Open Financial Exchange

Specification 2.1.1

May 1, 2006

© 2006 Intuit Inc., Microsoft Corp., CheckFree Corp. All rights reserved

10.6 Expired Models 183

Chapter 11 Banking 185

11.1 Consumer and Business Banking 185
11.1.1 Loan Data 185

11.2 Credit Card Data 185

11.3 Common Banking Aggregates 185
11.3.1 Banking Account <BANKACCTFROM> and <BANKACCTTO> 186
11.3.2 Credit Card Account <CCACCTFROM> and <CCACCTTO> 190
11.3.3 Bank Account Information <BANKACCTINFO> 191
11.3.4 Credit Card Account Information <CCACCTINFO> 192
11.3.5 Transfer Information <XFERINFO> 192
11.3.6 Transfer Processing Status <XFERPRCSTS> 195
11.3.7 Loan Account <LOANACCTFROM> and <LOANACCTTO> 196
11.3.8 Loan Account Information <LOANACCTINFO> 197
11.3.9 Loan Transaction Amount <LOANTRNAMT> 201

11.4 Downloading Transactions and Balances 202
11.4.1 Bank Statement Download 203
11.4.2 Credit Card Statement Download 205
11.4.3 Statement Transaction <STMTTRN> 209
11.4.4 Loan Statement Download 213
11.4.5 Loan Statement Transaction <LOANSTTMTTRN> 216
11.4.6 Amortization Schedule Download 219

11.5 Statement Closing Information 222
11.5.1 Statement Closing Download 222
11.5.2 Non-Credit Card Statement <CLOSING> 224
11.5.3 Credit Card Statement Closing Request <CCSTMTENDRQ> 225
11.5.4 Credit Card Statement Closing Response <CCSTMTENDRS> 226
11.5.5 Loan Statement End Request <LOANSTMTENDRQ> 229
11.5.6 Loan Statement End Response <LOANSTMTENDRS> 229
11.5.7 Status Codes 230
11.5.8 Loan Closing <LOANCLOSING> 231

11.6 Stop Check 234
11.6.1 Stop Check Add 235
11.6.2 Status Codes 238

11.7 Intrabank Funds Transfer 239
11.7.1 Intrabank Funds Transfer Addition 240
11.7.2 Intrabank Funds Transfer Modification 243

14.3.2 Account Inquiry 513
14.3.3 Service Activation 516
14.3.4 Service Status Update for Groups of Customers 518
14.4 Bill Delivery 522
14.4.1 Bill Delivery Process 522
14.4.2 Bill List Retrieval 522
14.4.3 Bill Detail Retrieval 536
14.4.4 Table Structure Definition 540
14.4.5 Delivery Notification 542
14.4.6 Bill Status Modification 545
14.5 Bill Payment 546
14.5.1 Remittance Information 546
14.5.2 Payee Identification 546
14.6 Bill Presentment E-Mail 547
14.6.1 Bill Presentment Mail Request <PRESMAILRQ> 548
14.6.2 Bill Presentment Mail Response <PRESMAILRS> 548
14.6.3 Status Codes <PRESMAILRS> 549
14.6.4 Request <PRESMAILSYNCRQ> 550
14.6.5 Response <PRESMAILSYNCRS> 551
14.7 Message Sets and Profile 552
14.7.1 Message Sets and Messages 552
14.7.2 Biller Directory Message Set Profile 556
14.7.3 Bill Delivery Message Set Profile 556
14.8 Bill Presentment Examples 558
14.8.1 Find Biller Examples 558
14.8.2 Enrollment Examples 565
14.8.3 Activation Example 567
14.8.4 Bill Delivery Examples 569

Chapter 15 Images 579

15.1 Overview 579
15.2 Image Download Options 579
15.2.1 Image Retrieval Method #1: ReferenceType OPAQUE 580
15.2.2 Image Retrieval Method #2: ReferenceType URL 582
15.2.3 Image Retrieval Method #3: ReferenceType FORMURL 583
15.3 Message Sets and Profile 584
15.3.1 Image Message Set Request Messages 584
15.3.2 Image Message Set Response Messages 584

15.3.3 Image Message Set Profile 584
15.4 Examples 586
15.4.1 Transaction Image Example 586
15.4.2 Statement Closing Image Example 590
15.4.3 Transaction FORMURL Example 593
15.4.4 Profile Request and Response Showing Image Support 595

Chapter 16 Automatic 1-Way OFX 599

16.1 RequestOFX Form 599
16.1.1 User Credentials <USERID>, <USERPASS>, <CRED2>, <CRED3>, <CLIENTUID> 600
16.1.2 Challenge Question Answers 600
16.1.3 Date Start and Date End <DTSTART>, <DTEND> 601
16.1.4 Include Flags <INCTRAN>, <INCPOS>, <INCBAL> 601
16.1.5 Account ID <ACCTID> 601
16.1.6 Account Type <ACCTTYPE> 602
16.2 HTTP Form Response 603
16.2.1 Response to RequestOFX Form 603
16.2.2 Status Responses 604
16.2.3 Challenge Response 604
16.3 Automatic 1-Way URL 605
16.4 Redirection and Cookies 605
16.5 Examples 606
16.5.1 Example - Failed Multiple Statements Request 606
16.5.2 Example - MFA Challenge Response 606
16.5.3 Banking and Credit Card Statements 607
16.5.4 Example - Banking Accounts with Multiple Accounts and Ampersand in Password 612
16.5.5 Example - Banking, Balance only 614
16.5.6 Investment Statement, Balances and Positions 616

Appendix A Status Codes 623

Appendix B Suggested Name Values for the Banking <BALLIST> 631
B.1 Usage Field Values 631
B.2 Name and Description Table 631

Appendix C Change History 635
C.1 OFX 1.6 to 2.0 635
C.1.1 Specification Changes by Chapter 635

1.1.1 Design Principles

The following principles were used in designing Open Financial Exchange:

- **Broad Range of Financial Activities** – Open Financial Exchange provides support for a ***broad*** range of financial activities. Open Financial Exchange 2.1.1 specifies the following services:
  - Bank statement download
  - Credit card statement download
  - Funds transfers including recurring transfers
  - Loan statement download
  - Consumer payments, including recurring payments
  - Business payments, including recurring payments
  - Brokerage and mutual fund statement download, including transaction history, current holdings, and balances for normal accounts and 401(k) accounts.
  - Bill presentment and payment
  - Tax form download, including 1099 and W2 (presented as a 2.0 addendum).
  - Image download for banking, credit cards, investments and loans
  - Automatic 1-Way OFX
- **Broad Range of Financial Institutions** – Open Financial Exchange supports communication with a ***broad*** range of financial institutions (FIs), including:
  - Banks
  - Brokerage houses
  - Merchants
  - Processors
  - Financial advisors
  - Government agencies
- **Broad Range of Front-End Applications** – Open Financial Exchange supports a ***broad*** range of front-end applications, including Web-based applications, covering all types of financial activities running on all types of platforms.
- **Extensible** – Open Financial Exchange has been designed to allow the easy addition of new services. Future versions will include support for many new services.
- **Open** – This specification is publicly available. You can build client and server applications using the Open Financial Exchange protocols independent of any specific technology, product, or company.
- **Multiple Client Support** – Open Financial Exchange allows a user to use multiple client applications to access the same data at a financial institution. With the popularity of the World Wide Web, customers are increasingly more likely to use multiple applications—either desktop-based or Web-based—to

perform financial activities. For example, a customer can track personal finances at home with a desktop application and occasionally pay bills while at work with a Web-based application. The use of data synchronization to support multiple clients is a key innovation in Open Financial Exchange.

Robust – Open Financial Exchange will be used for executing important financial transactions and for communicating important financial information. Assuring users that transactions are executed and information is correct is crucial. Open Financial Exchange provides robust protocols for error recovery.

Secure – Open Financial Exchange provides a framework for building secure online financial services. In Open Financial Exchange, security encompasses authentication of the parties involved, as well as secrecy and integrity of the information being exchanged.

Batch & Interactive – The design of request and response messages in Open Financial Exchange is for use in either batch or interactive style of communication. Open Financial Exchange provides for applying a single authentication context to multiple requests in order to reduce the overhead of user authentication.

International Support – Open Financial Exchange is designed to supply financial services throughout the world. It supports multiple currencies, country-specific extensions, and different forms of encoding such as UNICODE.

Platform Independent –Open Financial Exchange can be implemented on a wide variety of front-end client devices, including those running on Windows, Macintosh, or UNIX. It also supports a wide variety of Web-based environments and messaging frameworks, including those using SOAP, HTML, Java, JavaScript, or ActiveX. Similarly on the back end, Open Financial Exchange can be implemented on a wide variety of server systems, including those running UNIX, Windows NT, or OS/2.

Transport Independent – Open Financial Exchange is independent of the data communication protocol used to transport the messages between client and server computers. Open Financial Exchange 2.1.1 uses HTTP.

1.2 Open Financial Exchange at a Glance

The design of Open Financial Exchange is as a client and server system. An end-user uses a client application to communicate with a server at a financial institution. The form of communication is requests from the client to the server and responses from the server back to the client.

Open Financial Exchange uses the Internet Protocol (IP) suite to provide the communication channel between a client and a server. IP protocols are the foundation of the public Internet and a private network can also use them.

1.2.1 Data Transport

Clients use the HyperText Transport Protocol (HTTP) to communicate to an Open Financial Exchange server. The World Wide Web throughout uses the same HTTP protocol. In principle, a financial institution can use any off-the-shelf web server to implement its support for Open Financial Exchange.

To communicate by means of Open Financial Exchange over the Internet, the client must establish an Internet connection. This connection can be a dial-up Point-to-Point Protocol (PPP) connection to an Internet Service Provider (ISP) or a connection over a local area network that has a gateway to the Internet.

Clients use the HTTP POST command to send a request to the previously acquired Uniform Resource Locator (URL) for the desired financial institution. The URL presumably identifies a Common Gateway Interface (CGI) or other process on an FI server that can accept Open Financial Exchange requests and produce a response.

1.3.7 Tag

Tag is the generic name for either a start tag or an end tag. A *start tag* consists of an *element* or *aggregate* name surrounded by angle brackets. An *end tag* is the same as a start tag, with the addition of a forward slash immediately preceding the name. For example, the start tag for the aggregate named FOO looks like this:

```
<FOO>
```

The end tag for the same aggregate looks like this:

```
</FOO>
```

1.3.8 Element

An OFX document contains one or more *elements*. An element is some data bounded by a leading start tag and a trailing end tag. For example, an element named BAZ, containing data "bar," looks like this:

```
<BAZ>bar</BAZ><!-- An element ended by its own end tag-->
```

An OFX *element* must contain data (not just white space) and may *not* contain other elements. This is a refinement to the XML definition of an element which is more generic. An XML element containing other elements is defined in OFX as an *aggregate*. OFX specifically disallows empty elements and elements with *mixed* content.

1.3.9 Aggregate

An *aggregate* is a collection of elements and/or other aggregates. An aggregate may not contain any data itself, but rather contains elements containing data, and/or recursively contains aggregates.

OFX includes very few empty aggregates and clients and servers should not send an aggregate without content. In general, the entire aggregate should be left out of a request or response file when its (optional) content is missing. The few exceptions to these rules (such as <SECLISTRS>, described in section 13.8.3.3) are called out in the relevant sections of this document.

1.3.15 Message Set

Message sets are collections of messages. Generally they form all or part of a *service* (as defined in section 1.3.6). OFX utilizes these smaller groupings when wrapping request or response transactions, profiling server support for the wrappers and describing individual messages. The BANKSVC service, for example, is broken into the BANKMSGSET, CREDITCARDMSGSET, INTERXFERMSGSET and WIREXFERMSGSET message sets.

Please refer to section 2.4.5 , "Message Sets and Version Control" for additional information about message sets.

1.4 OFX Versions

There are several distinct versions of OFX clients and servers.

Version 1.0.2 supports any or all version 1 message sets except Bill Presentment. These message sets are defined by the OFX 1.0.2 Document Type Definition (DTD), which is used for parsing. Applications that conform to this version are referred to as 1.0.2 clients and 1.0.2 servers.

Version 1.0.3 extends 1.0.2 by adding support for Muli-Factor Authentication.

Version 1.5.1 supports all version 2 message sets, Bill Presentment, and all version 1 message sets. Because it supports all message sets, the OFX 1.5.1 DTD can be used to create and support OFX 1.0.2 and/or OFX 1.5.1 clients and servers.

Version 1.6 DTD supports all message sets available in the OFX 1.5.1 DTD. It adds specific enhancements to some of the aggregates. All of those enhancements are optional and should not be used by a client unless the server indicates support in its FI Profile. Applications that conform to this version are referred to as 1.6 clients and 1.6 servers. The OFX 1.6 DTD fully incorporates the OFX 1.0.2 and 1.5.1 message sets, so it can be used to support both 1.0.2 and 1.5.1 applications.

Version 2.0 supports all V1 message sets available in the OFX 1.6 DTD. It adds support for 401(k) investment statement download. The Tax OFX addendum to OFX 2.0 adds support for 1099 and W2 download. An important change for 2.0 is that it adds the requirement of XML compliance to OFX 2.0 clients and servers. See chapter 2 for more information.

Version 2.0.1 extends 2.0 by adding support for investment transaction reversals, by adding the capability to include an arbitrary list of balances in statement and credit card statement downloads, and by adding the ability to specify bill publisher information in the payment message.

Version 2.0.2 adds clarification to 2.0.1 as well as fixes some minor documentation bugs.

Version 2.0.3, an extension of 2.0.2, adds Multi-Factor Authentication (MFA) to the Signon Message Set and changes to the Profile Message Set to support MFA.

1.5 Conventions

The conventions used in the element and aggregate descriptions include the following:

- Required elements and aggregates are in **bold**. Regular face indicates elements and aggregates that are optional. Required means that a client must always include the element or aggregate in a request, and a server must always include the element or aggregate in a response.
- Required elements and aggregates occur once unless noted as one or more in the description, in which case the specification allows multiple occurrences.
- Optional elements and aggregates occur once if present unless noted as zero or more in the description, in which case the specification allows multiple occurrences.
- Character fields are identified with a data type of "*A-n"*, where n is the maximum number of allowed Unicode characters.

  Note: *n* refers to the number of characters in the resultant string. Each multi-byte or encoded character counts as a single character. UTF-8 encodes "high" Latin-1 characters (decimal 128-255) using two bytes, and double-byte characters using three bytes. In addition, XML encodes ampersands, less-than symbols, greater-than symbols, and spaces (where required) using multi-character escape strings (see section 2.3.1.1). Therefore, an element of type A-40 may require more than 40 bytes in a UTF-8-encoded XML stream.

- N-*n* identifies an element of numeric type where *n* is the maximum number of characters in the value. Values of this type are generally whole numbers, but the data type allows negative numbers. OFX includes a few fixed-position numeric values (such as <APPVER>, see section 2.5.1.2) called out in the text. In all cases, elements of this type may contain only the characters 0 through 9 and - (hyphen, the negative sign indicator). So an element of type "N-6" may take values from -99999 to 999999. The value "0000000" would be illegal for an N-6 element. White space is not allowed within the numeric value. Leading zeroes are allowed, but discouraged except where noted in the text. For example, a <MIN> element containing zero might be sent as "<MIN>0", "<MIN>00", "<MIN> 0", but not "<MIN>0 0".
- Common value types, such as a dollar amount, are referenced by name. Chapter 3, "Common Aggregates, Elements, and Data Types" lists value types that are referenced by name.

CHAPTER 2 STRUCTURE

This chapter describes the basic structure of an Open Financial Exchange request and response. Structure includes headers, basic syntax, and the Signon request and response. This chapter also describes how Open Financial Exchange encodes external data, such as bit maps.

Open Financial Exchange data consists of a declaration plus one Open Financial Exchange data block. This block consists of a signon message and zero or more additional messages. When sent over the Internet using HTTP, standard HTTP and (optionally) multipart MIME headers and formats surround the Open Financial Exchange data. A simple file that contained only Open Financial Exchange data would have the following form:

```
HTTP headers
MIME type application/x-ofx
XML declaration
Open Financial Exchange declaration
Open Financial Exchange XML block
```

A more complex file that contained additional Open Financial Exchange data would have this form:

```
HTTP headers
MIME type multipart/x-mixed-replace; boundary =XYZZY24x7
--XYZZY24x7
MIME type application/x-ofx
XML declaration
Open Financial Exchange declaration
Open Financial Exchange XML block

--XYZZY24x7
  MIME type image/jpeg
    FI logo
--XYZZY24x7--
```

Version 1.0.2 of the Open Financial Exchange specification did not specify how to properly separate the various components of an OFX request. In particular, separation of the HTTP headers, the MIME attachments, the OFX declaration, the OFX header elements, and the OFX SGML block.

OFX 1.0.2 clients used a mix of LF and CRLF constructs and OFX 1.0.2 servers handled either linefeed (LF) or carriage return/line feed (CRLF), but not often both. In the future, it is expected that 1.0.2 servers will be upgraded to handle both CRLF and LF.

OFX 2.1.1 clients and servers are expected to follow standard XML 1.0 conventions regarding the use of CR and LF. XML 1.0 is an accepted World Wide Web Consortium (W3C) recommendation.

http://www.w3.org	(W3C home page)
http://www.w3.org/TR/REC-xml	(XML 1.0 recommendation)

The text has been included below for ease of reference:

2.1 HTTP Headers

Data delivered by way of HTTP places the standard HTTP result code on the first line. HTTP defines a number of status codes. Servers can return any standard HTTP result. However, FIs should expect clients to collapse these codes into the following three cases:

	Meaning	*Action*
200	OK	The request was processed and a valid Open Financial Exchange result is returned.
400s	Bad request	The request was invalid and was not processed. Clients will report an internal error to the user. Invalid requests include: general HTTP transport errors, XML formatting errors, invalid OFX syntax, and invalid data values. This error should not appear for request files the server is able to parse.
500s	Server error	The server is unavailable. Clients should advise the user to retry shortly.

Note: The server must return a code in the 400s for any problem that prevents it from processing the request file. Processing problems include failures relating to security, communication, parsing, or the Open Financial Exchange declaration (for example, the client requested an unsupported language). For content errors such as wrong USERPASS or invalid account, the server must return a valid Open Financial Exchange response along with code 200. If a communication time-out error occurs while an OFX server and a back-end server are communicating to fill a request, then the server MUST return a code in the 500s.

Open Financial Exchange requires the following HTTP standard headers:

Code	*Value*	*Explanation*
Content-type	application/x-ofx	The MIME type for Open Financial Exchange
Content-length	length	Length of the data after removing HTTP headers

When responding with multipart MIME (likely only if the request included a <GETMIMERQ> request), the main type will be multipart/x-mixed-replace; one of the parts will use application/x-ofx.

2.2 Open Financial Exchange File Format

The contents of an Open Financial Exchange file consists of simple declarations followed by contents defined by those declarations.

The standard XML declaration must come first. This Processing Instruction (PI) includes an option to specify the version of XML being used, and may include options to show such things as the encoding declaration, and the standalone status of the document.

The XML declaration takes the following form:

> **Note:** White space requirements are not imposed by this specification beyond the standard XML 1.0 conventions; therefore, the white space formatting shown in these examples is not required.
> The encoding and standalone attributes are shown below for completeness and may be omitted. See XML 1.0 for a description of the default handling for these attributes.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

The OFX declaration must come next in the file. This PI identifies the contents as an Open Financial Exchange file and provides the version number of the Open Financial Exchange declaration itself (not the version number of the contents). The Open Financial Exchange PI contains the following attributes:

```
OFXHEADER
VERSION
SECURITY
OLDFILEUID
NEWFILEUID
```

All these attributes are required. "NONE" should be returned if client or server does not make use of an individual attribute, e.g., OLDFILEUID="NONE".

The entire declaration takes the form:

```
<?OFX OFXHEADER="200" VERSION="211" SECURITY="NONE" OLDFILEUID="NONE"
NEWFILEUID="NONE"?>
```

2.3 XML Details

2.3.1 Compliance

XML is the basis for Open Financial Exchange 2.0 and later. To enable OFX clients and servers to use off-the-shelf XML parsers, OFX 2.1.1 is fully XML compliant. Therefore, in contrast to the guidelines for OFX 1.6 and below, unrecognized tags may not be present. If clients and servers wish to extend OFX with private tags and true DTD validation is necessary, a modified OFX DTD which contains those new tags must be passed along with the OFX document.

2.3.1.1 Special Characters

Special characters in OFX 2.1.1 are handled according to the XML standard. Characters such as '<', '>', '&', '', and '"' are predefined in XML. Other character strings with many special characters should be enclosed in a CDATA section.

> **Note:** The space macro ( ) should be used if leading or trailing blanks are meant to be preserved as part of a data element's value. Alternatively, a CDATA block may be used to force the handling of leading or trailing spaces. No special formatting of space characters in the middle of an element's text value is needed.

2.4 Open Financial Exchange XML Structure

2.4.1 Overview

Open Financial Exchange hierarchically organizes request and response blocks:

```
Top Level <OFX>
  Message Set and Version <xxxMSGSVn>
    Synchronization Wrappers <xxxSYNCRQ>, <xxxSYNCRS>
      Transaction Wrappers <xxxTRNRQ>, <xxxTRNRS>
        Specific requests and responses
```

The following sections describe these levels.

2.4.2 Case Sensitivity

OFX requires upper case letters for tag names and enumerated values. In the example below, <SEVERITY> is an element with an enumerated value and <MESSAGE> is an element with a value that is not enumerated.

```
<STATUS>
  <CODE>2000</CODE>
```

```
   <SEVERITY>ERROR</SEVERITY>
   <MESSAGE>General Error</MESSAGE>
</STATUS>
```

2.4.3 Top Level

An Open Financial Exchange request or response has the following top-level form:

	Description
<OFX>	Opening tag
<SONRQ> or **<SONRS>**	Required signon request or response. See section 2.5.1.
... Open Financial Exchange requests or responses ...	0 or more transaction requests and responses inside appropriate message set aggregates
</OFX>	Closing tag for the Open Financial Exchange record

This chapter specifies the order of requests and responses.

A single file MUST contain only one OFX block.

2.4.4 Messages

A message is the unit of work in Open Financial Exchange. It refers to a request and response pair, and the status codes associated with that response. For example, the message to download a bank statement consists of the request <STMTRQ> and the response <STMTRS>.

OFX uses several common message types to perform specific functions. Within OFX, the following naming conventions are used, where the general *xxx* messages may be:

- Basic (or Add) request <*xxx*RQ> and response <*xxx*RS>
- Modify request <*xxx*MODRQ> and response <*xxx*MODRS>
- Delete request <*xxx*DELRQ> and response <*xxx*DELRS>
- Cancel request <*xxx*CANRQ> and response <*xxx*CANRS> (these pairs may also be named <*xxx*CANCRQ> and <*xxx*CANCRS)

2.4.4.1 Basic and Add Messages

The *basic* OFX message has a name structure of <*xxx*RQ>/<*xxx*RS>. It is used for read actions of a specific object (such as a bank statement using <STMTENDRQ>). It is encapsulated in a transaction wrapper <*xxx*TRNRQ> or <*xxx*TRNRS> (therefore, <STMTENDTRNRQ> and <STMTENDTRNRS> in the example above).

The *add* OFX message, like the Basic message, has a name structure of <*xxx*RQ>/<*xxx*RS>. It is used to create a new instance of object *xxx* (such as creating a new payment using <PMTRQ>). It is encapsulated in a transaction wrapper <*xxx*TRNRQ> or <*xxx*TRNRS> (therefore, <PMTTRNRQ> and <PMTTRNRS> in the example above).

2.4.4.2 Modify Message

The *modify* OFX message has a name structure of <*xxx*MODRQ>/<*xxx*MODRS>. It is used to modify an existing instance of object *xxx* (such as modifying an existing payment using <PMTMODRQ>). It is encapsulated in a transaction wrapper <*xxx*TRNRQ> or <*xxx*TRNRS> (therefore, <PMTTRNRQ> and <PMTTRNRS> in the example above).

The <*xxx*MODRQ> request contains the **complete replacement** data for an existing object *xxx*. Therefore, **both changed and unchanged elements** must be included in the request.

2.4.4.3 Delete and Cancel Messages

The *delete* and *cancel* OFX messages have a name structure of <*xxx*DELRQ>/<*xxx*DELRS> and <*xxx*CANRQ>/<*xxx*CANRS> or <*xxx*CANCRQ>/<*xxx*CANCRS>, respectively. They are used to delete an existing instance of object *xxx* (such as deleting a payee from a payee list using <PAYEEDELRQ>), or to cancel an existing scheduled object (such as canceling a pending payment using <PMTCANCRQ>). They are encapsulated in a transaction wrapper <*xxx*TRNRQ> or <*xxx*TRNRS> (therefore, <PAYEETRNRQ> and <PMTTRNRQ> in the examples above).

2.4.4.4 Inquiry Message

The *inquiry* OFX message sometimes has a name structure of <*xxx*INQRQ>/<*xxx*INQRS>. It is used to search for and/or gain information about (an) existing object(s) *xxx* (such as finding one or more existing

Note: A server is not obligated to support filtering on all selection criterion elements. If a server chooses not to support a particular element as a selection criterion, it **must** treat that element as if it were not present. That is, the server must return the appropriate record set for the elements on which it does support filtering. As a result, clients should be prepared to receive records outside the scope of the selection criteria submitted in the request.

Note: Many inquiry messages do not presently follow the naming conventions detailed above. They may be named <xxxINFORQ>/<xxxINFORS> (<ACCTINFORQ> and <ACCTINFORS> for example) or without reference to an obvious convention (<PRESLISTRQ> and <PRESLISTRS> for example).

2.4.5 Message Sets and Version Control

Message sets are collections of messages. Generally they form all or part of what a user would consider a *service*, something for which they might have signed up, such as "banking." Message sets are the basis of version control, routing, and security. They are also the basis for the required ordering in Open Financial Exchange files.

Within the OFX block, OFX organizes messages by message set. Message sets follow these rules:

- A request file may include at most one message set wrapper of each type.
- All messages within any message set must be from the same version of that message set.
- Servers must respond using the same message sets and versions as sent in the request file. For example, if <SIGNUPMSGSRQV1> appears in the request file, <SIGNUPMSGSRSV1> must appear in the response file. There is one exception to this rule: servers may return the <SECLISTMSGSRSV1> wrapper (see 13.7.2 and 13.8.4) in response to an investment statement download request that may or may not include <SECLISTMSGSRQV1>.

2.4.5.1 Message Set Aggregates

For each message set of *xxx* and version *n*, there are two aggregates, one for requests <*xxx*MSGSRQV*n*>) and one for responses <*xxx*MSGSRSV*n*>. All of the messages from that message set must be enclosed in the appropriate message set aggregate. In the following example, the Open Financial Exchange block contains a signon request inside the signon message set, and two statement requests and a transfer request inside the bank message set.

```
<OFX>
  <SIGNONMSGSRQV1>              <!-- Signon message set -->
    <SONRQ>                     <!-- Signon message -->
    ...
    </SONRQ>
  </SIGNONMSGSRQV1>

  <BANKMSGSRQV1>                <!-- Banking message set -->
```

```
    <STMTTRNRQ>                 <!-- Statement request -->
    ...
    </STMTTRNRQ>
    <STMTTRNRQ>                 <!-- Another stmt request -->
    ...
    </STMTTRNRQ>
    <INTRATRNRQ>                <!-- Intrabank transfer request -->
    ...
    </INTRATRNRQ>
  </BANKMSGSRQV1>
</OFX>
```

2.4.5.2 Message Set Ordering

Message sets must appear in the following order:

- Signon
- Signup
- Banking
- Credit card statements
- Loan statements
- Investment statements
- Interbank funds transfers
- Wire funds transfers
- Payments
- General e-mail
- Investment security list
- Biller Directory
- Bill Delivery
- FI Profile
- Image download

  Note: Image download is an exception to the above. An OFX file containing the Image message set includes only the Signon message set. No other message sets may be present in the file. See Chapter 15, "Image Download", for details.

The definition of each message set can further prescribe an order of its messages within that message set.

2.4.5.3 Message Set Version Numbers

The following table lists each message set, along with its aggregate name and the DTD/XML Schema versions that support it.

Note: Starting with OFX 2.0, a DTD is no longer maintained. Instead, an XML Schema is maintained.

	Message Set Aggregate	*DTD/Schema Support*
Signon	<SIGNONMSGSETV1>	1.0.2, 1.0.3, 1.5.1, 1.6, 2.0, 2.0.1, 2.0.2, 2.0.3, 2.1, 2.1.1
Signup	<SIGNUPMSGSETV1>	1.0.2, 1.0.3, 1.5.1, 1.6, 2.0, 2.0.1, 2.0.2, 2.0.3, 2.1, 2.1.1
Banking	<BANKMSGSETV1>	1.0.2, 1.0.3, 1.5.1, 1.6, 2.0, 2.0.1, 2.0.2, 2.0.3, 2.1, 2.1.1
Credit Card Statements	<CREDITCARDMSGSETV1>	1.0.2, 1.0.3, 1.5.1, 1.6, 2.0, 2.0.1, 2.0.2, 2.0.3, 2.1, 2.1.1
Loan Statements	<LOANMSGSETV1>	2.1, 2.1.1
Investment Statements	<INVSTMTMSGSETV1>	1.0.2, 1.0.3, 1.5.1, 1.6, 2.0, 2.0.1, 2.0.2, 2.0.3, 2.1, 2.1.1
Interbank Funds Transfers	<INTERXFERMSGSETV1>	1.0.2, 1.0.3, 1.5.1, 1.6, 2.0, 2.0.1, 2.0.2, 2.0.3, 2.1, 2.1.1
Wire Funds Transfers	<WIREXFERMSGSETV1>	1.0.2, 1.0.3, 1.5.1, 1.6, 2.0, 2.0.1, 2.0.2, 2.0.3, 2.1, 2.1.1
Payments	<BILLPAYMSGSETV1>	1.0.2, 1.0.3, 1.5.1, 1.6, 2.0, 2.0.1, 2.0.2, 2.0.3, 2.1, 2.1.1
General e-mail	<EMAILMSGSETV1>	1.0.2, 1.0.3, 1.5.1, 1.6, 2.0, 2.0.1, 2.0.2, 2.0.3, 2.1, 2.1.1
Investment security list	<SECLISTMSGSETV1>	1.0.2, 1.0.3, 1.5.1, 1.6, 2.0, 2.0.1, 2.0.2, 2.0.3, 2.1, 2.1.1
Biller directory	<PRESDIRMSGSETV1>	1.0.2, 1.0.3, 1.5.1, 1.6, 2.0, 2.0.1, 2.0.2, 2.0.3, 2.1, 2.1.1
Bill delivery	<PRESDLVMSGSETV1>	1.0.2, 1.0.3, 1.5.1, 1.6, 2.0, 2.0.1, 2.0.2, 2.0.3, 2.1, 2.1.1
FI Profile	<PROFMSGSETV1>	1.0.2, 1.0.3, 1.5.1, 1.6, 2.0, 2.0.1, 2.0.2, 2.0.3, 2.1, 2.1.1
Image download	<IMAGEMSGSETV1>	2.1, 2.1.1

Note: For each message set that it is supporting, a financial institution must indicate which version numbers of that message set it supports. The financial institution includes the message

set version number in the <MSGSETCORE> aggregate of the FI profile. For more information about the FI profile, refer to Chapter 7, "FI Profile." OFX 2.1.1 servers should use version number 1.

2.4.6 Transactions

Other than the signon message, each request is made as a transaction. Transactions contain a client-assigned globally-unique ID, optional client-supplied pass-back data, and the request aggregate. A transaction similarly wraps each response. The response transaction returns the client ID sent in the request, along with a status message, the pass-back data if present, and the response aggregate. This technique allows a client to track responses against requests. Section 3.1.2 provides more information about the format of information exchanged by the client and server.

The <STATUS> aggregate, defined in Chapter 3, "Common Aggregates, Elements, and Data Types," provides feedback on the processing of the request. If the <SEVERITY> of the status is ERROR, the server provides the transaction response without the nested response aggregate. Otherwise, the response must be complete even though a warning might have occurred.

Clients can send additional information in <CLTCOOKIE> that servers will return in the response. This allows clients that do not maintain state, and thus do not save <TRNUID>s, to cause some additional descriptive information to be present in the response. For example, a client might identify a request as relating to a user or a spouse.

<CLTCOOKIE> must only be returned by the server in the initial response to the client (and any crash recovery from that response). The <CLTCOOKIE> should not be present in a sync response, except for those transactions whose requests were wrapped in the sync request.

In some countries, some banks may require that a customer-supplied authorization number be included to authenticate certain kinds of individual transactions such as payment requests. For those banks, the <TAN> element passes this information to servers.

Note that if a <CLTCOOKIE> is given to an OFX server in a request, the OFX server is required to return it. This return of the <CLTCOOKIE> will necessitate server-side storage of <CLTCOOKIE> data. In the case of an OFX client getting a <CLTCOOKIE> that it didn't send in a request, the default behavior is to ignore it.

2.4.6.1 Transaction Wrapper

With the exception of the <SONRQ>/<SONRS> message, each message has a corresponding *transaction wrapper*. For requests, the transaction wrapper adds a transaction unique ID <TRNUID>. For responses, the transaction wrapper adds the same transaction unique ID <TRNUID> (an echo of that found in the request), plus a <STATUS> aggregate.

The *transaction wrapper* has a name structure of <*xxx*TRNRQ>/<*xxx*TRNRS>. A transaction wrapper pair encapsulates a single message (<*xxx*RQ>/<xxxRS>, <*xxx*MODRQ>/<*xxx*MODRS>, etc.).

While the same name may be used for addition, modification and deletion messages, a single transaction

A typical request is as follows:

	Description
<*xxx*TRNRQ>	Transaction-request aggregate
<TRNUID>	Client-assigned globally-unique ID for this transaction, *trnuid*
<CLTCOOKIE>	Data to be echoed in the transaction response, *A-32*
<TAN>	Transaction authorization number; used in some countries with some types of transactions. The FI Profile defines messages that require a <TAN>, *A-80*
Request aggregate	Aggregate for the request
</*xxx*TRNRQ>	

A typical response is as follows:

	Description
<*xxx*TRNRS>	Transaction-response aggregate
<TRNUID>	Client-assigned globally-unique ID for this transaction, *trnuid*
<STATUS>	Status aggregate
</STATUS>	
<CLTCOOKIE>	Client provided data, *A-32*
Response aggregate	Aggregate for the response
</*xxx*TRNRS>	

List of status code values for the <CODE> element of <STATUS>:

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2022	Invalid TAN (ERROR)

2.4.7 Synchronization Wrapper

The *synchronization wrapper* has a name structure of <*xxx*SYNCRQ>/<*xxx*SYNCRS>. It contains synchronization parameters and optionally encapsulates one or more transaction wrappers. For details on the use of synchronization wrappers, see Chapter 6.

When embedded transactions are not present, the synchronization request contains no transaction wrappers. If the client is up to date when the server processes such a request, the synchronization response also contains no transaction wrappers.

> **Note:** If a request/response is a sync request/response only, the transaction wrapper and request that it wraps are omitted.

2.4.8 Message Set Wrapper

The profile *message set wrappers* have a name structure of <*xxx*MSGSET> and <*xxx*MSGSETV1>.

The request and response *message set wrappers* have a name structure of <*xxx*MSGSRQV*n*> and <*xxx*MSGSRSV*n*> respectively. For OFX 2.1.1, "*n*" must be "1". This number indicates the version of the message set used by the contained messages.

2.5 The Signon Message Set

The Signon message set includes the signon message, USERPASS change message, challenge message, and multi-factor authentication (MFA) challenge message, which must appear in that order. The <SIGNONMSGSRQV1> and <SIGNONMSGSRSV1> aggregates wrap the message.

2.5.1 Signon <SONRQ> and <SONRS>

The signon record identifies and authenticates a user to an FI. It also includes information about the application making the request, because some services might be appropriate only for certain clients. Every Open Financial Exchange block contains exactly one <SONRQ>. Every response must contain exactly one <SONRS> record. Use of Open Financial Exchange presumes that FIs authenticate each customer and then give the customer access to one or more accounts or services. Authentication of a <SONRQ> is required, even when in Error Recovery. If passwords are specific to individual services or accounts, a separate Open Financial Exchange request must be made for each user ID or password required. This will not necessarily be in a manner visible to the user. Note that some situations, such as joint accounts or business accounts, will have multiple user IDs and multiple passwords that can access the same account.

FIs assign user IDs for the customer. The client must not make any assumptions about the syntax of the ID, add check-digits, or do similar processing. To ensure security and help prevent identity fraud, Financial Institutions are discouraged from using Social Security Number for Customer ID or PIN/Password.

To improve server efficiency in handling a series of Open Financial Exchange request files sent over a short period of time, clients can request that a server return a <USERKEY> in the signon response. If the server provides a user key, clients will send the <USERKEY> instead of the user ID and password in subsequent sessions, until the <USERKEY> expires. This allows servers to authenticate subsequent requests more quickly. Servers must accept a <GENUSERKEY> element in a <SONRQ>. However, a server may decide <USERKEY> does not afford sufficient security and may optionally not return a <USERKEY> in the <SONRS>.

The client returns <SESSCOOKIE> if the server sent one in a previous <SONRS>. Servers can use the value of <SESSCOOKIE> to track client usage but cannot assume that all requests come from a single client, nor can they deny service if they did not expect the returned cookie. Use of a backup file, for example, could lead to an unexpected <SESSCOOKIE> value that nevertheless should not stop a user from connecting.

A client may use an anonymous form of <USERID> and <USERPASS> on those rare occasions when a server need not authenticate the <SONRQ>. The only present situations in this class are first-time <PROFRQ>, <FINDBILLERRQ>, and all <ENROLLRQ> transactions. Any request sent by the client after a successful <ENROLLRQ> response (or out of band enrollment) for the service must provide the

2.5.1.1 Multi-Factor Authentication (MFA)

In addition to the previously existing method of authentication in OFX (username/password), OFX 2.1.1 introduces several ways for OFX to support additional authentication factors. This support is achieved through the addition of new tags in the OFX signon request and through a new message, the <MFACHALLENGERQ>/<MFACHALLENGERS> request/response.

The signon request includes several new fields that financial institutions can optionally require for user authentication.

2.5.1.1.1 Client Unique ID <CLIENTUID>

OFX servers can require OFX clients to include a client ID in each signon request. This client ID should be unique to the installation of the client software, but the method that the ID is generated is left up to the client. The server can specify that this field is required using the <CLIENTUIDREQ> tag in the applicable <SIGNONINFO> section of the profile. Servers should expect that users may connect via OFX from multiple locations and may need to associate more than one <CLIENTUID> value with their <USERID>.

The client may make this value user discoverable, so that the user can register the client ID with financial institutions.

2.5.1.1.2 Additional User Credentials <USERCRED1>, <USERCRED2>

The signon request contains two new user credential ID fields, <USERCRED1> and <USERCRED2>. Servers use the applicable <SIGNONINFO> aggregate in the profile to specify if one or both of these fields are required. The presence of <USERCRED1LABEL> and <USERCRED2LABEL> in <SIGNONINFO> specifies that these tags are required and also gives labels for these fields. OFX clients should use these labels to prompt the user for necessary signon information. For instance, a server may require <USERCRED1> and would specify its label as “Mother’s maiden name.”

Servers should assume that profile requests are made very infrequently. If the user credential ID or label is expected to change frequently, the MFACHALLENGE message is the most appropriate method to use. Use of <USERCRED1> and <USERCRED2> should be reserved for questions/prompts that will change rarely, if at all.

2.5.1.1.3 One-Time Authentication Token <AUTHTOKEN>

This authentication token is intended to be used in conjunction with the client ID functionality described in Section 2.5.1.1.1 although this is not required. By providing the user with this one-time value out of band and then requiring (and validating) it on either the initial (for first time setup) or next session (for existing users) an institution may establish the client ID received in the session as a “known” client ID value that is directly associated with the user.

The signon request contains an additional field, <AUTHTOKEN>. Servers use the Signon Realm’s <AUTHTOKENFIRST> tag to specify that the client is required to send this credential during the initial signon session. A server may also indicate that this credential is required on the next session by returning a 15512 error code in a signon response. If this credential is required by the OFX server under either of the

above conditions then the server must also use the Signon Realm's profile tags to specify a label for this value (<AUTHTOKENLABEL>) as well as a standard URL (<AUTHTOKENINFOURL>) where the one time authentication token is either directly provided to the user (e.g. they login to the institution's standard web banking system and request a credential) or information on how to acquire the credential is given (e.g. they are instructed to contact customer support).

During initial client software setup (if <AUTHTOKENFIRST> is set to Y in the profile) or upon receipt of a 15512 error code during a session the client software must inform the user that their institution requires additional information for the next session and display the <AUTHTOKENINFOURL> to the end-user. The client software must prompt the user for the entry of the <AUTHTOKEN> value using the <AUTHTOKENLABEL> as a caption for the data entry field.

This authentication token mechanism is intended for use on an infrequent basis and in conjunction with client ID functionality. If additional authentication would be required on a regular basis then the MFACHALLENGE messages would be a more appropriate implementation.

2.5.1.1.4 Access Key <ACCESSKEY>

If the client and server support the MFACHALLENGE request/response and/or the authentication token functionality, the signon request may include the <ACCESSKEY> tag. When provided by the server, the client will send the last value of the <ACCESSKEY> it has received.

2.5.1.2 Signon Request <SONRQ>

Unlike other requests, the signon request <SONRQ> does not appear within a transaction wrapper.

Tag	*Description*
<SONRQ>	Signon-request aggregate
<DTCLIENT>	Date and time of the request from the client computer, *datetime* This value should reflect the time (according to the client machine) when the request file is sent to the server, not the (original) creation time of the request file. While not required for existing software, OFX 2.1.1 clients must comply with this rule. This clarification is particularly important in error recovery situations in which the request file may be sent to the server after its initial creation.
User identification. Either <USERID> and <USERPASS> or <USERKEY>, but not both.	
<USERID>	User identification string. To ensure security and help prevent identity fraud, Financial Institutions are discouraged from using Social Security Number for Customer ID or PIN/Password. *A-32*

Tag	Description
<USERPASS>	User password on server, *A-171* **Note:** The maximum clear text length of USERPASS is 32 characters: a client must not send a longer password. However, when using Type 1 security, the encrypted value may extend to 171 characters.
<USERKEY>	Log in using previously authenticated context, *A-64*
<GENUSERKEY>	Request server to return a USERKEY for future use, *Boolean*
<LANGUAGE>	Requested language for text responses, *language*
<FI>	Financial-Institution-identification aggregate **Note:** The client will determine out-of-band whether a FI aggregate should be used and if so, the appropriate values for it. If the FI aggregate is to be used, then the client should send it in every request, and the server should return it in every response.
</FI>	
<SESSCOOKIE>	Session cookie value received in previous <SONRS>, not sent if first login or if none sent by FI, *A-1000*
<APPID>	ID of client application, *A-5*
<APPVER>	Version of client application, (6.00 encoded as 0600), *N-4*
<CLIENTUID>	Unique ID identifying OFX client, *A-36*
<USERCRED1>	Additional user credential required by server, *A-171* *Note: the effective size of USERCRED1 is A-32. However, if Type 1 security is used, then the actual field length is A-171.*
<USERCRED2>	Additional user credential required by server, *A-171* *Note: the effective size of USERCRED2 is A-32. However, if Type 1 security is used, then the actual field length is A-171.*
<AUTHTOKEN>	Authentication token required for this signon session only. Credential is provided to the user out of band, *A-171* *Note: the effective size of AUTHTOKEN is A-32. However, if Type 1 security is used, then the actual field length is A-171.*
<ACCESSKEY>	Access key value received in prevous <SONRS>, not sent if first login or none sent by FI, *A-1000*
<MFACHALLENGEA>	MFA challenge question/answer aggregates (0 or more). See section 2.5.4.5
</MFACHALLENGEA>	
</SONRQ>	

2.5.1.3 Signon Response <SONRS>

Unlike other responses, the signon response <SONRS> does not appear within a transaction wrapper.

Note: A client should use <DTPROFUP> and <DTACCTUP> only when the service provider that originated <SONRS> is the same provider that is specified by <SPNAME> in the profile message set. A client can determine if the service provider is the same by comparing the value of <SPNAME> in the appropriate message set with the value for <SPNAME> in the profile message set.

	Description
<SONRS>	Record-response aggregate
<STATUS>	Status aggregate, see section 3.1.5. See list of possible code values in section 2.5.1.4
</STATUS>	
<DTSERVER>	Date and time of the server response, *datetime*
	This value should reflect the time (according to the server) when the response file was originally created. While not required for existing software, OFX 2.1.1 servers must comply with this rule. This clarification is particularly important in error recovery situations: The server should (must for OFX 2.1.1 servers) return the time the request was first processed. If the previous attempt failed after transactions were processed, <DTSERVER> in the response file would reflect that processing time.
<USERKEY>	Use user key instead of USERID and USERPASS for subsequent requests. TSKEYEXPIRE can limit lifetime. *A-64*
<TSKEYEXPIRE>	Date and time that USERKEY expires, *datetime*
<LANGUAGE>	Language used in text responses, *language*
<DTPROFUP>	Date and time of last update to profile information for any service supported by this FI (see Chapter 7, "FI Profile"), *datetime*
<DTACCTUP>	Date and time of last update to account information (see Chapter 8, “Activation & Account Information”), *datetime*
<FI>	Financial-Institution-identification aggregate
	Note: The client will determine out-of-band whether an FI aggregate should be used and, if so, the appropriate values for it. If the FI aggregate is to be used, then the client should send it in every request, and the server should return it in every response.
</FI>	
<SESSCOOKIE>	Session cookie that the client should return on the next <SONRQ>,*A-1000*
<ACCESSKEY>	Access key that the client should send in the next <SONRQ>, *A-1000*
</SONRS>	

2.5.1.4 Status Codes

List of status code values for the <CODE> element of <STATUS>:

	Meaning
0	Success (INFO)
2000	General error (ERROR)
3000	User credentials are correct, but further authentication required (ERROR) *This notifies client to send <MFACHALLENGERQ>.*
3001	MFACHALLENGEA contains invalid information (ERROR)
13504	<FI> Missing or Invalid in <SONRQ> (ERROR)
15000	Must change USERPASS (INFO)
15500	Signon invalid (see section 2.5.1) (ERROR)
15501	Customer account already in use (ERROR)
15502	USERPASS Lockout (ERROR)
15506	Empty signon transaction not supported (ERROR)
15507	Signon invalid without supporting pin change request (ERROR)
15510	CLIENTUID error (ERROR)
15511	User should contact financial institution (ERROR)
15512	OFX server requires AUTHTOKEN in signon during the next session (ERROR)
15513	AUTHTOKEN invalid (ERROR)

2.5.1.5 Financial Institution ID <FI>

Some service providers support multiple FIs, and assign each FI an ID. The signon allows clients to pass this information along, so that providers know to which FI the user is signing on.

If a server does not require an FI aggregate in a request but receives one anyway, it should echo the FI aggregate back. This is compliant with the general rule that the server should echo elements and aggregates in the response if they are received and understood in the request.

If a server requires the <FI> aggregate in <SONRQ> requests and it contains incorrect information there are several different specification compliant ways to respond. These are given in the order of preference:

- Return a 2000 error with appropriate text message – since the FI aggregate information is incorrect the user's information (<USERID> and <USERPASS>) cannot be verified. Returning a 15500 might cause clients to display messages to the user that the attempt to communicate with the server failed. A client would probably suggest that the user verify their <USERID> and <USERPASS> values.
- Return a 15500 error – since the FI aggregate information is incorrect or unknown the server cannot verify the <USERID>, <USERPASS>, etc.
- Return an http 400 error – this is the least desirable option since it will provide no useful feedback to the client communicating with the server, however it is legal.

	Description
<FI>	FI-record aggregate
<ORG>	Organization defining this FI name space, *A-32*
<FID>	Financial Institution ID (unique within <ORG>), *A-32*
</FI>	

2.5.2 USERPASS Change <PINCHRQ> <PINCHRS>

The client sends a request to change the customer password as a separate request from the signon.

Password changes pose a special problem for error recovery. If the client does not receive a response, it cannot know whether or not the password change was successful. OFX recommends that servers accept either the old password or the new password on the connection following the one containing a password change. When file-based error recovery is in use, the server must reject the old password except when received with NEWFILEUID/OLDFILEUID headers indicating an error recovery attempt.

Also, if the client does not receive a response that has a status code of 15000 from a server, it cannot know that a password change is required. In this case, the server should not expect a pin change request in the signon when the NEWFILEUID/OLDFILEUID headers indicate an error recovery attempt.

Servers that do not support file-based error recovery (or, when interacting with a client that does not utilize file-based error recovery) must not complete a <PINCHRQ> until after the next request file arrives. If that request file uses the new password, the new password must be permanently associated with the <USERID>. Otherwise, the old password may authenticate the user. (For security, servers may return a signon error if the next request file uses the old password but does not include a <PINCHRQ>.) Conforming clients should re-send request files (unchanged beyond the <SONRQ>) after a failure whether or not file-based error recovery is in use.

2.5.2.1 <PINCHRQ>

A <PINCHRQ> request must appear within a <PINCHTRNRQ> transaction wrapper.

A USERPASS change request changes the customer's password for the specific realm associated with the messages contained in the OFX block. Based on the properties of an OFX profile, defined in Chapter 7, "FI Profile," a single OFX block contains instructions related to a single realm. The USERPASS change request thus changes the USERPASS for all message sets associated with one realm. For more information about signon realms, see section 7.2.2.

	Description
<PINCHRQ>	USERPASS-change-request aggregate
<USERID>	User identification string, *A-32* **Note:** The maximum clear text length of USERPASS is 32 characters: a client must not send a longer password. However, when using Type 1 security, the encrypted value may extend to 171 characters.
<NEWUSERPASS>	New user password, *A-171* **Note:** The effective size of NEWUSERPASS is A-32. However, if Type 1 security is used, then the actual field length is A-171.
</PINCHRQ>	

2.5.2.2 <PINCHRS>

A <PINCHRS> response must appear within a <PINCHTRNRS> transaction wrapper.

	Description
<PINCHRS>	USERPASS-change-response aggregate
<USERID>	User identification string, *A-32*
<DTCHANGED>	Date and time the password was changed, *datetime*
</PINCHRS>	

2.5.2.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
15503	Could not change USERPASS (ERROR)
15508	T ctiod n authorize( )dot ERROR8

2.5.3 <CHALLENGERQ> <CHALLENGERS>

A challenge request is the first step in Type 1 application-level security. Essentially, it asks for some random data from the server. The challenge response provides that server-generated random data and is the second step in Type 1 security.

The challenge message is part of the signon message set and is not subject to data synchronization.

2.5.3.1 <CHALLENGERQ>

A <CHALLENGERQ> is part of a <CHALLENGETRNRQ> transaction, a <CHALLENGERS> part of a <CHALLENGETRNRS>.

The client includes <FICERTID> in the request if it already has the server's certificate. If that is included and matches the server's current certificate, the server may omit the actual certificate from the response.

	Description
<**CHALLENGERQ**>	Opening tag for the challenge request.
<USERID>	User identification string, *A-32*
<FICERTID>	Optional server certificate ID. *A-64*
</CHALLENGERQ>	Closing tag for challenge request.

2.5.3.2 <CHALLENGERS>

Tag	*Description*
<**CHALLENGERS**>	Opening tag for the challenge response.
<USERID>	User identification string, *A-32*
<NONCE>	Server-generated random data. *A-16*
<FICERTID>	ID of server certificate used to encrypt. *A-64*
</**CHALLENGERS**>	Closing tag for challenge response.

When generating the <NONCE>, make sure the data is as unpredictable as possible. See RFC 1750 for recommendations.

The server includes <FICERTID> in the response to identify the certificate in a separate MIME part. Even if the certificate itself is not attached, <FICERTID> is still included in the response.

2.5.3.3 Status Codes

Status code values for the <CODE> element (contained within the <STATUS> aggregate):

	Meaning
0	Success (INFO)
2000	General error (ERROR)
15504	Could not provide random data (ERROR)
15508	Transaction not authorized (ERROR)

2.5.4 <MFACHALLENGERQ> <MFACHALLENGERS>

To support authentication mechanisms for which the additional signon information described above is not sufficient, OFX 2.1.1 introduces a new message, the MFACHALLENGE message. If the information in the signon request is correct, but it is not sufficient to authenticate the user, the server can reply with an error code of 3000, which indicates that the client must perform MFACHALLENGE authentication before proceeding with OFX requests. The server must not process any requests included as part of the message that resulted in the 3000 error code.

Following receipt of the 3000 error code, the client should request a list of challenge questions with an <MFACHALLEGERQ>. The server will then respond to this request with a <MFACHALLENGERS>, which includes the list of authentication questions, specified by ID and label. If for some reason the server cannot respond with a <MFACHALLENGERS> response, it should respond with an HTTP 400 error.

Note that some of these challenge questions may require user interaction and some may not (if the client already has access to the necessary information). It is up to the client to determine which questions require user interaction.

> **Note:** If the profile response contains <MFACHALLENGEFIRST>Y, the client must send an <MFACHALLENGERQ> request in the first connection with the server, before sending any other requests.

Once the client has retrieved the answers to the challenge questions (either from the user or another location), it will then include them within the signon request included as part of the next request message. If these answers are correct, the server will process the request file. If they are incorrect, the server will return an error code of 3001.

The client should not need to store the answers to the challenge questions. To prevent servers from needing

subsequent request with a 3000 error code, requiring the client to send a <MFACHALLENGERQ>. This allows the server to determine the lifetime of the <ACCESSKEY>.

The challenge message is part of the signon message set and is not subject to data synchronization.

2.5.4.1 <MFACHALLENGERQ>

<MFACHALLENGERQ> is a request for the server to send a list of challenge questions that must be correctly answered before the OFX client may proceed with further OFX requests. The <MFACHALLENGERQ> request should be sent in response to an error code of 3000 or on a first request to the server if the profile contains <MFACHALLENGEFIRST>Y.

The <MFACHALLENGERS> must appear within a <MFACHALLENGETRNRS> transaction wrapper. The <MFACHALLENGERQ> request must appear within a <MFACHALLENGETRNRQ> transaction wrapper.

	Description
<MFACHALLENGERQ>	MFA challenge request aggregate
<DTCLIENT>	Date and time of the request from the client computer, *datetime*
</MFACHALLENGERS>	

2.5.4.2 <MFACHALLENGERS>

The <MFACHALLENGERS> response contains the list of questions that must be correctly answered in the next OFX request. These questions may or may not require user interaction. See the table in Section 2.5.4.4 for more details.

While this specification imposes no upper limit on the number of challenge questions a server sends, financial institutions and servers should be aware that there may be a limit to the number of questions a client is able to collect.

The <MFACHALLENGERS> must appear within a <MFACHALLENGETRNRS> transaction wrapper following the <SONRS> aggregate within the Signon Message set.

Tag	*Description*
<MFACHALLENGERS>	MFA challenge response aggregate
<MFACHALLENGE>	Challenge question aggregate (1 or more)
<MFAPHRASEID>	Identifier for the challenge question. It should be unique for this challenge question but not unique for the user, session, etc. *A-32*. See section 2.5.4.4
<MFAPHRASELABEL>	The textual challenge question. This should be as appropriate as possible for display to the user. *A-64*

Tag	Description
</MFACHALLENGE> </MFACHALLENGERS>	

2.5.4.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)

2.5.4.4 <MFAPHRASEID>

The <MFAPHRASEID> tag uniquely identifies a challenge question. In addition to providing a way to correlate the answers to challenge questions with the questions themselves, it also provides the OFX client additional options for collecting the answer from the user.

2.5.4.4.1 Enumerated <MFAPHRASEID> Meanings

The following table details the list of reserved values for <MFAPHRASEID>s. Servers should use these values only when the question they are asking matches the associated question in the table below. Note that servers are never required to use the reserved values for the phrase ID, even when the challenge question the servers require does match the question in the table below, but they should be aware that using the reserved IDs when appropriate may result in a better customer experience.

MFAPHRASEID values above MFA100 are reserved for questions that the server expects the client to answer. These do not require customer responses. All other enumerated IDs as well as server specific IDs expect customer responses.

Clients may need to identify out of band which of the IDs above MFA100 they support.

Value	Meaning
MFA1	City of birth
MFA2	Date of birth, formatted MM/DD/YYYY
MFA3	Debit card number
MFA4	Father's middle name
MFA5	Favorite color
MFA6	First pet's name
MFA7	Five digit ZIP code

Value	*Meaning*
MFA8	Grandmother’s maiden name on your father’s side
MFA9	Grandmother’s maiden name on your mother’s side
MFA10	Last four digits of your cell phone number
MFA11	Last four digits of your daytime phone number
MFA12	Last four digits of your home phone number
MFA13	Last four digits of your social sescurity number
MFA14	Last four digits of your tax ID
MFA15	Month of birth of youngest sibling, *do not abbreviate*
MFA16	Mother’s maiden name
MFA17	Mother’s middle name
MFA18	Name of the company where you had your first job
MFA19	Name of the manufacturer of your first car
MFA20	Name of the street you grew up on
MFA21	Name of your high school football team, do not include high school name, e.g. "*Beavers*" rather than "*Central High Beavers*"
MFA22	Recent deposit or recent withdrawal amount
MFA23	Year of birth, formatted *YYYY*
MFA24	
MFA25	
MFA26	
MFA27	
MFA28	
MFA29	
MFA30	
MFA101	Datetime, formatted *YYYYMMDDHHMMSS*
MFA102	Host name
MFA103	IP Address
MFA104	MAC Address
MFA105	Operating System version
MFA106	Processor architecture, e.g. *I386*

```
          <CODE>15512</CODE>
          <SEVERITY>ERROR</SEVERITY>
          <MESSAGE>Please provide Authentication Token</MESSAGE>
        </STATUS>
        <DTSERVER>20060321083015</DTSERVER>
        <LANGUAGE>ENG</LANGUAGE>
        <FI>
          <ORG>ABC</ORG>
          <FID>000111222</FID>
        </FI>
      </SONRS>
  </SIGNONMSGSRSV1>
---<!--All other transaction responses return <CODE>15500</CODE>-->
</OFX>
```

Client collects the answers and returns them to server along with the original request.

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>
      <DTCLIENT>20060321083415</DTCLIENT>
      <USERID>12345</USERID>
      <USERPASS>MyPassword</USERPASS>
      <LANGUAGE>ENG</LANGUAGE>
      <FI>
        <ORG>ABC</ORG>
        <FID>000111222</FID>
      </FI>
      <APPID>MyApp</APPID>
      <APPVER>1600</APPVER>
      <CLIENTUID>22576921-8E39-4A82-9E3E-EDDB121ADDEE</CLIENTUID>
      <AUTHTOKEN>1234567890</AUTHTOKEN><!—Authentication token
  provided to user out of band-->
    </SONRQ>
  </SIGNONMSGSRQV1>
….                              <!--Other message sets-->
</OFX>
```

Server accepts requests and returns an ACCESSKEY.

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>
```

```
        <STATUS>
          <CODE>0</CODE>
          <SEVERITY>INFO</SEVERITY>
          <MESSAGE>Success</MESSAGE>
        </STATUS>
        <DTSERVER>20060321083445</DTSERVER>
        <LANGUAGE>ENG</LANGUAGE>
        <FI>
          <ORG>ABC</ORG>
          <FID>000111222</FID>
        </FI>
        <ACCESSKEY>EE225228-38E6-4E35-8266-CD69B5370675</ACCESSKEY>
    </SONRS>
    </SIGNONMSGSRSV1>
---                              <!--All other transaction responses-->
</OFX>
```

On subsequent calls, client will return ACCESSKEY in SONRQ.

The following series shows the OFX 2.1.1 exchanges that occur when a server requires the client to collect answers to MFA Challenge questions.

Client sends OFX request to server.

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>
      <DTCLIENT>20060321083010</DTCLIENT>
      <USERID>12345</USERID>
      <USERPASS>MyPassword</USERPASS>
      <LANGUAGE>ENG</LANGUAGE>
      <FI>
        <ORG>ABC</ORG>
        <FID>000111222</FID>
      </FI>
      <APPID>MyApp</APPID>
      <APPVER>1600</APPVER>
      <CLIENTUID>22576921-8E39-4A82-9E3E-EDDB121ADDEE</CLIENTUID>
    </SONRQ>
  </SIGNONMSGSRQV1>
```

```
    <MFACHALLENGETRNRQ><!--MFA Challenge Transaction aggregate-->
      <TRNUID>66D3749F-5B3B-4DC3-87A3-8F795EA59EDB</TRNUID>
      <MFACHALLENGERQ><!--MFA Challenge aggregate-->
        <DTCLIENT>20060321083020</DTCLIENT
      </MFACHALLENGERQ>
    </MFACHALLENGETRNRQ>
  </SIGNONMSGSRQV1>
</OFX>
```

Server returns challenge answers which include:

• Enumerated phrase ID requiring a user response

• Enumerated phrase ID requiring only client response

• Custom question.

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <DTSERVER>20060321083025</DTSERVER>
      <LANGUAGE>ENG</LANGUAGE>
      <FI>
        <ORG>ABC</ORG>
        <FID>000111222</FID>
      </FI>
    </SONRS>
    <MFACHALLENGETRNRS><!--MFA Challenge Transaction aggregate-->
      <TRNUID>66D3749F-5B3B-4DC3-87A3-8F795EA59EDB</TRNUID>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
        <MESSAGE>SUCCESS</MESSAGE>
      </STATUS>
      <MFACHALLENGERS><!--MFA Challenge aggregate-->
        <MFACHALLENGE>
          <MFAPRHASEID>MFA13</MFAPRHASEID>
```

```
          <MFAPHRASELABEL>Please enter the last four digits of your
                  social security number.</MFAPHRASELABEL>
        </MFACHALLENGE>
        <MFACHALLENGE><!--MFA Challenge aggregate w/o a label-->
          <MFAPRHASEID>MFA107</MFAPRHASEID>
        </MFACHALLENGE>
        <MFACHALLENGE><!--MFA Challenge aggregate-->
          <MFAPRHASEID>123</MFAPRHASEID>
          <MFAPHRASELABEL>With which branch is your account
                  associated?</MFAPHRASELABEL>
        </MFACHALLENGE>
      </MFACHALLENGERS>
    <MFACHALLENGETRNRS>
  </SIGNONMSGSRSV1>
</OFX>
```

```
          <MFAPRHASEID>123</MFAPRHASEID>
          <MFAPHRASEA>Anytown</MFAPHRASEA>
        </MFACHALLENGEA>
      </SONRQ>
    </SIGNONMSGSRQV1>
….                                <!--Other message sets-->
</OFX>
```

Server accepts requests and returns an ACCESSKEY.

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
        <MESSAGE>Success</MESSAGE>
      </STATUS>
      <DTSERVER>20060321083445</DTSERVER>
      <LANGUAGE>ENG</LANGUAGE>
      <FI>
        <ORG>ABC</ORG>
        <FID>000111222</FID>
      </FI>
      <ACCESSKEY>EE225228-38E6-4E35-8266-CD69B5370675</ACCESSKEY>
    </SONRS>
  </SIGNONMSGSRSV1>
---                               <!--All other transaction responses-->
</OFX>
```

On subsequent calls, client will return ACCESSKEY in SONRQ.

2.6 External Data Support

Some data, such as binary data, cannot easily be sent within XML. For these situations, the specification defines an element that references some external data. The way that clients pick up the external data depends on the transport used. For the HTTP-based transport described in this document, servers can send the data in one of two ways:

- Send the same response, using multipart MIME types to separate the response into the Open Financial Exchange file and one or more external data files

- Client can make a separate HTTP get against the supplied URL, if it really needs the data

For example, to retrieve a logo, a <GETMIMERS> might answer a <GETMIMERQ> as follows:

```
<GETMIMERS>
   <URL>https://www.fi.com/xxx/yyy/zzz.jpg</URL>
</GETMIMERS>
```

If the file includes the same response using multipart MIME, clients must have the local file, zzz.jpg.

2.8.2 XML Compliant Header

Any client or server using OFX 2.1.1 will have to use the XML compliant header. Mapping between the old and new style of OFX headers is straightforward.

The old OFX header looks like:

```
OFXHEADER:100
DATA:OFXSGML
VERSION:102
SECURITY:NONE
ENCODING:USASCII
CHARSET:NONE
COMPRESSION:NONE
OLDFILEUID:NONE
NEWFILEUID:NONE
```

The new XML compliant OFX header looks like:

```
<?OFX OFXHEADER="200" VERSION="211" SECURITY="NONE" OLDFILEUID="NONE"
NEWFILEUID="NONE"?>
```

The old OFX header maps to the new header as follows:

- OFXHEADER has the same meaning in both versions.
- DATA is not necessary because XML is assumed.
- VERSION has the same meaning in both versions.
- SECURITY has the same meaning in both versions.
- ENCODING is not necessary because it is specified in the standard XML declaration.
- CHARSET is not necessary because it is handled by the XML declaration.
- COMPRESSION is not necessary because it will not be handled at this data level.
- OLDFILEUID has the same meaning in both versions.
- NEWFILEUID has the same meaning in both versions.

2.8.3 International Support

XML supports many different types of character encoding. An OFX 2.1.1 server would have to support the full range of encoding specified in the XML 1.0 recommendation to be fully XML compliant. However, OFX 1.x only required support for USASCII and UTF-8. Therefore, to guarantee compatibility with older servers, it will be necessary to limit the encoding of characters to USASCII and UTF-8.

2.8.4 Message Set Versioning

OFX 2.1.1 supports all V1 message sets found in the OFX 1.6 specification plus the Loan and Image message sets.

CHAPTER 3 COMMON AGGREGATES, ELEMENTS, AND DATA TYPES

3.1 Common Aggregates

This section describes aggregates used in more than one service of Open Financial Exchange (for example, investments and payments).

3.1.1 Identification of Financial Institutions and Accounts

Open Financial Exchange does not provide a universal space for identifying financial institutions, accounts, or types of accounts. The way to identify an FI and an account at that FI depends on the service. For information about service-specific ID aggregates, see Chapter 11, "Banking," Chapter 12, "Payments," and Chapter 13, "Investments."

3.1.2 Punctuation in Certain User-Supplied Values

This section discusses the addition or removal of punctuation in certain user-supplied values by a client or server. The term punctuation is loosely used to pertain to the manipulation of these values in such a way as to make them more readable to either a user or processor, or make them more precise or correct. Making user-supplied values more readable to the user or processor involves the utilization of punctuation characters, for example, the stripping out of dashes in a user-supplied account number. Making the values more precise or correct might involve an actual syntactic change to data, for example, the extension of a zip code to use the full zip+4 value.

3.1.2.1 Manipulation of User-Supplied Values by a Client

The user-supplied values under consideration here fall into three broad groups:

- Values provided for security reasons
- Values of critical ID fields
- Values of non-critical fields

3.1.2.1.1 Values provided for security reasons

This group pertains to values provided for security reasons such as <USERID> and <USERPASS> elements. These values must never be manipulated by a client; they are sent without change to the server.

3.1.2.1.2 Values of critical ID fields

This group pertains to critical ID fields, generally account numbers, routing numbers and the like. These values also should never be manipulated by a client unless the server has supplied the client with a

3.1.3 Echoing in Responses

A server should echo back unedited element values in the immediate response, but may store values in edited form. In the cases where the stored value is changed, it is recommended that the server respond with an out-of-band modification synchronization response whenever possible. For example, if a client sends a payee name of “Sears” but the server stores it as “SEARS”, the server should send a <PAYEEMODRS> in the next sync response. (See Chapter 12, "Payments" for clarification of payee issues.) However, if the server simply edits punctuation in or out of client-supplied numbers such as account numbers and will match both forms in future requests, it is not required to notify the client.

Any intermediate software should avoid any modifications to these values, thus avoiding the need to resolve this issue out-of-band.

3.1.4 Balance Records <BAL>

Several responses allow FIs to send an arbitrary set of balance information as part of a response, for example a bank statement download. FIs might want to send information on outstanding balances, payment dates, interest rates, and so forth. Balances can report the date the given balance reflects in <DTASOF>.

	Description
<BAL>	Balance-response aggregate
<NAME>	Balance name, *A-32*
<DESC>	Balance description, *A-80*
<BALTYPE>	Balance type. DOLLAR = dollar (value formatted DDDD.cc) PERCENT = percentage (value formatted XXXX.YYYY) NUMBER = number (value formatted as is)
<VALUE>	Balance value. Interpretation depends on <BALTYPE> field, *amount*
<DTASOF>	Effective date of the given balance, *datetime*
<CURRENCY>	If dollar formatting, can optionally include currency, see section 5.2
</CURRENCY>	
</BAL>	

3.1.5 Error Reporting <STATUS>

To provide as much feedback as possible to clients and their users, Open Financial Exchange defines a <STATUS> aggregate. The most important element is the code that identifies the error. Each response defines the codes it uses. Codes 0 through 2999 have common meanings in all Open Financial Exchange transactions. Codes from 3000 and up have meanings specific to each transaction.

Clients should assume the burden of checking the profile and not sending a transaction which the server does not support. If the client goes ahead and sends such a transaction, the server may either return an HTTP 400 syntax error, or ignore unsupported elements and aggregates. In the latter case, assuming no other problems occur in processing that request, servers may return warning code 2028 (Request element unknown). The response file should not contain the unsupported elements or aggregates.

The last 200 error codes in each assigned range of 1000 are reserved for server-specific status codes. For example, of the general status codes, 2800-2999 are reserved for status codes defined by the server. Of the banking status codes, codes 10800-10999 are reserved for the server. If a client receives a server-specific status code of <SEVERITY> ERROR that it does not know, it must handle it as a general error 2000.

	Description
<STATUS>	Error-reporting aggregate.
<CODE>	Error code, *N-6*
<SEVERITY>	Severity of the error: INFO = Informational only WARN = Some problem with the request occurred but a valid response still present ERROR = A problem severe enough that response could not be made
<MESSAGE>	A textual explanation from the FI. Note that clients will generally have messages of their own for each error ID. Use this element only to provide more details or for the general errors. *A-255*
</STATUS>	

For general errors, the server can respond with one of the following <CODE> values. However, not all codes are possible in a specific context. See section 2.5.1.4 for a complete list.

	Meaning
0	Success (INFO)
2000	General error (ERROR) **Note:** Servers should provide a more specific error whenever possible. Error 2000 should be reserved for cases in which a more specific code is not available.
2021	Unsupported version (ERROR)
2028	Requested element unknown (WARNING)
3000	MFA challenge authentication is required (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
15500	Signon invalid (see section 2.5.1) (ERROR)
15512	OFX server requires AUTHTOKEN in signon during the next session (ERROR)

Note: Clients will generally have error messages that are based on <CODE>. Therefore, do not use <MESSAGE> to replace that text. Use <MESSAGE> only to explain an error not well described by one of the defined codes, or to provide some additional information. <MESSAGE> should be returned whenever the <CODE> can be refined. For example, <CODE>2000 should always be accompanied with a <MESSAGE> explaining the problem.

3.1.6 Common Aggregates related to Images

3.1.6.1 Image Data Aggregate <IMAGEDATA>

Several OFX request and response pairs allow clients to request, and FIs to send, instructions for accessing images. These instructions contain image retrieval information but not the actual images themselves. This information is returned in the <IMAGEDATA> aggregate, and is used by the client to retrieve the actual images, as described in Chapter 15, “Image Download”.

Tag	*Description*
<IMAGEDATA>	Image information
<IMAGETYPE>	Type of image. Use one of the following: STATEMENT: the image is a statement image TRANSACTION: the image is a transaction image (e.g. a check image) TAX: the image is a tax image

3.1.6.1.1 Values for <IMAGEREFTYPE> Element

	Description
OPAQUE	The image is accessed via an explicit OFX <IMAGERQ> request, which will be followed by the image data. See section 15.2.1 for more information
URL	The image is accessed directly via the URL provided. The image cannot be retrieved via an OFX image request. The expectation is that the client will not provide authentication and will simply follow the URL provided. See section 15.2.2 for more information.
FORMURL	The image is accessed directly via an encoded URL. The image cannot be retrieved via an OFX image request. The expectation is that the client will send authentication to the server. See section 15.2.3 for more information.

3.1.6.2 Image Profile Aggregate <IMAGEPROF>

The following aggregate may appear in various profile responses and will be referred to in those profile sections where appropriate.

Tag	*Description*
<IMAGEPROF>	Image Profile (if supported)
<CLOSINGIMGAVAIL>	Whether the server supports closing statement images, *Boolean*
<TRANIMGAVAIL>	Whether the server supports transaction images, *Boolean*
</IMAGEPROF>	

3.2 Common Elements

This section defines elements used in several services of Open Financial Exchange. The format of the value is either character (A-*n*) or numeric (N-*n*) with a maximum length *n*; or as a named type. Section 3.2.8 describes the named types.

3.2.1 Client-Assigned Transaction UID <TRNUID>

Format: *A-36*

Open Financial Exchange uses <TRNUID>s to identify transactions within transaction wrappers (<*xxx*TRNRQ>, </*xxx*TRNRQ>).

In most cases, clients originate <TRNUID>s. When a client originates a <TRNUID>, the value of the <TRNUID> is always set to a unique identifier. The server must return the same <TRNUID> in the corresponding response and any later synchronization responses that include this response. Clients may use this <TRNUID> to match up requests and responses or to recognize synchronized responses for transactions they did not initiate. Servers can use <TRNUID>s to reject duplicate requests. Because multiple clients might be generating requests to the same server, transaction IDs must be unique across

A <SRVRTID> is a server-assigned ID for an object that is stored on the server. It should remain constant throughout the lifetime of the object on the server. The client will consider the SRVRTID as its "receipt" or confirmation and will use this ID in any subsequent requests to change, delete, or inquire about this object.

A <SRVRTID> is not unique across FI's or Service Providers, and clients might need to use FI + <SPNAME> + <SRVRTID> when a unique key is necessary.

A <SRVRTID> must be unique across users for joint accounts. Therefore, it is not sufficient to just keep the <SRVRTID> unique to an account if it is shared by more than one user.

Where the context allows, a server may use the same *value* for a given server object for both <SRVRTID> and <FITID>, but the client will not know this. In this case, the server must assign <SRVRTID> and <FITID> values that are more unique than otherwise required. Because of the differing uniqueness constraints on the individual elements, such a reused value must be unique throughout the FI.

For example: The server creates the new recurring model from the example in section 3.2.1 with <RECSRVRTID>1234:5687. The server uses this identifier in the initial <RECPMTRS> and any synchronization responses that reference this model. The client references the same <RECSRVRTID> in the later <RECPMTCANCRQ>.

If any payments are spawned from this model before it is cancelled, they would each have their own <SRVRTID> value (for example, <SRVRTID>8765:4321 and <SRVRTID>8765:4322). The <SRVRTID> value for one of the spawned payments may match the <RECSRVRTID> of the model. Such a match is not required for any spawned payment. To guarantee uniqueness of the payment identifiers, no more than one spawned payment may use the <RECSRVRTID> value of its model.

Usage: Banking, Payments, Investments, Bill Presentment, Tax

Elements of this type: RECSRVRTID and SRVRTID

3.2.3 Financial Institution Transaction ID <FITID>

Format: *A-255*

An FI (or its Service Provider) assigns an <FITID> to uniquely identify a financial transaction that can appear in an account statement. Its primary purpose is to allow a client to detect duplicate responses. Open Financial Exchange intends <FITID> for use in statement download applications, where every transaction (not just those that are client-originated or server-originated) requires a unique ID.

An <FITID> also uniquely identifies the closing statement in <CLOSINGRS> and <CCCLOSINGRS>. Again, the OFX client should detect repeated closing statements (duplicate downloads) using these identifiers.

FITIDs must be unique within the scope of an account but need not be sequential or even increasing. Clients should be aware that FITIDs are not unique across FIs. If a client performs the same type of request within the same scope at two different FIs, clients will need to use FI + <ACCTID> + <FITID> as a

globally unique key in a client database. That is, the <FITID> value must be unique within the account and Financial Institution (independent of the service provider).

> **Note:** Although the specification allows FITIDs of up to 255 characters, client performance may significantly improve if servers use fewer characters. It is recommended that servers use 32 characters or fewer.

For example: The two spawned payments mentioned in section 3.2.1 are processed and later downloaded in a <STMTRS>. The first payment's <STMTTRN> would list <SRVRTID>8765:4321, <RECSRVRTID>1234:5678, and <FITID>9999:8888:7777. The second payment would be described in a <STMTTRN> containing <SRVRTID>8765:4322, <RECSRVRTID>1234:5678, and <FITID>6666:5555:4444.

Usage: Bank statement download, investment statement download

Elements of this type: <CORRECTFITID>, <FITID>, <RELFITID>, and <REVERSALFITID>

3.2.4 Token <TOKEN>

Format: *A-10 for <TOKEN>, used in V1 message sets*

Open Financial Exchange uses <TOKEN> as part of data synchronization requests to identify the point in history that the client has already received data, and in responses to identify the server's current end of history. See Chapter 6, "Data Synchronization," for more information.

<TOKEN> is unique within an FI and the scope of the synchronization request. For example, if the synchronization request includes an account ID, the <TOKEN> needs to be unique only within an account. Servers are free to use a <TOKEN> that is unique across the entire FI. Clients must save separate <TOKEN>s for each account, FI, and type of synchronization request.

Usage: All synchronization requests and responses

3.2.5 Transaction Amount <TRNAMT>

Format: *Amount*

Open Financial Exchange uses <TRNAMT> in any request or response that reports the total amount of an individual transaction.

Usage: Bank statement download, investment statement download, payments

3.2.6 Memo <MEMO>

Format: *A-255 for <MEMO>, used in V1 message sets*

A <MEMO> provides additional information about a transaction.

Usage: Bank statement download, transfers, payments, investment statement download, bill presentment tables

3.2.7 Date Start and Date End <DTSTART> <DTEND>

Format: *Datetime*

Clients use these elements in requests to indicate the range of response that is desired. Servers use these elements in responses to let clients know what the FI was able to produce.

In requests, the following rules apply:

- If <DTSTART> is absent, the client is requesting all available history (up to the <DTEND>, if specified). Otherwise, it indicates the *inclusive* date and time in history where the client expects servers to start sending information.
- If <DTSTART> is absent, the client is requesting all available history (up to the <DTEND>, if specified). Otherwise, it indicates the *inclusive* date and time in history where the client expects servers to start sending information. This inclusiveness also applies to the time element. And, because servers and clients may store time elements to differing degrees of precision, the server should include transactions in the response using the highest degree of precision possible. For example:
  - If the request does not include a time element, then all transactions on or after the date portion should be returned.
  - If the request includes time specified to milliseconds but the bank system only has time specified to seconds on its transactions, then all transactions on or after the date/time (specified to seconds) should be returned.
- If <DTEND> is absent, the client is requesting all available history (starting from <DTSTART>, if specified). Otherwise, it indicates the *exclusive* date and time in history where the client expects servers to stop sending information. This exclusiveness also applies to the time element. And, because servers and clients may store time elements to differing degrees of precision, the server should include transactions in the response using the highest degree of precision possible. For example:
  - If the request does not include a time element, then all transactions before the date portion should be returned.
  - If the request includes time specified to milliseconds but the bank system only has time specified to seconds on its transactions, then all transactions before the date/time (specified to seconds) should be returned.

In responses, the following rules apply:

- <DTSTART> is the date and time where the server began *looking* for information, not necessarily the date of the earliest returned information. If the response <DTSTART> is later than the requested <DTSTART>, clients can infer that the user has not signed on frequently enough to ensure that the client has retrieved all information. If the user has been calling frequently enough, <DTSTART> in the response will match <DTSTART> in the request.
- <DTEND> is the date and time that, if used by the client as the next requested <DTSTART>, it would pick up exactly where the current response left off. It is the *exclusive* date and time in history where the server stopped *looking* for information, based on the request <DTEND> rules.

Because the system add date for a transaction is not necessarily the post date for the transaction (the latter occurring when the account is actually debited or credited), a server should consider the <DTSTART> and <DTEND> dates in a <(CC)STMTRQ> as a request for any transactions that were posted or added to the FI system at that time. In addition, the transactions returned should probably span a greater window of time than that included in the <DTSTART>/<DTEND> dates since a transaction might be added to the system after a statement download request was made for that time period. (Clients should be able to filter out the unnecessary transactions.) If a client is always requesting a download sequentially, through time, is never requesting an end date using <DTEND> and is always substituting <DTEND> in the response for <DTSTART> in the next request, it is safe for a server to return only those transactions that had a system add date on or after <DTSTART> in the request. In all cases, servers are minimally **required** to use a "system add datetime" as the basis for deciding which details match the requested date range. For example, if an FI posts a transaction dated Jan 3 to a user's account on Jan 5, and a client connects on Jan 4 and again on Jan 6, the server is **required** to return that Jan 3-dated transaction when the client calls on Jan 6.

Usage: Bank statement download, investment statement download, 401(k) summary, bill presentment list request

3.2.8 Common Data Types

3.2.8.1 Dates, Times, and Time Zones

There is one format for representing dates, times, and time zones. The complete form is:

YYYYMMDDHHMMSS.XXX [*gmt offset*[*:tz name*]]

3.2.8.1.1 Ranges for Years, Months, Days, Hours, Seconds

	Range
YYYY	0000 - 9999
MM	1 - 12
DD	1 - 31
HH	0 - 23
MM	0 - 59
SS	0 - 60 60 is only used in the case of the leap second

3.2.8.2 Date and Datetime

Elements specified as type *date or datetime* and generally starting with the letters “DT” accept a fully formatted date-time-timezone string. For example, “19961005132200.124[-5:EST]” represents October 5, 1996, at 1:22 and 124 milliseconds p.m., in Eastern Standard Time. This is the same as 6:22 p.m. Greenwich Mean Time (GMT).

Date and *datetime* also accept values with fields omitted from the right. They assume the following defaults if a field is missing:

Specified date or datetime	*Assumed defaults*
YYYYMMDD	12:00 AM (the start of the day), GMT
YYYYMMDDHHMMSS	GMT
YYYYMMDDHHMMSS.XXX	GMT

Note that times zones are specified by an offset and optionally, a time zone name. The offset defines the time zone. Valid offset values are in the range from –12 to +12 for whole number offsets. Formatting is +12.00 to -12.00 for fractional offsets, plus sign may be omitted.

Take care when specifying an ending date without a time. For example, if the last transaction returned for a bank statement download was Jan 5 1996 10:46 am and if the <DTEND> was given as just Jan 6, the next statement download request would have a <DTSTART> of just Jan 6, causing any transactions posted

on Jan 5 after 10:46 am to be missed. If results are available only daily, then just using dates and not times will work correctly.

> **Note:** Open Financial Exchange does not require servers or clients to use the full precision specified. However, they are REQUIRED to accept any of these forms without complaint.

Some services extend the general notion of a *date* by adding special values, such as “Tc[(8rD)-(A)935.4Y special value s-1.6(arecJa)-4.3llue“sste.” Sspecifcf r(qu)]TJ214.1694 0 TD-0.0031 Tc0.0007 Tw[sess indicatse hnD to(u

3.2.8.3 Time

Elements specified as type *time* and generally ending with the letters “TM” accept times in the following format:

HHMMSS.XXX[*gmt offset*[*:tz name*]]

The milliseconds and time zone are still optional, and default to GMT.

3.2.8.4 Time Zone Issues

Several issues arise when a customer and FI are not in the same time zone, or when a customer moves a computer into new time zones. In addition, it is generally unsafe to assume that computer users have correctly set their time or time zone.

Although most transactions are not sensitive to the exact time, they often are sensitive to the date. In some cases, time zone errors lead to actions occurring on a different date than intended by the customer. For this reason, servers should always use a complete local time plus GMT offset in any datetime values in a response. If a customer’s request is for 5 p.m. EST, and a server in Europe responds with 1 a.m. MET the next day, a smart client can choose to warn the customer about the date shift.

Clients that maintain local state, especially of long-lived server objects, should be careful how they store datetime values. If a customer initiates a repeating transaction for 5 p.m. EST, then moves to a new time zone, the customer might have intended that the transaction remain 5 p.m. in the new local time, requiring a change request to be sent to the server. If, however, the customer intended it to remain fixed in server time, this would require a change in the local time stored in the client.

Client software that doesn’t know the current local time zone for the user, or client proxies that don’t know the current local time zone of their end users, should maintain and display the datetime value in the time zone indicated by the originator of the value and explicitly marked with that time zone. As an example, consider <DTPMTDUE> in section 11.5.4.2. If the biller gave a due date of 23:59pm EST on Dec. 29, 1997, this is best displayed as 23:59pm EST rather than rendered in local time if there is any doubt at all as to the current local time zone of the end user looking at the due date.

When considering timezone conversions, remember the following differences between the *date* and *datetime* datatypes:

- *Date* = A date without time; this date is explicit. Clients and servers will not convert the value in any way. Examples include birth date and billing date.
- *Datetime* = A date and time format; clients and servers may convert this date to their local timezone. Examples include last account update date and bill summary fetch date.

  Note: Developers should consider the possibility of a date change due to timezone conversion. A *datetime* value in the GMT timezone with a time of 12:00:00 (noon) would be converted to another time on the same date in every timezone. For example, 199812251200 remains Christmas Day in every timezone.

3.2.9 Amounts, Prices, and Quantities

3.2.9.1 Basic Format

Format: *A-32*

This section describes the format of numerical values used for amounts, prices, and quantities. In all cases, a numerical value that does not contain a decimal point has an implied decimal point at the end of the value. For example, a numerical value of "550" is equivalent to "550." Trailing and leading spaces should be stripped. Number format uses a leading sign. Negative number format uses a minus sign (-). Positive number format uses a plus sign (+). The plus sign is implied for all amounts and can be omitted.

The following types are defined to have a maximum of 32 characters, including alphabetic characters, digits and punctuation. However, clients and servers may have specific limits for the maximum number of digits to the left or right of a decimal point. If a server cannot support a client request due to the size or precision of a number, the server should return status code 2012.

Amount: Amounts that do not represent whole numbers (for example, 540.32), must include a decimal point or comma to indicate the start of the fractional amount. Amounts should not include any punctuation separating thousands, millions, and so forth. The maximum value accepted depends on the client.

Quantity: Use decimal notation.

Unitprice: Use decimal notation. Unless specifically noted, prices should always be positive.

Rate: Use decimal notation, with the rate specified out of 100%. For example, 5.2 is 5.2%. Rates can be greater than 100 and can be negative.

Some services define special values, such as INFLATION, which you can use instead of a designated value. Open Financial Exchange refers to these as "smart types," and identifies them in the specification.

3.2.9.2 Positive and Negative Signs

Most OFX transaction aggregates describe the flow of funds. Amounts in transactions which clearly describe the flow of funds should normally be positive. For example, bank transfers (<INTRARQ>), bill payments (<PMTRQ>) and investment buys/sells (<BUYSTOCK>, <SELLSTOCK>) should all have positive amounts. An exception to this is the sign of the amount in statement download transactions, wrapped within <STMTTRN></STMTTRN> tags. The amounts in these transactions should be signed on the basis of how the account is affected, e.g. a <TRNTYPE>DEBIT should have a negative <TRNAMT> value.

Servers should sign amounts from the perspective of the user in cases where the flow of funds cannot be determined from the transaction aggregate alone. For example, interest amounts can be either positive or negative, depending on whether the interest is earned or paid.

3.2.10 Language

Language identifies the human-readable language used for such things as status messages and e-mail. *Language* is specified as a three-letter code based on ISO-639.

3.2.11 Other Basic Data Types

Boolean: Y = yes or true, N = no or false.

currsymbol: A three-letter code that identifies the currency used for a request or response. The currency codes are based on ISO-4217. For more information about currencies, refer to section 5.2.

URL: String form of a World Wide Web Uniform Resource Location. It should be fully qualified including protocol, host, and path. *A-255*

CHAPTER 4 OFX SECURITY

OFX provides several options for ensuring the security of customer transactions. This chapter describes the OFX security framework, security goals, types of security, and financial institution (FI) responsibilities.

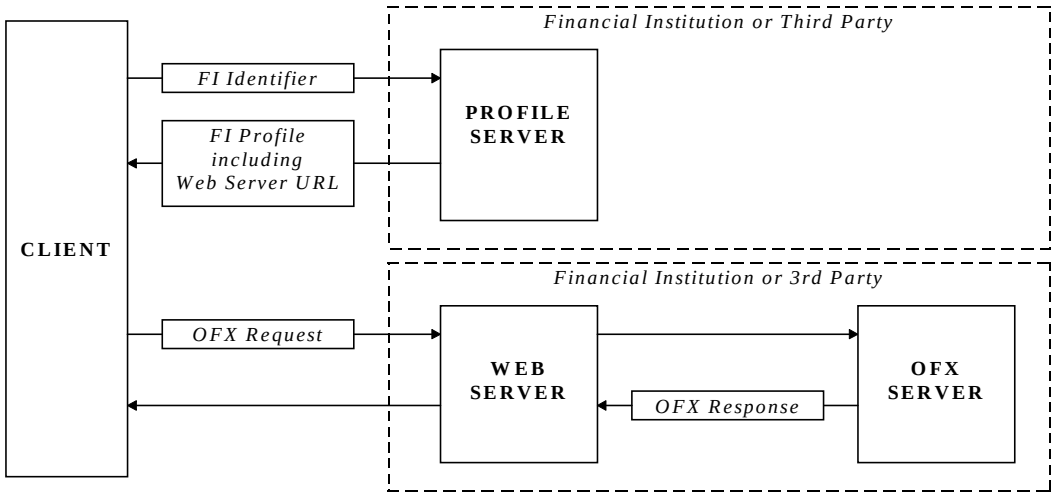
4.1 Security Concepts in OFX

4.1.1 Architecture

OFX security applies to the communication paths between a client and the profile server, a client and the Web server, and, when the OFX server is separate from the Web server, a client and the OFX server. The diagram below illustrates the initial order in which these communications occur, assuming that the client already has the URL for the FI profile server.

The bootstrap process for a client is:

- From the FI Profile Server, the client gets the URL of the FI Web server, so that it can retrieve a particular message set.
- The client sends an OFX request to the FI Web Server URL, from which it is forwarded to the OFX Server.
- The OFX Server sends back a response to the client via the Web Server.



4.1.2 Security Goals

The main goals of OFX security are:

- **Privacy:** Only the intended recipient can read a message. *Encryption* is a technique often used to ensure privacy.
- **Authentication:** The recipient of a message can verify the identity of the sender. In OFX, *passwords* allow an FI to authenticate a client, and *certificates* allow a client to authenticate a server.
- **Integrity:** A message cannot be altered after it is created A cryptographic *hash* is often used to assist integrity verification.

OFX specifies the minimum security required for Internet transactions and provides several security options, based on existing standards. Through its choice of security techniques and related options, an FI can achieve privacy, authentication, and integrity with varying degrees of assurance. For example, there are many kinds of encryption algorithms, most of which can be strengthened or weakened by changing the key size.

4.1.3 Security Standards

Several standards underlie Type 1 security:

- Certificates (X.509 v3) are used to identify and authenticate servers, and to convey their public keys.
- PKCS #1 block type 2 is the encryption format specified by the recipe (See section 4.2.2.4.3).
- RSA is the encryption algorithm.

4.1.3.1 Certificates and Certification Authorities

A certificate is a digitally signed document that binds a public key to an identity. It contains a public key that identifies information such as the name of the person or organization to whom the key belongs, an expiration date, a unique serial number, and additional descriptive information.

A certificate is useful for authentication because it is signed by a trusted third-party. This assures the verifier that the certificate has not been changed since it was signed. The entity which signs certificates is called a *certification authority,* or CA. A CA acts somewhat like a notary public: the reader of a document stamped by a notary public knows that the notary has checked the identity of the person who originated the document. By digitally signing someone's identity and public key, the CA affirms that the two go together.

If the client and server do not share a common CA, the client cannot validate the server's certificate. For this reason, OFX specifies a number of trusted CAs that all clients must accept and all servers must use.

Certificates are used in Type 1 security, as well as channel-level security through SSL. The format for these is defined by X.509 version 3. For more information, refer to ITU-T Rec. X.509, ISO/IEC 9594-8.

4.1.3.2 PKCS #1

The acronym, PKCS, stands for “Public Key Cryptography Standards,” a set of standards developed by a consortium and hosted by RSA. PKCS #1 is the RSA Encryption Standard, the rules for using RSA public key encryption. For the complete syntax of the PKCS #1 standard, refer to “Public-Key Cryptography Standards (PKCS)” published by RSA Data Security, Inc. at http://www.rsa.com/.

4.1.4 FI Responsibilities

OFX is designed with the understanding that there must be a security policy in place at each supporting financial institution. That policy must clearly delineate how customer data is secured, and how transactions are managed such that all parties to the transaction are protected according to accepted and recognized best common practices.

The decision regarding which users may perform a given operation on a given account must be determined by the financial institution. For example, is the specified user authorized to perform a transfer from the specified account? The financial institution must also determine whether the user has exceeded allowed limits on withdrawals, whether the activity on this account is unusual given past history, and other context-sensitive issues.

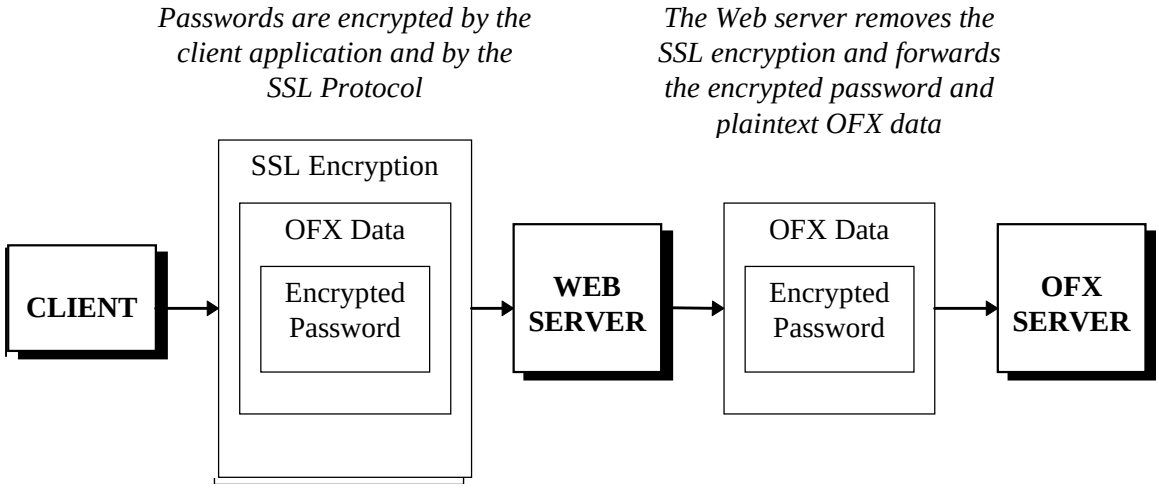
Although OFX provides many security options, an FI must support a minimal level of security. Tmal X-73.9v2 o b<</MC/

4.1.5 Security Levels: Channel vs. Application

With OFX, security can be applied at two different levels in the message exchange process.

- **Channel level:** Generally transparent to a client or server, channel-level security is built into the communication process, protecting messages between two ends of the "pipe." To secure messages during HTTP transport, client and server applications use the Secure Sockets Layer (SSL) protocol. SSL transparently protects messages exchanged between the client and the destination Web server. SSL authenticates the destination Web server using the Web server's certificate. Additionally, it provides privacy via encryption, and SSL-record integrity, i.e. the block of data sent in each transmission cannot be altered without detection.
- **Application level:** Transparent to and independent of the transport process, application-level security protects the user password sent from the client application all the way to the server application that handles the OFX messages. The server application typically resides beyond the destination Web server, secured behind an Internet firewall. Application-level security requires channel-level security.

The following diagram illustrates how channel-level and application-level security relate. The diagram shows the path of a request from the client to the server when application-level encryption is used.



Channel-level security is sufficient for most message sets, provided that the network architecture at the destination is adequately secure; however, application-level password encryption can allow a more flexible back-end architecture with a high level of security.

4.2.1.3 Trusted Certificate Authorities

4.2.2 Application-Level Security

4.2.2.1 Specification in FI Profile

For each message set listed in the FI profile response, the <MSGSETCORE> aggregate describes the security required for that message set.

The <OFXSEC> element defines the type of application-level security required for the message set. <OFXSEC> can have one of the following values, which also are used in the SECURITY element of the OFX headers:

	Description
NONE	Do not use any application-level security
TYPE1	Use Type 1 application-level security

Application-level security requires channel-level security.

4.2.2.2 Type 1 Protocol Overview

The goal of the Type 1 protocol is to protect the user password all the way to the destination OFX server. In the absence of client certificates, this password is the primary vehicle for client authentication and is therefore worthy of special consideration.

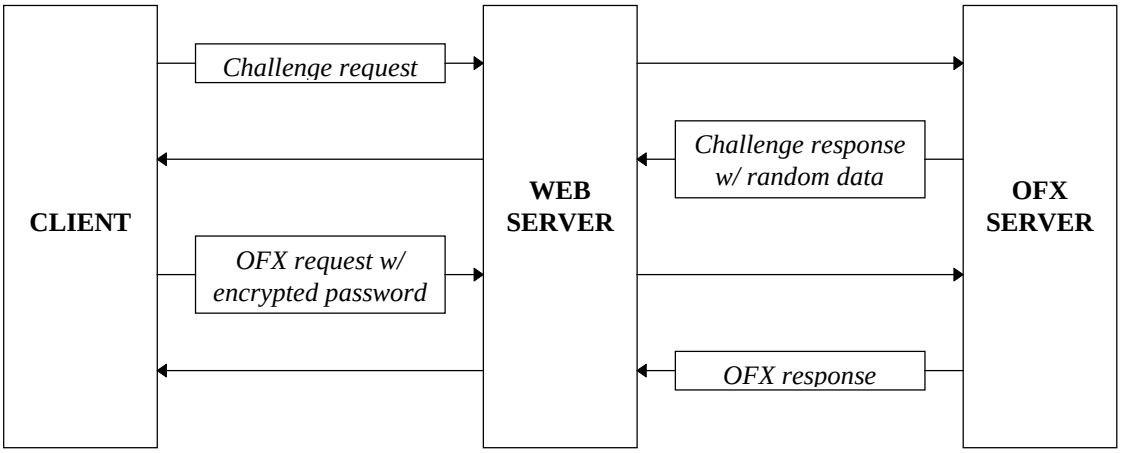
Type 1 requires channel-level security, *i.e.* SSL. Though the password is well protected by SSL alone in the client to Web server connection, the server-side network architecture may render the password less secure while it is in transit between the Web and OFX servers. With Type 1, the user password is not decrypted until the request reaches the OFX server.

Type 1 applies only to the request part of a message; the server response is unaffected.

A simple approach would be to deliver the server's Type 1 certificate in the profile and use it to encrypt the password, but that would permit a *replay attack*. An attacker could capture a transaction, including encrypted password, and replay it to the server. It wouldn't matter that the password remained unknown.

To prevent the *replay attack*, the server introduces some random data to the process, data which is unpredictably different for each transmission. The client asks for the random data with a challenge request. The server sends it, along with its Type 1 certificate, in the challenge response. The client then uses that random data in the encryption process, thereby assuring the server that the client response is associated with this and only this interaction.

The following diagram illustrates:



4.2.2.3 Type 1 Protocol Notation

In this section, the expression, $C = E_A (M)$, means that plain text M is encrypted either symmetrically or asymmetrically with key A into ciphertext C. The expression, $M = D_A(C)$ signifies the inverse operation (decryption), in which ciphertext C is decrypted into plain text M using key A. If C was encrypted asymmetrically, then A in the latter case is understood to be the private component of the key. The expression, A || B, indicates that B is concatenated to A.

4.2.2.4 Type 1 Protocol Implementation

Type 1 application-level security provides additional password secrecy. These are the steps for conducting a Type 1 transaction (unless otherwise noted, the term “Server” in this section refers to the Financial Institution Server):

1. Client obtains the Server’s profile from the Profile Server (see Chapter 7, "FI Profile")
2. Client establishes an SSL connection with the Server (see section 4.2.1)
3. Client sends <CHALLENGERQ> to Server (see section 4.2.2.4.1)
4. Server sends <CHALLENGERS> which contains a nonce and the Server’s Type 1 certificate (see section 4.2.2.4.2)
5. Client builds a transaction request and sends it to the Server (see section 4.2.2.4.3)
6. Server parses the request, verifying the user password, and either rejects or processes the transaction (see section 4.2.2.4.4)

The following table lists data elements used in the Type 1 protocol:

	Type	*Description*
BT	octet, length 1	Block Type byte. BT = 0x02
CT1	octet string, length 128	Ciphertext: the PKCS #1 RSA encryption of EB with KS. $CT1 = E_{KS}(EB)$
CT2	printable ASCII, length 171	Encoded Ciphertext: the RADIX-64 encoding of CT1 (see RFC 1113, §4.3.2.4 and §4.3.2.5). CT2 = RADIX64(CT1)
D	octet string, length 68	Data: the user data to be encrypted. D = NC || P || T
EB	octet string, length 128	Encryption Block: the formatted plain text block, ready for encryption. EB = 0x00 || BT || PS || 0x00 || D
KS	RSA key, modulus length 1,024 bits	Server's Type 1 RSA key
NC	octet string, length 16	Client Nonce: string of random octets generated by the Client
NS	octet string, length 16	Server Nonce: string of random octets generated by the Server
P	printable ASCII, null-padded, length 32	Password: shared by the Client and Financial Institution, null-padded on the right
PS	octet string, length 57	Padding String: each octet is pseudo-random and non-zero
T	octet string, length 20	Authentication Token. T = SHA1(NS || P || NC)

```
struct {
  unsigned char nc[16];
  unsigned char p[32];
  unsigned char t[20];
} D;
struct {
  unsigned char null1 = 0x00;
  unsigned char bt = 0x02;
  unsigned char ps[57];
  unsigned char null2 = 0x00;
  struct D d;
} EB;
```

4.2.2.4.1 Challenge request

Client sends a <CHALLENGERQ> to the Server.

4.2.2.4.2 Challenge response

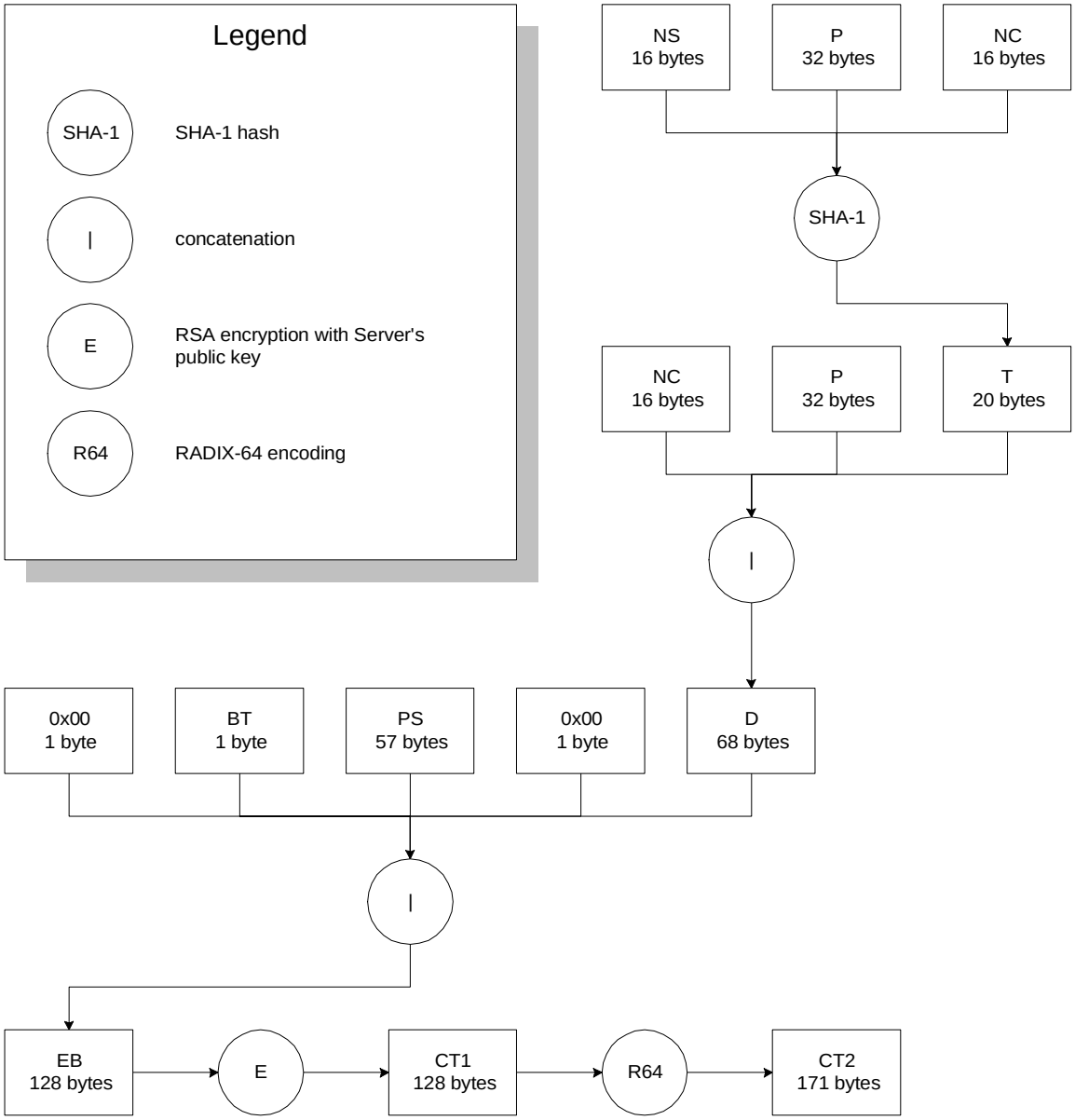
Server sends a <CHALLENGERS> to the client. This response contains the Server's Type 1 certificate and NS.

4.2.2.4.3 Building the OFX Request

1. Client generates 16 random octets and places them in NC (see RFC 1750 for recommendations on entropy generation)
2. Client obtains the User's password (P)
3. Client computes T = SHA1(NS || P || NC)
4. Client generates 57 pseudo-random, non-zero octets and places them in PS (NC may be used to seed the pseudo-random number generator)
5. Client sets D = NC || P || T
6. Client sets EB = 0x00 || BT || PS || 0x00 || D
7. Client RSA-encrypts EB using the Server's Type 1 public key (obtained from the Server's Type 1 certificate): CT1 = E_{KS}(EB) (see PKCS #1, §§8.2-8.4)
8. Client encodes the ciphertext for transport: CT2 = RADIX64(CT1). See RFC 1113, §4.3.2.4 and §4.3.2.5. This is a standard encoding method supported by RSA's Bsafe library and others.
9. Client constructs the body of its OFX request
10. Client copies CT2 to the <USERPASS> field of the OFX <SONRQ>
11. Client sends the complete OFX request to the Server

In <PINCHRQ>, the steps are identical, except that in step 2, P is set to <NEWUSERPASS> and in step 10, CT2 is copied to the <NEWUSERPASS> field of the <PINCHRQ>.

The diagram below illustrates the creation of CT2.



4.2.2.4.4 Parsing the OFX Request

1. Server reads the OFX SECURITY header in the request file to ascertain whether Type 1 processing should be used on this message. If Type 1 is not used, skip to step 6.
2. Server extracts CT2 from the <USERPASS> field of the OFX <SONRQ> and removes the encoding to obtain CT1 (see RFC 1113, §4.3.2.4 and §4.3.2.5)
3. Server decrypts CT1 to obtain EB: EB = D_{KS}(CT1) (see PKCS #1, §9)
4. Server extracts D from EB, then extracts NC, P, and T from D
5. Server looks up the Client's password in its database, and computes SHA1(NS || P || NC). If the result does not match T, Server terminates the session and reports the error to the client
6. Server processes the request and returns confirmation to the Client

In <PINCHRQ>, the steps are identical except that in step 2, CT2 is obtained from the <NEWUSERPASS> field of the <PINCHRQ> and in step 5, the server does not look up the extracted new password in a database.

CHAPTER 5 INTERNATIONAL SUPPORT

5.1 Language and Encoding

Most of the content in OFX is language-neutral. However, some error messages, balance descriptions, and similar elements contain text meant to appear to the financial institution customers. There are also cases, such as e-mail records, where customers need to send text in other languages. To support worldwide languages, OFX relies on standard XML mechanisms to encode text.

The encoding declaration of the standard XML declaration specifies the character set being used. Servers should respond to clients using the same encoding as was sent in the client's request.

Clients identify the language in the signon request. OFX specifies languages by three-letter codes as defined in ISO-639. Servers report their supported languages in the profile (see Chapter 7, "FI Profile"). If a server cannot support the language requested by the client, it must return an error and not process the rest of the transactions.

5.2 Currency <CURDEF> <CURRENCY> <ORIGCURRENCY>

In each transaction involving amounts, responses include a default currency identification, <CURDEF>. The values are based on the ISO-4217 three-letter currency identifiers.

Within each transaction, specific parts of the response might need to report a different currency. Where appropriate, aggregates include an optional <CURRENCY> aggregate. The scope of a <CURRENCY> aggregate is everything within the same aggregate that the <CURRENCY> aggregate appears in, including nested aggregates, unless overridden by a nested <CURRENCY> aggregate. For example, specifying a <CURRENCY> aggregate in an investment statement detail means that the unit price, transaction total, commission, and all other amounts are in terms of the given currency, not the default currency.

Note that there is no way for two or more individual elements that represent amounts—and are directly part of the same aggregate—to have different currencies. For example, there is no way in a statement download to have a different currency for the <LEDGERBAL> and the <AVAILBAL>, because they are both directly members of <STMTRS>. In most cases, you can use the optional <BAL> aggregates to overcome this limitation, since <BAL> aggregates accept individual <CURRENCY> aggregates.

The default currency for a request is the currency of the source account. For example, the currency for <BANKACCTFROM>.

The <CURRATE> should be the one in effect throughout the scope of the <CURRENCY> aggregate. It is not necessarily the current rate. Note that the <CURRATE> needs to take into account the choice of the FI for formatting of amounts (that is, where the decimal is) in both default and overriding currency, so that a

5.3 Country-Specific Element Values

Some of the elements in OFX have values that are country-specific. For example, <USPRODUCTTYPE> is useful only within the United States. OFX will extend in each country as needed to provide elements that accept values useful to that country. Clients in other countries that do not know about these elements must simply skip them.

In some cases, an element value represents a fundamental way of identifying something, yet there does not exist a world-wide standard for such identification. Examples include bank accounts and securities. In these cases, OFX must define a single, extensible approach for identification. For example, CUSIPs are used within the U.S., but not in other countries. However, CUSIPs are fundamental to relating investment securities, holdings, and transactions. Thus, a security ID consists of a two-part aggregate: one to identify the naming scheme, and one to provide a value. OFX will define valid naming schemes as necessary for each country.

CHAPTER 6 DATA SYNCHRONIZATION

6.1 Overview

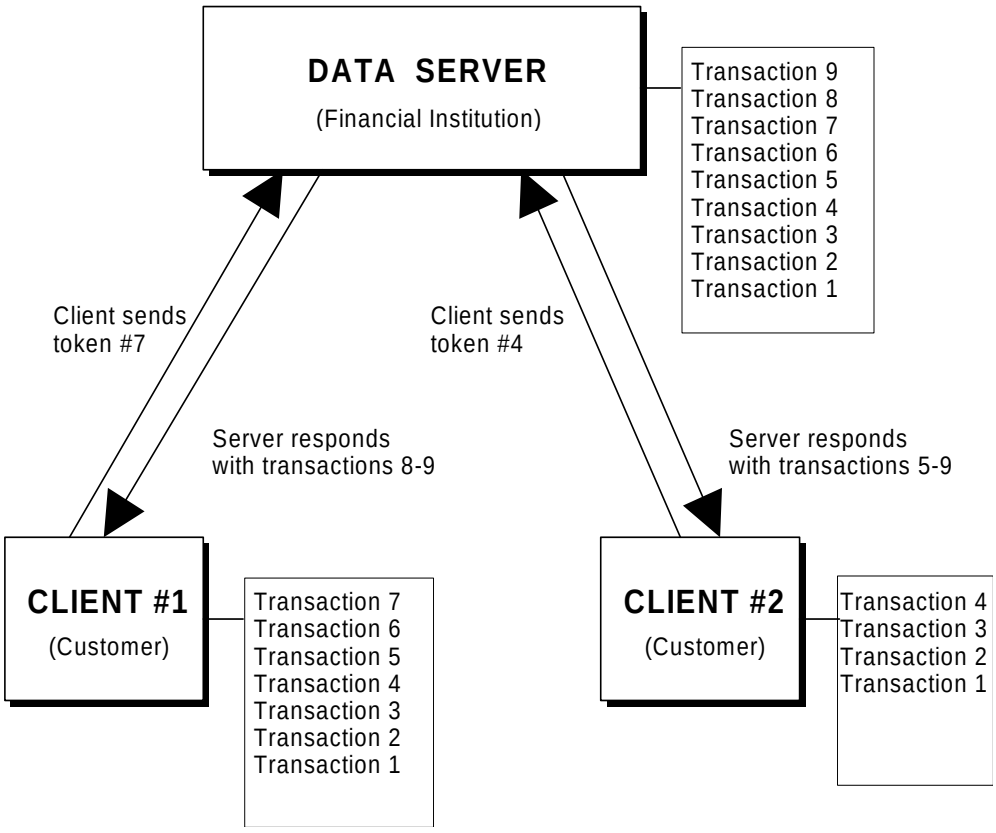
6.3 Data Synchronization Approach

A simple solution is to make sure that clients can always obtain information from the server for a reasonable length of time after it is initially sent. Clients can request recent responses—whether due to client-initiated work or other status changes on the server—by supplying the previous endpoint in the response history. Servers should always supply a new endpoint whenever they supply responses. These endpoints are described by the <TOKEN> element.

To ensure a consistent state after a failure (for example, dropped client connections or a client crash before updating its database), the client must store all data returned in a sync response before updating the saved token for that account and object type. After a failure, the next sync attempt using the old token might download information already reflected in the client database. But, re-integration of that data is much preferred over losing all changes between the old and new token values.

If a user switches to an outdated backup file, then the most recent endpoint known to the client will be older than the most recent endpoint known to the server.

If multiple clients are in use, each will send requests based on its own current endpoint, so that both clients will obtain complete information from the server. This is one reason why OFX responses carry enough information from the request to enable them to be processed independent from the requests. The diagram below shows the interaction between clients and servers.



OFX relieves the server from maintaining any special error-recovery state information. However, OFX requires the server to maintain a history of individual responses and a <TOKEN> to identify a position in the history. This token is commonly a time stamp, but it need not be. Because of the freedom a server has in choosing values for its <TOKEN>s, a client must not assume any sequential relationship between <TOKEN>s based on the <TOKEN> values.

> **Note:** OFX does not require servers to store responses based on individual connections. Also, not all requests are subject to synchronization. For example, OFX does not require servers to store statement-download responses separately for data synchronization.

6.4 Data Synchronization Specifics

OFX performs synchronization separately for each type of response. In addition, a synchronization request might include further identifying information, such as a specific account number. This specification defines the additional information for each synchronization request.

Each OFX service identifies the objects that are subject to data synchronization. For example, a bank-statement download is a read-only operation from the server. A client can request it again; consequently, there is no data synchronization for this type of response.

6.4.1 Tokens

The basis for synchronization is a *token* as defined by the <TOKEN> element. The server can create a token in any way it wishes. The client simply holds the token for possible use in a future synchronization request.

The server can derive a token from one of the following:

- Time stamp
- Sequential number
- Unique nonsequential number
- Other convenient values for a server

OFX reserves the following tokens:

- <TOKEN>0 (zero) from the client requests all available history for the referenced account (if specified) and object type. Servers should send all relevant transactions that are accessible, allowing a new client to know about work done by other clients. If a user's account has never been used with OFX, the server returns no history.
- Servers should return <TOKEN>–1 (negative one) in the event they must respond with an error. For more information, see section 6.4.4.

In all other cases, the server can use different types of tokens for different types of responses, if suitable for the server.

Clients must send either a <REFRESH>Y request (if supported by the server) or <TOKEN>0 in their initial synchronization request for each account (if necessary) and object type. As described in section 6.6, a server's response to either request should bring the client up-to-date. The <REFRESH>Y response would not detail how or when an object reached its current state. But, the <TOKEN>0 response might not list every relevant object (for example, some early history that the server has already purged, which might include a payment that was scheduled far in the past, but not yet due.) Should the client require full history information initially, OFX recommends a <REFRESH>Y request together with a <TOKEN>0 request.

Tokens can contain up to 10 characters in V1 message sets; see Chapter 3, "Common Aggregates, Elements, and Data Types." Tokens must be unique only with respect to the type of synchronization request and the additional information in that request. For example, a bill payment synchronization request takes an account number; therefore, a token needs to be unique only within payments for the account. In sync requests which do not include an account number, token values are scoped to the current user. For example, a token in a payee synchronization request needs to be unique only within payees for the signed on user.

The server can use different types of tokens for different types of responses, if suitable for the server.

Servers will not have infinite history available, so synchronization responses can optionally include a <LOSTSYNC>Y (yes) if the old token in the synchronization request was older than the earliest available history. This element allows clients to alert users that some responses have been lost.

> **Note:** Tokens are unrelated to <TRNUID>s, <SRVRTID>s, and <FITID>s, each of which serves a specific purpose and has its own scope and lifetime.

A <SRVRTID> is not appropriate as a <TOKEN> for bill payment. A single payment has a single <SRVRTID>, but it can undergo several state changes over its life and thus have several entries in the token history.

6.4.2 The Synchronization Process

There are three different ways a client and a server can conduct their requests and responses:

- Explicit synchronization—A client can request synchronization without sending any other OFX requests. The client sends a synchronization request, including the current token for that type of request. The response includes responses more recent than the given token, along with the current token.
- Synchronization with new requests—A client can request synchronization as part of any new request. The client gives the latest token it has. The response includes responses to the new requests plus any others that became available since the time of the token in the request, along with the current token. An aggregate contains the requests so that the server can process the new requests and update the token as a single action.
- New requests without synchronization—A client can make new requests without providing a token. In this case, it expects only responses to the new requests. A subsequent request for synchronization will cause the server to send this response again, because the client did not receive the current token.

SYNC responses should return a new <TOKEN> only if new activity was generated for a set of transactions (e.g. payee, payment, intrabank transfers, payment email, etc.). Alternatively, if a server always returns a new <TOKEN> even if no new activity was generated, the server should remember that the old and new <TOKEN> values are both up-to-date with respect to <REJECTIFMISSING>Y

Each request and response that requires data synchronization will define a synchronization aggregate. The aggregate tells the server which kind of data it should synchronize. By convention, these aggregates always have SYNC as part of their names, for example, <PMTSYNCRQ>. These aggregates can be used on their own to perform explicit synchronization, or as wrappers around one or more new transactions. For example, <PMTSYNCRQ> aggregates request synchronization and may include new work.

Some clients can choose to perform an explicit synchronization before sending any new requests. This practice allows clients to be up-to-date before sending any new requests. Other clients can simply send new requests as part of the synchronization request.

If a client synchronizes in one file, then sends new work inside a synchronization request in a second file, there is a small chance that additional responses became available between the two connections. There is an even smaller chance that these would be conflicting requests, such as modifications to the same object. However, some clients and some requests might require absolute control, so that the user can be certain that they are changing known data. To support this, synchronization requests can optionally specify <REJECTIFMISSING> element. The element tells a server that it should reject all enclosed requests if the supplied <TOKEN> is out of date before considering the new requests. That is, if any new responses became available, whether related to the incoming requests or not (but in scope of the synchronization request), the server should immediately reject the requests. It should still return the new responses. A client can then try again until it finds a stable window to submit the work. See section 6.5 for more information about conflict detection and resolution.

Note: If <REJECTIFMISSING>Y causes enclosed requests to be rejected, this rejection can be done in one of two ways:

- Embedded requests are completely ignored – they are not included in the response.
- Embedded requests are returned with a 2000 (or 6502 for recent servers) error. This is the preferred approach.

The password change request and response present a special problem. See section 2.5.2 for further information.

6.4.3 Synchronizable Objects

OFX allows synchronization of email (in all message sets), service activations, changes to user information, stop checks, banking notifications, transfers (both types), recurring transfers (both types), wire transfers, payees, payments, and recurring payments. OFX includes the following synchronization request/response pairs.

	Request	*Response*
8.6.4	<ACCTSYNCRQ>	<ACCTSYNCRS>
8.7	<CHGUSERINFOSYNCRQ>	<CHGUSERINFOSYNCRS>
9.2.4	<MAILSYNCRQ>	<MAILSYNCRS>
11.12.1	<STPCHKSYNCRQ>	<STPCHKSYNCRS>
11.12.2	<INTRASYNCRQ>	<INTRASYNCRS>
11.12.3	<INTERSYNCRQ>	<INTERSYNCRS>
11.12.4	<WIRESYNCRQ>	<WIRESYNCRS>
11.12.5	<RECINTRASYNCRQ>	<RECINTRASYNCRS>
11.12.6	<RECINTERSYNCRQ>	<RECINTERSYNCRS>
11.12.7	<BANKMAILSYNCRQ>	<BANKMAILSYNCRS>
11.12.8	<LOANMAILSYNCRQ>	<LOANMAILSYNCRS>
12.8.2	<PMTMAILSYNCRQ>	<PMTMAILSYNCRS>
12.9.4	<PAYEESYNCRQ>	<PAYEESYNCRS>
12.10.1	<PMTSYNCRQ>	<PMTSYNCRS>
12.10.2	<RECPMTSYNCRQ>	<RECPMTSYNCRS>
13.11.2	<INVMAILSYNCRQ>	<INVMAILSYNCRS>
14.6	<PRESMAILSYNCRQ>	<PRESMAILSYNCRS>

6.4.4 Token and Full Synchronization Summary

In review, tokens are used to identify a point in an activity continuum. Each client maintains a current token that identifies a place on that continuum. When sent to the server, the server can determine whether or not the client is up-to-date and send history if not. For instance, if ten activities have occurred for a particular type of synchronized activity and a client knows about the first eight activities, the token sent in the request will show this and the server will respond with the missing two, along with the newest token, thus bringing the client up to date. Several clients may be kept up-to-date with each other in this way, presuming all are accessing the same userid/accountid (depending on the activity) within the same FI.

There is no need to recreate a stream of responses that emulate the entire history of the object. An add response that reflects the current state is sufficient. For example, if you create a model and then modify it several times, even if this history would have been available for a regular synchronization, servers should only send a single add that reflects the current state.

Due to the large volume of data which might be included in the response, clients should not perform <MAILSYNCRQ> (or, one of the service-specific equivalents such as <BANKMAILSYNCRQ>) with <REFRESH>Y.

A client that wants only the current token, without refresh or synchronization, makes requests with <TOKENONLY>Y.

In all cases, servers should send the current ending <TOKEN> for the synchronization request in refresh responses. This allows a client to perform regular synchronization requests in the future.

The following table summarizes the options in a client synchronization request:

	Description
Client synchronization option; <TOKEN>, <TOKENONLY>, or <REFRESH>	
<TOKEN>	Previous value of <TOKEN> received for this type of synchronization request from server; 0 for first-time requests; *token*
<TOKENONLY>	Request for just the current <TOKEN> without the history, *Boolean*
<REFRESH>	Request for refresh of current state, *Boolean*
<REJECTIFMISSING>	If Y, do not process requests if client <TOKEN> is out of date, *Boolean*

Note: Compliant clients should not send synchronization requests matching those listed below. Nonetheless, servers should handle such requests and respond as described.

- <TOKENONLY>N has no useful meaning and clients should not send such a request. Servers may interpret <TOKENONLY>N in the sync request the same as <TOKENONLY>Y.
- <REFRESH>N has no useful meaning and clients should not send such a request. Servers may interpret <REFRESH>N in the sync request the same as <REFRESH>Y.
- If a client embeds transaction requests in a <REFRESH> or <TOKENONLY> sync request, the server should respond in such a way that the <REFRESH> data or returned <TOKEN> reflects a specific state, after the transactions have processed. Since servers are not required to reduce the data about any particular object to a single addition response, embedded transactions may be processed before or after the <REFRESH> data is retrieved. As with all synchronization responses, the returned <TOKEN> must reflect the actions of all embedded transactions.

- <REJECTIFMISSING>Y is illegal unless accompanied by <TOKEN>. If received in the same wrapper as <TOKENONLY> or <REFRESH>, the server should fail that synchronization request (as described in section 6.6.1).

6.6.1 Synchronization Errors

When a client sends an unrecognized or “bad” token, the server response should be one of the following. (Note that <LOSTSYNC> is an optional element):

- Return <TOKEN>-1, <LOSTSYNC>Y, with no history
- Return <TOKEN>X (the current token), <LOSTSYNC>Y, with no history
- Return <TOKEN>X (the current token), <LOSTSYNC>Y, with full history (i.e. treat as if it were a <TOKEN>0)

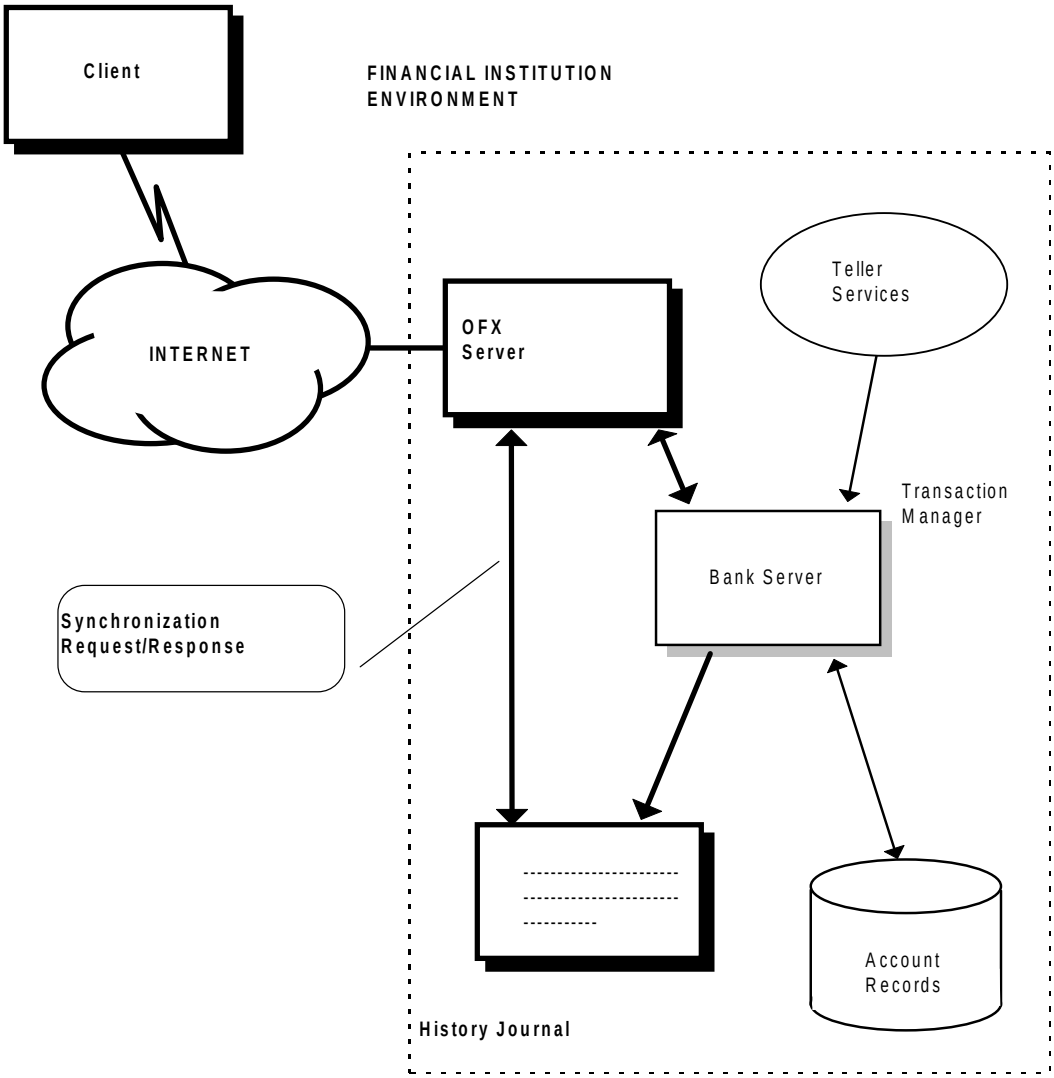
If the synchronization request included a bad account number or BANKID, or signon failed, or an account was closed, etc. the response should include <TOKEN>–1, optionally <LOSTSYNC>N, and no history.

6.7 Typical Server Architecture for Synchronization

This section describes how an FI can approach supporting synchronization based on the assumption that modifications to an existing financial server will be kept to a minimum.

The simplest approach is to create a history database separate from the existing server. This history could consist of the actual OFX transaction responses (<*xxx*TRNRS> aggregates) that are available to a synchronization request, or simply the information required to re-create the responses upon request from the client. The history database could index records by token, response type, and any other identifying information for that type, such as account number. Clearly, this database must include all <TRNUID>s for all transactions it contains. OFX recommends that <TRNUID>s be stored for as long as possible so that they may be used to detect duplicate client requests even after the original requests have been purged from the synch database.

The diagram below shows a high-level model of the OFX architecture for a financial institution. Notice that the diagram shows the presence of a history journal.



The server adds responses to the history journal for any action that takes place on the existing server. This is true whether the OFX requests initiate the action or, in the case of recurring payments, it happens automatically on the server. Once added to the history journal, the server can forget them.

The areas of the OFX server that process synchronization requests need only search this history database for matching responses that are more recent than the incoming token.

For a refresh request, an OFX server would access the actual bank server to obtain the current state rather than recent history.

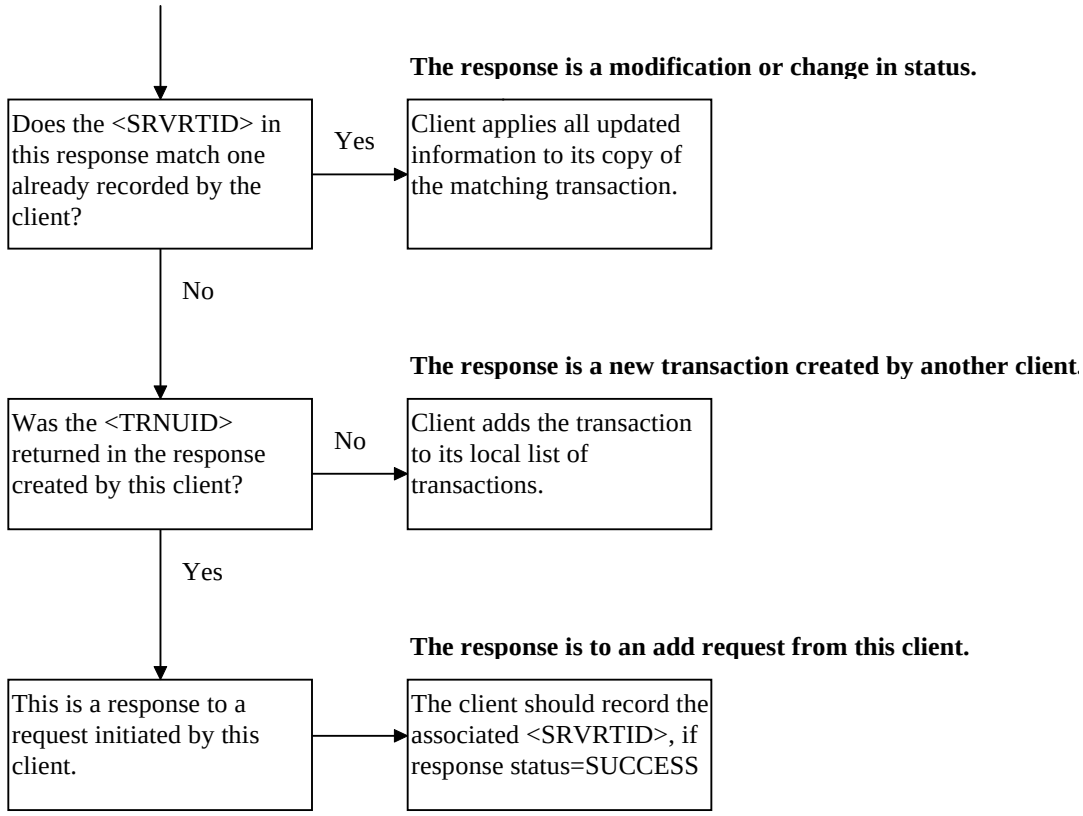
Periodically the bank server would purge the history server of older entries.

Only requests that are subject to synchronization need to have entries in the history database. Statement downloads do not involve synchronization; therefore, the FI server should not add these responses to the history database. Since statement downloads are usually the largest in space and the most frequent, eliminating these saves much of the space a response history might otherwise require.

More sophisticated implementations can save even more space. The history database could save responses in a coded binary form that is more compact than the full OFX response format. Some FIs might have much or all of the necessary data already in their servers; consequently, they would not require new data. An FI could regenerate synchronization responses rather than recall them from a database.

6.8 Typical Client Processing of Synchronization Results

The diagram below shows a general flowchart of what an OFX client would do with the results of a synchronization request. Most requests and responses subject to data synchronization contain both <TRNUID> and <SRVRTID>.



6.9 Simultaneous Connections

It is increasingly common for a server to get simultaneous or overlapping requests from the same user from two different front ends. OFX requires a server to process each set of requests sent in a file as an atomic action. Servers can deal with the problems that arise with simultaneous use in two ways:

- Allow simultaneous connections, ensure each is processed atomically, and use the data synchronization mechanism to bring the two clients up to date. This is the preferred method.
- Lock out all but one user at a time, returning the error code 15501 for multiple users.

6.10 Synchronization Alternatives

Although it is **RECOMMENDED** that OFX servers implement full synchronization as described in this chapter, an alternate approach, “lite synchronization,” could be easier for some servers to support. This approach focuses only on error recovery and does not provide any support for multiple clients, multiple data files, or use of backup files. The approach is to preserve the message sets while simplifying the implementation.

In addition, some clients might prefer to use file-based error recovery with all servers, even if the client and some servers support full synchronization. This section first describes file-based error recovery and lite synchronization, and then explains the rules that clients and servers use to decide how to communicate.

Lite synchronizing servers may support both file-based error recovery *and* <REFRESH>Y. This type of server is called a Refresh-capable Lite Synchronizing Server.

For information on how these types of synchronization are profiled, see section 7.2.1.

6.10.1 File-Based Error Recovery

Because only full synchronization supports error recovery, an alternative is needed for lite synchronization. Servers using lite synchronization keep a copy of the last response file they sent. This is the basis for what is often called “file-based error recovery.” Clients requesting that servers prepare for error recovery generate a globally unique ID for each file they send. Two OFX headers are associated with error recovery:

OLDFILEUID—UID of the last request and response that was successfully received and processed by the client

NEWFILEUID—UID of the current file

The format of these is the same as used with <TRNUID> as documented in section 2.4.6.

Servers use the following rules:

If NEWFILEUID is set to NONE, the client is not requesting file-based error recovery for this session. The server does not need to save the response file. If NEWFILEUID is set to NONE and OLDFILEUID matches a previous request file (see below), the client may be ending use of file-based error recovery.

If NEWFILEUID matches a previous request file, the client is requesting error recovery. The server should send the matching saved response file.

Note: If NEWFILEUID matches a previous request file, then the request identified by the NEWFILEUID must contain exactly the same set of transactions as the previous request file. Servers can reject the file if it contains new or modified transactions. In particular, clients should disallow new <PINCHRQ> transactions during error recovery. For more information about <PINCHRQ> and synchronization, see section 2.5.2.

If NEWFILEUID is not set to NONE and does not match a previous request file, the client is preparing for error recovery. The server should save the response file in case the data does not reach the client.

If OLDFILEUID is set to NONE, the server may ignore the presence of this header. The server should not search for a response file to delete. Clients should initiate file-based error recovery by sending OLDFILEUID set to NONE and NEWFILEUID set to a unique value.

If OLDFILEUID matches a file saved on the server, then OLDFILEUID is a file that the client has successfully processed and the server can delete it.

If OLDFILEUID is not set to NONE and does not match a previous request file, the server should ignore the presence of this header. Either the server has purged the associated request file without explicit request from the client or the client is requesting error recovery with identical headers to the initial request attempt (NEWFILEUID should not match a previous request file in this case).

Note: While it may indicate a client error for OLDFILEUID and NEWFILEUID to hold identical values other than NONE, the server should ignore this OLDFILEUID header. Earlier rules in this list detail how the server should handle the request file (based solely upon the NEWFILEUID value).

A server should not save more than one file per client data file thread (history of FILEUID values). Servers should purge response files in response to an explicit client request (reference in the OLDFILEUID header) or after some long period (at least 2 months). Clients must not abuse this storage requirement by (for example) setting OLDFILEUID to the header used three request files previously. The server should preserve response files on a per-thread basis. This approach would support multiple clients or data files per user. But, the server has the option to ignore these needs and purge response files as soon as another valid request arrives for the same <USERID>. In either case, if an error recovery attempt comes after the corresponding error recovery file is purged, the server will not recognize the request as an attempt at error recovery. The server would simply process it as a new request. In this case, the server should recognize duplicate transaction UIDs for client-initiated work, such as payments, and then reject them individually. Server-generated responses would be lost to the client.

A server should not save a response file when it is useless to do so. Specifically, the server should not save a response file when the request fails parsing or when the request was rejected due to a <SONRQ> problem (e.g. invalid <USERID>).

If all accounts are shared between two (or more) users (for example, husband and wife have separate online access to the same list of joint accounts and none others), some identifiers may differ and should be maintained separately by the client. Thus, clients should initiate error recovery and maintain/generate *xxx*FILEUID values on a per-user basis.

6.10.1.1 File-Based Error Recovery and Authentication

There are two aspects of error recovery authentication which must be considered, request validation and password validation.

6.10.1.1.1 Request Validation

Clients request file-based error recovery by including the old and new session UIDs in the header. If these are absent, servers need not save the response file for error recovery. Clients request synchronization by using those synchronization requests defined throughout this specification.

6.11 Examples

Here is an example of full synchronization using bill payment as the service. OFX Payments provides two different synchronization requests and responses, each with their own token; one for payment requests and one for repeating payment model requests. Note that these simplified examples do not include the outer <OFX> layer, <SONRQ>, and so forth.

Client A requests synchronization:

```
<PMTSYNCRQ>
  <TOKEN>123</TOKEN>
  <REJECTIFMISSING>N</REJECTIFMISSING>
  <BANKACCTFROM>
    <BANKID>121000248</BANKID>
    <ACCTID>123456789</ACCTID>
    <ACCTTYPE>CHECKING</ACCTTYPE>
  </BANKACCTFROM>
</PMTSYNCRQ>
```

The server sends in response:

```
<PMTSYNCRS>
  <TOKEN>125</TOKEN>
  <LOSTSYNC>N</LOSTSYNC>
  <BANKACCTFROM>
    <BANKID>121000248</BANKID>
    <ACCTID>123456789</ACCTID>
    <ACCTTYPE>CHECKING</ACCTTYPE>
  </BANKACCTFROM>
  <PMTTRNRS>
    <TRNUID>123</TRNUID>
    <STATUS>
      ... status details
    </STATUS>
    <PMTRS>
      ... details on a payment response
    </PMTRS>
  </PMTTRNRS>
  <PMTTRNRS>
    <TRNUID>546</TRNUID>
    <STATUS>
      ... status details
    </STATUS>
```

```
    <PMTRS>
      ... details on another payment response
    </PMTRS>
  </PMTTRNRS>
</PMTSYNCRS>
```

Client A was missing two payment responses, which the server provides. At this point, client A is synchronized with the server. Client A now makes a new payment request, and includes a synchronization update as part of the request. This update avoids having to re-synchronize the expected response at a later time.

```
<PMTSYNCRQ>
  <TOKEN>125</TOKEN>
  <REJECTIFMISSING>N</REJECTIFMISSING>
  <BANKACCTFROM>
    <BANKID>121000248</BANKID>
    <ACCTID>123456789</ACCTID>
    <ACCTTYPE>CHECKING</ACCTTYPE>
  </BANKACCTFROM>
  <PMTTRNRQ>
    <TRNUID>12345</TRNUID>
    <PMTRQ>
      ... details of a new payment request
    </PMTRQ>
  </PMTTRNRQ>
</PMTSYNCRQ>
```

The response to this new request:

```
<PMTSYNCRS>
  <TOKEN>126</TOKEN>
  <LOSTSYNC>N</LOSTSYNC>
  <BANKACCTFROM>
    <BANKID>121000248</BANKID>
    <ACCTID>123456789</ACCTID>
    <ACCTTYPE>CHECKING</ACCTTYPE>
  </BANKACCTFROM>
  <PMTTRNRS>
    ... details on a payment response to the new request
  </PMTTRNRS>
</PMTSYNCRS>
```

The client now knows that the server has processed the payment request it just made, and that nothing else has happened on the server since it last synchronized with the server.

CHAPTER 7 FI PROFILE

7.1 Overview

OFX clients use the profile to learn the capabilities of an OFX server. This information includes general properties such as account types supported, user password requirements, specific messages supported, and how the client should batch requests and where to send the requests. A client obtains a portion of the profile when a user first selects an FI. The client obtains the remaining information prior to sending any actual requests to that FI. The server uses a time stamp to indicate whether the server has updated the profile, and the client checks periodically to see if it should obtain a new profile.

In more detail, a profile response contains the following sections, which a client can request independently:

- Message Sets – list of services and any general attributes of those services. Message sets are collections of messages that are related functionally. They are generally subsets of what users see as a service.
- Signon realms – FIs can require different signons (user ID and/or password) for different message sets. Because there can only be one signon per <OFX> block, a client needs to know which signon the server requires and then provide the right signon for the right batch of messages.

The profile message is itself a message set. In files, OFX uses the <PROFMSGSETV1> aggregate to identify this profile message set.

The following sections describe the general use of profile information.

7.1.1 Message Sets

A message set may be thought of as representing an available financial service. A message set itself is a collection of related messages. For example, Chapter 11, "Banking," defines several message sets: statement download, credit card statement download, intrabank transfers, and so forth. A server may route all of the messages in a message set to a single URL and merge their versions together.

Clients and servers generally use message sets as the granularity to decide what functionality they will support. A “banking” server can choose to support the statement download and intrabank transfer message sets, but not the wire transfer message set. Attributes are available in many cases to further define how OFX supports a message set.

The profile applies only to the requests a client might expect the server to honor. That is, clients should not send requests to servers unless support is indicated. However, the server may send unsupported responses in a sync response as information is entered out of band. A client is required to at least parse such a file.

Clients should assume the burden of checking the profile and not sending a transaction which the server does not support. If the client goes ahead and sends such a transaction, the server may either return an HTTP 400 syntax error, or ignore unsupported elements and aggregates. In the latter case, assuming no

other problems occur in processing that request, servers may return warning code 2028 (Request element unknown). The response file should not contain the unsupported elements or aggregates.

Each portion of the OFX specification that defines messages also defines the message set to which those messages belong. This includes what additional attributes are available for those messages and whether OFX requires the message set or it is optional.

7.1.1.1 Profile Support for Downloading Images

Clients may request images in statement download and/or closing requests in various message sets. Prior to requesting these images, clients must verify that support exists on the server for image download. This is indicated by the presence of the <IMAGEMSGSET> aggregate in the profile response, as well as the <IMAGEPROF> aggregate in the profile response for the specific message set in question. For instance, if a client wishes to request transaction images in the banking statement download request, the client must verify the presence of <IMAGEMSGSET> in the profile as well as transaction image support in the <IMAGEPROF> aggregate in the <BANKMSGSET> in the profile. Image download requests are allowed in OFX 2.1.1 in the Banking, Credit Card, Loan and Investments message sets.

7.1.2 Version Control

Message sets are the basis of version control. Over time there will be new versions of the message sets, and at any given time servers will likely want to support more than one version of a message set. Clients should also be capable of supporting as many versions as possible. Through the profile, clients discover which versions are supported for each message set. Clients and servers exchange messages at the highest common level for each message set.

If banking version 1 is at one URL (A) and billpay version 1 is at another URL (B), both may need version 1 of signon to be used. In that case, <MSGSETCORE> inside <BANKMSGSETV1> would refer to <URL>A and <MSGSETCORE> inside <BILLPAYMSGSETV1> would refer to <URL>B, but <MSGSETCORE> inside <SIGNONMSGSETV1>may refer to either URL or to some other. As mentioned in Section 2.5.4, the <URL> included in <SIGNONMSGSETV1> does not restrict where the <SIGNotherh may be

Servers correcting problems with per-user <PROFRQ> requests (which previously caused error responses) must update the FI Profile to tell compliant clients to retry.

For more information about signon requests, refer to section 2.5.

7.1.5 Profile Request <PROFRQ>

A profile request indicates which profile components a client desires. It also indicates what the client's routing capability is. Profiles returned by the FI must be compatible with the requested routing style, or the server returns an error.

Profile requests are not subject to synchronization.

Profile requests must appear within a <PROFTRNRQ> transaction wrapper.

	Description
<PROFRQ>	Profile-request aggregate
<CLIENTROUTING>	Identifies client routing capabilities, see table below
<DTPROFUP>	Date and time client last received a profile update, *datetime*
</PROFRQ>	

	Description
NONE	Client cannot perform any routing. All URLs must be the same. All message sets share a single signon realm.
SERVICE	Client can perform limited routing. See details below.
MSGSET	Client can route at the message-set level. Each message set can have a different URL and/or signon realm.

The SERVICE option supports clients that can route bill payment messages to a separate URL from the rest of the messages. Because the exact mapping of message sets to the general concept of bill payment can vary by client and by locale, this specification does not provide precise rules for the SERVICE option. Each client will define its requirements.

7.2 Profile Response <PROFRS>

To determine whether the client has the latest version of the FI profile, the server checks the date and time passed by the client in <DTPROFUP>.

If the client has the latest version of the FIs profile, the server returns status code 1 in the <STATUS> aggregate of the profile-transaction aggregate <PROFTRNRS>. The server does not return a profile-response aggregate <PROFRS>.

> **Note:** Not sending a response aggregate in this case is an exception to rules outlined in sections 2.4.6 and 3.1.5.

If the client does not have the latest version of the FI profile, the server responds with the profile-response aggregate <PROFRS> in the profile-transaction aggregate <PROFTRNRS>. The response includes message set descriptions, signon information, and general contact information.

Tag	*Description*
<PROFRS>	Profile-response aggregate
<MSGSETLIST>	Beginning list of message set information
<xxxMSGSET>	One or more message set aggregates
</xxxMSGSET>	
</MSGSETLIST>	
<SIGNONINFOLIST>	Beginning of signon information
<SIGNONINFO>	Zero or more signon information aggregates. Though the DTD allows an empty <SIGNONINFOLIST>, servers should provide at least one signon realm (include a minimum of one <SIGNONINFO> aggregate in the <PROFRS> response).
</SIGNONINFO>	
</SIGNONINFOLIST>	
<DTPROFUP>	Time this was updated on server, *datetime*
<FINAME>	Name of institution, *A-32*
<ADDR1>	FI address, line 1, *A-32*
<ADDR2>	FI address, line 2, *A-32*
<ADDR3>	FI address, line 3. Use of <ADDR3> requires the presence of <ADDR2>, *A-32*
<CITY>	FI address city, *A-32*
<STATE>	FI address state, *A-5*
<POSTALCODE>	FI address postal code, *A-11*
<COUNTRY>	FI address country; 3-letter country code from ISO/DIS-3166, *A-3*

The common message set information <MSGSETCORE> is as follows:

	Description
<MSGSETCORE>	Common-message-set-information aggregate
<VER>	Version number of the message set, (for example, <VER>1 for version 1 of the message set), *N-5* Because this information is already provided by the surrounding <*xxx*MSGSETV*n*> wrapper, <VER> should be ignored by OFX clients. Nonetheless, servers should use the supported value (<VER>1) consistent with that wrapper.
<URL>	URL where messages in this set are to be sent, *URL*
<OFXSEC>	Security level required for this message set; see Chapter 4, "OFX Security." *NONE or TYPE 1.*
<TRANSPSEC>	Y if transport-level security must be used, N if not used; see Chapter 4, "OFX Security." *Boolean*
<SIGNONREALM>	Signon realm to use with this message set, *A-32*
<LANGUAGE>	1 or more. Language supported, *language.* If more than one language is supported, multiple <LANGUAGE> elements can be sent.
<SYNCMODE>	FULL for full synchronization capability LITE for lite synchronization capability See Chapter 6, "Data Synchronization," for more information.
<REFRESHSUPT>	Y if server supports <REFRESH>Y within synchronizations. This option is irrelevant for full synchronization servers. Clients must ignore <REFRESHSUPT> (or its absence) if the profile also specifies <SYNCMODE>FULL. For lite synchronization, the default is N. Without <REFRESHSUPT>Y, lite synchronization servers are not required to support <REFRESH>Y requests, *Boolean*
<RESPFILEER>	Y if server supports file-based error recovery, *Boolean* See Chapter 6, "Data Synchronization," for more information.
<SPNAME>	Service provider name, *A-32* Some financial institutions out-source their OFX servers to a service provider. In such cases, the SPNAME element should be included in the MSGSETCORE.
</MSGSETCORE>	

Note: For all message sets currently defined in OFX, <TRANSPSEC>Y must be specified.

Note: Within a <MSGSETCORE> aggregate, the <VER> element defines the version number of that message set. It does not refer to the version number of the OFX specification or the DTD files. For more information about message sets and version numbers, refer to section 2.4.5.

Note: Within a message set, there can be more than one <MSGSETCORE> aggregate with the same value for <VER>, or the same value for <URL>, but not the same value for both. The pair must be unique for each instance of <MSGSETCORE> within a message set. Multiple <MSGSETCORE>s with the same value for <VER> are used in instances such as signon or registration, which may have the same version sent to multiple URLs for different services.

7.2.2 Signon Realms

A signon realm identifies a set of messages that can be accessed using the same password. Realms are used to disassociate signons from specific services, allowing FIs to require different signons for different message sets. In practice, FIs will want to use the absolute minimum number of realms possible to reduce the user's workload.

Tag	*Description*	
<SIGNONINFO>	Signon-information aggregate	
<SIGNONREALM>	Identifies this realm, *A-32*	
<MIN>	Minimum number of password characters, *N-2*	
<MAX>	Maximum number of password characters, *N-2*	
<CHARTYPE>	Type of characters allowed in password:	
	ALPHAONLY	Password may not contain numeric characters. The server would allow "abbc", but not "1223" or "a122".
	NUMERICONLY	Password may not contain alphabetic characters. The server would allow "1223", but not "abbc" or "a122".
	ALPHAORNUMERIC	Password may contain alphabetic or numeric characters (or both). The server would allow "abbc", "1223", or "a122".
	ALPHAANDNUMERIC	Password must contain both alphabetic and numeric characters. The server would allow "a122", but not "abbc" or "1223".
<CASESEN>	Y if password is case-sensitive, *Boolean*	
<SPECIAL>	Y if special characters are allowed over and above those characters allowed by <CHARTYPE> and <SPACES>, *Boolean*	
<SPACES>	Y if spaces are allowed over and above those characters allowed by <CHARTYPE> and <SPECIAL>, *Boolean*	

Tag	Description
<PINCH>	Y if server supports <PINCHRQ> (PIN change requests), *Boolean*
<CHGPINFIRST>	Y if server requires clients to change USERPASS as part of first signon. Clients must ignore <CHGPINFIRST> if the profile indicates <PINCH>N. Note that if <MFACHALLENGEFIRST> is also Y, this pin change request should be sent immediately *after* the session containing MFACHALLENGE authentication. *Boolean*
<USERCRED1LABEL>	Text prompt for user credential. If it is present, a third credential (USERCRED1) is required in addition to USERID and USERPASS. *A-64*
<USERCRED2LABEL>	Text prompt for user credential. If it is present, a fourth credential (USERCRED2) is required in addition to USERID, USERPASS and USERCRED1. If present, USERCRED1LABEL must also be present. *A-64*
<CLIENTUIDREQ>	Y if CLIENTUID is required, *Boolean*
<AUTHTOKENFIRST>	Y if server requires clients to send AUTHTOKEN as part of the first signon, *Boolean*
<AUTHTOKENLABEL>	Text label for the AUTHTOKEN. Required if server supports AUTHTOKEN, *A-64*
<AUTHTOKENINFOURL>	URL where AUTHTOKEN information is provided by the institution operating the OFX server. Required if server supports AUTHTOKEN, *A-255*
<MFACHALLENGESUPT>	Y if the server supports MFACHALLENGE functionality, *Boolean*
<MFACHALLENGEFIRST>	Y if the client is required to send MFACHALLENGERQ as part of the first signon, before sending any other requests, *Boolean*
</SIGNONINFO>	

7.2.3 Status Codes

	Meaning
0	Success (INFO)
1	Client is up-to-date (INFO)
2000	General error (ERROR)

7.3 Profile Message Set Profile Information

The profile message set functions the same way as all other message sets; therefore, it contains a profile description for that message set. Because <PROFMSGSET> is always part of a message set response, it is described here. Servers must include the <PROFMSGSET> as part of the profile response

CHAPTER 8 ACTIVATION & ACCOUNT INFORMATION

8.1 Overview

The Signup message set defines three messages to help users get setup with their FI:

- Enrollment – informs FI that a user wants to use OFX and requests that a password be returned
- Accounts – asks the FI to return a list of accounts and the services supported for each account
- Activation – allows a client to tell the FI which services a user wants on each account

There is also a message to request name and address changes.

Clients use the account information request on a regular basis to look for changes in a user's account information. A time stamp is part of the request so that a server has to report only new changes. Account activation requests are subject to data synchronization, and will allow multiple clients to learn how the other clients have been enabled.

In OFX request files, the <SIGNUPMSGSRQV1> aggregate identifies the Signup messages.

8.2 Approaches to User Sign-Up with OFX

The message sets in this chapter are designed to allow both FIs and clients to support a variety of sign-up procedures. There are four basic steps a user needs to go through to complete the sign-up:

1. **Select the FI.** OFX does not define this step or provide message sets to support it. Client developers and FIs can let a user browse or search this information on a web site, or might define additional message sets to do this within the client. At the conclusion of this step, the client will have some minimal profile information about the FI, including the set of services supported and the URL to use for the next step.
2. **Enrollment and password acquisition.** In this step, the user identifies and authenticates itself to the FI *without a password*. In return, the user obtains a password (possibly temporary) to use with OFX. FIs can perform this entire step over the telephone, through a combination of telephone requests and a mailed response, or at the FI web site. FIs can also use the OFX enrollment message to do this by means of the client. The response can contain a temporary password or users can wait for a mailed welcome letter containing the password.
3. **Account Information.** In this step, the user obtains a list of accounts available for use with OFX, and which specific services are available for each account. Even if users have enrolled over the telephone, clients will still use this message set to help users properly set up the accounts within the client. Clients periodically check back with the FI for updates.
4. **Service Activation.** The last step is to activate specific services on specific accounts. The activation messages support this step. Synchronization is applied to these messages to ensure that other clients are aware of activated services.

The combination of media-interface through which an FI accomplishes these steps can vary. FIs might wish to do steps two through four over the telephone. Clients will still use OFX messages in steps 3 and 4 to automatically set up the client based on the choices made by the user over the phone. Other FIs might wish to have the entire user experience occur within the client. Either way, the OFX sign-up messages support the process.

8.3 Users and Accounts

To support the widest possible set of FIs, OFX assumes that individual users and accounts are in a many-to-many relationship. Consider a household with three accounts:

- Checking 1 – held individually by one spouse
- Checking 2 – held jointly by both
- Checking 3 – held individually by the other spouse

Checking 2 should be available to either spouse, and the spouse holding Checking 1 should be able to see both Checking 1 and 2.

OFX expects FIs to give each user their own user ID and password. Each user will go through the enrollment step separately. A given account need only be activated once for a service; not once for each user. Clients will use the account information and activation messages to combine information about jointly held accounts.

If an FI prefers to have a single user ID and password per household or per master account, it will have to make this clear to users through the enrollment process. It is up to the FI to assign a single user ID and password that can access all three of the checking accounts described above.

8.4 Enrollment and Password Acquisition

The main purpose of the enrollment message is to communicate a user's intent to access the FI by way of OFX and to acquire a password for future use with OFX. Some FIs might return a user ID and an initial password in the enrollment response, while others will send them by way of regular mail.

> **Note:** The client may not know the user ID and password when it sends the enrollment request, in such a case the <USERID> and/or <USERPASS> may be set to lower case "anonymous" followed by 23 zeroes.

Enrollment requests are not subject to synchronization. If the client does not receive a response, it will simply re-request the enrollment. If a user successfully enrolls from another client before the first client obtains a response, the server should respond to subsequent requests from the first client with status code:

13501 - user already enrolled.

8.4.1 User IDs

The OFX <SONRQ> requires a user ID to uniquely identify a user to an FI. The server must accept the user ID with or without punctuation.

UserIDs are assigned in various ways. Some FIs have existing user IDs that they use for other online activities that they want to use for OFX as well. Others might create new IDs specifically for OFX. Finally, some FIs might assign IDs while others might allow users to create them. To ensure security and help prevent identity fraud, Financial Institutions are discouraged from using Social Security Number for Customer ID or PIN/ Password.

Because users do not usually know either their OFX sign-on user ID or their password at time of enrollment, the enrollment response is designed to return both. The enrollment request allows users to optionally provide a user ID, which an FI can interpret as their existing online ID or a suggestion for what their new user ID should be. Ideally, the enrollment process should explain ID syntax to users.

8.4.2 Enrollment Request <ENROLLRQ>

The enrollment request captures enough information to identify and authenticate a user as being legitimate and that it has a relationship with the FI.

FIs might require that an account number be entered as part of the identification process. However, this is discouraged since the account information request is designed to automatically obtain all account information, avoiding the effort and potential mistakes of a user-supplied account number.

It is **RECOMMENDED** that FIs provide detailed specifications for user IDs and passwords along with information about the services available when a user is choosing an FI.

The enrollment request must appear within an <ENROLLTRNRQ> transaction wrapper.

	Description
<ENROLLRQ>	Enrollment-request aggregate
<FIRSTNAME>	First name of user, *A-32*
<MIDDLENAME>	Middle name of user, *A-32*
<LASTNAME>	Last name of user, *A-32*
<ADDR1>	Address line 1, *A-32*
<ADDR2>	Address line 2, *A-32*
<ADDR3>	Address line 3. Use of <ADDR3> requires the presence of <ADDR2>, *A-32*
<CITY>	City, *A-32*
<STATE>	State or province, *A-5*
<POSTALCODE>	Postal code, *A-11*
<COUNTRY>	3-letter country code from ISO/DIS-3166, *A-3*
<DAYPHONE>	Daytime telephone number, *A-32*
<EVEPHONE>	Evening telephone number, *A-32*
<EMAIL>	Electronic e-mail address, *A-80*
<USERID>	Actual user ID if already known, or preferred user ID if user can choose, *A-32*
<TAXID>	ID used for tax purposes (such as SSN), may be same as user ID, *A-32*
<SECURITYNAME>	Mother's maiden name or equivalent, *A-32*
<DATEBIRTH>	Date of birth, *date*
<*xxx*ACCTFROM>	An account description aggregate for an existing account at the FI, for identification purposes only. For example, <BANKACCTFROM> or <INVACCTFROM>.
</*xxx*ACCTFROM>	
</ENROLLRQ>	

This enrollment request is intended for use only by individuals. Business enrollment will be defined in a later release.

8.4.3 Enrollment Response <ENROLLRS>

The main purpose of the enrollment response is to acknowledge the request. In those cases where FIs permit delivery of an ID and a temporary password, the response also provides for this. Otherwise the server will send the real response to the user by way of regular mail, electronic mail, or over the telephone.

If enrollment is successful, but the server does not return the ID and password in the response, a server is REQUIRED to use status code 13000 and provide some information to the user by means of the <MESSAGE> element in the <STATUS> aggregate about what to expect next.

The enrollment response must appear within an <ENROLLTRNRS> transaction wrapper.

	Description
<ENROLLRS>	Enrollment-response aggregate
<TEMPPASS>	Temporary password, *A-32*
<USERID>	User ID, *A-32*
<DTEXPIRE>	Time the temporary password expires (if <TEMPPASS> included), *datetime*
</ENROLLRS>	

8.4.4 Enrollment Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
13000	User ID & password will be sent out-of-band (INFO)
13500	Unable to enroll user (ERROR)
13501	User already enrolled (ERROR)
15508	Transaction not authorized (ERROR)

8.4.5 Examples

An enrollment request:

```
<ENROLLTRNRQ>
  <TRNUID>12345</TRNUID>
  <ENROLLRQ>
    <FIRSTNAME>Joe</FIRSTNAME>
    <MIDDLENAME>Lee</MIDDLENAME>
    <LASTNAME>Smith</LASTNAME>
    <ADDR1>21 Main St.</ADDR1>
    <CITY>Anytown</CITY>
    <STATE>TX</STATE>
    <POSTALCODE>87321</POSTALCODE>
    <COUNTRY>USA</COUNTRY>
    <DAYPHONE>123-456-7890</DAYPHONE>
    <EVEPHONE>987-654-3210</EVEPHONE>
    <EMAIL>jsmith@isp.com</EMAIL>
    <USERID>jls</USERID>
    <TAXID>123-456-1234</TAXID>
    <SECURITYNAME>jbmam</SECURITYNAME>
    <DATEBIRTH>19530202</DATEBIRTH>
  </ENROLLRQ>
</ENROLLTRNRQ>
```

And the reply might be:

```
<ENROLLTRNRS>
  <TRNUID>12345</TRUNID>
  <STATUS>
    <CODE>0</CODE>
    <SEVERITY>INFO</SEVERITY>
  </STATUS>
  <ENROLLRS>
    <TEMPPASS>changeme</TEMPPASS>
    <USERID>jls</USERID>
    <DTEXPIRE>20060305</DTEXPIRE
  </ENROLLRS>
</ENROLLTRNRS>
```

8.5.1 Request <ACCTINFORQ>

The <ACCTINFORQ> request must appear within an <ACCTINFOTRNRQ> transaction wrapper.

	Description
<ACCTINFORQ>	Account-information-request aggregate
<DTACCTUP>	Last <DTACCTUP> received in a response, *datetime*
</ACCTINFORQ>	

8.5.2 Response <ACCTINFORS>

The <ACCTINFORS> response must appear within an <ACCTINFOTRNRS> transaction wrapper.

	Description
<ACCTINFORS>	Account-information-response aggregate
<DTACCTUP>	Date and time of last update to this information on the server, *datetime*
<ACCTINFO>	Zero or more account information aggregates Left out of the response when nothing is found for the current user. **Note:** When <DTACCTUP> indicates the client is up-to-date, server should not return surrounding <ACCTINFORS>.
</ACCTINFO>	
</ACCTINFORS>	End of account information response

8.5.3 Account Information Aggregate <ACCTINFO>

	Description
<ACCTINFO>	Account-information-record aggregate
<DESC>	Description of the account, *A-80*
<PHONE>	Telephone number for the account, *A-32*
<xxxACCTINFO>	Service-specific account information, defined in each service chapter. Some services may include additional elements. Refer to service chapters for details.
<xxxACCTFROM>	Service-specific account identification. For a given service *xxx*, there can be at most one <*xxx*ACCTINFO> returned. For example, you cannot return two <BANKACCTINFO> aggregates.
</xxxACCTFROM>	
<SVCSTATUS>	AVAIL = Available, but not yet requested PEND = Requested, but not yet available

8.5.5 Examples

An account information request:

```
<ACCTINFOTRNRQ>
  <TRNUID>12345</TRNUID>
  <ACCTINFORQ>
    <DTACCTUP>20050101</DTACCTUP>
  </ACCTINFORQ>
</ACCTINFOTRNRQ>
```

And a response for a user with access to one account, supporting banking:

```
<ACCTINFOTRNRS>
  <TRNUID>12345</TRNUID>
  <STATUS>
    <CODE>0</CODE>
    <SEVERITY>INFO</SEVERITY>
  </STATUS>
  <ACCTINFORS>
    <DTACCTUP>20050301</DTACCTUP>
    <ACCTINFO>
      <DESC>Power Checking</DESC>
      <PHONE>8002223333</PHONE>
      <BANKACCTINFO>
        <BANKACCTFROM>
          <BANKID>1234567789</BANKID>
          <ACCTID>12345</ACCTID>
          <ACCTTYPE>CHECKING</ACCTTYPE>
        </BANKACCTFROM>
        <SUPTXDL>Y</SUPTXDL>
        <XFERSRC>Y</XFERSRC>
        <XFERDEST>Y</XFERDEST>
        <SVCSTATUS>ACTIVE</SVCSTATUS>
      </BANKACCTINFO>
    </ACCTINFO>
  </ACCTINFORS>
</ACCTINFOTRNRS>
```

8.6 Service Activation

Clients inform FIs that they wish to start, modify, or terminate a service for an account by sending service activation requests. These are subject to data synchronization, and servers should send responses to inform clients of any changes, even if the changes originated on the server.

Clients use these records during the initial user sign-up process. Once a client learns about the available accounts and services (by using the account information request above, or by having a user directly enter the required information), it sends a series of service ADD requests.

If a user changes any of the identifying information about an account, the client sends a service activation request containing both the old and the new account information. Servers should interpret this as a change in the account, not a request to transfer the service between two existing accounts, and all account-based information such as synchronization tokens should continue. If a user or FI is reporting that a service should be moved between two existing accounts, service must be terminated for the old account and started for the new account. The new account will have reset token histories, as with any new service.

Each service to be added, changed, or removed is contained in its own request because the same real-world account might require different <*xxx*ACCTFROM> aggregates depending on the type of service.

8.6.1 Activation Request <ACCTRQ>

The <ACCTRQ> request must appear within an <ACCTTRNRQ> transaction wrapper.

	Description
<ACCTRQ>	Account-service-request aggregate
Action identification. Specify either <SVCADD>, <SVCCHG>, or <SVCDEL>	Action aggregate, either <SVCADD>, <SVCCHG>, or <SVCDEL>
<SVCADD> **</SVCCADD>**	Service-addition aggregate
-or-	
<SVCCHG> **</SVCCHG>**	Service-change aggregate
-or-	
<SVCDEL> **</SVCDEL>**	Service-deletion aggregate
<SVC>	Service to be added/changed/deleted BANKSVC = Banking service BPSVC = Payments service INVSVC = Investments PRESSVC = Bill presentment service
</ACCTRQ>	

8.6.1.1 Service Add Aggregate <SVCADD>

When a client sends a <SVCADD> to a financial institution routing particular messages to another service provider, it is up to the financial institution to determine whether or not an <ENROLLRQ> needs to be sent to the service provider along with the <SVCADD>. The FI may choose to always send an <ENROLLRQ> and ignore the 13550 error message responses, though this would only be reliable if <xxxACCTFROM> is included in the <ENROLLRQ>. The FI may also choose to keep a database of enrolled services, so as to send an <ENROLLRQ> only when the client is sending a <SVCADD> for a new service. The FI also has the option of sending <ENROLLRQ>s to all service providers when the client sends the initial <ENROLLRQ> to the FI.

	Description
<SVCADD>	Service-addition aggregate
<*xxx*ACCTTO>	Service-specific-account-identification aggregate (for example, <BANKACCTTO> or <INVACCTTO>)
</*xxx*ACCTTO>	
</SVCADD>	

8.6.1.2 Service Change Aggregate <SVCCHG>

	Description
<SVCCHG>	Service-change aggregate
<*xxx*ACCTFROM>	Service-specific-account-identification aggregate (for example, <BANKACCTFROM> or <INVACCTFROM>)
</*xxx*ACCTFROM>	
<*xxx*ACCTTO>	Service-specific-account-identification aggregate (for example, <BANKACCTTO> or <INVACCTTO>)
</*xxx*ACCTTO>	
</SVCCHG>	

8.6.1.3 Service Delete Aggregate <SVCDEL>

	Description
<SVCDEL>	Service-deletion aggregate
<*xxx*ACCTFROM>	Service-specific-account-identification aggregate (for example, <BANKACCTFROM> or <INVACCTFROM>)
</*xxx*ACCTFROM>	
</SVCDEL>	

8.6.2 Activation Response <ACCTRS>

The <ACCTRS> response must appear within an <ACCTTRNRS> transaction wrapper.

	Description
<ACCTRS> *Action identification. Specify either <SVCADD>, <SVCCHG>, or <SVCDEL>*	Account-service-response aggregate
<SVCADD> **</SVCADD>** -or- **<SVCCHG>** **</SVCCHG>** -or- **<SVCDEL>** **</SVCDEL>**	Service-addition aggregate Service-change aggregate Service-deletion aggregate
<SVC>	Service to be added/changed: BANKSVC = Banking service BPSVC = Payments service INVSVC = Investments PRESSVC = Bill Presentment service
<SVCSTATUS>	AVAIL = Available, but not yet requested PEND = Requested, but not yet available ACTIVE = In use
</ACCTRS>	

8.6.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2006	Source account not found (ERROR)
2007	Source account closed (ERROR)
2008	Source account not authorized (ERROR)
2009	Destination account not found (ERROR)
2010	Destination account closed (ERROR)
2011	Destination account not authorized (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
13502	Invalid service (ERROR)
15508	Transaction not authorized (ERROR)

8.6.4 Service Activation Synchronization

Service activation requests are subject to the standard data synchronization protocol. The scope of these requests and the <TOKEN> is the user ID. The request and response tags are <ACCTSYNCRQ> and <ACCTSYNCRS>.

8.6.4.1 Request <ACCTSYNCRQ>

	Description
<ACCTSYNCRQ> *Client synchronization option; <TOKEN>, <TOKENONLY>, or <REFRESH>*	Activation synchronization request aggregate
<TOKEN>	Previous value of <TOKEN> received for this type of synchronization request from server; 0 for first-time requests; *token*
<TOKENONLY>	Request for just the current <TOKEN> without the history, *Boolean*
<REFRESH>	Request for refresh of current state, *Boolean*
<REJECTIFMISSING>	If Y, do not process requests if client <TOKEN> is out of date, *Boolean*
<ACCTTRNRQ>	Account-service-request transactions (0 or more)
</ACCTTRNRQ>	
</ACCTSYNCRQ>	

8.6.4.2 Response <ACCTSYNCRS>

	Description
<ACCTSYNCRS>	Payee-list-request aggregate
<TOKEN>	New synchronization token, *token*
<ACCTTRNRS>	Account-service-response transactions (0 or more)
</ACCTTRNRS>	
<LOSTSYNC>	Y if the token in the synchronization request is older than the earliest entry in the server's history table. In this case, some responses have been lost. N if the token in the synchronization request is newer than or matches a token in the server's history table. *Boolean*
</ACCTSYNCRS>	

8.7 Name and Address Changes

Users may request that an FI update the official name, address, phone, and e-mail information using the <CHGUSERINFORQ>. All modified and unmodified elements are submitted in a change user information request, <CHGUSERINFORQ>. The lack of inclusion of a field in a change user request when that field was previously populated implies its deletion on the server. The response reports all of the current values. If the USERID element is not present in CHGUSERINFO, then the USERID from the SONRQ is assumed to be the identifier for the user in question. For security reasons, some of the fields in the <ENROLLRQ> cannot be changed online, such as tax ID and userID.

The transaction tags are <CHGUSERINFOTRNRQ> and <CHGUSERINFOTRNRS>. These messages are subject to synchronization, <CHGUSERINFOSYNCRQ>, and <CHGUSERINFOSYNCRS>.

8.7.1 Change User Information Request <CHGUSERINFORQ>

	Description
<CHGUSERINFORQ>	Change-user-information-request aggregate
<FIRSTNAME>	First name of user, *A-32*
<MIDDLENAME>	Middle name of user, *A-32*
<LASTNAME>	Last name of user, *A-32*
<ADDR1>	Address line 1, *A-32*
<ADDR2>	Address line 2. Use of <ADDR2> requires the presence of <ADDR1>, *A-32*
<ADDR3>	Address line 3. Use of <ADDR3> requires the presence of <ADDR2>, *A-32*
<CITY>	City, *A-32*
<STATE>	State or province, *A-5*
<POSTALCODE>	Postal code, *A-11*
<COUNTRY>	3-letter country code from ISO/DIS-3166, *A-3*
<DAYPHONE>	Daytime telephone number, *A-32*
<EVEPHONE>	Evening telephone number, *A-32*
<EMAIL>	Electronic e-mail address, *A-80*
</CHGUSERINFORQ>	

8.7.2 Change User Information Response <CHGUSERINFORS>

Description

8.7.4 Change User Information Synchronization

Change user information requests are subject to the standard data synchronization protocol. The scope of these requests and the <TOKEN> is the user ID. The request and response tags are <CHGUSERINFOSYNCRQ> and <CHGUSERINFOSYNCRS>.

8.7.4.1 Request <CHGUSERINFOSYNCRQ>

	Description
<CHGUSERINFOSYNCRQ> *Client synchronization option; <TOKEN>, <TOKENONLY>, or <REFRESH>*	Activation synchronization request aggregate
<TOKEN>	Previous value of <TOKEN> received for this type of synchronization request from server; 0 for first-time requests; *token*
<TOKENONLY>	Request for just the current <TOKEN> without the history, *Boolean*
<REFRESH>	Request for refresh of current state, *Boolean*
<REJECTIFMISSING>	If Y, do not process requests if client <TOKEN> is out of date, *Boolean*
<CHGUSERINFOTRNRQ>	Change user information request transactions (0 or more)
</CHGUSERINFOTRNRQ>	
</CHGUSERINFOSYNCRQ>	

8.7.4.2 Response <CHGUSERINFOSYNCRS>

	Description
<CHGUSERINFOSYNCRS>	Payee-list-request aggregate
<TOKEN>	New synchronization token, *token*
<LOSTSYNC>	Y if the token in the synchronization request is older than the earliest entry in the server's history table. In this case, some responses have been lost. N if the token in the synchronization request is newer than or matches a token in the server's history table. *Boolean*
<CHGUSERINFOTRNRS>	Change user information response transactions (0 or more)
<CHGUSERINFOTRNRS>	
</CHGUSERINFOSYNCRS>	

8.8 Signup Message Set Profile Information

A server which supports this message set must include the following aggregates as part of the profile <MSGSETLIST> response, since every server must support at least the account information and service activation messages. Servers indicate how enrollment should proceed: via the client, a given web page, or a text message directing users to some other method (such as a phone call).

	Description
<SIGNUPMSGSET>	Signup-message-set-profile-information aggregate
<SIGNUPMSGSETV1>	Opening tag for V1 of the message set profile information
<MSGSETCORE>	Common message set information, defined in Chapter 7, "FI Profile"
</MSGSETCORE>	
Enrollment options. Choose one of <CLIENTENROLL>, <WEBENROLL>, or <OTHERENROLL>.	
<CLIENTENROLL>	Client-based enrollment supported
<ACCTREQUIRED>	Y if account number is required as part of enrollment, *Boolean*
</CLIENTENROLL>	
-or-	
<WEBENROLL>	Web-based enrollment supported
<URL>	URL to start enrollment process, *URL*
</WEBENROLL>	
-or-	
<OTHERENROLL>	Some other enrollment process
<MESSAGE>	Message to consumer about what to do next (for example, a phone number), *A-80*
</OTHERENROLL>	
<CHGUSERINFO>	Y if server supports client-based user information changes, *Boolean*
<AVAILACCTS>	Y if server can provide information on accounts with SVCSTATUS available, N means client should expect to ask user for specific account information, *Boolean*
<CLIENTACTREQ>	Y if server allows clients to make service activation requests (<ACCTRQ>), N if server will only advise clients via synchronization of service additions, changes, or deletions. *Boolean*
</SIGNUPMSGSETV1>	
</SIGNUPMSGSET>	

CHAPTER 9 CUSTOMER TO FI COMMUNICATION

9.1 The E-Mail Message Set

The e-mail message set includes two messages: generic e-mail and generic MIME requests by way of URLs. In OFX files, the message set name is EMAILMSGSV1.

9.2 E-Mail Messages

OFX allows consumers and FIs to exchange messages. The message body can be placed in HTML so that FIs can provide some graphic structure to the message. Keep in mind that, as with regular World Wide Web browsing, an OFX client might not support some or all of the HTML formatting, so the text of the message must be clear on its own. Clients can request the server to send graphics (the images referenced in an <IMG> tag) as part of the response file, or clients can separately request those elements. If a server sends images, it should use the standard procedure for incorporating external data as described in Chapter 2, "Structure." Servers are not required to support HTML or to send images, even if the client asks.

A user or an FI can originate a message. E-mail messages are subject to data synchronization so that a server can send a response again if it is lost or if multiple clients use it.

Because e-mail messages cannot be replied to immediately, the response should just echo back the original message (so that data synchronization will get this original e-mail message to other clients). When the FI is ready to reply, it should generate an unsolicited response (<TRNUID>0) and the client will pick this up during synchronization.

	Server Responds
Account information From, To Subject Message	
	Account information From, To Subject Message Type

9.2.1 Regular vs. Specialized E-Mail

Several services with OFX define e-mail requests and responses that contain additional information specific to that service. To simplify implementation for OFX clients and servers, this section defines a <MAIL> aggregate that OFX uses in all e-mail requests and responses. For regular e-mail, the only additional information is an account-from aggregate and whether to include images in the e-mail response or not.

When users want to send messages about service-specific problems, service-specific messages are best. However, when service-specific mail transactions are not available, general mail is acceptable.

9.2.2 Basic <MAIL> Aggregate

	Description
<MAIL>	Core e-mail aggregate
<USERID>	User identification, *A-32*
<DTCREATED>	When message was created, *datetime*
<FROM>	Who the message is from, *A-32*
<TO>	Who the message should be delivered to, *A-32*
<SUBJECT>	Subject of message (plain text, not HTML), *A-60*
<MSGBODY>	Body of message, HTML-encoded or plain text depending on <USEHTML>, HTML-encoded text - *A-10000* Plain text - *A-2000*
<INCIMAGES>	Include images in the message body. *Boolean*
<USEHTML>	Y for HTML-formatted text. N for plain text. See section 9.2.2.2 for more information. *Boolean*
</MAIL>	

9.2.2.1 <INCIMAGES>

The meaning of the <INCIMAGES> element depends on whether the element appears in a request or response.

When used in a request, <INCIMAGES> indicates whether the client accepts mail that includes images in the message body.

	Description
<INCIMAGES>Y	The client accepts mail that includes images in the message body. In this case, the server can choose whether to send images in the response.
<INCIMAGES>N	The client does not accept mail that includes images in the message body. In this case, the server must not send images in the response.

When used in a response, <INCIMAGES> indicates whether the server included images in the message body.

When used in a response…	*Description*
<INCIMAGES>Y	The server included images in the message body.
<INCIMAGES>N	The server did not include images in the message body.

9.2.2.2 <USEHTML>

The meaning of the <USEHTML> element depends on whether the element appears in a request or response.

When used in a request, <USEHTML> indicates whether the client sends and accepts HTML-formatted text in the message body. If a server receives a <xxxMAILSYNCRQ> request with <USEHTML>Y set, the server should process the request whether or not it supports HTML mail. If a server does not support HTML mail, it should simply set the <USEHTML> flag to N in any transactions which are returned in the sync response.

When used in a request…	*Description*
<USEHTML>Y	The client is including HTML-formatted text in the message body. In addition, the client will accept mail responses that include HTML-formatted text in the message body. In this case, a server can choose whether to respond with HTML-formatted text or plain text.
<USEHTML>N	The client is not including HTML-formatted text in the message body. In addition the client will not accept mail responses that include HTML-

When used in a response, <USEHTML> indicates whether the message body includes HTML-formatted text or plain text.

	Description
<USEHTML>Y	The server is including HTML-formatted text in the message body.
<USEHTML>N	The server is including only plain text in the message body.

Note: When using HTML for the message body, clients and servers are REQUIRED to enclose the HTML in a CDATA section to protect the HTML markup: <![CDATA[... html ...]]>. For an example, see section 9.2.5.

9.2.3 E-Mail <MAILRQ> <MAILRS>

E-mail is subject to synchronization. The transaction aggregate is <MAILTRNRQ> / <MAILTRNRS> and the synchronization aggregate is <MAILSYNCRQ> / <MAILSYNCRS>.

Tag	Description
<MAILRQ>	E-mail-message-request aggregate
<MAIL>	Core e-mail aggregate
</MAIL>	
</MAILRQ>	

In a response, the <TRNUID> is zero if this is an unsolicited message or an out-of-band reply to a prior email request. Immediate responses (acknowledgments) to a request should contain the <TRNUID> of the user's original message. It is RECOMMENDED that servers include the <MESSAGE> of the user's message as part of the reply <MESSAGE>. The <MESSAGE> contents can include carriage returns to identify desired line breaks.

	Description
<MAILRS>	E-mail-message-response aggregate
<MAIL>	Core e-mail aggregate
</MAIL>	
</MAILRS>	

9.2.4 E-Mail Synchronization <MAILSYNCRQ> <MAILSYNCRS>

E-mail presents a special case with regards to synchronization. Since FIs will not immediately reply to a user's e-mail, the response to the user's e-mail only echoes the request and confirms that the e-mail was successfully received. The client receives the real response to the e-mail following a synchronization request.

Note that this synchronization action expects only the basic <MAILRS> responses. Specialized e-mail is received by means of their own synchronization requests.

	Description
<MAILSYNCRQ> *Client synchronization option; <TOKEN>, <TOKENONLY>, or <REFRESH>*	E-mail-synchronization-request aggregate
<TOKEN>	Previous value of <TOKEN> received for this type of synchronization request from server; 0 for first-time requests; *token*
<TOKENONLY>	Request for just the current <TOKEN> without the history, *Boolean*
<REFRESH>	Request for refresh of current state, *Boolean*
<REJECTIFMISSING>	If Y, do not process requests if client <TOKEN> is out of date, *Boolean*
<INCIMAGES>	Y if the client accepts mail with images in the message body, N if the client does not accept mail with images in the message body, *Boolean*
<USEHTML>	Y if client wants an HTML response, N if client wants plain text, *Boolean*
<MAILTRNRQ>	Mail-transaction-request aggregate (0 or more)
</MAILTRNRQ>	
</MAILSYNCRQ>	

	Description
<MAILSYNCRS>	E-mail-synchronization-response. aggregate
<TOKEN>	Server history marker, *token*
<LOSTSYNC>	Y if the token in the synchronization request is older than the earliest entry in the server's history table. In this case, some responses have been lost. N if the token in the synchronization request is newer than or matches a token in the server's history table. *Boolean*
<MAILTRNRS>	Missing e-mail response transactions (0 or more)
</MAILTRNRS>	
</MAILSYNCRS>	

9.2.5 E-Mail Example

In this example, a consumer requests information about the checking statement just downloaded. Since the financial institution will not immediately answer the inquiry, the immediate response only echoes the consumer's request and confirms that the request was successfully received.

The client receives the real response at a later time following a mail synchronization request. For an example of the mail synchronization request and response, see section 9.2.5.1.

> **Note:** This example omits the <OFX> top level and the signon <SONRQ>. Since this example uses HTML for the message body, it must protect the HTML content in an CDATA-marked section.

The request:

```
<MAILTRNRQ>
  <TRNUID>54321</TRNUID>
  <MAILRQ>
    <MAIL>
      <USERID>123456789</USERID>
      <FROM>James Hackleman</FROM>
      <TO>Noelani Federal Savings</TO>
      <SUBJECT>What do I need to earn interest?</SUBJECT>
      <DTCREATED>20050305</DTCREATED>
      <MSGBODY><![CDATA[<HTML><BODY>I didn't earn any interest this
month. Can you please tell me what I need to do to earn interest on this
account?</BODY></HTML>
]]></MSGBODY>
      <INCIMAGES>N</INCIMAGES>
      <USEHTML>Y</USEHTML>
    </MAIL>
```

```
  </MAILRQ>
</MAILTRNRQ>
```

The response from the FI:

```
<MAILTRNRS>
  <TRNUID>54321</TRNUID>
  <STATUS>
    <CODE>0</CODE>
    <SEVERITY>INFO</SEVERITY>
  </STATUS>
  <MAILRS>
    <MAIL>
      <USERID>123456789</USERID>
      <FROM>James Hackleman</FROM>
      <TO>Noelani Federal Savings</TO>
      <SUBJECT>What do I need to earn interest?</SUBJECT>
      <DTCREATED>20050305</DTCREATED>
      <MSGBODY><![CDATA[<HTML><BODY>I didn't earn any interest this
month. Can you please tell me what I need to do to earn interest on this
account?</BODY></HTML>]]></MSGBODY>
      <INCIMAGES>N</INCIMAGES>
      <USEHTML>Y</USEHTML>
    </MAIL>
  </MAILRS>
</MAILTRNRS>
```

9.2.5.1 E-Mail Synchronization Example

In the following example, the client has not yet received the reply to the e-mail sent in the previous example, so its <TOKEN> is one less than the server's. The server replies by giving the current <TOKEN> and the missed response.

```
<MAILSYNCRQ>
  <TOKEN>101</TOKEN>
  <REJECTIFMISSING>N</REJECTIFMISSING>
  <INCIMAGES>N</INCIMAGES>
  <USEHTML>Y</USEHTML>
</MAILSYNCRQ>

<MAILSYNCRS>
  <TOKEN>102</TOKEN>
   <MAILTRNRS>
```

```
    <TRNUID>0</TRNUID>          <!-- server initiated response -->
<STATUS>
       <CODE>0</CODE>
       <SEVERITY>INFO</SEVERITY>
    </STATUS>
    <MAILRS>
       <MAIL>
         <USERID>123456789</USERID>
         <DTCREATED>20050307</DTCREATED>
         <FROM>Noelani Federal Savings</FROM>
         <TO>James Hackleman</TO>
         <SUBJECT>Re: What do I need to earn interest?</SUBJECT>
         <MSGBODY><![CDATA[<HTML><BODY>You need to maintain $1000 in
this account to earn interest. Because your balance was only $750 this
month, no interest was earned. You could also switch to our new Checking
Extra plan that always pays interest. Call us or check our web page
http://www.fi.com/check-plans.html for more information.
Sincerely,
Customer Service Department


Original message:
I didn't earn any interest this month. Can you please tell me what I
need to do to earn interest on this account?</BODY></HTML>]]></MSGBODY>
         <INCIMAGES>N</INCIMAGES>
         <USEHTML>Y</USEHTML>
       </MAIL>
    </MAILRS>
  </MAILTRNRS>
</MAILSYNCRS>
```

9.3 Get HTML Page

Some responses (<PROFRS> and <FINDBILLERRS> for example) contain values that are URLs intended to be separately fetched by clients. Clients can use their own HTTP libraries to perform this fetch outside of the OFX specification. However, to insulate clients against changes in transport technology, and to allow for fetches that require the protection of an authenticated signon by a specific user, OFX defines a transaction roughly equivalent to an HTTP Get. Any MIME type can be retrieved, including images as well as HTML pages.

When a <GETMIMERQ> request appears in a request file and no error occurs in processing, the server must return a response file containing multiple entities (defined in the MIME protocol to include the MIME headers and content for one part of the transmission). Such a response file has content type "multipart/x-mixed-replace", as discussed in section 2.1. One entity contains the OFX response. Other entities contain the content of individual retrievals corresponding to each <GETMIMERS> in the OFX entity.

When multiple <GETMIMERS> responses (corresponding to successful <GETMIMERQ> requests) appear in an OFX response entity, the server must return individual entities in the same order as the corresponding response aggregates. Since the OFX response itself should be the only entity with content type "application/x-ofx" in the response file, the client may find the retrieved information in predictable locations within the multipart response.

9.3.1 MIME Get Request and Response <GETMIMERQ> <GETMIMERS>

The following table lists the components of a request:

	Description
<GETMIMERQ>	Get-MIME-request aggregate
<URL>	URL, *URL*
</GETMIMERQ>	

The response simply echoes the URL. The actual response, whether HTML, an image, or some other type, is always sent as a separate part of the file using multipart MIME.

	Description
<GETMIMERS>	Get-MIME-response aggregate
<URL>	URL, *URL*
</GETMIMERS>	

9.3.1.1 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2019	Duplicate request (ERROR)
16502	Invalid URL (ERROR)
16503	Unable to get URL (ERROR)

9.3.2 MIME Example

A request:

```
<GETMIMETRNRQ>
  <TRNUID>54321</TRNUID>
  <GETMIMERQ>
    <URL>http://www.fi.com/apage.html</URL>
  </GETMIMERQ>
</GETMIMETRNRQ>
```

A response – the full file is shown here to illustrate the use of multipart MIME:

```
HTTP 1.0 200 OK
Content-Type: multipart/x-mixed-replace; boundary =boundary =XYZZY24x7

--XYZZY24x7
Content-Type: application/x-ofx
Content-Length: 8732

<?xml version="1.0"?>

<?OFX OFXHEADER="200" VERSION="211" SECURITY="NONE" OLDFILEUID="NONE"
NEWFILEUID="NONE"?>

<OFX>
    <!-- signon not shown
    message set wrappers not shown -->
<GETMIMETRNRS>
  <TRNUID>54321</TRNUID>
  <STATUS>
    <CODE>0</CODE>
```

```
    <SEVERITY>INFO</SEVERITY>
  </STATUS>
  <GETMIMERS>
    <URL>http://www.fi.com/apage.html</URL>
  </GETMIMERS>
</GETMIMETRNRS>
</OFX>

--XYZZY24x7
Content-Type: text/html
<HTML>
  <!-- standard HTML page -->
</HTML>

--XYZZY24x7--
```

9.4 Message Sets and Profile

9.4.1 Message Set and Messages

The following tables show the OFX Email transactions and their corresponding transaction wrappers.

9.4.1.1 Email Message Set and Messages

9.4.1.1.1 Email Message Set Request Messages

Message	
<EMAILMSGSRQV1>	
	MAILTRNRQ
	MAILRQ
	GETMIMETRNRQ
	GETMIMERQ
	MAILSYNCRQ
</EMAILMSGSRQV1>	

9.4.1.1.2 Email Message Set Response Messages

Message	
<EMAILMSGSRSV1>	
	MAILTRNRS
	MAILRS
	GETMIMETRNRS
	GETMIMERS
	MAILSYNCRS
</EMAILMSGSRSV1>	

9.4.2 E-Mail Message Set Profile

If either or both of the messages in the e-mail message set are supported, the following aggregate must be included in the profile <MSGSETLIST> response. If <EMAILMSGSET> is supported by the server, you

must also support <MAILSYNCRQ>.

	Description
<EMAILMSGSET>	E-mail-message-set-profile-information aggregate
<EMAILMSGSETV1>	Opening tag for V1 of the message set profile information
<MSGSETCORE>	Common message set information, defined in Chapter 7, "FI Profile"
</MSGSETCORE>	
<MAILSUP>	Y if server supports <MAILRQ> request. N if server supports only the <MAILSYNCRQ> request. *Boolean*
<GETMIMESUP>	Y if server supports get MIME message, *Boolean*
</EMAILMSGSETV1>	
</EMAILMSGSET>	

CHAPTER 10 RECURRING TRANSACTIONS

OFX enables users to automate transactions that occur on a regular basis. Recurring transactions are useful when a customer has payments or transfers, for example, that repeat at regular intervals. The customer can create a “model” at the server for automatic generation of these instructions. The model in turn creates payments or transfers until it is canceled or expires. After the user creates a recurring model at the server, the server can relieve the user from the burden of creating these transactions; it generates the transactions on its own, based on the operating parameters of the model.

10.1 Creating a Recurring Model

10.2 Recurring Instructions <RECURRINST>

The Recurring Instructions aggregate is used to specify the schedule for a repeating instruction. It is passed to the server when a recurring transfer or payment model is first created.

	Description
<RECURRINST>	Recurring-Instructions aggregate
<NINSTS>	Number of instructions If this element is absent, the schedule is open-ended, *N-3*
<FREQ>	Frequency, see section 10.2.1
</RECURRINST>	

10.2.1 Values for <FREQ>

	Description
WEEKLY	Weekly
BIWEEKLY	Biweekly
TWICEMONTHLY	Twice a month
MONTHLY	Monthly
FOURWEEKS	Every four weeks
BIMONTHLY	Bimonthly
QUARTERLY	Quarterly
SEMIANNUALLY	Semiannually
ANNUALLY	Annually

Rules for calculating recurring dates of WEEKLY, BIWEEKLY, and TWICEMONTHLY are as follows:

- WEEKLY = starting date for first transaction, starting date + 7 days for the second
- TWICEMONTHLY = starting date for first, starting date + 15 days for the second
- BIWEEKLY = starting date for first, starting date + 14 days for the second

Note that servers are free to adjust the payment date of a spawned payment to accommodate weekends and holidays. For instance, if August 1 falls on a Saturday, a MONTHLY model generating payments on the 1st of each month might change the due date of its August 1 payment to July 31.

Examples:

Start date of May 2: next transaction date for WEEKLY is May 9; TWICEMONTHLY is May 17; next transfer date for BIWEEKLY is May 16.

Start date of May 20: next date for WEEKLY is May 27; TWICEMONTHLY is June 4; next date for BIWEEKLY is June 3.

TWICEMONTHLY recurring transactions will occur each month on those days adjusting for weekends and holidays. BIWEEKLY will occur every 14 days.

10.2.2 Examples

The following example illustrates the creation of a repeating payment. The payment repeats on a monthly basis for 12 months. All payments are for $395.

The request:

```
.
.
.
<RECPMTRQ>
  <RECURRINST>
    <NINSTS>12</NINSTS>
    <FREQ>MONTHLY</FREQ>
  </RECURRINST>
  <PMTINFO>
    <BANKACCTFROM>
      <BANKID>555432180</BANKID>
      <ACCTID>763984</ACCTID>
      <ACCTTYPE>CHECKING</ACCTTYPE>
    </BANKACCTFROM>
    <TRNAMT>395.00</TRNAMT>
    <PAYEEID>77810</PAYEEID>
    <PAYACCT>444-78-97572</PAYACCT>
    <DTDUE>20051115</DTDUE>
    <MEMO>Auto loan payment</MEMO>
  </PMTINFO>
</RECPMTRQ>
.
.
.
```

The response includes the <RECSRVRTID> that the client can use to cancel or modify the model:

```
.
.
.
<RECPMTRS>
  <RECSRVRTID>387687138</RECSRVRTID>
  <RECURRINST>
    <NINSTS>12</NINSTS>
    <FREQ>MONTHLY</FREQ>

  </RECURRINST>
  <PMTINFO>
    <BANKACCTFROM>
      <BANKID>555432180</BANKID>
      <ACCTID>763984</ACCTID>
      <ACCTTYPE>CHECKING</ACCTTYPE>
    </BANKACCTFROM>
    <TRNAMT>395.00</TRNAMT>
    <PAYEEID>77810</PAYEEID>
    <PAYACCT>444-78-97572</PAYACCT>
    <DTDUE>20051115</DTDUE>
    <MEMO>Auto loan payment</MEMO>
  </PMTINFO>
</RECPMTRS>
.
.
.
```

10.3 Retrieving Transactions Generated by a Recurring Model

Once created, a recurring model independently generates instructions. At the time the instance is generated, its status is pending. At this point, the pending/spawned transaction is treated as a single transaction, and the rules for what happens to this transaction are the same as if it had been generated from an explicit request. Since the client has not directly generated these transactions, the client has no record of their creation. To enable users to modify and/or cancel these transactions, the client must use data synchronization in order to retrieve these transactions. (Some message sets also support an inquiry request, which may be used once the SRVRTID of the transaction is obtained via synchronization.)

The client has two purposes for synchronizing state with the server with respect to recurring models:

- Retrieve any added, modified, or canceled recurring models
- Retrieve any added, modified, or canceled transactions generated by any models

The client must be able to synchronize with the state of any models at the server, as well as the state of any transactions generated by the server.

10.3.1 Models and Sync Behavior

When a model spawns a payment, the corresponding change may be reflected in the sync in the form of a <RECPMTMODRS> with <TRNUID>0. The change would be reflected in a decremented <NINSTS> as well as a changed <DTDUE> which would contain the date of the next payment to be spawned. If the server chooses to send a last <RECPMTMODRS> containing <NINSTS>0, the client should use this to know that the model is finished.

A recurring payments refresh should cause the current, not original, state of the model to be sent. The remaining number of instances (<NINSTS>) and the date due (<DTDUE>) of the next payment to be spawned should be returned. This assures that another client who may not know about payment history relevant to the state of the model (such as payment cancellations) can know the current state of the model.

10.4 Modifying and Canceling Individual Transactions

Once created and retrieved by the customer, recurring payments and transfers are almost identical to customer-created payments or transfers. As with ordinary payments or transfers, you can cancel or modify transactions individually. However, because servers generate these transfers, they are different in the following respects:

- Recurring transactions must be retrieved as part of a synchronization request.
- Recurring transactions are related to a model. A server can modify or cancel transactions if the model is modified or canceled.

10.5 Modifying and Canceling Recurring Models

A recurring model can be modified or canceled. When a model is modified, all transactions that it generates in the future will change as well. The client can indicate whether transactions that have been generated, but have not been sent, should be modified as well. The actual elements within a transaction that can be modified differ by service. See the recurring sections within Chapter 11, "Banking," and Chapter 12, "Payments," for details. When a model is cancelled, the server cancels any transactions that it has not yet sent.

If a client indicates that the modification or cancellation of a model should also affect its pending transactions, those individual modifications/cancellations must appear in the appropriate synchronization response the next time a synchronization request is made. For example, a recurring payment cancellation request that affects pending payments should cause payment cancellation responses to show up in the payment synchronization response for all pending payments belonging to the model.

10.5.1 Examples

Canceling a recurring payment model requires the client to pass the <RECSRVRTID> of the model. The client requests that pending payments also be canceled. The server cancels the model immediately and notifies the client that the model was canceled.

The request:

```
.
.
.
  <RECPMTCANCRQ>
    <RECSRVRTID>387687138</RECSRVRTID>
    <CANPENDING>Y</CANPENDING>
  </RECPMTCANCRQ>
.
.
.
```

The response:

```
.
.
.
  <RECPMTCANCRS>
    <RECSRVRTID>387687138</RECSRVRTID>
    <CANPENDING>Y</CANPENDING>
  </RECPMTCANCRS>
.
.
```

.

The server also cancels any payments that have been generated but not executed. In the example shown above, the client would not learn of this immediately. To receive notification that all pending payments were canceled, the client would need to send a synchronization request in the file. The following example illustrates this.

The next request file contains a synchronization request:

```
.
.
.
  <PMTSYNCRQ>
    <TOKEN>12345</TOKEN>
    <REJECTIFMISSING>N</REJECTIFMISSING>
    <BANKACCTFROM>
      <BANKID>123432123</BANKID>
      <ACCTID>516273</ACCTID>
      <ACCTTYPE>CHECKING</ACCTTYPE>
    </BANKACCTFROM>
  </PMTSYNCRQ>
.
.
.
```

The response file contains one response (assuming one payment was pending).

```
.
.
.
  <PMTSYNCRS>
    <TOKEN>123456</TOKEN>
    <BANKACCTFROM>
      <BANKID>123432123</BANKID>
      <ACCTID>516273</ACCTID>
      <ACCTTYPE>CHECKING</ACCTTYPE>
    </BANKACCTFROM>
    <PMTTRNRS>
      <TRNUID>0</TRNUID>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <PMTCANCRS>
        <SRVRTID>1030155</SRVRTID>
      </PMTCANCRS>
    </PMTTRNRS>
  </PMTSYNCRS>
.
.
```

Note that because requests are not guaranteed to be executed in order, a PMTSYNCRQ in the same file as the RECPMTCANCRQ would not guarantee that the cancelled payments would be returned, since the PMYSYNCRQ might be executed first. This is the reason two OFX files are required in the example above.

10.6 Expired Models

A model should (preferably) expire after the last pending transfer/payment has been executed for that model, rather than when the last transfer/payment has been spawned. This enables the user to change and/or cancel the model (possibly, with the <MODPENDING>Y or <CANPENDING>Y flags) during this period.

Models should show up in synchronization responses even after they have expired (at least for a time), since the RECSRVRTID will be in payment synchronization responses and a client needs to find the corresponding model. Servers may safely remove this information shortly after the final payment or transfer has posted to the source account.

CHAPTER 11 BANKING

OFX enables financial insititution (FI) customers to keep their finances up-to-date and to manage their bank accounts conveniently in several ways. Customers can download transactions and update account balances on a daily basis. They can retrieve a closing statement that contains the same information that they are accustomed to seeing on a paper statement. They can transfer funds between accounts at a financial institution, either immediately upon going online or on a regular schedule. Customers can schedule transfers between accounts on a recurring basis and can transfer funds between accounts at different financial institutions. If necessary, customers can request a wire funds transfer. OFX also enables requests to stop payment on pending checks.

Using customer notification, an FI can notify customers of important events regarding their accounts, such as returned checks or deposits.

11.1 Consumer and Business Banking

OFX supports banking for both consumers and businesses. Some customers might use some areas more heavily within OFX Banking (such as credit card download); other areas might be more appropriate for businesses (such as wire transfers). Yet all of the functionality defined for Banking is appropriate to some extent for both consumer and business applications.

11.1.1 Loan Data

Loan data is available to OFX clients through the statement download facility. Statement download provides a way to download loan transaction data and balances on an as-needed basis. Statement closing information can be made available to clients as well. Loan amortization schedules are also available through the amortization download facility.

11.2 Credit Card Data

Credit card data is available to OFX clients through the statement download facility. Statement download provides a way to download credit card transaction data and balances on an as-needed basis. Statement closing information can be made available to clients as well.

11.3 Common Banking Aggregates

This section describes several aggregates used throughout the Banking portion of OFX.

11.3.1 Banking Account <BANKACCTFROM> and <BANKACCTTO>

OFX uses the Banking Account aggregates to identify an account at an FI. The aggregates contain enough information to uniquely identify an account for the purposes of statement download, bill payment, and funds transfer. <CCACCTFROM> identifies credit card accounts; see section 11.3.2. <LOANACCTFROM> identifies loan accounts; see section 11.3.7.

Tag	Description
<BANKACCTFROM>	Bank-account-from aggregate
<BANKID>	Bank identifier, *A-9* Use of this field by country: COUNTRY — Interpretation BEL — Bank code CAN — Routing and transit number CHE — Clearing number DEU — *Bankleitzahl* ESP — *Entidad* FRA — *Banque* GBR — Sort code ITA — *ABI* NLD — Not used (field contents ignored) USA — Routing and transit number
<BRANCHID>	Branch identifier. May be required for some non-US banks, *A-22* Use of this field by country: COUNTRY — Interpretation BEL — Not present CAN — Not present CHE — Not present DEU — Not present ESP — *Oficina* FRA — *Agence* GBR — Not present ITA — *CAB* NLD — Not present USA — Not present
<ACCTID>	Account number, *A-22*

Tag	Description
<ACCTTYPE>	Type of account, see section 11.3.1.1
<ACCTKEY>	Checksum, *A-22* Use of this field by country:
</BANKACCTFROM>	

COUNTRY	Interpretation
BEL	Check digits
CAN	Not present
CHE	Not present
DEU	Not present
ESP	*D.C.*
FRA	*Clé*
GBR	Not present
ITA	*CIN*
NLD	Not present
USA	Not present

Tag	Description
<BANKACCTTO>	Bank-account-to aggregate
<BANKID>	Bank identifier, *A-9* Use of this field by country: COUNTRY — Interpretation BEL — Bank code CAN — Routing and transit number CHE — Clearing number DEU — *Bankleitzahl* ESP — *Entidad* FRA — *Banque* GBR — Sort code ITA — *ABI* NLD — Not used (field contents ignored) USA — Routing and transit number
<BRANCHID>	Branch identifier. May be required for some banks, *A-22* Use of this field by country: COUNTRY — Interpretation BEL — Not present CAN — Not present CHE — Not present DEU — Not present ESP — *Oficina* FRA — *Agence* GBR — Not present ITA — *CAB* NLD — Not present USA — Not present
<ACCTID>	Account number, *A-22*
<ACCTTYPE>	Type of account, see section 11.3.1.1
<ACCTKEY>	Checksum, *A-22* Use of this field by country:

Tag	Description	
	COUNTRY	Interpretation
	BEL	Check digits
	CAN	Not present
	CHE	Not present
	DEU	Not present
	ESP	*D.C.*
	FRA	*Clé*
	GBR	Not present
	ITA	*CIN*
	NLD	Not present
	USA	Not present
</BANKACCTTO>		

11.3.1.1 Account Types for <ACCTTYPE> Elements

	Description
CHECKING	Checking
SAVINGS	Savings
MONEYMRKT	Money Market
CREDITLINE	Line of credit

11.3.4 Credit Card Account Information <CCACCTINFO>

OFX uses the credit card account information aggregate to download account information from an FI. It includes credit card number specification in <CCACCTFROM> as well as the status of the service.

	Description
<CCACCTINFO>	Credit-card-account-information aggregate
<CCACCTFROM>	Credit-card-account-from aggregate

Tag	Description
<XFERINFO> *Account-from options. Choose either <BANKACCTFROM> or <CCACCTFROM> or <LOANACCTFROM>*	Transfer-information aggregate
<BANKACCTFROM> ***</BANKACCTFROM>*** -or- ***<CCACCTFROM>*** ***</CCACCTFROM>*** -or- ***<LOANACCTFROM>*** ***</LOANACCTFROM>***	Bank account-from aggregate, see section 11.3.1 Credit-card-account-from aggregate, see section 11.3.2 Loan-account-from aggregate, see section 11.3.7
Account-to options. Choose either <BANKACCTTO> or <CCACCTTO> or <LOANACCTTO>	
<BANKACCTTO> ***</BANKACCTTO>*** -or- ***<CCACCTTO>*** **</CCACCTTO>** -or- ***<LOANACCTTO>*** ***</LOANACCTTO>***	Bank account-to aggregate, see section 11.3.1 Credit-card account-to aggregate, see section 11.3.2 Loan account-to aggregate, see section 11.3.7
<TRNAMT> <LOANTRNAMT> </LOANTRNAMT> <DTDUE>	Amount of the transfer, *amount* This amount should be specified as a positive number. Loan transaction amount aggregate, see section 11.3.9. Total of all these items must equal <TRNAMT> value. Date that the transfer is to be sent. If the client does not specify <DTDUE>, the transfer occurs as soon as possible. <DTDUE> is required for scheduled or repeating transfers, *datetime*

Tag	Description
</XFERINFO>	

If the server has the breakdown available at the time of the transfer, <XFERINFO> should include the <LOANTRNAMT> aggregate in <INTRATRNRS>. Otherwise, the breakdown of the loan payment may be returned in <LOANSTMTTRN>.

11.3.6 Transfer Processing Status <XFERPRCSTS>

The Transfer Processing Status aggregate contains the current processing status for a transfer. This aggregate is intended to describe status changes to the associated transfer after creation. The interpretation of the date value depends on the value of <XFERPRCCODE>.

	Description
<XFERPRCSTS>	Transfer processing status aggregate
<XFERPRCCODE>	See section 11.3.6.1
<DTXFERPRC>	Transfer processing date; value depends on <XFERPRCCODE>
</XFERPRCSTS>	

11.3.6.1 Transfer Processing Status Values <XFERPRCCODE>

	Description
WILLPROCESSON	Will be processed on <DTXFERPRC>
POSTEDON	Posted on <DTXFERPRC>
NOFUNDSON	Funds not available to make transfer on <DTXFERPRC>
CANCELEDON	User canceled payment on <DTXFERPRC>
FAILEDON	Unable to make transfer for unspecified reasons on <DTXFERPRC>

11.3.7 Loan Account <LOANACCTFROM> and <LOANACCTTO>

Open Financial Exchange uses the Loan Account aggregate to identify a loan account at a financial institution. The aggregate contains enough information to uniquely identify an account for the purpose of statement download and funds transfer.

	Description
<LOANACCTFROM>	Transfer processing status aggregate
<LOANACCTID>	Account number, *A-32*
<LOANACCTTYPE>	Loan account type, see section 11.3.7.1
</LOANACCTFROM>	

The <LOANACCTTO> aggregate contains the same elements as <LOANACCTFROM>.

11.3.7.1 Loan Account Type <LOANACCTTYPE>

	Description
AUTO	Automobile Loan
CONSUMER	Consumer Loan
MORTGAGE	Mortgage Loan
COMMERCIAL	Commercial Loan
STUDENT	Student Loan
MILITARY	Military Loan
SMB	Small Business Loan
CONSTR	Construction Loan
HOMEEQUITY	Home Equity Loan

11.3.8 Loan Account Information <LOANACCTINFO>

Tag	Description
<LOANACCTINFO>	
<LOANACCTFROM>	Loan-account-from aggregate, see section 11.3.7
</LOANACCTFROM>	
<LOANTYPE**>**	Type of term, use one of: *FIXED = Payments made at fixed intervals, also known as "installment"* *REVOLVE = Revolving term* *OPEN = Open-ended loan such as a line of credit, home equity line of credit, etc.* *COMBO = Combination of above*
<LOANINITNUMPMTS**>**	Initial number of loan payments. *N-5*
<LOANINITBAL>	Initial loan balance, *amount*
<LOANFREQ>	Frequency of payments, see section 11.3.8.1
<DTLOANSTART>	Start date of loan, *date*
<DTLOANMATURITY**>**	Expected loan end date. *date* Must be returned if <LOANFREQ>MATURITY
<PRINBAL>	Loan Princiapl Balance aggregate, see section 11.3.8.2
</PRINBAL>	
<BALLOONAMT**>**	Not included or zero for regular loans, otherwise Balloon amount, *amount*
<LOANINT>	Loan Interest aggregate, see section 11.3.8.3
</LOANINT>	
<LOANIRATE>	Loan Rate aggregate, see section 11.3.8.4
</LOANIRATE>	
<LOANPMT>	Loan Payment aggregate, see section 11.3.8.5
</LOANPMT>	
<LOANRMNPMTS>	Remaining number of loan payments. *N-5*
<SUPTXDL>	Y if account supports transaction detail downloads, N if it is balance-only, *Boolean*
<XFERSRC>	Y if account is enabled as a source for an intrabank or interbank transfer, *Boolean*
<XFERDEST>	Y if account is enabled as a destination for an intrabank or interbank transfer, *Boolean*

Tag	Description
<SVCSTATUS>	Status of the account, choose one of the following: AVAIL = Available, but not yet requested PEND = Requested, but not yet available ACTIVE = In use
</LOANACCTINFO>	

11.3.8.1 Values for <LOANFREQ>

	Description
WEEKLY	Weekly
BIWEEKLY	Biweekly
TWICEMONTHLY	Twice a month
MONTHLY	Monthly
FOURWEEKS	Every four weeks
BIMONTHLY	Bimonthly
QUARTERLY	Quarterly
SEMIANNUALLY	Semiannually
ANNUALLY	Annually
MATURITY	Payment is only due upon maturity of loan

When using frequency of MATURITY, server must also return <DTLOANMATURITY>.

Refer to section 10.2.1for rules for calculating recurring dates of WEEKLY, BIWEEKLY, and TWICEMONTHLY.

11.3.8.2 Principal Balance <PRINBAL>

	Description
<PRINBAL>	Loan Principal Balance aggregate
<BALAMT>	Current loan principal balance, *amount*
<PRINYTD>	Total principal paid year to date, *amount*
<PRINLTD>	Total principal paid loan to date, *amount*
<DTASOF>	Date and time of the current loan balance. *datetime*
</PRINBAL>	

11.3.8.3 Loan Interest <LOANINT

	Description
<LOANINT>	
Loan interest paid. Return at least one of the following:	
<LOANINTYTD>	Total interest paid year to date, *amount*
-or-	
<LOANINTLTD>	Total interest paid loan to date, *amount*
-or-	
<LOANINTPRJ>	Total projected interest to be paid on this loan, *amount*
<DTASOF>	Date and time of the current interest information, *datetime*

Tag	Description
<DTASOF>	Date and time of the current interest rate, *datetime*
</LOANIRATE>	

11.3.8.5 Loan Payment Amount <LOANPMT>

	Description
<LOANPMT>	
<PMTAMT>	Next loan payment amount, *amount*
<DTPMTDUE>	Next payment due date, *datetime*
<ESCRWAMT>	Escrow aggregate for mortgages, see section 11.3.9.1
</ESCRWAMT>	
<LOANPMTTYPE>	Type of payment, choose one of the following INTONLY = Interest only payments required PRNANDINT = Principal and interest included in set payment PRNPLUSINT = Principal plus accrued interest owing
</LOANPMT>	

11.3.9 Loan Transaction Amount <LOANTRNAMT>

	Description
<LOANTRNAMT>	
<PRINAMT>	Amount of payment applied to Principal. *amount*
<INTAMT>	Amount of payment applied to interest. *amount*
<ESCRWAMT>	Escrow aggregate for mortgages, see section 11.3.9.1
</ESCRWAMT>	
<INSURANCE>	Life, accident, health insurance on loan, *amount*
</LOANTRNAMT>	

11.3.9.1 Escrow Amount <ESCRWAMT>

	Description
<ESCRWAMT>	
<ESCRWTOTAL>	Total amount of payment applied to Escrow, *amount*
<ESCRWTAX>	Amount of tax set aside in Escrow, *amount*
<ESCRWINS>	Amount of insurance set aside in Escrow, *amount*
<ESCRWPMI>	Amount of PMI set aside in Escrow, *amount*
<ESCRWFEES>	Amount of fees set aside in Escrow, *amount*
<ESCRWOTHER>	Amount of other amount set aside in Escrow, *amount*
</ESCRWAMT>	

The amount of ESCRWTAX + ESCRWINS + ESCRWPMI + ESCRWFEES + ESCRWOTHER should normally equal ESCRWTOTAL. However, if values do not compute, client should roll remainder into <ESCRWOTHER>.

11.4 Downloading Transactions and Balances

Statement download allows a customer to receive transactions and balances that are typically part of a regular paper statement. Clients can retrieve transactions and balances on a daily basis if they wish. Coupled with the information returned by statement closing information request (see section 11.5), a client can construct an “electronic statement” that contains all of the information that appears on a regular paper statement.

Clients typically allow customers to view these transactions and guide customers through a process of updating their account registers based on the downloaded transactions. By using transaction IDs supplied by financial institutions, OFX makes it possible for clients to ensure that a server downloads each transaction only once. The request also contains starting and ending dates to limit the amount of downloaded data. Clients can remember the last date they received data and use it as the starting date in the next request.

The messages in this chapter are appropriate for checking, savings, money market, credit card, and line of credit accounts. Investment statement download is a superset of bank statement download. Chapter 13, "Investments," describes the messages specific to investment statement download.

Statement download requires the client to designate an account for the download, and to indicate if the server should download transactions and/or balances. If the client wishes to download transactions, it can specify a date range that the transactions fall within.

The server returns transactions that match the date range (if the client specifies one), and balance information for the account.

	Server Responds
Account information	
Include transactions?	
Date range	
	Transactions
	Cycle-ending information

11.4.1 Bank Statement Download

A client can request a download of balances separately from transaction detail. The server downloads transactions only if the <INCTRAN> aggregate is present and the <INCLUDE> flag is set to Y. The current ledger balance (and balance date) are always downloaded.

If a statement download request does not contain <DTSTART> or <DTEND> elements but does request transactions and if no transactions are found on the server, the response may or may not include a <BANKTRANLIST>. If the <BANKTRANLIST> is included, the server should leave out the empty <STMTTRN> aggregate(s).

You can use the <STMTRQ> … <STMTRS> request and response pair to download transactions and balances for checking, savings, money market, and line of credit accounts. Section 11.4.2 describes download for credit card accounts.

Clients and servers should interpret <DTSTART> and <DTEND> as described in Chapter 3, "Common Aggregates, Elements, and Data Types."

Image transaction information can be requested by including <INCTRANIMG>Y in a <STMTRQ>, requesting the server to return image data corresponding to transaction images, if available. The image identification data returned in any <STMTTRN> aggregate is not the actual image, but will be used to access the image through subsequent client sessions. See Chapter 15, “Image Download”, for more information about retrieving images from the server.

11.4.1.1 Request <STMTRQ>

The <STMTRQ> request must appear within a <STMTTRNRQ> transaction wrapper.

	Description
<STMTRQ>	Statement-request aggregate
<BANKACCTFROM>	Bank-account-from aggregate, see section 11.3.1
</BANKACCTFROM>	
<INCTRAN>	Include-transactions aggregate
<DTSTART>	Start date of statement requested, *datetime*
<DTEND>	End date of statement requested, *datetime*
<INCLUDE>	Include transactions flag, *Boolean*
</INCTRAN>	
<INCTRANIMG>	Include transaction image information, *Boolean*
</STMTRQ>	

11.4.1.2 Response <STMTRS>

A statement response comprises elements supplying various balances, plus zero or more <STMTTRN> aggregates, each describing one statement transaction.

The response can contain a <BALLIST> aggregate that allows an FI to send any number of <BAL> aggregates (see Chapter 3, “Common Aggregates, Elements, and Data Types”) to the user, complete with description and Help text.

The <STMTRS> response must appear within a <STMTTRNRS> transaction wrapper.

See Chapter 3, "Common Aggregates, Elements, and Data Types," for size and type information for common elements (such as currency values).

Tag	*Description*
<STMTRS>	Statement-response aggregate
<CURDEF>	Default currency for the statement, *currsymbol*
<BANKACCTFROM>	Account-from aggregate, see section 11.3.1
</BANKACCTFROM>	
<BANKTRANLIST>	Statement-transaction-data aggregate
<DTSTART>	Start date for transaction data, *date*
<DTEND>	Value that client should send in next <DTSTART> request to ensure that it does not miss any transactions, *date*
<STMTTRN>	Opening tag for each statement transaction (0 or more), see section 11.4.3
</STMTTRN>	End tag for each statement transaction
</BANKTRANLIST>	
<LEDGERBAL>	Ledger balance aggregate
<BALAMT>	Ledger balance amount, *amount*
<DTASOF>	Balance date, *datetime*
</LEDGERBAL>	
<AVAILBAL>	Available balance aggregate
<BALAMT>	Available balance amount, *amount*
<DTASOF>	Balance date, *datetime*
</AVAILBAL>	
<BALLIST>	List of miscellaneous other balances

Tag	Description
<BAL>	Balance aggregates (0 or more), see section 3.1.4 See Appendix B, "Suggested Name Values for the Banking <BALLIST>," for a list of suggested common values for use with the <NAME> and <DESC> tags in these <BAL> aggregates. Note: Should the balance type not map to any of those shown in B.2, any value for the <NAME> and <DESC> fields may be used.
</BAL>	
</BALLIST>	
<MKTGINFO>	Marketing information (at most 1), *A-360*
</STMTRS>	

11.4.1.3 Status Codes

	Meaning
0	Success
2000	General error (ERROR)
2002	General account error (ERROR)
2003	Account not found (ERROR)
2004	Account closed (ERROR)
2005	Account not authorized (ERROR)
2019	Duplicate request (ERROR)
2020	Invalid date (ERROR)
2027	Invalid date range (ERROR)

11.4.2 Credit Card Statement Download

The credit card download request is semantically identical to the bank statement download request. However, the <CCSTMTRQ> aggregate contains the credit card request, not the <STMTRQ> aggregate.

If a statement download request does not contain <DTSTART> or <DTEND> elements but does request transactions and if no transactions are found on the server, the response may or may not include a <BANKTRANLIST>. If the <BANKTRANLIST> is included, the server should leave out the empty <STMTTRN> aggregate(s).

Tag	Description
<BAL>	Balance aggregates (0 or more), see section 3.1.4 See Appendix B, "Suggested Name Values for the Banking <BALLIST>," for a list of suggested common values for use with the <NAME> and <DESC> tags in these <BAL> aggregates. Note: Should the balance type not map to any of those shown in B.2, any value for the <NAME> and <DESC> fields may be used.
</BAL>	
</BALLIST>	
<MKTGINFO>	Marketing information (at most 1), *A-360*
</CCSTMTRS>	

11.4.2.3 Status Codes

	Meaning
0	Success
2001	Invalid account (ERROR)
2002	General account error (ERROR)
2003	Account not found (ERROR)
2004	Account closed (ERROR)
2005	Account not authorized (ERROR)
2019	Duplicate request (ERROR)
2020	Invalid date (ERROR)
2027	Invalid date range (ERROR)

11.4.3 Statement Transaction <STMTTRN>

A <STMTTRN> aggregate describes a single transaction. It identifies the type of the transaction and the date it was posted. The aggregate can also provide additional information to help the customer recognize the transaction: check number, payee name, and memo. The transaction can have a Standard Industrial Code that a client can use to categorize the transaction.

Each <STMTTRN> contains an <FITID> that the client uses to detect whether the server has previously downloaded the transaction.

Transaction amounts are signed from the perspective of the customer. For example, a credit card payment is positive while a credit card purchase is negative.

Tag	Description
<STMTTRN>	Statement-transaction aggregate
<TRNTYPE>	Transaction type, see section 11.4.3.1 for possible values. This element does not change the effect of the transaction upon the balance (increases and decreases are indicated by the sign of the <TRNAMT>).
<DTPOSTED>	Date transaction was posted to account, *datetime*
<DTUSER>	Date user initiated transaction, if known, *datetime*
<DTAVAIL>	Date funds are available (value date), *datetime*
<TRNAMT>	Amount of transaction, *amount*
<FITID>	Transaction ID issued by financial institution. Used to detect duplicate downloads, *FITID*
<CORRECTFITID>	If present, the FITID of a previously sent transaction that is corrected by this record. This transaction replaces or deletes the transaction that it corrects, based on the value of <CORRECTACTION> below, *FITID*
<CORRECTACTION>	Actions can be REPLACE or DELETE. REPLACE replaces the transaction referenced by CORRECTFITID; DELETE deletes it.
<SRVRTID>	Server assigned transaction ID; used for transactions initiated by client, such as payment or funds transfer. If a transaction associated with a transfer contains a SRVRTID, the associated account is either the transfer's "to" or "from" account. Transactions for both accounts in a transfer would contain the same SRVRTID. *SRVRTID*
<CHECKNUM>	Check (or other reference) number, *A-12*
<REFNUM>	Reference number that uniquely identifies the transaction. Can be used in addition to or instead of a <CHECKNUM>, *A-32*
<SIC>	Standard Industrial Code, *N-6*
<PAYEEID>	Payee identifier if available, *A-12*
Payee options. Choose either <NAME> or <PAYEE>.	
<NAME>	Name of payee or description of transaction, *A-32* **Note:** Provide NAME or PAYEE, not both
-or-	
<PAYEE>	Payee aggregate, see section 12.5.2.1
</PAYEE>	
<EXTDNAME>	Extended name of payee or description of transaction, *A-100*

Tag	*Description*
Account-to options. Choose either <BANKACCTTO> or <CCACCTTO>.	
<BANKACCTTO> *</BANKACCTTO>* -or- *<CCACCTTO>* *</CCACCTTO>*	If this was a transfer to an account and the account information is available, see section 11.3.1
<MEMO>	Extra information (not in <NAME>), *MEMO*
<IMAGEDATA> </IMAGEDATA>	Image data aggregate, up to 2 allowed. See section 3.1.6.1
Currency options. Choose either <CURRENCY> or <ORIGCURRENCY>.	
<CURRENCY> *</CURRENCY>* -or- *<ORIGCURRENCY>* *</ORIGCURRENCY>*	Currency, if different from CURDEF, see section 5.2.
<INV401KSOURCE>	Source of cash for this transaction. See section 13.9.2.4.2.
</STMTTRN>	

11.4.3.1 Transaction types used in <TRNTYPE>

Transfers generated from a model are treated identically to individually requested transfers by OFX. Therefore, they should have the transaction types listed below once they are processed. Transfers initiated out of band with respect to OFX should also be handled in this fashion when they appear in a statement download.

	Description
CREDIT	Generic credit
DEBIT	Generic debit
INT	Interest earned or paid
	Note: Depends on signage of amount
DIV	Dividend
FEE	FI fee
SRVCHG	Service charge
DEP	Deposit
ATM	ATM debit or credit
	Note: Depends on signage of amount
POS	Point of sale debit or credit
	Note: Depends on signage of amount
XFER	Transfer
CHECK	Check
PAYMENT	Electronic payment
CASH	Cash withdrawal
DIRECTDEP	Direct deposit
DIRECTDEBIT	Merchant initiated debit
REPEATPMT	Repeating payment/standing order
OTHER	Other

11.4.4 Loan Statement Download

The loan download request is semantically identical to the bank and credit card statement download requests. However, the <LOANSTMTTRNRQ> aggregate contains the loan request, not the <STMTRQ> or <CCSTMTRQ> aggregate.

Image transaction information can be requested by including <INCTRANIMG>Y in a <LOANSTMTRQ>, requesting the server to return image data corresponding to transaction images, if available. The image identification data returned in any <LOANSTMTTRN> aggregate is not the actual image, but will be used to access the image through subsequent client sessions. See Chapter 15, "Image Download", for more information about retrieving images from the server.

11.4.4.1 Request <LOANSTMTRQ>

The <LOANSTMTRQ> request must appear within a <LOANSTMTTRNRQ> transaction wrapper.

	Description
<LOANSTMTRQ>	Loan-download-request aggregate
<LOANACCTFROM>	Loan-account-from aggregate, see section 11.3.7
<LOANACCTFROM>	
<INCTRAN>	Include transactions
<DTSTART**>**	Start date of statement requested, *datetime*
<DEND>	Ending date of statement requested, *datetime*
<INCLUDE>	Include transactions flag, *Boolean*
</INCTRAN>	
<INCTRANIMG>	Include data for transaction images, *Boolean*
</**LOANSTMTRQ**>	

11.4.4.2 Response <LOANSTMTRS>

The loan download response is semantically similar to the bank and credit card statement download request with the addition of new tags relating to interest rates, and various balances. However, the <LOANSTMTTRNRS> aggregate contains the loan response, not the <STMTRS> or <CCSTMTRS> aggregate.

The <LOANSTMTRS> response must appear within a <LOANSTMTTRNRS> transaction wrapper.

Tag	*Description*
<LOANSTMTRS>	Loan-download-request aggregate

Tag	Description
<CURDEF>	Default currency for the statement, *currsymbol*
<LOANACCTFROM>	Loan-account-from aggregate, see section 11.3.7
</**LOANACCTFROM**>	
<LOANTRANLIST>	Opening tag for statement transaction data
<DTSTART>	Start date for transaction data, *date*
<**DTEND**>	Value client should send in next <DSTART> request to ensure that it does not miss any transactions, *date*
<LOANSTMTTRN**>**	Opening tag for each statement transaction (0 or more), see section 11.4.5
</LOANSTMTTRN>	End tag for each statement transaction
</LOANTRANLIST>	
<PRINBAL>	Principal-balance aggregate, see section 11.3.8.2
</PRINBAL>	
<AVAILBAL>	Available balance aggregate
<BALAMT>	Available credit balance amount, *amount*
<DTASOF>	Balance date, *datetime*
</AVAILBAL>	
<MKTGINFO>	Marketing information (at most 1), *A-360*
</LOANSTMTRS>	

11.4.5 Loan Statement Transaction <LOANSTTMTTRN>

A <LOANSTMTTRN> aggregate describes a single transaction. It identifies the type of the transaction and the date it was posted. The aggregate can also provide additional information to help the customer recognize the transaction: check number, name, memo and images. Each <LOANSTMTTRN> contains an <FITID> that the client uses to detect whether the server has previously downloaded the transaction.

Transaction amounts are signed from the perspective of the customer. For example, a loan payment is positive while a loan advance is negative.

If the server has available the breakdown of payment into principal, interest and escrow, <LOANSTMTTRN> should include <LOANTRNAMT> When provided, the sum of the <LOANTRNAMT> aggregate should equal <TRNAMT>.

Tag	*Description*
<LOANSTMTTRN>	Loan-statement-transaction aggregate
<LOANTRNTYPE>	Transaction type, see section 11.4.5.1 for possible values
<DTPOSTED>	Date transaction was posted to account, *datetime*
<DTUSER>	Date user initiated transaction, if known, *datetime*
<TRNAMT>	Amount of transaction, *amount*
<LOANTRNAMT>	Loan transaction amount aggregate, see section 11.3.9
</LOANTRNAMT**>**	
<FITID>	Transaction ID issued by financial institution. Used to detect duplicate transactions, *FITID*
<CORRECTFITID>	If present, the FITID of a previously sent transaction that is the transaction that it corrects, based on the value of <CORRECTACTION>, below, *FITID*
<CORRECTACTION>	Corrected actions; use one of REPLACE or DELETE. REPLACE replaces the transaction referenced by CORRECTFITID; DELETE deletes it.
<SRVRTID>	Server assigned transaction ID; used for transactons initiated by client, such as funds transfer, *SRVRTID*
<CHECKNUM>	Check (or other reference) number, *A-12*
<REFNUM>	Reference number that uniquely identifies the transaction. Can be used in addition to or instead of a <CHECKNUM>, *A-32*
<NAME>	Name of payee or description of transaction, *A-32*
<EXTDNAME>	Extended name of payee or description of transaction, *A-100*
Account-to options. Choose either <BANKACCTTO> or <LOANACCTTO>	

11.4.5.1 Transaction types used in <LOANTRNTYPE>

	Description
PAYMENT	Loan Payment - total of principal, interest, escrow
FEE	Loan Fee
LATE	Late Fee
INT	Interest charged, adds to loan balance
ADVANCE	Cash advance which adds to Principal Balance
XFER	Principal transfer to/from another account, loan or DDA
BALLOON	Payment of Balloon amount
OTHER	Other

Tag	Description
</AMRTSTMTRS>	

11.4.6.3 Status Codes

	Meaning
0	Success
2001	Invalid account (ERROR)
2002	General account error (ERROR)
2003	Account not fouond (ERROR)
2004	Accouont closed (ERROR)
2005	Account not authorized (ERROR)
2019	Duplicate request (ERROR)
2020	Invalid date (ERROR)
2027	Invalid date range (ERROR)

11.4.6.4 Amortization Statement Transaction <AMRTSTMTTRN>

	Description
<AMRTSTMTTRN>	Amortization transaction aggregate
<PMTNUMBER>	Loan payment number, *N-3*
<LOANINITBAL>	Beginning balance before payment, *amount*
<PRINBAL>	Principal Balance aggregate - see section 11.3.8.2 Ending balance after payment, includes date of payment
</PRINBAL>	
<LOANTRNAMT>	Loan transaction amount aggregate, see section 11.3.9
</LOANTRNAMT>	
<LOANIRATE>	Loan interest rate aggregate, see section 11.3.8.4
</LOANIRATE>	
<AMRTTYPE>	Amortization type, choose one of: ORIGINAL = Original amortization ADJUSTED = Adjusted amortization PROJECTED = Projected amortization
</AMRTSTMTTRN>	

11.5 Statement Closing Information

OFX provides a way for customers to receive closing statement information that typically appears as part of a paper statement. This information includes opening and closing dates and balances for a statement period, as well as a detailed breakdown of debits, credits, fees, and interest that are usually part of a paper statement. In addition to this information, clients receive a date range for transactions that correspond to the closing statement. Clients might wish to use this date range to retrieve transactions through statement download in order to present the user with an "electronic" statement.

To request statement information, the client is **REQUIRED** to designate an account for the download. The client can also specify a date range to limit the number of closing information aggregates that the server returns. If the client does not specify a date range, the server returns as many closing information aggregates as it can.

Image download information can be requested in a statement closing request by including <INCSTMTIMG>Y, which requests the server to return image data corresponding to closing statements, if available. The image identification data returned in the <CLOSING>, <CCCLOSING> or <LOANCLOSING> aggregate will not be the actual image, but will be used to access the image through subsequent client sessions. See Chapter 15, "Image Download", for more information on retrieving images from the server.

	Server Responds
Account Information Date range	 Cycle-ending information (0 or more)

11.5.1 Statement Closing Download

You can use the <STMTENDRQ> …<STMTENDRS> request and response pair to download statement closing information for checking, savings, money market, and line of credit accounts. Section 11.5.3 describes download for credit card accounts.

11.5.1.1 Request <STMTENDRQ>

The <STMTENDRQ> request must appear within a <STMTENDTRNRQ> transaction wrapper.

Tag	***Description***
<STMTENDRQ>	Closing-statement-request aggregate
<BANKACCTFROM>	Bank-account-from aggregate
</BANKACCTFROM>	

11.5.1.2 Response <STMTENDRS>

The <STMTENDRS> response must appear within a <STMTENDTRNRS> transaction wrapper.

	Description
<STMTENDRS>	Closing-statement-response aggregate
<CURDEF>	Default currency used for closing information, *currsymbol*
<BANKACCTFROM>	Account from aggregate, see section 11.3.1
</BANKACCTFROM>	
<CLOSING>	Statement information (0 or more), see section

11.5.1.3 Status Codes

	Meaning
0	Success
2000	General error (ERROR)
2002	General account error (ERROR)
2003	Account not found (ERROR)
2004	Account closed (ERROR)
2005	Account not authorized (ERROR)
2019	Duplicate request (ERROR)
2020	Invalid date (ERROR)
2027	Invalid date range (ERROR)

11.5.2 Non-Credit Card Statement <CLOSING>

A checking, savings, or money market account uses the <CLOSING> aggregate to describe statement closing information.

The <FITID> provides a way for the client to distinguish one closing statement from another.

For each <CLOSING> aggregate returned, clients can retrieve corresponding transactions by using <DTPOSTSTART> and <DTPOSTEND> as <DTSTART> and <DTEND> in a <STMTRQ> request.

Tag	*Description*
<CLOSING>	Non-credit-card-account-types aggregate
<FITID>	Unique identifier for this statement, *FITID*
<DTOPEN>	Opening statement date, *date*
<DTCLOSE>	Closing statement date, *date*
<DTNEXT>	Closing date of next statement, *date*
<BALOPEN>	Opening statement balance, *amount*
<BALCLOSE>	Closing statement balance, *amount*
<BALMIN>	Minimum balance in statement period, *amount*
<DEPANDCREDIT>	Total of deposits and credits, including interest, *amount*
<CHKANDDEB>	Total of checks and debits, including fees, *amount*
<TOTALFEES>	Total of all fees, *amount*

<TOTALINT>	Total of all interest, *amount*
<DTPOSTSTART>	Start date of transaction data for this statement, *date* A client should be able to use this date in a <STMTRQ> to request transactions that match this statement.
<DTPOSTEND>	End date of transaction data for this statement, *date* A client should be able to use this date in a <STMTRQ> to request transactions that match this statement.
<MKTGINFO>	Marketing information (at most 1), *A-360*
<IMAGEDATA> </IMAGEDATA>	Image data aggregate, see section 3.1.6.1
Currency options. Choose either <CURRENCY> or <ORIGCURRENCY>	
<CURRENCY> *</CURRENCY>* -or-	Currency, if different from CURDEF, see section 5.2.

11.5.3 Credit Card Statement Closing Request <CCSTMTENDRQ>

The credit card statement closing request is semantically identical to the bank statement closing request. However, the <CCSTMTENDRQ> aggregate contains the credit card request, not the <STMTENDRQ> aggregate.

The <CCSTMTENDRQ> request must appear within a <CCSTMTENDTRNRQ> transaction wrapper.

	Description
<CCSTMTENDRQ>	Credit-card-closing-statement-request aggregate
<CCACCTFROM>	Credit-card-account-from aggregate
</CCACCTFROM>	
<DTSTART>	Start date for statement closing information, *datetime*
<DTEND>	End date of statement closing information, *datetime*
<INCSTMTIMG>	Include data for closing statement images, *Boolean*
</CCSTMTENDRQ>	

11.5.4 Credit Card Statement Closing Response <CCSTMTENDRS>

The credit card statement closing response is semantically identical to the bank statement closing response. However, the <CCSTMTENDRS> aggregate contains the credit card response, not the <STMTENDRS> aggregate.

The <CCSTMTENDRS> response must appear within a <CCSTMTENDTRNRS> transaction wrapper.

	Description
<CCSTMTENDRS>	Credit-card-closing-statement-response aggregate
<CURDEF>	Default currency for closing information, *currsymbol*
<CCACCTFROM>	Account from aggregate, see section 11.3.2
</CCACCTFROM>	
<CCCLOSING>	Statement information (0 or more). See section 11.5.4.2
</CCCLOSING>	
</CCSTMTENDRS>	

11.5.4.1 Status Codes

	Meaning
0	Success
2000	General error (ERROR)
2002	General account error (ERROR)
2003	Account not found (ERROR)
2004	Account closed (ERROR)
2005	Account not authorized (ERROR)
2019	Duplicate request (ERROR)
2020	Invalid date (ERROR)
2027	Invalid date range (ERROR)

11.5.4.2 Credit Card Statement <CCCLOSING>

A credit card account uses the <CCCLOSING> aggregate to describe statement closing information.

The <FITID> provides a way for the client to distinguish one closing statement from another.

For each <CCCLOSING> returned, clients should be able to retrieve corresponding transactions by using <DTPOSTSTART> and <DTPOSTEND> as <DTSTART> and <DTEND> in a <CCSTMTRQ> request.

Tag	Description
<CCCLOSING>	Credit-card-statement-information aggregate
<FITID>	Unique identifier for this statement, *FITID*
<DTOPEN>	Opening statement date, *date*
<DTCLOSE>	Closing statement date, *date*
<DTNEXT>	Closing date of next statement, *date*
<BALOPEN>	Opening statement balance, *amount*
<BALCLOSE>	Closing statement balance, *amount*
<DTPMTDUE>	Payment due date, *date*
<MINPMTDUE>	Minimum amount due, *amount*
<FINCHG>	Finance charges, *amount*
<PAYANDCREDIT>	Total of payments and credits, *amount*
<PURANDADV>	Total of purchases and cash advances, *amount*
<DEBADJ>	Debit adjustments, *amount*
<CREDITLIMIT>	Current credit limit, *amount*
<DTPOSTSTART>	Start date of transaction data for this statement, *date* A client should be able to use this date in a <CCSTMTRQ> to request transactions that match this statement.
<DTPOSTEND>	End date of transaction data for this statement, *date* A client should be able to use this date in a <CCSTMTRQ> to request transactions that match this statement.
<MKTGINFO>	Marketing information (at most 1), *A-360*
<IMAGEDATA> </IMAGEDATA>	Image data aggregate, see section 3.1.6.1
Currency options. Choose either <CURRENCY> or <ORIGCURRENCY>.	
<CURRENCY> </CURRENCY> -or- *<ORIGCURRENCY> </ORIGCURRENCY>*	Currency, if different from CURDEF, see section 5.2.

Tag	*Description*
</CCCLOSING>	

11.5.5 Loan Statement End Request <LOANSTMTENDRQ>

The loan statement closing request is semantically identical to the bank and credit card statement closing requests. However, the <LOANSTMTENDTRNRQ> aggregate contains the loan request, not the <STMTENDRQ> or <CCSTMTENRQ> aggregate.

The <LOANSTMTENDRQ> request must appear within a <LOANSTMTENDTRNQ> transaction wrapper.

	Description
<LOANSTMTENDRQ>	Loan-dosing-statement-request aggregate
<LOANACCTFROM>	Loan-account-from aggregate, see section 11.3.7
</LOANACCTFROM>	
<DTSTART>	Start date for statement closing information, *datetime*
<DTEND>	End date of statement closing information, *datetime*
<INCSTMTIMG>	Include data for closing mages, *Boolean*
</LOANSTMTENDRQ>	

11.5.6 Loan Statement End Response <LOANSTMTENDRS>

The loan statement closing response is semantically identical to the bank and credit card statement closing response. However, the <LOANSTMTENDRS> aggregate contains the loan response, not the <STMTENDRS> or <CCSTMTENDRS> aggregate.

The <LOANSTMTENDRS> response must appear within a <LOANSTMTENDTRNRS> transaction wrapper.

	Description
<LOANSTMTENDRS>	Loan-closing-statement-response aggregate
<CURDEF>	Default currency for closing
<LOANACCTFROM>	Loan-account-from aggregate, see section 11.3.7
</LOANACCTFROM>	
<LOANCLOSING>	Statement information (0 or more). See section 11.5.8
</LOANCLOSING>	
</LOANSTMTENDRS>	

11.5.7 Status Codes

	Meaning
0	Success
2000	General error (ERROR)
2002	General account error (ERROR)
2003	Accouont not found (ERROR)
2004	Account closed (ERROR)
2005	Accouont not authorized (ERROR)
2019	Duplicate request (ERROR)
2020	Invalid date (ERROR)
2027	Invalid date range (ERROR)

11.5.8 Loan Closing <LOANCLOSING>

A loan account uses the <LOANCLOSING> aggregate to describe statement closing information. The <FITID> provides a way for the client to distinguish one closing statement from another. <DTSTART> and <DTEND> should be interpreted by clients and servers as described in Chapter 3.

For each <LOANCLOSING> returned, clients should be able to retrieve corresponding transactions by using <DTPOSTSTART> and <DTPOSTEND> as <DTSTART> and <DTEND> in a <LOANSTMTRQ> request.

Tag	*Description*
<LOANCLOSING>	Loan-statement-information aggregate
<FITID>	Unique identifier for this statement, *FITID*
<DTOPEN>	Opening statement date, *date*
<DTCLOSE>	Closing statement date, *date*
<DTNEXT>	Closing date of next statement, *date*
<BALOPEN>	Opening statement principal balance, *amount*
<PRINBAL>	Current Principal Balance aggregate - see 11.3.8.2
</PRINBAL>	
<LOANINT>	Loan interest aggregate - see section 11.3.8.3
</LOANINT>	
<LOANIRATE>	Current interest rate aggregate - see section 11.3.8.4
</LOANIRATE>	
<ESTPAYOFF>	Estimated payoff balance aggregate, see section 11.5.8.1
</ESTPAYOFF>	
<BALLOONAMT>	Not included or zero for regular loans, otherwise balloon payment amount, *amount*
<LOANPMT>	Loan Payment aggregate - see section 11.3.8.5
</LOANPMT>	
<LOANRMNPMTS>	Remaining number of loan payments, *N-5*
<BALLIST>	List of miscellaneous other balances
<BAL>	Balance aggregates (0 or more), see section 3.1.4
</BAL>	
</BALLIST>	
<ESCRWBAL>	Escrow balance aggregate, see section 11.5.8.2.

Tag	Description
</ESCRWBAL>	
<DTPOSTSTART>	Start date of transaction data for this statement, *date* A client should be able to use this date in a <LOANSTMTRQ> to request transactions that match this statement.
<DTPOSTEND>	End date of transaction data for this statement, *date* A client should be able to use this date in a <LOANSTMTRQ> to request transactions that match this statement.
<MKTGINFO>	Marketing information (at most 1), *A-360*
<IMAGEDATA>	Image data aggregate, see section 3.1.6.1
</IMAGEDATA>	
Currency options. Choose either <CURRENCY> or <ORIGCURRENCY>.	
<CURRENCY>	Currency, if different from CURDEF
</CURRENCY>	
-or-	
<ORIGCURRENCY>	
</ORIGCURRENCY>	
</LOANCLOSING>	

11.5.8.1 Estimated Payoff Balance <ESTPAYOFF>

	Description
<ESTPAYOFF>	
<ESTPAYOFFBAL>	Estimated Payoff amount, *amount*
<DTASOF>	Payoff date, *datetime*
</ESTPAYOFF>	

11.5.8.2 Escrow Balance <ESCRWBAL>

	Description
<ESCRWBAL>	
<BALAMT>	Total amount held in escrow, *amount*
<ESCRWTAXBAL>	Balance of taxes held in escrow, *amount*
<ESCRWINSBAL>	Balance of insurance held in escrow, *amount*
<ESCRWPMIBAL>	Balance of PMI held in escrow, *amount*
<ESCRWFEESBAL>	Balance of fees held in escrow, *amount*
<ESCRWOTHERBAL>	Balance of other amount held in escrow, *amount*
<DTASOF>	Date of escrow balance, *datetime*
</ESCRWBAL>	

The amount of ESCRWTAXBAL + ESCRWINSBAL + ESCRWPMIBAL + ESCRWFEESBAL + ESCRWOTHERBAL should normally equal BALAMT. However, if values do not compute, client should roll remainder into <ESCRWOTHERBAL>.

11.6 Stop Check

OFX supports a request to issue a stop payment for one or more outstanding checks. The stop request can be for a single check or for a range of checks. There must be one request for each check or range of checks the user wants to stop.

When stopping a single check, the client can provide a payee name and optionally an amount instead of a check number to describe the check to stop. Not all servers can support this behavior.

Examples:

Stop check 22	– one request
Stop check to “Acme Lighting”	– one request
Stop checks 200-224	– one request
Stop checks 275-280, 283	– two requests (first stops 275-280, the next stops 283)

	Server Responds
Account information	
Check number(s) to stop -or- Check description	
	Status for each check

11.6.1 Stop Check Add

Stop Check Add is subject to synchronization.

11.6.1.1 Request <STPCHKRQ>

The <STPCHKRQ> request must appear within a <STPCHKTRNRQ> transaction wrapper.

	Description
<STPCHKRQ>	Stop-check-request aggregate
<BANKACCTFROM>	Account-from aggregate, see section 11.3.1
</BANKACCTFROM>	
Check options. Choose either<CHKRANGE> or <CHKDESC>.	
<CHKRANGE>	Check range aggregate, see section 11.6.1.1.1
</CHKRANGE>	
-or-	
<CHKDESC>	Check description aggregate, see section 11.6.1.1.2
</CHKDESC>	
</STPCHKRQ>	

11.6.1.1.1 Check Range <CHKRANGE>

	Description
<CHKRANGE>	Check-range aggregate
<CHKNUMSTART>	Start check number to cancel, *A-12*
<CHKNUMEND>	Ending check number to cancel; omit if only one check is to be stopped, *A-12*
</CHKRANGE>	

11.6.1.2.1 Stopped Check <STPCHKNUM>

This aggregate contains a status code that indicates whether or not a specific check was canceled.

	Description
<STPCHKNUM>	Stopped-check-item aggregate
<CHECKNUM>	Check number, *A-12*
<NAME>	Payee name or description, *A-32*
<DTUSER>	Date on check, *datetime*
<TRNAMT>	Amount, *amount*
<CHKSTATUS>	Status code for individual stop check request 0 = OK 1 = rejected 100 = check not found 101 = check already posted
<CHKERROR>	Further textual explanation, *A-255*
Currency options. Choose either <CURRENCY> or <ORIGCURRENCY>.	
<CURRENCY> *</CURRENCY>*	Currency, if different from a7tjEeme
-or-	

11.6.2 Status Codes

	Meaning
0	Success
2000	General error (ERROR)
2002	General account error (ERROR)
2003	Account not found (ERROR)
2004	Account closed (ERROR)
2005	Account not authorized (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10000	Stop check in process (INFO)
10500	Too many checks to process (ERROR)

11.7 Intrabank Funds Transfer

OFX supports transferring funds between two accounts at the same financial institution. Funds transfers in OFX can be immediate or scheduled. Scheduled transfers can repeat at specified intervals.

Financial institutions can choose to support:

- Immediate transfers
- Immediate and scheduled transfers
- Immediate, scheduled, and recurring transfers

Recurring transfers require support for scheduled transfers.

In general, an OFX server may not choose which transactions to support unless the profile can be used to indicate to the client that a transaction is not supported. However, immediate intrabank funds transfers usually cannot be modified or canceled, so a server that does not support scheduled transfers may return an error code on any request for cancel or modify. A preferred approach would be to return status code 2016, which means the transfer may not be modified or canceled because it is already committed. (An immediate transfer may not actually commit until the end of the business day. For more information, see the discussion on the support of INTRASYNCRQ in section 11.12.2.)

After a transfer has executed, the server can either issue a transfer modification response in the sync or it can do nothing. In the latter case, it would be up to the client to get status from a statement download. If a transfer fails, it is recommended, but not required, that a transfer modification response with the appropriate XFERPRCCODE be sent in the sync.

In general, all Intrabank Funds Transfer requests are subject to synchronization. The only exception occurs when the request is for an immediate transfer and the server is able to successfully perform the transfer in real time. In that case, the server may choose whether or not the transfer affects the sync history. After receiving an immediate response indicating that a transfer took place in real time, the client must not expect the relevant token to change or to receive information about that transfer in a later sync response. Servers choosing to ignore real time immediate transfers in the sync history force additional clients to wait until the transfer appears in a statement download for information about the transfer. Note that a server that does not implement file based error recovery should include immediate transfers in the sync history since error recovery attempts will be token-based rather than file based.

Note: If a server batches up immediate transfers, to be processed that night or possibly the next day, it should return a <WILLPROCESSON> status in the immediate transfer response. At that point it is up to the server whether or not to send the <INTRARS> in the sync before the transfer actually occurs. *It should be noted that servers that don't sync such "batched, but not yet transferred" responses prevent other clients accessing the same account from getting accurate balance information during this time.* After the transfer, the up-to-date balance information can be obtained from either the sync (if the server supports this) or the statement downloaded.

11.7.1 Intrabank Funds Transfer Addition

The Intrabank Funds Transfer Add request provides a way for a client to set up a single transfer. The request designates source and destination accounts and the amount of the transfer. The client must provide a date if it has scheduled the transfer. Immediate funds transfers cannot be modified or canceled.

	Server Responds
Source account Destination account Amount Date of transfer (optional)	
	Server ID for the transfer Source account Destination account Amount Expected/actual posting date

Intrabank Funds Transfer Add is subject to synchronization.

11.7.1.1 Request <INTRARQ>

The <INTRARQ> request must appear within an <INTRATRNRQ> transaction wrapper.

Tag	***Description***
<INTRARQ>	Intrabank-transfer-request aggregate
<XFERINFO>	Transfer information aggregate, see section 11.3.5
</XFERINFO>	
</INTRARQ>	

11.7.1.2 Response <INTRARS>

A server cannot, in all cases, provide complete confirmation for the transfer. The server can confirm only that it received the transfer instruction; and possibly whether it validated the accounts, amount, and date specified in the transfer. For any transfer where the client does not know the status at the time of the response, a server should confirm that it accepted the instruction and indicate the expected posting date of the transfer. A client can pick up the confirmation at a later date through a synchronization request. Servers should inform clients of any errors found while processing this transaction using the <STATUS> aggregate. A response containing <STATUS><CODE>0 and

<XFERPRCSTS><XFERPRCCODE>FAILEDON should be avoided for problems such as an invalid account or amount.

If the request is for an immediate transfer and the server can perform the transfer in real time, the server should indicate whether the transfer succeeded and should return the date of the transfer in <DTPOSTED>. In this case, synchronization is not required.

The <INTRARS> response must appear within an <INTRATRNRS> transaction wrapper.

	Description
<INTRARS>	Intrabank-transfer-response aggregate
<CURDEF>	Default currency for the intrabank transfer response, *currsymbol*
<SRVRTID>	Server ID for this transfer, *SRVRTID*
<XFERINFO>	Transfer information aggregate, see section 11.3.5
</XFERINFO>	
Transfer-date options. Choose either <DTXFERPRJ> or <DTPOSTED>	
<DTXFERPRJ>	Projected date of the transfer; response can contain either a <DTXFERPRJ> or a <DTPOSTED> but not both; *datetime*
-or-	
<DTPOSTED>	Actual date of the transfer, *datetime*
<RECSRVRTID>	If the response is generated by a recurring transfer model, this ID references it, see section 11.10, *SRVRTID*
<XFERPRCSTS>	Transfer-processing status, see section 11.3.6
</XFERPRCSTS>	
</INTRARS>	

Note: The server can deliver this response to a client immediately after the request is made (for an immediate or one-time scheduled transfer). The server should also return this response for any transfers that were generated by a model.

11.7.1.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2006	Source account not found (ERROR)
2007	Source account closed (ERROR)
2008	Source account not authorized (ERROR)
2009	Destination account not found (ERROR)
2010	Destination account closed (ERROR)
2011	Destination account not authorized (ERROR)
2012	Invalid amount (ERROR)
2014	Date too soon (ERROR)
2015	Date too far in future (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10504	Insufficient funds (ERROR)

11.7.2.2 Response <INTRAMODRS>

This response normally just echoes the values passed by the client. However, if the status of a scheduled transfer changes in any way, clients may expect to receive modification responses when they synchronize with the server. For example, when a server completes a transfer, the status of the transfer goes from *pending* to *posted*. Servers may notify clients of this status change if they wish.

The <INTRAMODRS> response must appear within an <INTRATRNRS> transaction wrapper.

	Description
<INTRAMODRS>	Modification-response aggregate
<SRVRTID>	ID assigned by the server to the transfer being modified, *SRVRTID*
<XFERINFO>	Transfer information aggregate, see section 11.3.5
</XFERINFO>	
<XFERPRCSTS>	Transfer processing status, see section 11.3.6
</XFERPRCSTS>	
</INTRAMODRS>	

11.7.2.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2006	Source account not found (ERROR)
2007	Source account closed (ERROR)
2008	Source account not authorized (ERROR)
2009	Destination account not found (ERROR)
2010	Destination account closed (ERROR)
2011	Destination account not authorized (ERROR)
2012	Invalid amount (ERROR)
2014	Date too soon (ERROR)
2015	Date too far in future (ERROR)
2016	Transaction already committed (ERROR)
2017	Already canceled (ERROR)
2018	Unknown server ID (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10500	Too many checks to process (ERROR)
10505	Cannot modify element (ERROR)
10514	Transaction already processed (ERROR)

11.7.3 Intrabank Funds Transfer Cancellation

The client sends a Transfer Cancellation request to cancel a scheduled transfer, where <SRVRTID> identifies the transfer. Immediate transfers cannot be canceled, so this request should be used only for scheduled transfers.

Intrabank Funds Transfer Cancellation is subject to synchronization.

11.7.3.1 Request <INTRACANRQ>

The <INTRACANRQ> request must appear within an <INTRATRNRQ> transaction wrapper.

	Description
<INTRACANRQ>	Transfer-cancellation-request aggregate
<SRVRTID>	ID of the transfer the user wants to cancel. The server must have previously assigned this ID to a transfer. *SRVRTID*
</INTRACANRQ>	

11.7.3.2 Response <INTRACANRS>

The <INTRACANRS> response must appear within an <INTRATRNRS> transaction wrapper.

	Description
<INTRACANRS>	Transfer-cancellation-response aggregate
<SRVRTID>	ID of the transfer the user wants to cancel. The server must have previously assigned this ID to a transfer. *SRVRTID*
</INTRACANRS>	

11.7.3.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2016	Transaction already committed (ERROR)
2017	Already canceled (ERROR)
2018	Unknown server ID (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10514	Transaction already processed (ERROR)

11.7.3.4 DTDUE, DTPOSTED, and DTXFERPRJ in Immediate Transfers

The following is a list of what might be returned in an immediate mode transfer response. The interpretation of each response is provided. All responses referenced in this section are immediate and describe success conditions. That is, we are not attempting to describe the INTRARS aggregates that may be returned within a INTRASYNCRS response. Further, successful INTRARS aggregates for immediate transfers are not expected to appear in lite synchronization INTRASYNCRS responses.

- DTDUE and INTRARQ
  DTDUE should not be present if the client is requesting an immediate mode transfer. If it is, then this is a client error, and will be treated by the server as if the client were attempting to create a scheduled transfer.
- DTDUE, DTPOSTED, and DTXFERPRJ NOT returned in INTRARS
  The client should interpret this from the server as indicating that the immediate transfer request was processed in real-time.
- DTDUE only returned
  DTDUE should not be present in the response to a request for an immediate transfer. If it were present, the XFERINFO aggregate would not match that found in the request.
- DTPOSTED only returned
  If this is not equal to today's date and an earlier time than "now", then this is a server error. Otherwise, the client should interpret this as confirmation from the server that the request was processed in real-time.
- DTXFERPRJ only returned
  The client should interpret this as indication that the transfer request will be completed by the specified date. The client may receive an INTRAMODRS or INTRARS with updated information about this transfer when it is actually processed. That future INTRARS or INTRAMODRS will contain the DTPOSTED to reflect when the transfer occurred. This response is not required for successful transfers processed at the originally specified projected date and time.

11.8 Interbank Funds Transfer

The Interbank Funds Transfer Add request provides a way for a client to set up a single transfer between accounts at different financial institutions. Like intrabank funds transfers, the request designates source and destination accounts and the amount of the transfer. Also, as in intrabank funds transfers, the FI must be able to authenticate the source account. However, interbank funds transfers differ from intrabank funds transfers in the following respects:

- The routing and transit number of the destination account differs from the source account.
- At the discretion of an FI, the destination account can be subject to pre-notification.
- Source and destination accounts must be enabled for the Automated Clearing House (ACH).

Use the ACH system to implement the Interbank Funds Transfer, which is subject to the rules and regulations governing the ACH network.

In all other respects, interbank funds transfers function like intrabank funds transfers. The user can schedule them for a future date or request an immediate transfer. The user can modify or cancel scheduled transfers, but not immediate transfers. Scheduled transfers can recur at regular intervals.

11.8.1 Interbank Funds Transfer – US

In the United States, interbank funds transfers usually use only the <XFERINFO> portion of the request and response.

	Server Responds
Source account	
Destination account	
Amount	
Date of transfer (optional)	
	Server ID for the transfer
	Source account
	Destination account
	Amount
	Expected/actual posting date

Interbank Funds Transfer Add is subject to synchronization.

11.8.2 Interbank Funds Transfer – International Usage

In countries where the funds transfer is the basis of the payments system, the OFX payments messages allow specifying payees by destination account (see Chapter 12, "Payments").

Interbank Funds Transfer Add is subject to synchronization.

11.8.2.1 Interbank Funds Transfer Request <INTERRQ>

The <INTERRQ> request must appear within an <INTERTRNRQ> transaction wrapper.

	Description
<INTERRQ>	Interbank-transfer-request aggregate
<XFERINFO>	Transfer information aggregate, see section 11.3.5
</XFERINFO>	
</INTERRQ>	

11.8.2.2 Interbank Funds Transfer Response <INTERRS>

The server cannot provide complete confirmation for interbank transfer. It can confirm only that the FI received the transfer instruction and possibly validated the source account, amount, and date specified in the transfer. Since the client does not know the status of the transfer at the time of the response, the server should confirm that it accepted the instruction and indicate the expected posting date of the transfer. The client can pick up the confirmation at a later date through a synchronization request. Servers should inform clients of any errors found while processing this transaction using the <STATUS> aggregate. A response containing <STATUS><CODE>0 and <XFERPRCSTS><XFERPRCCODE>FAILEDON should be avoided for problems such as an invalid account or amount.

The <INTERRS> response must appear within an <INTERTRNRS> transaction wrapper.

	Description
<INTERRS>	Interbank-transfer-response aggregate
<CURDEF>	Currency used in transfer, *currsymbol*
<SRVRTID>	Server ID for this transfer, *SRVRTID*
<XFERINFO>	Transfer information aggregate, see section 11.3.5
</XFERINFO>	
Transfer-date options. Choose either <DTXFERPRJ> or <DTPOSTED>	
<DTXFERPRJ>	Projected date of the transfer; response can contain either a <DTXFERPRJ> or a <DTPOSTED> but not both; *datetime*
-or-	
<DTPOSTED>	Actual date of the transfer, *datetime*
<REFNUM>	Server can generate a reference or check for the transfer, *A-32*
<RECSRVRTID>	If server generates the response by a recurring transfer model, this ID references it. *SRVRTID*
<XFERPRCSTS>	Transfer-processing status, see section 11.3.6
</XFERPRCSTS>	
</INTERRS>	

Note: A server can deliver this response to a client immediately after the client makes the request (for an immediate or one-time scheduled transfer). In response to a synchronization request by a client, the server should provide a second response containing complete status regarding the transfer. It should also return any transfers that it generates by a model.

11.8.2.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2006	Source account not found (ERROR)
2007	Source account closed (ERROR)
2008	Source account not authorized (ERROR)
2009	Destination account not found (ERROR)
2010	Destination account closed (ERROR)
2011	Destination account not authorized (ERROR)
2012	Invalid amount (ERROR)
2014	Date too soon (ERROR)
2015	Date too far in future (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10504	Insufficient funds (ERROR)

11.8.3 Interbank Funds Transfer Modification

The client sends a Transfer Modification request to modify a scheduled transfer. Immediate transfers cannot be modified, so this request should only be used for scheduled transfers. Once created and retrieved by the customer, spawned transfers are almost identical to customer-created transfers. (The exception is when a spawned transfer is modified or cancelled due to a recurring modification or cancellation request.) As with ordinary transfers, you can cancel or modify transactions individually. When modifying a transfer, the client must specify all of the elements and aggregates within the <XFERINFO> aggregate that were specified when the transfer was created, not just the elements and aggregates that the client wants to modify. <SRVRTID> specifies which transfer to modify. Some servers cannot support the modification of certain values. Servers must indicate this by returning status code 10505 when the client requests an unsupported modification. Clients must not change <BANKACCTFROM> or <CCACCTFROM> in a funds transfer modification.

Interbank Funds Transfer Modification is subject to synchronization.

11.8.3.1 Request <INTERMODRQ>

The <INTERMODRQ> request must appear within an <INTERTRNRQ> transaction wrapper.

	Description
<INTERMODRQ>	Modification-request aggregate
<SRVRTID>	ID assigned by the server to the transfer being modified, *SRVRTID*

11.8.3.2 Response <INTERMODRS>

The <INTERMODRS> response must appear within an <INTERTRNRS> transaction wrapper.

	Description
<INTERMODRS>	Modification-response aggregate
<SRVRTID>	ID assigned by the server to the transfer being modified, *SRVRTID*
<XFERINFO>	Transfer information aggregate; server returns if client provided an <XFERINFO> in the request, see section 11.3.5
</XFERINFO>	
<XFERPRCSTS>	Processing status for transfer, see section 11.3.6
</XFERPRCSTS>	
</INTERMODRS>	

11.8.3.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2006	Source account not found (ERROR)
2007	Source account closed (ERROR)
2008	Source account not authorized (ERROR)
2009	Destination account not found (ERROR)
2010	Destination account closed (ERROR)
2011	Destination account not authorized (ERROR)
2012	Invalid amount (ERROR)
2014	Date too soon (ERROR)
2015	Date too far in future (ERROR)
2016	Transaction already committed (ERROR)
2017	Already canceled (ERROR)
2018	Unknown server ID (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10504	Insufficient funds (ERROR)
10505	Cannot modify element (ERROR)
10514	Transaction already processed (ERROR)

11.9 Wire Funds Transfer

OFX enables clients to set up wire funds transfers. Wire funds transfers are similar to other types of funds transfers. Clients designate a source account that the FI can authenticate and a destination account at the same or a different institution. Clients also designate an amount and an optional date.

The FI must know the originator of the transfer. The beneficiary of the transfer might be an established customer at the same institution.

OFX implements wire funds transfers using the FedWire system, and is subject to its rules and regulations.

In almost all respects, wire funds transfers work like interbank funds transfers. A user can schedule and cancel them. Unlike interbank funds transfers, a user cannot modify Wire funds transfers once they have been set up. A user cannot set up wire funds transfers to recur at regular intervals.

	Server Responds
Source account	
Originator	
Receiver	
Amount	
Date of transfer (optional)	
	Server ID for the transfer
	Originator
	Receiver
	Amount
	Expected/actual posting date

11.9.1.1.1 Wire Beneficiary Aggregate <WIREBENEFICIARY>

The wire beneficiary aggregate describes the receiver of a wire transfer.

	Description
<WIREBENEFICIARY>	Wire-beneficiary aggregate
<NAME>	Name of beneficiary, *A-32*
<BANKACCTTO>	Bank details for beneficiary, see section 11.3.1
</BANKACCTTO>	
<MEMO>	Information for the beneficiary, *memo*
</WIREBENEFICIARY>	

11.9.1.1.2 Extended Bank Description aggregate <EXTBANKDESC>

	Description
<EXTBANKDESC>	Extended-bank-description aggregate
<NAME>	Abbreviated name of bank, *A-32*
<BANKID>	Routing: ABA number or S.W.I.F.T. number, *A-9*
<ADDR1>	Bank's address line 1, *A-32*
<ADDR2>	Bank's address line 2, *A-32*
<ADDR3>	Bank's address line 3. Use of <ADDR3> requires the presence of <ADDR2>, *A-32*
<CITY>	Bank's city, *A-32*
<STATE>	Bank's state or province, *A-5*
<POSTALCODE>	Bank's postal code, *A-11*
<COUNTRY>	Bank's country; 3-letter country code from ISO/DIS-3166, *A-3*
<PHONE>	Bank's phone number, *A-32*
</EXTBANKDESC>	

11.9.1.2 Response <WIRERS>

The server cannot provide complete confirmation for the transfer. It can confirm only that the server received the transfer instruction and possibly that it validated the source account, amount, and date specified in the transfer. For any transfer where the client does not know the status at the time of the response, the server should confirm that it accepted the instruction and indicate the expected posting date of the transfer. The client can pick up the confirmation at a later date through a synchronization request.

The server can indicate the fee assessed for the transfer by using the <FEE> element in the response. The server can also include a confirmation message in the response.

The <DTDUE> in a response may have been adjusted by a server. For example, the server may adjust <DTDUE> to comply with non-processing days. If a client sends a request to make a transfer on July 4 and July 4 happens to be a non-processing day, the <DTDUE> in the response may be July 4 (because the server hasn't adjusted it yet), July 5 (because this server rolls dates forward), or some other date. For this reason, a client should pay attention to the <DTDUE> in the response.

The <WIRERS> response must appear within a <WIRETRNRS> transaction wrapper.

	Description
<WIRERS>	Wire-transfer-response aggregate
<CURDEF>	Currency used in transfer, *currsymbol*
<SRVRTID>	Server ID for this transfer, *SRVRTID*
<BANKACCTFROM>	Source of funds, see section 11.3.1
</BANKACCTFROM>	
<WIREBENEFICIARY>	Wire transfer beneficiary, see section 11.9.1.1.1
</WIREBENEFICIARY>	
<WIREDESTBANK>	Beneficiary's bank
<EXTBANKDESC>	Extended bank description, see section 11.9.1.1.2
</EXTBANKDESC>	
</WIREDESTBANK>	
<TRNAMT>	Transfer amount, *amount*
<DTDUE>	Date to occur, echoed if provided in request, *datetime*
<PAYINSTRUCT>	Payment instructions, echoed if provided in request, *A-255*
Transfer-date options. Choose either <DTXFERPRJ> or <DTPOSTED>	
<DTXFERPRJ>	Projected date of the transfer; response can contain either a <DTXFERPRJ> or a <DTPOSTED> but not both; *datetime*
-or-	
<DTPOSTED>	Actual date of the transfer, *datetime*
<FEE>	Fee assessed for the transfer, *amount*
<CONFMSG>	Confirmation message, *A-255*
</WIRERS>	

11.9.1.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2006	Source account not found (ERROR)
2007	Source account closed (ERROR)
2008	Source account not authorized (ERROR)
2012	Invalid amount (ERROR)
2014	Date too soon (ERROR)
2015	Date too far in future (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10504	Insufficient funds (ERROR)
10516	Wire beneficiary invalid (ERROR)

11.9.2 Wire Funds Transfer Cancellation

The client sends a Wire Funds Transfer Cancellation Request to cancel a scheduled transfer, where <SRVRTID> identifies the transfer.

Wire Funds Transfer Cancellation is subject to synchronization.

11.9.2.1 Request <WIRECANRQ>

The <WIRECANRQ> request must appear within a <WIRETRNRQ> transaction wrapper.

Tag	*Description*
<WIRECANRQ>	Wire-transfer-cancellation-request aggregate
<SRVRTID>	ID of the transfer to cancel; server must have previously assigned this ID to a transfer, *SRVRTID*
</WIRECANRQ>	

11.10 Recurring Funds Transfer

OFX uses a Recurring Funds Transfer Add request to set up a recurring transfer model. The transfer model generates transfers according to its schedule. Transfers created by a model and retrieved by a customer can be modified or canceled without impacting the model.

A user can create recurring funds transfer models to generate two types of scheduled transfers: interbank and intrabank. You cannot set up recurring wire funds transfers.

For more information on recurring transactions, see Chapter 10, "Recurring Transactions."

11.10.1 Recurring Intrabank Funds Transfer Addition

A Recurring Intrabank Funds Transfer Add request sets up an intrabank funds transfer that repeats at a specified interval for a specified period of time.

Model-created transfers are retrieved by means of a synchronization request.

	Server Responds
Source account	
Destination account	
Amount	
Date of first transfer	
Frequency	
Duration	
	Server ID for the model
	Source account
	Destination account
	Amount
	Date of first transfer
	Frequency
	Duration

Recurring Intrabank Funds Transfer Add is subject to synchronization.

11.10.1.1 Request <RECINTRARQ>

The <RECINTRARQ> request must appear within a <RECINTRATRNRQ> transaction wrapper.

	Description
<RECINTRARQ>	Recurring-transfer-request aggregate
<RECURRINST>	Recurring-instructions aggregate, see section
</RECURRINST>	
<INTRARQ>	Intrabank-transfer-request aggregate, see section 11.7.1.1
</INTRARQ>	
</RECINTRARQ>	

11.10.1.2 Response <RECINTRARS>

The <RECINTRARS> response must appear within a <RECINTRATRNRS> transaction wrapper.

For version 1 of the message set, the <SRVRTID> included in the <INTRARS> should be set to the same value as the <RECSRVRTID>.

Note: This is the response to the recurring model only. Servers must still generate an INTRARS for each instance of the recurring transfer.

	Description
<RECINTRARS>	Recurring-transfer-response aggregate
<RECSRVRTID>	Server-assigned ID for this model, *SRVRTID*
<RECURRINST>	Recurring-instructions aggregate
</RECURRINST>	
<INTRARS>	Intrabank-transfer-response aggregate, see section 11.7.1.2
</INTRARS>	
</RECINTRARS>	

11.10.1.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2006	Source account not found (ERROR)
2007	Source account closed (ERROR)
2008	Source account not authorized (ERROR)
2009	Destination account not found (ERROR)
2010	Destination account closed (ERROR)
2011	Destination account not authorized (ERROR)
2014	Date too soon (ERROR)
2015	Date too far in future (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10508	Invalid frequency (ERROR)

11.10.2 Recurring Intrabank Funds Transfer Modification

The client sends a Recurring Intrabank Funds Transfer Modification request to modify a recurring intrabank transfer model.

Recurring Intrabank Funds Transfer Modification is subject to synchronization.

Clients must not change <BANKACCTFROM> in a recurring funds transfer modification.

11.10.2.1 Request <RECINTRAMODRQ>

<RECSRVRTID> identifies the model. The client can indicate whether the changes should apply to pending transfers.

The <RECINTRAMODRQ> request must appear within a <RECINTRATRNRQ> transaction wrapper.

	Description
<RECINTRAMODRQ>	Recurring-modification-request aggregate
<RECSRVRTID>	ID assigned by the server to the model being modified, *SRVRTID*
<RECURRINST>	Recurring-instructions aggregate
</RECURRINST>	
<INTRARQ>	Intrabank-transfer-request aggregate, see section 11.7.1.1
</INTRARQ>	
<MODPENDING>	Modify pending flag, *Boolean* If the client sets this flag, the server must modify pending and future transfers.
</RECINTRAMODRQ>	

11.10.2.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2006	Source account not found (ERROR)
2007	Source account closed (ERROR)
2008	Source account not authorized (ERROR)
2009	Destination account not found (ERROR)
2010	Destination account closed (ERROR)
2011	Destination account not authorized (ERROR)
2012	Invalid amount (ERROR)
2014	Date too soon (ERROR)
2015	Date too far in future (ERROR)
2016	Transaction already committed (ERROR)
2017	Already canceled (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10500	Too many checks to process (ERROR)
10505	Cannot modify element (ERROR)
10508	Invalid frequency (ERROR)
10514	Transaction already processed (ERROR)
10518	Unknown model ID (ERROR)

11.10.3 Recurring Intrabank Funds Transfer Cancellation

The client sends a Recurring Intrabank Funds Transfer Cancellation request to cancel a recurring intrabank transfer model.

Recurring Intrabank Funds Transfer Cancellation is subject to synchronization.

11.10.3.1 Request <RECINTRACANRQ>

<RECSRVRTID> identifies the model the user wants to cancel. The client can indicate whether the cancel should apply to pending transfers.

The <RECINTRACANRQ> request must appear within a <RECINTRATRNRQ> transaction wrapper.

	Description
<RECINTRACANRQ>	Recurring-transfer-cancellation-request aggregate
<RECSRVRTID>	ID assigned by the server to the model being canceled, *SRVRTID*
<CANPENDING>	Cancel pending flag, *Boolean* If Y, server should cancel all pending and unspawned transfers. If N, server should cancel only the model (and unspawned transfers).
</RECINTRACANRQ>	

11.10.3.2 Response <RECINTRACANRS>

The <RECINTRACANRS> response must appear within a <RECINTRATRNRS> transaction wrapper.

	Description
<RECINTRACANRS>	Recurring-transfer-cancellation-response aggregate
<RECSRVRTID>	ID assigned by the server to the model being canceled, *SRVRTID*
<CANPENDING>	Cancel pending flag, *Boolean* Y if the client requested that the server cancel all pending and unspawned transfers. N if the client requested that the server cancel only unspawned transfers.
</RECINTRACANRS>	

11.10.3.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2016	Transaction already committed (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10509	Model already canceled (ERROR)
10514	Transaction already processed (ERROR)
10518	Unknown model ID (ERROR)

11.10.4 Recurring Interbank Funds Transfer Addition

A Recurring Interbank Funds Transfer Add request sets up an interbank funds transfer that repeats at a specified interval for a specified period of time.

The client retrieves model-created transfers by means of a synchronization request.

Client Sends	*Server Responds*
Source account	
Destination account	
Amount	
Date of first transfer	
Frequency	
Duration	
	Server ID for the model
	Source account
	Destination account
	Amount
	Date of first transfer
	Frequency
	Duration

Recurring Interbank Funds Transfer Add is subject to synchronization

11.10.4.1 Request <RECINTERRQ>

The <RECINTERRQ> request must appear within a <RECINTERTRNRQ> transaction wrapper.

	Description
<RECINTERRQ>	Recurring-transfer-request aggregate
<RECURRINST>	Recurring-instructions aggregate
</RECURRINST>	
<INTERRQ>	Interbank-transfer-request aggregate, see section 11.8.2.1
</INTERRQ>	
</RECINTERRQ>	

11.10.4.2 Response <RECINTERRS>

The <RECINTERRS> response must appear within a <RECINTERTRNRS> transaction wrapper.

For version 1 of the message set, the <SRVRTID> included in the <INTERRS> should be set to the same value as the <RECSRVRTID>.

Note: This is the response to the recurring model only. Servers must still generate an <INTERRS> for each instance of the recurring transfer.

	Description
<RECINTERRS>	Recurring-transfer-response aggregate
<RECSRVRTID>	Server-assigned ID for this model, *SRVRTID*
<RECURRINST>	Recurring-instructions aggregate, see section 10.2
</RECURRINST>	
<INTERRS>	Interbank funds transfer response, see section 11.8.2.2
</INTERRS>	
</RECINTERRS>	

11.10.4.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2006	Source account not found (ERROR)
2007	Source account closed (ERROR)
2008	Source account not authorized (ERROR)
2009	Destination account not found (ERROR)
2010	Destination account closed (ERROR)
2011	Destination account not authorized (ERROR)
2012	Invalid amount (ERROR)
2014	Date too soon (ERROR)
2015	Date too far in future (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10504	Insufficient funds (ERROR)
10508	Invalid frequency (ERROR)

11.10.5 Recurring Interbank Funds Transfer Modification

The client sends a Recurring Interbank Funds Transfer Modification request to modify a recurring interbank transfer model.

Recurring Interbank Funds Transfer Modification is subject to synchronization.

Clients must not change <BANKACCTFROM> in a recurring funds transfer modification.

11.10.5.1 Request <RECINTERMODRQ>

<RECSRVRTID> identifies the model. The client can indicate whether the changes should apply to pending transfers.

The <RECINTERMODRQ> request must appear within a <RECINTERTRNRQ> transaction wrapper.

	Description
<RECINTERMODRQ>	Recurring-modification-request aggregate
<RECSRVRTID>	ID assigned by the server to the model being modified, *SRVRTID*
<RECURRINST>	Recurring-instructions aggregate
</RECURRINST>	
<INTERRQ>	Interbank-funds-transfer-request aggregate, see section 11.8.2.1.
</INTERRQ>	
<MODPENDING>	Modify pending flag If the client sets this flag, the server must modify pending and future transfers. *Boolean*
</RECINTERMODRQ>	

11.10.5.2 Request <RECINTERMODRS>

The <RECINTERMODRS> response must appear within a <RECINTERTRNRS> transaction wrapper.

	Description
<RECINTERMODRS>	Recurring-transfer-modification-response aggregate
<RECSRVRTID>	ID assigned by the server to the model being modified, *SRVRTID*
<RECURRINST>	Recurring-instructions aggregate
</RECURRINST>	
<INTERRS>	Interbank-funds-transfer-response, see section 11.8.2.2
</INTERRS>	
<MODPENDING>	Modify pending flag, *Boolean* Y if the client requested that the server modify pending and future transfers. N if the client did not request that the server modify pending and future transfers.
</RECINTERMODRS>	

11.10.5.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2006	Source account not found (ERROR)
2007	Source account closed (ERROR)
2008	Source account not authorized (ERROR)
2009	Destination account not found (ERROR)
2010	Destination account closed (ERROR)
2011	Destination account not authorized (ERROR)
2012	Invalid amount (ERROR)
2014	Date too soon (ERROR)
2015	Date too far in future (ERROR)
2016	Transaction already committed (ERROR)
2017	Already canceled (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10504	Insufficient funds (ERROR)
10505	Cannot modify element (ERROR)
10508	Invalid frequency (ERROR)
10510	Invalid payee ID (ERROR)
10514	Transaction already processed (ERROR)
10518	Unknown model ID (ERROR)

11.10.6 Recurring Interbank Funds Transfer Cancellation

The client sends a Recurring Transfer Cancellation request to cancel a recurring transfer model.

Recurring Transfer Cancellation is subject to synchronization.

11.10.6.1 Request <RECINTERCANRQ>

<RECSRVRTID> identifies the model the client wants to cancel. The client can indicate whether the cancel should apply to pending transfers.

The <RECINTERCANRQ> request must appear within a <RECINTERTRNRQ> transaction wrapper.

	Description
<RECINTERCANRQ>	Recurring-transfer-cancellation-request aggregate
<RECSRVRTID>	ID assigned by the server to the model being canceled, *SRVRTID*
<CANPENDING>	Cancel pending flag, *Boolean* If Y, server should cancel all pending and unspawned transfers. If N, server should cancel only the model (and unspawned transfers).
</RECINTERCANRQ>	

11.10.6.2 Response <RECINTERCANRS>

The <RECINTERCANRS> response must appear within a <RECINTERTRNRS> transaction wrapper.

	Description
<RECINTERCANRS>	Recurring-transfer-cancellation-response aggregate
<RECSRVRTID>	ID assigned by the server to the model being canceled, *SRVRTID*
<CANPENDING>	Cancel pending flag, *Boolean* Y if the client requested that the server cancel all pending and unspawned transfers. N if the client requested that the server cancel only unspawned transfers.
</RECINTERCANRS>	

11.10.6.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2016	Transaction already committed (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10509	Model already canceled (ERROR)
10514	Transaction already processed (ERROR)
10518	Unknown model ID (ERROR)

11.11.1.2 Response <BANKMAILRS>

The <BANKMAILRS> response must appear within a <BANKMAILTRNRS> transaction wrapper.

	Description
<BANKMAILRS> *Account-from options. Choose either <BANKACCTFROM> or <CCACCTFROM>.*	Bank-e-mail-response aggregate
<BANKACCTFROM> ***</BANKACCTFROM>*** -or- ***<CCACCTFROM>*** ***</CCACCTFROM>***	Account-from aggregate, see section 11.3.1 Credit-card-account-from aggregate, see section 11.3.2
<MAIL> **</MAIL>** </**BANKMAILRS**>	To, from, message information, see Chapter 9, "Customer to FI Communication"

11.11.1.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2003	Account not found (ERROR)
2004	Account closed (ERROR)
2005	Account not authorized (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
15508	Transaction not authorized (ERROR)
16500	HTML not allowed (ERROR)
16501	Unknown mail To: (ERROR)

11.11.3.2 Response <DEPMAILRS>

The server returns this response (when a deposit has been returned), if it receives a banking e-mail synchronization message.

The <DEPMAILRS> response must appear within a <BANKMAILTRNRS> transaction wrapper.

	Description
<DEPMAILRS>	Notification-message-response aggregate
<BANKACCTFROM>	Account-from aggregate, see section 11.3.1
</BANKACCTFROM>	
<MAIL>	To, from, message information, see Chapter 9, "Customer to FI Communication"
</MAIL>	
<TRNAMT>	Amount of deposit, *amount*
<DTUSER>	Customer date of deposit, *date*
<FEE>	Fee assessed for NSF, *amount*
</**DEPMAILRS**>	

11.11.4 Loan E-Mail

With this message, the user can prepare a message to the FI regarding one of his loan accounts. The server acknowledges receipt of the message. The FI prepares the response that the client picks up when it synchronizes with the server.

11.11.4.1 Request <LOANMAILRQ>

Loan mail request is semantically identical to bank mail request. However, the <LOANMAILRQ> includes loan account information, not <BANKACCTFROM> or <CCACCTFROM>.

The <LOANMAILRQ> request must appear within a <LOANMAILTRNRQ> transaction wrapper.

	Description
<LOANMAILRQ>	Loan e-mail request aggregate
<LOANACCTFROM>	Loan-account-from aggregate, see section 11.3.7
</LOANACCTFROM>	
<MAIL>	To, from, message information, see Chapter 9, "Customer to FI Communication"
</MAIL>	
</**LOANMAILRQ**>	

11.11.4.2 Response <LOANMAILRS>

The <LOANMAILRS> response must appear within a <LOANMAILTRNRS> transaction wrapper.

	Description
<LOANMAILRS>	Loan e-mail request aggregate
<LOANACCTFROM>	Loan-account-from aggregate, see section 11.3.7
</LOANACCTFROM>	
<MAIL>	To, from, message information, see Chapter 9, "Customer to FI Communication"
</MAIL>	
</**LOANMAILRS**>	

11.12.1.2 Response <STPCHKSYNCRS>

	Description
<STPCHKSYNCRS>	Synchronization-response aggregate
<TOKEN>	New synchronization token, *token*
<LOSTSYNC>	Y if the token in the synchronization request is older than the earliest entry in the server's history table. In this case, some responses have been lost. N if the token in the synchronization request is newer than or matches a token in the server's history table. *Boolean*
<BANKACCTFROM>	Bank account of interest; token must be interpreted in terms of this account, see section 11.3.1
</BANKACCTFROM>	
<STPCHKTRNRS>	Stop-check transactions (0 or more)
</STPCHKTRNRS>	
</STPCHKSYNCRS>	

11.12.2 Data Synchronization for Intrabank Funds Transfers

<INTRASYNCRQ> must be supported by all servers, even if it will always return <TOKEN>0 without sync history, because a client cannot know whether or not the server would ever return updated transfer information. Specifically, the client cannot know if a transfer will be processed immediately or at the end of the business day until it has performed at least one transfer operation (then DTPOSTED vs. DTXFERPRJ indicates which "mode" the server operates in). As such, the client must always send an <INTRASYNCRQ> in case the server has updated information about a transfer, including immediate transfers which were actually batch processed at the end of the business day or the next day and which may have failed due to other account activity.

Transfers into an account do not show up in the sync for the recipient account. Only transfers out of an account show up in the sync for that account.

11.12.2.1 Request <INTRASYNCRQ>

	Description
<INTRASYNCRQ> *Client synchronization option; <TOKEN>, <TOKENONLY>, or <REFRESH>*	Synchronization-request aggregate
<TOKEN>	Previous value of <TOKEN> received for this type of synchronization request from server; 0 for first-time requests; *token*
<TOKENONLY>	Request for just the current <TOKEN> without the history, *Boolean*
<REFRESH>	Request for refresh of current state, *Boolean*
<REJECTIFMISSING>	If Y, do not process requests if client <TOKEN> is out of date, *Boolean*
Account-from options. Choose either <BANKACCTFROM> or <CCACCTFROM> or <LOANACCTFROM>	
<BANKACCTFROM> ***</BANKACCTFROM>*** -or- ***<CCACCTFROM>*** ***</CCACCTFROM>*** -or- ***<LOANACCTFROM>*** ***</LOANACCTFROM>***	Account-from aggregate, see section 11.3.1 Credit-card-account-from aggregate, see section 11.3.2 Loan-account-from aggregate, see section 11.3.7
<INTRATRNRQ> </INTRATRNRQ>	Intrabank-funds-transfer transactions (0 or more)
</INTRASYNCRQ>	

11.12.2.2 Response <INTRASYNCRS>

	Description
<INTRASYNCRS>	Synchronization-response aggregate
<TOKEN>	New synchronization token, *token*
<LOSTSYNC>	Y if the token in the synchronization request is older than the earliest entry in the server's history table. In this case, some responses have been lost. N if the token in the synchronization request is newer than or matches a token in the server's history table. *Boolean*
Account-from options. Choose either <BANKACCTFROM> or <CCACCTFROM> or <LOANACCTFROM>	
<BANKACCTFROM>	Account-from aggregate, see section 11.3.1
</BANKACCTFROM>	
-or-	
<CCACCTFROM>	Credit-card-account-from aggregate, see section 11.3.2
</CCACCTFROM>	
-or-	
</LOANACCTFROM>	Loan-account-from aggregate, see section 11.3.7
</LOANACCTFROM>	
<INTRATRNRS>	Intrabank-funds-transfer transactions (0 or more)
</INTRATRNRS>	
</INTRASYNCRS>	

The <INTRASYNCRS> responses contain only intrabank transfers where the <xxxACCTFROM> matches that submitted in the sync request. In other words, only transfer-from accounts are synchronized.

11.12.3.2 Response <INTERSYNCRS>

	Description
<INTERSYNCRS>	Synchronization-response aggregate
<TOKEN>	New synchronization token, *token*
<LOSTSYNC>	Y if the token in the synchronization request is older than the earliest entry in the server's history table. In this case, some responses have been lost. N if the token in the synchronization request is newer than or matches a token in the server's history table. *Boolean*
Account-from options. Choose either <BANKACCTFROM> or <CCACCTFROM> or <LOANACCTFROM>	
<BANKACCTFROM>	Account-from aggregate, see section 11.3.1
</BANKACCTFROM>	
-or-	
<CCACCTFROM>	Credit-card-account-from aggregate, see section 11.3.2
</CCACCTFROM>	
-or-	
<LOANACCTFROM>	Credit-card-account-from aggregate, see section 11.3.7
</LOANACCTFROM>	
<INTERTRNRS>	Interbank-funds-transfer transactions (0 or more)
</INTERTRNRS>	
</INTERSYNCRS>	

The <INTERSYNCRS> responses contain only intrabank transfers where the <xxxACCTFROM> matches that submitted in the sync request. In other words, only transfer-from accounts are synchronized

11.12.5 Data Synchronization for Recurring Intrabank Funds Transfers

11.12.5.1 Request <RECINTRASYNCRQ>

This request will synchronize the client with the server in relation to recurring intrabank transfer models. To synchronize individual transfers that were created by the model (and perhaps canceled by another client), the client must also issue an <INTRASYNCRQ>.

	Description
<RECINTRASYNCRQ> *Client synchronization option; <TOKEN>, <TOKENONLY>, or <REFRESH>*	Synchronization request
<TOKEN>	Previous value of <TOKEN> received for this type of synchronization request from server; 0 for first-time requests; *token*
<TOKENONLY>	Request for just the current <TOKEN> without the history, *Boolean*
<REFRESH>	Request for refresh of current state, *Boolean*
<REJECTIFMISSING> *Account-from options. Choose either <BANKACCTFROM> or <CCACCTFROM> or <LOANACCTFROM>*	If Y, do not process requests if client <TOKEN> is out of date, *Boolean*
<BANKACCTFROM> ***</BANKACCTFROM>*** -or- ***<CCACCTFROM>*** ***</CCACCTFROM>*** -or- ***<LOANACCTFROM>*** ***</LOANACCTFROM>***	Account-from aggregate, see section 11.3.1 Credit-card-account-from aggregate, see section 11.3.2 Loan-account-from aggregate, see section 11.3.7
<RECINTRATRNRQ> </RECINTRATRNRQ>	Recurring-intrabank-funds-transfer transactions (0 or more)
</RECINTRASYNCRQ>	

11.12.5.2 Response <RECINTRASYNCRS>

	Description
<RECINTRASYNCRS>	Synchronization-response aggregate
<TOKEN>	New synchronization token, *token*
<LOSTSYNC>	Y if the token in the synchronization request is older than the earliest entry in the server's history table. In this case, some responses have been lost. N if the token in the synchronization request is newer than or matches a token in the server's history table. *Boolean*
Account-from options. Choose either <BANKACCTFROM> or <CCACCTFROM> or <LOANACCTFROM>	
<BANKACCTFROM>	Account-from aggregate, see section 11.3.1
</BANKACCTFROM>	
-or-	
<CCACCTFROM>	Credit-card-account-from aggregate, see section 11.3.2
</CCACCTFROM>	
-or-	
<LOANACCTFROM>	Credit-card-account-from aggregate, see section 11.3.7
</LOANACCTFROM>	
<RECINTRATRNRS>	Recurring-intrabank-funds-transfer transactions (0 or more)
</RECINTRATRNRS>	
</RECINTRASYNCRS>	

The <RECINTRASYNCRS> responses contain only intrabank transfer models where the <xxxACCTFROM> matches that submitted in the sync request.

11.12.6 Data Synchronization for Recurring Interbank Funds Transfers

11.12.6.1 Request <RECINTERSYNCRQ>

This request will synchronize the client with the server in relation to recurring interbank transfer models. To synchronize individual funds transfers that were created by the model (and perhaps canceled by another client), the client must also issue an <INTERSYNCRQ>.

	Description
<RECINTERSYNCRQ> *Client synchronization option; <TOKEN>, <TOKENONLY>, or <REFRESH>*	Synchronization-request aggregate
<TOKEN>	Previous value of <TOKEN> received for this type of synchronization request from server; 0 for first-time requests; *token*
<TOKENONLY>	Request for just the current <TOKEN> without the history, *Boolean*
<REFRESH>	Request for refresh of current state, *Boolean*
<REJECTIFMISSING> *Account-from options. Choose either <BANKACCTFROM> or <CCACCTFROM> or <LOANACCTFROM>*	If Y, do not process requests if client <TOKEN> is out of date, *Boolean*
<BANKACCTFROM> ***</BANKACCTFROM>*** -or- ***<CCACCTFROM>*** ***</CCACCTFROM>*** -or- ***<LOANACCTFROM>*** ***</LOANACCTFROM>***	Account-from aggregate, see section 11.3.1 Credit-card-account-from aggregate, see section 11.3.2 Loan-account-from aggregate, see section 11.3.7
<RECINTERTRNRQ> </RECINTERTRNRQ>	Recurring-transfer transactions (0 or more)
</RECINTERSYNCRQ>	

11.12.6.2 Response <RECINTERSYNCRS>

	Description
<RECINTERSYNCRS>	Synchronization-response aggregate
<TOKEN>	New synchronization token, *token*
<LOSTSYNC>	Y if the token in the synchronization request is older than the earliest entry in the server's history table. In this case, some responses have been lost. N if the token in the synchronization request is newer than or matches a token in the server's history table. *Boolean*
Account-from options. Choose either <BANKACCTFROM> or <CCACCTFROM> or <LOANACCTFROM>	
<BANKACCTFROM>	Account-from aggregate, see section 11.3.1
</BANKACCTFROM>	
-or-	
<CCACCTFROM>	Credit-card-account-from aggregate, see section 11.3.2
</CCACCTFROM>	
-or-	
<LOANACCTFROM>	Credit-card-account-from aggregate, see section 11.3.7
</LOANACCTFROM>	
<RECINTERTRNRS>	Recurring-interbank-funds-transfer transactions (0 or more)
</RECINTERTRNRS>	
</RECINTERSYNCRS>	

11.12.7 Data Synchronization for Bank Mail

11.12.7.1 Request <BANKMAILSYNCRQ>

	Description
<BANKMAILSYNCRQ>	Synchronization-request aggregate

11.12.7.2 Response <BANKMAILSYNCRS>

	Description
<BANKMAILSYNCRS>	Synchronization-response aggregate
<TOKEN>	New synchronization token, *token*
<LOSTSYNC>	Y if the token in the synchronization request is older than the earliest entry in the server's history table. In this case, some responses have been lost. N if the token in the synchronization request is newer than or matches a token in the server's history table. *Boolean*
Account-from options. Choose either <BANKACCTFROM> or <CCACCTFROM>.	
<BANKACCTFROM>	Account-from aggregate, see section 11.3.1
</BANKACCTFROM>	
-or-	
<CCACCTFROM>	Credit-card-account-from aggregate, see section 11.3.2
</CCACCTFROM>	
<BANKMAILTRNRS>	Bank-mail transactions (0 or more)
</BANKMAILTRNRS>	
</BANKMAILSYNCRS>	

11.12.8 Data Synchronization for Loan Mail

11.12.8.1 Loan Mail Sync Request <LOANMAILSYNCRQ>

	Description
<LOANMAILSYNCRQ> *Client synchronization option; <TOKEN>, <TOKENONLY>, or <REFRESH>*	Synchronization-request aggregate
<TOKEN>	Previous value of <TOKEN> received for this type of synchronization request from server; 0 for first-time requests; *token*
<TOKENONLY>	Request for just the current <TOKEN> without the history, *Boolean*
<REFRESH>	Request for refresh of current state, *Boolean*
<REJECTIFMISSING>	If Y, do not process requests if client <TOKEN> is out of date, *Boolean*
<INCIMAGES>	Y if the client accepts mail with images in the message body. N if the client does not accept mail with images in the message body. *Boolean*
<USEHTML>	Y if client wants an HTML response, N if client wants plain text, *Boolean*
<LOANACCTFROM>	Loan-account-from aggregate, see section 11.3.7
</LOANACCTFROM>	
<LOANMAILTRNRQ>	Loan-mail transactions (0 or more)
</LOANMAILTRNRQ>	
</LOANMAILSYNCRQ>	

11.12.8.2 Loan Mail Sync Response <LOANMAILSYNCRS>

	Description
<LOANMAILSYNCRS>	Synchronization-response aggregate
<TOKEN>	New synchronization token, *token*
<LOSTSYNC>	Y if the token in the synchronization request is older than the earliest entry in the server's history table. In this case, some responses have been lost. N if the token in the synchronization request is newer than or matches a token in the server's history table. *Boolean*
<LOANACCTFROM>	Loan-account-from aggregate, see section 11.3.7
</LOANACCTFROM>	
<LOANMAILTRNRS>	Loan-mail transactions (0 or more)
</LOANMAILTRNRS>	
</LOANMAILSYNCRS>	

11.13

11.13.1 Message Sets and Messages

The following tables show the OFX banking transactions and their corresponding transaction wrappers for a particular message set.

11.13.1.1 Bank Message Set and Messages

11.13.1.1.1 Bank Message Set Request Messages

Message	
<BANKMSGSRQV1>	
	STMTTRNRQ
	STMTRQ
	STMTENDTRNRQ
	STMTENDRQ
	STPCHKTRNRQ
	STPCHKRQ
	INTRATRNRQ
	INTRARQ
	INTRAMODRQ
	INTRACANRQ
	RECINTRATRNRQ
	RECINTRARQ
	RECINTRAMODRQ
	RECINTRACANRQ
	BANKMAILTRNRQ
	BANKMAILRQ
	STPCHKSYNCRQ
	INTRASYNCRQ
	RECINTRASYNCRQ
	BANKMAILSYNCRQ
</BANKMSGSRQV1>	

11.13.1.1.2 Bank Message Set Response Messages

Message	
<BANKMSGSRSV1>	
	STMTTRNRS
	STMTRS
	STMTENDTRNRS
	STMTENDRS
	STPCHKTRNRS
	STPCHKRS
	INTRATRNRS
	INTRARS
	INTRAMODRS
	INTRACANRS
	RECINTRATRNRS
	RECINTRARS
	RECINTRAMODRS
	RECINTRACANRS
	BANKMAILTRNRS
	BANKMAILRS
	CHKMAILRS
	DEPMAILRS
	STPCHKSYNCRS
	INTRASYNCRS
	RECINTRASYNCRS
	BANKMAILSYNCRS
</BANKMSGSRSV1>	

11.13.1.2 Credit Card Message Set and Messages

11.13.1.2.1 Credit Card Message Set Request Messages

Message	
<CREDITCARDMSGSRQV1>	
	CCSTMTTRNRQ
	CCSTMTRQ
	CCSTMTENDTRNRQ
	CCSTMTENDRQ
</CREDITCARDMSGSRQV1>	

11.13.1.2.2 Credit Card Message Set Response Messages

Message	
<CREDITCARDMSGSRSV1>	
	CCSTMTTRNRS
	CCSTMTRS
	CCSTMTENDTRNRS
	CCSTMTENDRS
</CREDITCARDMSGSRSV1>	

11.13.1.3.2 Interbank Transfer Message Set Response Messages

Message	
<INTERXFERMSGSRSV1>	
	INTERTRNRS
	INTERRS
	INTERMODRS
	INTERCANRS
	RECINTERTRNRS
	RECINTERRS
	RECINTERMODRS
	RECINTERCANRS
	INTERSYNCRS
	RECINTERSYNCRS
</INTERXFERMSGSRSV1>	

11.13.1.4 Wire Transfer Message Set and Messages

11.13.1.4.1 Wire Transfer Message Set Request Messages

Message	
<WIREXFERMSGSRQV1>	
	WIRETRNRQ
	WIRERQ
	WIRECANRQ
	WIRESYNCRQ
</WIREXFERMSGSRQV1>	

11.13.1.4.2 Wire Transfer Message Set Response Messages

Message	
<WIREXFERMSGSRSV1>	
	WIRETRNRS
	WIRERS
	WIRECANRS
	WIRESYNCRS
</WIREXFERMSGSRSV1>	

11.13.1.5 Loan Message Set and Messages

11.13.1.5.1 Loan Message Set Request Messages

Message	
<LOANMSGSRQV1>	
	LOANSTMTTRNRQ
	LOANSTMTRQ
	AMRTSTMTTRNRQ
	AMRTSTMTRQ
	LOANSTMTENDTRNRQ
	LOANSTMTENDRQ
	LOANMAILTRNRQ
	LOANMAILRQ
	LOANMAILSYNCRQ
</LOANMSGSRQV1>	

11.13.1.5.2 Loan Message Set Response Messages

Message	
<LOANMSGSRSV1>	
	LOANSTMTTRNRS
	LOANSTMTRS
	AMRTSTMTTRNRS
	AMRTSTMTRS
	LOANSTMTENDTRNRS
	LOANSTMTENDRS
	LOANMAILTRNRS
	LOANMAILRS
	LOANMAILSYNCRS
</LOANMSGSRSV1>	

11.13.2 Bank Message Set Profile

11.13.2.1 <BANKMSGSET>, <BANKMSGSETV1>

	Description
<BANKMSGSET>	Message set for banking
<BANKMSGSETV1>	Version 1 of message set
<MSGSETCORE>	Common message-set core
</MSGSETCORE>	
<INVALIDACCTTYPE>	Account type not supported in <BANKACCTFROM>; 0 or more of account types, see section 11.3.1.1 for values
<CLOSINGAVAIL>	Closing statement information available, *Boolean*
<XFERPROF>	Intrabank transfer profile (if supported), see section 11.13.2.2
</XFERPROF>	
<STPCHKPROF>	Stop check profile (if supported), see section 11.13.2.3
</STPCHKPROF>	
<EMAILPROF>	E-mail profile, see section 11.13.2.4
</EMAILPROF>	
<IMAGEPROF>	Image profile (if supported), see section 3.1.6.2
</IMAGEPROF>	
</BANKMSGSETV1>	End of bank message set version 1
</BANKMSGSET>	

11.13.2.2 Banking Profile, Funds Transfer <XFERPROF>

	Description
<XFERPROF>	Intrabank transfer profile (if supported)
<PROCDAYSOFF>	Days of week that no processing occurs: MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, or SUNDAY. 0 or more <PROCDAYSOFF> can be sent.
<PROCENDTM>	Time of day that day's processing ends, *time*
<CANSCHED>	Supports scheduled transfers, *Boolean*
<CANRECUR>	Supports recurring transfers, *Boolean*. Requires <CANSCHED>
<CANLOAN>	Supports loan transfers. <CANLOAN>Y must be present for transfers to involve loans, *Boolean*
<CANSCHEDLOAN>	Supports scheduled transfers of loans (requires <CANLOAN>Y), *Boolean*
<CANRECURLOAN>	Supports recurring transfers of loans (requires <CANSCHEDLOAN>Y), *Boolean*
<CANMODXFERS>	Permit modifications to transfers, i.e. <INTRAMODRQ>, *Boolean*
<CANMODMDLS>	Permit modifications to models, i.e. <RECINTRAMODRQ>, *Boolean*
<MODELWND>	Model window; the number of days before a recurring transaction is scheduled to be processed that it is instantiated on the system, *N-3*
<DAYSWITH>	Number of days before processing date that funds are withdrawn, *N-3*
<DFLTDAYSTOPAY>	Default number of days to pay, *N-3*
</XFERPROF>	

11.13.2.3 Banking Profile, Stop Checks <STPCHKPROF>

	Description
<STPCHKPROF>	Stop check profile (if supported)
<PROCDAYSOFF>	Days of week that no processing occurs: MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, or SUNDAY. 0 or more <PROCDAYSOFF> can be sent.
<PROCENDTM>	Time of day that day's processing ends, *time*
<CANUSERANGE>	Can stop a range of checks, *Boolean.*
<CANUSEDESC>	Can stop by descripTm,

11.13.2.4 Banking Profile, Email <EMAILPROF>

	Description
<EMAILPROF>	E-mail profile
<CANEMAIL>	Supports generalized banking e-mail, *Boolean*
<CANNOTIFY>	Server may return DEPMAILRS or CHKMAILRS in synchronization responses. See section 11.11.3 , *Boolean*
</EMAILPROF>	

11.13.3 Credit Card Message Set Profile

	Description
<CREDITCARDMSGSET>	Beginning tag for credit card message set
<CREDITCARDMSGSETV1>	Version 1 of message set
<MSGSETCORE>	Common message-set core
</MSGSETCORE>	
<CLOSINGAVAIL>	Closing statement information available, *Boolean*
<IMAGEPROF>	Image profile (if supported), see section 3.1.6.2
</IMAGEPROF>	
</CREDITCARDMSGSETV1>	Ending tag of credit card message set version 1
</CREDITCARDMSGSET>	Ending tag of credit card message set

11.13.4 Interbank Funds Transfer Message Set Profile

	Description
<INTERXFERMSGSET>	Beginning tag for interbank transfers message set
<INTERXFERMSGSETV1>	Version 1 of message set
<MSGSETCORE>	Common message-set core
</MSGSETCORE>	
<XFERPROF>	Interbank transfer profile, same as XFERPROF in banking, see section 11.13.2.2
</XFERPROF>	
<CANBILLPAY>	Server is capable of handling bill payment as a form of transfers, *Boolean*
<CANCELWND>	Number of days after an interbank transfer occurs that it can be canceled, *N-3*
<DOMXFERFEE>	Standard fee for a domestic interbank transfer, *amount*
<INTLXFERFEE>	Standard fee for an international interbank transfer, *amount*
</INTERXFERMSGSETV1>	End of interbank transfer message set version 1
</INTERXFERMSGSET>	End of interbank transfer message set

11.13.5 Wire Transfer Message Set Profile

	Description
<WIREXFERMSGSET>	Core message set for wire transfers
<WIREXFERMSGSETV1>	Version 1 of message set
<MSGSETCORE>	Common message-set core
</MSGSETCORE>	
<PROCDAYSOFF>	Days of week that no processing occurs: MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, or SUNDAY. 0 or more <PROCDAYSOFF> can be sent.
<PROCENDTM>	Time of day that day's processing ends, *time*
<CANSCHED>	Supports scheduled transfers, *Boolean*
<DOMXFERFEE>	Standard fee for a domestic wire transfer, *amount*
<INTLXFERFEE>	Standard fee for an international wire transfer, *amount*
</WIREXFERMSGSETV1>	End of wire transfer message set version 1
</WIREXFERMSGSET>	Ending tag of wire transfer message set

11.13.6 Loan Message Set Profile

	Description
<LOANMSGSET>	Core message set for Loans
<LOANMSGSETV1>	Version 1 of message set
<MSGSETCORE>	Common message-set core
</MSGSETCORE>	
<TRANDNLD>	Loan statement transactions available, *Boolean*
<CLOSINGAVAIL>	Closing statement information available, *Boolean*
<AMRTAVAIL>	Amortization statement information available, *Boolean*
<CANEMAIL>	Supports generalized loan e-mail, *Boolean*
<IMAGEPROF>	Image profile (if supported), see section 3.1.6.2
</IMAGEPROF>	
</LOANMSGSETV1>	End of Loan message set version 1
</LOANMSGSET>	Ending tag of Loan message set

11.14 Examples

11.14.1 Statement Download

This example represents a customer who requests a statement download for a checking account. The request omits <DTSTART> and <DTEND> because the client is interested in getting all available data. The response contains an updated balance for the account and two transactions.

The request file:

```
<OFX>                                    <!-- Begin request data -->
  <SIGNONMSGSRQV1>
    <SONRQ>                              <!-- Begin signon -->
      <DTCLIENT>20051029101000</DTCLIENT><!-- Oct. 29, 2005, 10:10:00
am -->
      <USERID>MyUserID</USERID>          <!-- User ID -->
      <USERPASS>MyPassword</USERPASS>    <!--Password(SSLencrypts
whole) -->
      <LANGUAGE>ENG</LANGUAGE>  <!-- Language used for text -->
      <FI>                      <!-- ID of receiving institution -->
        <ORG>NCH</ORG>          <!-- Name of ID owner -->
        <FID>1001</FID>         <!-- Actual ID -->
      </FI>
      <APPID>MyApp</APPID>
      <APPVER>0500</APPVER>
    </SONRQ>                    <!-- End of signon -->
  </SIGNONMSGSRQV1>

  <BANKMSGSRQV1>
    <STMTTRNRQ>                 <!-- Begin request -->
      <TRNUID>1001</TRNUID>
      <STMTRQ>                  <!-- Begin statement request -->
        <BANKACCTFROM>          <!-- Identify the account -->
          <BANKID>121099999</BANKID><!-- Routing transit or other
                                    FI ID -->
          <ACCTID>999988</ACCTID><!-- Account number -->
          <ACCTTYPE>CHECKING</ACCTTYPE><!-- Account type -->
        </BANKACCTFROM>         <!-- End of account ID -->
        <INCTRAN>               <!-- Begin include transaction -->
          <INCLUDE>Y</INCLUDE>  <!-- Include transactions -->
        </INCTRAN>              <!-- End of include transaction -->
      </STMTRQ>                 <!-- End of statement request -->
    </STMTTRNRQ>                <!-- End request -->
```

```
  </BANKMSGSRQV1>
</OFX>                          <!-- End of request data -->
```

The response file:

```
<OFX>                           <!-- Begin response data -->
  <SIGNONMSGSRSV1>
    <SONRS>                     <!-- Begin signon -->
      <STATUS>                  <!-- Begin status aggregate -->
        <CODE>0</CODE>          <!-- OK -->
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <DTSERVER>20051029101003</DTSERVER><!-- Oct. 29, 2005, 10:10:03
am -->
      <LANGUAGE>ENG</LANGUAGE>          <!-- Language used in response
-->
      <DTPROFUP>19991029101003</DTPROFUP><!-- Last update to profile--
>
      <DTACCTUP>20031029101003</DTACCTUP><!-- Last account update -->
      <FI>                      <!-- ID of receiving institution -->
        <ORG>NCH</ORG>          <!-- Name of ID owner -->
        <FID>1001</FID>         <!-- Actual ID -->
      </FI>
    </SONRS>                    <!-- End of signon -->
  </SIGNONMSGSRSV1>

  <BANKMSGSRSV1>
    <STMTTRNRS>                 <!-- Begin response -->
      <TRNUID>1001</TRNUID>     <!-- Client ID sent in request -->
      <STATUS>                  <!-- Start status aggregate -->
        <CODE>0</CODE>          <!-- OK -->
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <STMTRS>                  <!-- Begin statement response -->
        <CURDEF>USD</CURDEF>
        <BANKACCTFROM>          <!-- Identify the account -->
```

```
          <DTEND>20051028</DTEND><!-- End date: Oct. 28, 2005 -->
          <STMTTRN>            <!-- First statement transaction -->
            <TRNTYPE>CHECK</TRNTYPE><!--Check -->
            <DTPOSTED>20051004</DTPOSTED><!-- Posted on Oct. 4, 2005
-->
            <TRNAMT>-200.00</TRNAMT><!-- $200.00 -->
            <FITID>00002</FITID><!-- Unique ID -->
            <CHECKNUM>1000</CHECKNUM><!-- Check number -->
          </STMTTRN>           <!-- End statement transaction -->
          <STMTTRN>            <!-- Second transaction -->
            <TRNTYPE>ATM</TRNTYPE><!-- ATM transaction -->
            <DTPOSTED>20051020</DTPOSTED><!-- Posted on Oct. 20, 2005
-->
            <DTUSER>20051020</DTUSER><!-- User date of Oct. 20, 2005 -
->
            <TRNAMT>-300.00</TRNAMT><!-- $300.00 -->
            <FITID>00003</FITID><!-- Unique ID -->
          </STMTTRN>           <!-- End statement transaction -->
        </BANKTRANLIST>        <!-- End list of statement trans. -->
        <LEDGERBAL>            <!-- Ledger balance aggregate -->
          <BALAMT>200.29</BALAMT><!-- Bal amount: $200.29 -->
          <DTASOF>200510291120</DTASOF><!-- Bal date: 10/29/05, 11:20
am -->
        </LEDGERBAL>           <!-- End ledger balance -->
        <AVAILBAL>             <!-- Available balance aggregate -->
          <BALAMT>200.29</BALAMT><!-- Bal amount: $200.29 -->
          <DTASOF>200510291120</DTASOF><!-- Bal date: 10/29/05, 11:20
am -->
        </AVAILBAL>            <!-- End available balance -->
      </STMTRS>                <!-- End statement response -->
    </STMTTRNRS>               <!-- End of transaction -->
  </BANKMSGSRSV1>
</OFX>                         <!-- End of response data -->
```

11.14.2 Intrabank Funds Transfer

This example is for a customer who requests an immediate funds transfer of $200.00 from a checking account to a savings account.

The request file:

```
<OFX>                          <!-- Begin request data -->
  <SIGNONMSGSRQV1>
```

```
    <SONRQ>                     <!-- ...Sign on request.
                                For a complete example,
                                see section 11.14.1-->
    </SONRQ>                    <!-- End of signon -->
  </SIGNONMSGSRQV1>

  <BANKMSGSRQV1>
    <INTRATRNRQ>                <!-- Begin request -->
      <TRNUID>1001</TRNUID>     <!-- Client's ID for this request -->
      <INTRARQ>                 <!-- Begin transfer request -->
        <XFERINFO>              <!-- Begin transfer aggregate -->
          <BANKACCTFROM>        <!-- Identify the account -->
            <BANKID>121099999</BANKID><!-- Routing transit or other
                                FI ID -->
            <ACCTID>999988</ACCTID><!-- Account number -->
            <ACCTTYPE>CHECKING</ACCTTYPE><!-- Account type -->
          </BANKACCTFROM>       <!-- End of account ID -->
          <BANKACCTTO>          <!-- Identify the account -->
            <BANKID>121099999</BANKID><!-- Routing transit or other
                                FI ID -->
            <ACCTID>999977</ACCTID><!-- Account number -->
            <ACCTTYPE>SAVINGS</ACCTTYPE><!-- Account type -->
          </BANKACCTTO>         <!-- End of account ID -->
          <TRNAMT>200.00</TRNAMT><!-- Amount of transfer -->
        </XFERINFO>             <!-- End of transfer aggregate -->
      </INTRARQ>                <!-- End of transfer request -->
    </INTRATRNRQ>               <!-- End request -->
  </BANKMSGSRQV1>
</OFX>                          <!-- End of request data -->
```

The response file:

```
<OFX>                           <!-- Begin response data -->
  <SIGNONMSGSRSV1>
    <SONRS>                     <!-- ...Sign on response.
                                For a complete example,
                                see section 11.14.1-->
    </SONRS>                    <!-- End of signon -->
  </SIGNONMSGSRSV1>


 <BANKMSGSRSV1>
    <INTRATRNRS>                <!-- Begin response -->
      <TRNUID>1001</TRNUID>     <!-- Client ID sent in request -->
```

```
      <STATUS>                    <!-- Start status aggregate -->
        <CODE>0<CODE>             <!-- OK -->
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <INTRARS>                   <!-- Begin transfer response -->
        <CURDEF>USD</CURDEF>
        <SRVRTID>1001</SRVRTID> <!-- Server assigned ID -->
        <XFERINFO>                <!-- Begin transfer aggregate -->
          <BANKACCTFROM>          <!-- Identify the account -->
            <BANKID>121099999</BANKID><!-- Routing transit or other
                                    FI ID -->
            <ACCTID>999988</ACCTID><!-- Account number -->
            <ACCTTYPE>CHECKING</ACCTTYPE><!-- Account type -->
          </BANKACCTFROM>         <!-- End of account ID -->
          <BANKACCTTO>            <!-- Identify the account -->
            <BANKID>121099999</BANKID><!-- Routing transit or other
                                    FI ID -->
            <ACCTID>999977</ACCTID><!-- Account number -->
            <ACCTTYPE>SAVINGS</ACCTTYPE><!-- Account type -->
          </BANKACCTTO>           <!-- End of account ID -->
          <TRNAMT>200.00</TRNAMT><!-- Amount of transfer -->
        </XFERINFO>               <!-- End of transfer aggregate -->
        <DTXFERPRJ>20050829100000</DTXFERPRJ><!-- Projected posting
date -->
      </INTRARS>                  <!-- End of transfer response -->
    </INTRATRNRS>                 <!-- End response -->
  </BANKMSGSRSV1>
</OFX>                            <!-- End of response data -->
```

11.14.3 Stop Check

This example represents a customer who requests a stop for checks 200 through 202. The response indicates that the first check (200) has already posted; the server has stopped the rest of the checks in the range.

The request file:

```
<OFX>                             <!-- Begin request data -->
  <SIGNONMSGSRQV1>
    <SONRQ>                       <!-- ...Sign on request.
                                  For a complete example,
                                  see section 11.14.1-->
```

```
  <BANKMSGSRQV1>
    <STPCHKTRNRQ>                <!-- Begin request -->
      <TRNUID>1001</TRNUID>      <!-- Client's ID for this request -->
      <STPCHKRQ>                 <!-- Begin stop check request -->
        <BANKACCTFROM>           <!-- Identify the account -->
          <BANKID>121099999</BANKID><!-- Routing transit or other
                                       FI ID -->
          <ACCTID>999988</ACCTID><!-- Account number -->
          <ACCTTYPE>CHECKING</ACCTTYPE><!-- Account type -->
        </BANKACCTFROM>          <!-- End of account ID -->
        <CHKRANGE>               <!-- Cancel a range of checks -->
          <CHKNUMSTART>200</CHKNUMSTART><!-- Starting check number -->
          <CHKNUMEND>202</CHKNUMEND><!-- Ending check number -->
        </CHKRANGE>              <!-- End range -->
      </STPCHKRQ>                <!-- End of stop check request -->
    </STPCHKTRNRQ>               <!-- End request -->
  </BANKMSGSRQV1>
</OFX>                           <!-- End of request data -->
```

The response file:

```
<OFX>                            <!-- Begin response data -->
  <SIGNONMSGSRSV1>
    <SONRS>                      <!-- ...Sign on response.
                                 For a complete example,
                                 see section 11.14.1-->
    </SONRS>                     <!-- End of signon -->
  </SIGNONMSGSRSV1>

  <BANKMSGSRSV1>
    <STPCHKTRNRS>                <!-- Begin response -->
      <TRNUID>1001</TRNUID>      <!-- Client ID sent in request -->
      <STATUS>                   <!-- Begin status aggregate -->
        <CODE>0<CODE>            <!-- OK -->
        <SEVERITY>INFO</SEVERITY>
      </STATUS>                  <!-- End of status aggregate -->
      <STPCHKRS>                 <!-- Begin stop check response -->
        <CURDEF>USD</CURDEF>
        <BANKACCTFROM>           <!-- Identify the account -->
          <BANKID>121099999</BANKID><!-- Routing transit or other
                                       FI ID -->
          <ACCTID>999988</ACCTID><!-- Account number -->
          <ACCTTYPE>CHECKING</ACCTTYPE><!-- Account type -->
```

```
        </BANKACCTFROM>         <!-- End of account ID -->
        <STPCHKNUM>             <!-- First stopped check -->
          <CHECKNUM>200</CHECKNUM><!-- Check 200 -->
          <CHKSTATUS>101</CHKSTATUS><!-- Too late - already posted -->
        </STPCHKNUM>            <!-- End of first stopped check -->
        <STPCHKNUM>             <!-- Second stopped check -->
          <CHECKNUM>201</CHECKNUM><!-- Check 201 -->
          <CHKSTATUS>0</CHKSTATUS><!-- OK -->
        </STPCHKNUM>            <!-- End of second stopped check -->
        <STPCHKNUM>             <!-- Third stopped check -->
          <CHECKNUM>202</CHECKNUM><!-- Check 202 -->
          <CHKSTATUS>0</CHKSTATUS><!-- OK -->
        </STPCHKNUM>            <!-- End of third stopped check -->
        <FEE>10.00</FEE>
        <FEEMSG>Fee for stop payment</FEEMST>
      </STPCHKRS>               <!-- End stop check response -->
    </STPCHKTRNRS>              <!-- End of transaction -->
  </BANKMSGSRSV1>
</OFX>                          <!-- End of response data -->
```

```
            <DTDUE>20051115</DTDUE><!-- First transfer - Nov.15 -->
          </XFERINFO>           <!-- End transfer aggregate -->
          </INTRARQ>
      </RECINTRARQ>             <!-- End transfer request -->
    </RECINTRATRNRQ>            <!-- End request -->
  </BANKMSGSRQV1>
</OFX>
```

The response file shows that the model has been successfully created:

```
<OFX>                           <!-- Begin response data -->
  <SIGNONMSGSRSV1>
    <SONRS>                     <!-- ...Sign on response.
                                For a complete example,
                                see section 11.14.1-->
    </SONRS>                    <!-- End of signon -->
  </SIGNONMSGSRSV1>

  <BANKMSGSRSV1>
    <RECINTRATRNRS>             <!-- Begin response -->
      <TRNUID>1001</TRNUID>     <!-- Client ID sent in request -->
      <STATUS>                  <!-- Start of status aggregate -->
        <CODE>0</CODE>          <!-- OK -->
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <RECINTRARS>              <!-- Begin response -->
        <RECSRVRTID>20000</RECSRVRTID><!-- Server assigned ID -->
        <RECURRINST>            <!-- Begin recurring aggregate -->
          <FREQ>MONTHLY</FREQ>  <!-- Monthly schedule -->
        </RECURRINST>           <!-- End of recurring aggregate -->
          <INTRARS>
          <CURDEF>USD</CURDEF?
          <SRVRTID>20000</SRVRTID>
          <XFERINFO>            <!-- Begin transfer aggregate -->
            <BANKACCTFROM>      <!-- Identify the account -->
            <BANKID>121099999</BANKID><!-- Routing transit or other
                                        FI ID -->
            <ACCTID>999988</ACCTID><!-- Account number -->
            <ACCTTYPE>CHECKING</ACCTTYPE><!-- Account type -->
            </BANKACCTFROM>     <!-- End of account ID -->
            <BANKACCTTO>        <!-- Identify the account -->
            <BANKID>121099999</BANKID><!-- Routing transit or other
                                FI ID -->
            <ACCTID>999977</ACCTID><!-- Account number -->
```

```
            <ACCTTYPE>SAVINGS</ACCTTYPE><!-- Account type -->
            </BANKACCTTO>      <!-- End of account ID -->
            <TRNAMT>1000.00</TRNAMT><!-- Amount of transfer -->
            <DTDUE>20051115</DTDUE><!-- First transfer - Nov. 15 -->
          </XFERINFO>          <!-- End of transfer aggregate -->
        </INTRARS>
      </RECINTRARS>            <!-- End of response -->
    </RECINTRATRNRS>           <!-- End of response -->
  </BANKMSGSRSV1>
</OFX>                         <!-- End of response data -->
```

The client sends the payment synchronization request later on November 1:

```
<OFX>                          <!-- Begin request data -->
  <SIGNONMSGSRQV1>
    <SONRQ>                    <!-- ...Sign on request.
                               For a complete example,
                               see section 11.14.1-->
    </SONRQ>                   <!-- End of signon -->
  </SIGNONMSGSRQV1>
  <BANKMSGSRQV1>
    <INTRASYNCRQ>              <!-- Sync intrabank transfers -->
      <TOKEN>0</TOKEN>         <!-- Token held by client -->
      <REJECTIFMISSING>N</REJECTIFMISSING>
          <BANKACCTFROM>       <!-- Identify the account -->
            <BANKID>121099999</BANKID>
            <ACCTID>999988</ACCTID>
            <ACCTTYPE>CHECKING<ACCTTYPE>
          </BANKACCTFROM>
    </INTRASYNCRQ>             <!-- End of sync request -->
  </BANKMSGSRQV1>
</OFX>                         <!-- End of request data -->
```

Assuming that the server creates transfers 30 days prior to posting, the server returns status for one pending transfer. This response comes back since the first transfer is scheduled to occur on November 15 and this date falls within 30 days of our session. Had the starting date been more than 30 days from our signon date, the response would not have contained any pending transfers since the model would not have generated any yet.

The response file from the server shows one pending transfer:

```
<OFX>                          <!-- Begin response data -->
  <SIGNONMSGSRSV1>
    <SONRQ>                    <!-- ...Sign on request.
                               For a complete example,
                               see section 11.14.1-->
    </SONRQ>                   <!-- End of signon -->
  </SIGNONMSGSRSV1>
  <BANKMSGSRSV1>
    <INTRASYNCRS>              <!-- Sync intrabank transfers -->
      <TOKEN>22243</TOKEN>     <!-- Token updated -->
        <BANKACCTFROM>         <!-- Identify the account -->
          <BANKID>121099999</BANKID>
          <ACCTID>999988</ACCTID>
          <ACCTTYPE>CHECKING</ACCTTYPE>
        </BANKACCTFROM>
        <INTRATRNRS>           <!-- begin response -->
          <TRNUID>0</TRNUID>   <!-- Server generated, so 0-->
          <STATUS>             <!-- Success -->
            <CODE>0</CODE>     <!-- OK -->
            <SEVERITY>INFO</SEVERITY>
          </STATUS>
          <INTRARS>            <!-- Begin transfer response -->
            <CURDEF>USD</CURDEF>
            <SRVRTID>100100000</SRVRTID> <!-- Server assigned ID -->
            <XFERINFO>         <!-- Begin transfer aggregate -->
            <BANKACCTFROM>     <!-- Identify the account -->
              <BANKID>121099999</BANKID> <!-- Routing transit or
                                  other FI ID -->
              <ACCTID>999988</ACCTID> <!-- Account number -->
              <ACCTTYPE>CHECKING</ACCTTYPE><!-- Account type -->
            </BANKACCTFROM>    <!-- End of account ID -->
            <BANKACCTTO>       <!-- Identify the account -->
              <BANKID>121099999</BANKID> <!-- Routing transit or
                                  other FI ID -->
              <ACCTID>999977</ACCTID> <!-- Account number -->
              <ACCTTYPE>SAVINGS</ACCTTYPE> <!-- Account type -->
            </BANKACCTTO>      <!-- End of account ID -->
            <TRNAMT>1000.00</TRNAMT> <!-- Amount of transfer -->
            </XFERINFO>        <!-- End transfer aggregate -->
          <DTXFERPRJ>20051115</DTXFERPRJ> <!-- Projected date of the
                                  transfer -->
```

```
          <RECSRVRTID>20000</RECSRVRTID> <!-- Model that created this
xfer -->
          </INTRARS>            <!-- End of transfer response -->
        </INTRATRNRS>           <!-- End of response -->
      </INTRASYNCRS>            <!-- End of sync response -->
  </BANKMSGSRSV1>
</OFX>                          <!-- End of response data -->
```

Suppose the customer does not attempt to connect between November 16 and January 1. When the customer does attempt to connect, it is to cancel the recurring transfer model. The client also sets the <CANPENDING> flag, causing any pending transfers to be immediately cancelled as well. In order to get all synchronization information (since requests are not guaranteed to be executed in order), the client sends two request files, the first to cancel the model and the next to retrieve all transfer activity. This time, the recurring request is wrapped in synchronization wrappers. It should be assumed that the token below was received in a previous RECPMTSYNCRS. (The use of synchronization wrappers in requests is entirely up to the client. Both ways are shown here for explanatory purposes.)

The request file:

```
<OFX>                              <!-- Begin request data -->
  <SIGNONMSGSRQV1>
    <SONRQ>                        <!-- ...Sign on request.
                                   For a complete example,
                                   see section 11.14.1-->
    </SONRQ>                       <!-- End of signon -->
  </SIGNONMSGSRQV1>

  <BANKMSGSRQV1>
    <RECINTRASYNCRQ>               <!-- Sync recurring transfers -->
      <TOKEN>324789987</TOKEN>     <!-- Token held by the client -->
      <REJECTIFMISSING>Y</REJECTIFMISSING><!-- Cancel only if up to
date -->
      <BANKACCTFROM>
        <BANKID>121099999</BANKID>
        <ACCTID>99998</ACCTID>
        <ACCTTYPE>CHECKING</ACCTTYPE>
      </BANKACCTFROM>
      <RECINTRATRNRQ>              <!-- Begin request -->
        <TRNUID>1005</TRNUID>      <!-- Client's ID for this request -->
        <RECINTRACANRQ>            <!-- Begin recur transfer cancel -->
          <RECSRVRTID>20000</RECSRVRTID><!-- ID of the model -->
          <CANPENDING>Y</CANPENDING><!-- Cancel pending transfers -->
        </RECINTRACANRQ>           <!-- End request -->
      </RECINTRATRNRQ>             <!-- End request -->
    </RECINTRASYNCRQ>              <!-- End of sync request -->
  </BANKMSGSRQV1>
</OFX>                             <!-- End of request data -->
```

The response file:

```
<OFX>                              <!-- Begin response data -->
  <SIGNONMSGSRSV1>
    <SONRS>                        <!-- ...Sign on response.
                                   For a complete example,
                                   see section 11.14.1-->
    </SONRS>                       <!-- End of signon -->
  </SIGNONMSGSRSV1>

  <BANKMSGSRSV1>
    <RECINTRASYNCRS>               <!-- Sync response -->
      <TOKEN>324789988</TOKEN      <!-- New token -->
```

```
      <BANKACCTFROM>
        <BANKID>121099999</BANKID>
        <ACCTID>99998</ACCTID>
        <ACCTTYPE>CHECKING</ACCTTYPE>
      </BANKACCTFROM>
      <RECINTRATRNRS>           <!-- Begin response -->
        <TRNUID>1005</TRNUID>   <!-- Client ID sent in request -->
        <STATUS>                <!-- Start of status aggregate -->
          <CODE>0</CODE>        <!-- OK -->
          <SEVERITY>INFO</SEVERITY>
        </STATUS>
        <RECINTRACANRS>         <!-- Begin cancel model -->
          <RECSRVRTID>20000</RECSRVRTID><!-- Model that was canceled -
->
          <CANPENDING>Y</CANPENDING>
        </RECINTRACANRS>        <!-- End of cancel model -->
      </RECINTRATRNRS>          <!-- End response -->
    </RECINTRASYNCRS>           <!-- End sync response -->
  </BANKMSGSRSV1>
</OFX>                          <!-- End response -->
```

Next request file:

```
<OFX>                           <!-- Begin request data -->
  <SIGNONMSGSRQV1>
    <SONRQ>                     <!-- ...Sign on request.
                                For a complete example,
                                see section 11.14.1-->
    </SONRQ>                    <!-- End of signon -->
  </SIGNONMSGSRQV1>
  <BANKMSGSRQV1>
    <INTRASYNCRQ>               <!-- Sync intrabank transfers -->
      <TOKEN>22243</TOKEN>      <!-- Token held by the client -->
      <REJECTIFMISSING>N</REJECTIFMISSING>
      <BANKACCTFROM>
        <BANKID>121099999</BANKID>
        <ACCTID>99998</ACCTID>
        <ACCTTYPE>CHECKING</ACCTTYPE>
      </BANKACCTFROM>
    </INTRASYNCRQ>              <!-- End of sync request -->
  </BANKMSGSRQV1>
</OFX>                          <!-- End of request data -->
```

Since the customer last connected, the November 15 transfer has posted, the December 15 transfer has been scheduled, the December 15 transfer has posted and a transfer has been scheduled for January 15. The response file shows these four transfer responses and the cancellation response for the January 15 transfer. Note that servers are not required to show the post of transfers via a transfer modification response in the sync. Alternatively, a client may need to note that the transfer happened in a subsequent statement download.

The response file:

```
<OFX>                          <!-- Begin response data -->
  <SIGNONMSGSRSV1>
    <SONRS>                    <!-- ...Sign on response.
                               For a complete example,
                               see section 11.14.1-->
    </SONRS>                   <!-- End of signon -->
  </SIGNONMSGSRSV1>
  <BANKMSGSRSV1>
    <INTRASYNCRS>
      <TOKEN>22244</TOKEN>     <!-- New token -->
      <BANKACCTFROM>
        <BANKID>121099999</BANKID>
        <ACCTID>99998</ACCTID>
        <ACCTTYPE>CHECKING</ACCTTYPE>
      </BANKACCTFROM>
      <INTRATRNRS>             <!-- 11/15 post -->
        <TRNUID>0</TRNUID>     <!-- Server generated, so 0 -->
        <STATUS>
          <CODE>0</CODE>       <!-- OK -->
          <SEVERITY>INFO
        </STATUS>
        <INTRAMODRS>           <!-- This is the Nov. 15 post -->
          <SRVRTID>100100000</SRVRTID><!-- Server assigned ID -->
          <XFERINFO>           <!-- Begin transfer aggregate -->
          <BANKACCTFROM>       <!-- Identify the account -->
            <BANKID>121099999</BANKID><!-- Routing transit or other
                               FI ID -->
            <ACCTID>999988</ACCTID><!-- Account number -->
            <ACCTTYPE>CHECKING</ACCTTYPE><!-- Account type -->
            </BANKACCTFROM>    <!-- End of account ID -->
            <BANKACCTTO>       <!-- Identify the account -->
            <BANKID>121099999</BANKID><!-- Routing transit or other
                               FI ID -->
            <ACCTID>999977</ACCTID><!-- Account number -->
            <ACCTTYPE>SAVINGS</ACCTTYPE><!-- Account type -->
```

```
            </BANKACCTTO>      <!-- End of account ID -->
            <TRNAMT>1000.00</TRNAMT><!-- Amount of transfer -->
          </XFERINFO>          <!-- End of transfer aggregate -->
          <XFERPRCSTS>         <!-- Status of transfer -->
            <XFERPRCCODE>POSTEDON</XFERPRCCODE><!-- Status code -->
            <DTXFERPRC>20051115</DTXFERPRC<!-- Date transfer was
posted -->
          </XFERPRCSTS>        <!-- End of transfer status -->
      </INTRAMODRS>            <!-- End of Nov. 15 post -->
    </INTRATRNRS>              <!-- End of response -->


    <INTRATRNRS>               <!--12/15 pending transfer -->
      <TRNUID>0</TRNUID>
      <STATUS>                 <!-- Success -->
        <CODE>0</CODE>         <!-- OK -->
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <INTRARS>                <!-- This is the Dec. 15 pending -->
        <CURDEF>USD</CURDEF>
        <SRVRTID>112233</SRVRTID> <!-- Server assigned ID -->
        <XFERINFO>             <!-- Begin transfer aggregate -->
          <BANKACCTFROM>       <!-- Identify the account -->
            <BANKID>121099999</BANKID><!-- Routing transit or other
                                 FI ID -->
            <ACCTID>999988</ACCTID> <!-- Account number -->
            <ACCTTYPE>CHECKING</ACCTTYPE><!-- Account type -->
          </BANKACCTFROM>      <!-- End of account ID -->
          <BANKACCTTO>         <!-- Identify the account -->
            <BANKID>121099999</BANKID><!-- Routing transit or other
                                 FI ID -->
            <ACCTID>999977</ACCTID> <!-- Account number -->
            <ACCTTYPE>SAVINGS</ACCTTYPE><!-- Account type -->
          </BANKACCTTO>        <!-- End of account ID -->
          <TRNAMT>1000.00</TRNAMT> <!-- Amount of transfer -->
        </XFERINFO>             <!-- End of transfer aggregate -->
        <DTXFERPRJ>20051215</DTXFERPRJ> <!-- Projected date of the
                                 transfer -->
        <RECSRVRTID>20000</RECSRVRTID> <!-- Model -->
      </INTRARS>               <!-- End of Dec. 15 pending -->
    </INTRATRNRS>              <!-- End response -->
    <INTRATRNRS>               <!-- 12/15 post -->
      <TRNUID>0</TRNUID>       <!-- Client ID sent in request -->
```

```
      <STATUS>
        <CODE>0</CODE>          <!-- OK -->
        <SEVERITY>INFO
      </STATUS>
      <INTRAMODRS>              <!-- This is the Dec. 15 post -->
        <SRVRTID>112233</SRVRTID><!-- Server assigned ID -->
        <XFERINFO>              <!-- Begin transfer aggregate -->
          <BANKACCTFROM>        <!-- Identify the account -->
            <BANKID>121099999</BANKID><!-- Routing transit or other
                                FI ID -->
            <ACCTID>999988</ACCTID><!-- Account number -->
            <ACCTTYPE>CHECKING</ACCTTYPE><!-- Account type -->
          </BANKACCTFROM>       <!-- End of account ID -->
          <BANKACCTTO>          <!-- Identify the account -->
            <BANKID>121099999</ACCTTYPE><!-- Routing transit or other
                                FI ID -->
            <ACCTID>999977</ACCTID><!-- Account number -->
            <ACCTTYPE>SAVINGS</ACCTTYPE><!-- Account type -->
          </BANKACCTTO>         <!-- End of account ID -->
          <TRNAMT>1000.00</TRNAMT><!-- Amount of transfer -->
        </XFERINFO>             <!-- End of transfer aggregate -->
        <XFERPRCSTS>            <!-- Status of transfer -->
          <XFERPRCCODE>POSTEDON</XFERPRCCODE><!-- Status code -->
          <DTXFERPRC>20051215</DTXFERPRC><!-- Date transfer was posted
-->
        </XFERPRCSTS>           <!-- End of transfer status -->
      </INTRAMODRS>             <!-- End of Dec. 15 post -->
    </INTRATRNRS>               <!-- End of response -->

    <INTRATRNRS>                <!—This is the 1/15 pending -->
      <TRNUID>0</TRNUID>        <!-- Client ID sent in request -->
      <STATUS>
        <CODE>0</CODE>          <!-- OK -->
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <INTRARS>                 <!-- This is the Jan. 15 pending -->
        <CURDEF>USD</CURDEF>
        <SRVRTID>112255</SRVRTID><!-- Server assigned ID -->
        <XFERINFO>              <!-- Begin transfer aggregate -->
          <BANKACCTFROM>        <!-- Identify the account -->
            <BANKID>121099999</BANKID><!-- Routing transit or other
                                FI ID -->
            <ACCTID>999988</ACCTID><!-- Account number -->
```

```
            <ACCTTYPE>CHECKING</ACCTTYPE><!-- Account type -->
          </BANKACCTFROM>       <!-- End of account ID -->
          <BANKACCTTO>          <!-- Identify the account -->
            <BANKID>121099999</BANKID><!-- Routing transit or other
                                FI ID -->
            <ACCTID>999977</ACCTID><!-- Account number -->
            <ACCTTYPE>SAVINGS</ACCTTYPE><!-- Account type -->
          </BANKACCTTO>         <!-- End of account ID -->
          <TRNAMT>1000.00</TRNAMT><!-- Amount of transfer -->
        </XFERINFO>             <!-- End of transfer aggregate -->
        <DTXFERPRJ>20050115</DTXFERPRJ><!-- Projected date of transfer
-->
        <RECSRVRTID>20000</RECSRVRTID><!-- Model -->
      </INTRARS>                <!-- End of Jan. 15 pending -->
    </INTRATRNRS>               <!-- Cancellation of 1/15 pending-->
    <INTRATRNRS>                <!-- response -->
        <TRNUID>0</TRNUID>      <!-- Client ID sent in this
                                request -->
        <STATUS>
          <CODE>0</CODE>        <!-- OK -->
          <SEVERITY>INFO</SEVERITY>
        </STATUS>
        <INTRACANRS>            <!-- This is the Jan. 15 cancel -->
          <SRVRTID>112255</SRVRTID> <!-- Server ID for Jan. 15 xfer --
>
        </INTRACANRS>           <!-- End of Jan. 15 cancel -->
      </INTRATRNRS>             <!-- End of response -->
    </INTRASYNCRS>              <!-- End of sync response -->
  </BANKMSGSRSV1>
</OFX>                          <!-- End of response -->
```

11.14.5 Loans

The following are OFX fragments; the missing OFX is covered in other request/response examples.

- **LOAN ACCOUNT INFORMATION**

```
<LOANACCTINFO>
  <LOANACCTFROM>
    <LOANACCTID>123456789</LOANACCTID>
    <LOANACCTTYPE>CONSUMER</LOANACCTTYPE>
  </LOANACCTFROM>
  <LOANTYPE>FIXED</LOANTYPE>
  <LOANINITNUMPMTS>6</LOANINITNUMPMTS>
```

```
    </LOANACCTFROM>
    <INCTRAN>Y
    <DTSTART>20050101120000</DTSTART>
    <DTEND>20050301120000</DTEND>
  </LOANSMTRQ>
</LOANSMTTRNRQ>
```

The response file:

```
<LOANSMTTRNRS>
  <TRNUID>ABCDEF-12345-54321-FEDGBA</TRNUID>
  <STATUS>
    <CODE>0</CODE>
    <SEVERITY>INFO</SEVERITY>
  </STATUS>
  <LOANSMTRS>
    <CURDEF>USD</CURDEF>
    <LOANACCTFROM>
      <LOANACCTID>123456789</LOANACCTID>
      <LOANACCTTYPE>CONSUMER</LOANACCTTYPE>
    </LOANACCTFROM>
    <LOANTRANLIST>
      <DTSTART>20050101120000</DTSTART>
      <DTEND>20050325120000</DTEND>
      <LOANSMTTRN>
        <LOANTRNTYPE>PAYMENT</LOANTRNTYPE>
        <DTPOSTED>20050208120000</DTPOSTED>
        <TRNAMT>887.89</TRNAMT>
        <LOANTRNAMT>
          <PRINAMT>818.92</PRINAMT>
          <INTAMT>33.97</INTAMT>
          <INSURANCE>35.00</INSURANCE>
        </LOANTRNAMT>
        <FITID>A45GC86565
      </LOANSMTTRN>
      <LOANSMTTRN>
        <LOANTRNTYPE>INT</LOANTRNTYPE>
        <DTPOSTED>20050208120000</DTPOSTED>
        <TRNAMT>33.97</TRNAMT>
        <FITID>A45GE86565
      </LOANSMTTRN>
      <LOANSMTTRN>
```

```
          <LOANTRNTYPE>PAYMENT</LOANTRNTYPE>
          <DTPOSTED>20050304120000</DTPOSTED>
          <TRNAMT>887.89</TRNAMT>
          <LOANTRNAMT>
            <PRINAMT>830.82</PRINAMT>
            <INTAMT>22.68</INTAMT>
            <INSURANCE>35.00</INSURANCE>
          </LOANTRNAMT>
        </LOANSMTTRN>
        <LOANSMTTRN>
          <LOANTRNTYPE>INT</LOANTRNTYPE>
          <DTPOSTED>20050304120000</DTPOSTED>
          <TRNAMT>22.68</TRNAMT>
          <FITID>A45GE86565
        </LOANSMTTRN>
        <LOANSMTTRN><!-- Extra principal only payment-->
          <LOANTRNTYPE>PAYMENT</LOANTRNTYPE>
          <DTPOSTED>20050315120000</DTPOSTED>
          <TRNAMT>500.00</TRNAMT>
          <LOANTRNAMT>
            <PRINAMT>500</PRINAMT>
            <INTAMT>0.00</INTAMT>
            <INSURANCE>0.00</INSURANCE>
          </LOANTRNAMT>
        </LOANSMTTRN>
      </LOANTRANLIST>
      <PRINBAL>
        <BALAMT>2850.26</BALAMT>
        <PRINYTD>2149.74</PRINYTD>
        <PRINLTD>2149.74</PRINLTD>
        <DTASOF>20050301120000</DTASOF>
      </PRINBAL>
      <MKTGINFO>New fixed rate loans at 6%. Contact branch</MKTGINFO>
    </LOANSTMTRS>
</LOANSTMTTRNRS>
```

- AMORTIZATION STATEMENT DOWNLOAD

The request file:

```
<AMRTSMTTRNRQ>
  <TRNUID>ABCDEF-12345-54321-FEDCBA</TRNUID>
```

```
  <AMRTSMTRQ>
    <LOANACCTFROM>
      <LOANACCTID>123456789</LOANACCTID>
      <LOANACCTTYPE>CONSUMER</LOANACCTTYPE>
    </LOANACCTFROM>
  </AMRTSMTRQ>
</AMRTSMTTRNRQ>
```

The response file:

```
<AMRTSMTTRNRS>
  <TRNUID>ABCDEF-12345-54321-FEDCBA</TRNUID>
  <STATUS>
    <CODE>0</CODE>
    <SEVERITY>INFO</SEVERITY>
  </STATUS>
  <AMRTSMTRS>
    <CURDEF>USD</CURDEF>
    <LOANACCTFROM>
      <LOANACCTID>123456789</LOANACCTID>
      <LOANACCTTYPE>CONSUMER</LOANACCTTYPE>
    </LOANACCTFROM>
    <DTSTART>20050101120000</DTSTART>
    <DTEND>20050704120000</DTEND>
    <AMRTSMTTRN>
      <PMTNUMBER>1</PMTNUMBER>
      <LOANINITBAL>5000.00</LOANINITBAL>
      <PRINBAL>
        <BALAMT>4181.08</BALAMT>
        <DTASOF>20050208120000</DTASOF>
      </PRINBAL>
      <LOANTRNAMT>
        <PRINAMT>818.92</PRINAMT>
        <INTAMT>33.97</INTAMT>
        <INSURANCE>35.00</INSURANCE>
      </LOANTRNAMT>
      <LOANIRATE>
        <LOANINTRATE>8</LOANINTRATE>
        <RATETYPE>FLOATING</RATETYPE>
        <DTASOF>20050101120000</DTASOF>
      </LOANIRATE>
      <AMRTTYPE>ADJUSTED</AMRTTYPE>
```

```
      <BALAMT>2016.73</BALAMT>
      <DTASOF>20050404120000</DTASOF>
    </PRINBAL>
    <LOANTRNAMT>
      <PRINAMT>833.53</PRINAMT>
      <INTAMT>19.97</INTAMT>
      <INSURANCE>35.00</INSURANCE>
    </LOANTRNAMT>
    <LOANIRATE>
      <LOANINTRATE>8.5</LOANINTRATE>
      <RATETYPE>FLOATING</RATETYPE>
      <DTASOF>20050301120000</DTASOF>
    </LOANIRATE>
    <AMRTTYPE>PROJECTED</AMRTTYPE>
  </AMRTSMTTRN>
  <AMRTSMTTRN>
    <PMTNUMBER>4</PMTNUMBER>
    <LOANINITBAL>2016.73</LOANINITBAL>
    <PRINBAL>
      <BALAMT>1176.91</BALAMT>
      <DTASOF>20050504120000</DTASOF>
    </PRINBAL>
    <LOANTRNAMT>
      <PRINAMT>839.82</PRINAMT>
      <INTAMT>13.68</INTAMT>
      <INSURANCE>35.00</INSURANCE>
    </LOANTRNAMT>
    <LOANIRATE>
      <LOANINTRATE>8.5</LOANINTRATE>
      <RATETYPE>FLOATING</RATETYPE>
      <DTASOF>20050301120000</DTASOF>
    </LOANIRATE>
    <AMRTTYPE>PROJECTED</AMRTTYPE>
  </AMRTSMTTRN>
      <AMRTSMTTRN>
    <PMTNUMBER>5</PMTNUMBER>
    <LOANINITBAL>1176.91</LOANINITBAL>
    <PRINBAL>
      <BALAMT>331.66</BALAMT>
      <DTASOF>20050604120000</DTASOF>
    </PRINBAL>
    <LOANTRNAMT>
```

```
        <PRINAMT>845.25</PRINAMT>
        <INTAMT>8.25</INTAMT>
        <INSURANCE>35.00</INSURANCE>
      </LOANTRNAMT>
      <LOANIRATE>
        <LOANINTRATE>8.5</LOANINTRATE>
        <RATETYPE>FLOATING</RATETYPE>
        <DTASOF>20050301120000</DTASOF>
      </LOANIRATE>
      <AMRTTYPE>PROJECTED</AMRTTYPE>
    </AMRTSMTTRN>
        <AMRTSMTTRN>
      <PMTNUMBER>6</PMTNUMBER>
      <LOANINITBAL>331.66</LOANINITBAL>
      <PRINBAL>
        <BALAMT>0</BALAMT>
        <DTASOF>20050704120000</DTASOF>
      </PRINBAL>
      <LOANTRNAMT>
        <PRINAMT>368.91</PRINAMT>
        <INTAMT>2.25</INTAMT>
        <INSURANCE>35.00</INSURANCE>
      </LOANTRNAMT>
      <LOANIRATE>
        <LOANINTRATE>8.5</LOANINTRATE>
        <RATETYPE>FLOATING</RATETYPE>
        <DTASOF>20050301120000</DTASOF>
      </LOANIRATE>
      <AMRTTYPE>PROJECTED</AMRTTYPE>
    </AMRTSMTTRN>
    <MKTGINFO>New fixed rate loans at 6%. Contact branch</MKTGINFO>
  </AMRTSMTRS>
</AMRTSMTTRNRS>
```

- **LOAN CLOSING STATEMENT DOWNLOAD**

The request file:

```
<LOANSTMTENDTRNRQ>
  <TRNUID>ABCDEF-12345-54321-FEDCCBA</TRNUID>
  <STATUS>
    <CODE>0</CODE>
```

```
    <SEVERITY>INFO</SEVERITY>
  </STATUS>
  <LOANSTMTENDRQ>
    <LOANACCTFROM>
      <LOANACCTID>123456789</LOANACCTID>
      <LOANACCTTYPE>CONSUMER</LOANACCTTYPE>
    </LOANACCTFROM>
    <DTSTART>20050101120000</DTSTART>
    <DTEND>20050704120000</DTEND>
  </LOANSTMTENDRQ>
</LOANSTMTENDTRNRQ>
```

The response file:

```
<LOANSTMTENDTRNRS>
  <TRNUID>ABCDEF-12345-54321-FEDCCBA</TRNUID>
  <STATUS>
    <CODE>0</CODE>
    <SEVERITY>INFO</SEVERITY>
  </STATUS>
  <LOANSTMTENDRS>
    <CURDEF>USD</CURDEF>
    <LOANACCTFROM>
      <LOANACCTID>123456789</LOANACCTID>
      <LOANACCTTYPE>CONSUMER</LOANACCTTYPE>
    </LOANACCTFROM>
    <LOANCLOSING>
      <FITID>1234</FITID>
      <DTOPEN>20050104120000</DTOPEN>
      <DTCLOSE>20050207120000</DTCLOSE>
      <DTNEXT>20050304120000</DTNEXT>
      <BALOPEN>5000.00</BALOPEN>
      <PRINBAL>
        <BALAMT>4181.08</BALAMT>
        <PRINYTD>818.92</PRINYTD>
        <PRINLTD>818.92</PRINLTD>
        <DTASOF>20050208120000</DTASOF>
      </PRINBAL>
      <LOANINT>
        <LOANINTYTD>33.97</LOANINTYTD>
        <LOANINTLTD>33.97</LOANINTLTD>
        <LOANINTPRJ>116.04</LOANINTPRJ>
```

```
        <DTASOF>20050208120000</DTASOF>
      </LOANINT>
      <LOANIRATE>
        <LOANINTRATE>8</LOANINTRATE>
        <RATETYPE>FLOATING</RATETYPE>
        <DTASOF>20050101120000</DTASOF>
      </LOANIRATE>
      <ESTPAYOFF>
        <ESTPAYOFFBAL>4422.91</ESTPAYOFFBAL>
        <DTASOF>20050208120000</DTASOF>
      </ESTPAYOFF>
      <BALLOONAMT>0</BALLOONAMT>
      <LOANPMT>
        <PMTAMT>887.89</PMTAMT>
        <DTPMTDUE>20050301120000</DTPMTDUE>
        <ESCRWAMT>0</ESCRWAMT>
        <LOANPMTTYPE> PRNANDINT</LOANPMTTYPE>
      </LOANPMT>
      <LOANRMNPMTS>5</LOANRMNPMTS>
      <DTPOSTSTART>20050208120000</DTPOSTSTART>
      <DTPOSTEND>20050303120000</DTPOSTEND>
      <MKTGINFO>New fixed rate loans at 6%. Contact branch</MKTGINFO>
    </LOANCLOSING>
  </LOANSTMTENDRS>
</LOANSTMTENDTRNRS>
```

CHAPTER 12 PAYMENTS

This section describes the Payments portion of OFX. OFX Payments consists of a set of functions for scheduling and maintaining payment transactions, and for synchronizing with the server to obtain an accurate status of all recent and scheduled transactions.

Clients use payment requests to schedule payments and to modify or delete payments if necessary. OFX also supports business payments, as described in section 12.1.

The recurring payments function allows the client to schedule automatic generation of a series of recurring payments by means of a single request. As with individual payments, the client can modify or delete these requests.

The payments function incorporates the synchronization features of OFX, allowing multiple client applications to synchronize with the server to obtain the current status of all payment transactions known to the server.

In many international environments, payments are performed using interbank funds transfers. OFX Payments supports this by allowing a payee to be designated as a destination bank account. Servers can implement these messages as transfers where appropriate.

12.1 Consumer and Business Payments

OFX Payments is designed to support both consumer and business payments. Businesses have additional requirements for payments. In particular, there is a need to include itemized instructions that specify how a payment should be disbursed across multiple invoices and/or line items. OFX supports this requirement through the inclusion of the <EXTDPMT> aggregate within payment requests. The Payment Modification Request <PMTMODRQ> also supports changes to <EXTDPMT> data.

12.2 The Payee Model

The payee model in OFX is designed to provide support for both “pay-some” and “pay-any” payment systems. “Pay-some” systems are those that restrict users to only make payments to payees that appear on an approved list. Such payees are often referred to as “standard payees,” or “standard merchants.” These are generally larger corporations that receive high volumes of payments, such as telephone companies or power utilities. In contrast, **“**pay-any” systems allow payments to any payee for which the user provides accurate billing information. These systems often also include a list of standard payees.

12.2.1 Payee Identifiers

OFX is designed to be flexible in the requirements for payee identifiers. It supports systems where all, some, or no payees are assigned a payee ID. In addition, it enables the server to assign an ID to a payee that was previously being paid by billing address.

You must implement the scope of payee such that the ID is at least global across the user's set of payments-enabled accounts with the payments provider. For example, if the user has both a checking and a money market account enabled for payments with the payments provider, then a payee ID obtained for a payment made from one of these accounts should identify the same payee if used for a payment drawn on the other account. This simplifies client support for allowing a user to choose from which account to make a payment.

OFX requires payee identifiers to have a one-to-one relationship with the corresponding <PAYEE> information. In other words, different payee IDs must also differ in their corresponding payee billing description or payee name <NAME>. Similarly, a payee ID must be independent of a user's account number with the payee. However, the payment system is free to use the user's account number in combination with the payee ID to determine the routing of a payment. These rules are intended to simplify the payee model for the user, insuring that different payee IDs will have discernibly different descriptions associated with them. They also insure that the user will not be required to maintain multiple payee entries for a payee with which the user holds multiple accounts.

OFX includes an element for indicating the scope of a payee ID returned from the server. This allows clients to adapt by expanding or restricting their functionality depending on the scope of payee IDs it encounters.

A payee list for each user, maintained on the server, allows the server to manage the identifiers assigned to a user's payees. This functionality is described in section 12.2.2.

12.2.2 Payee Lists

OFX specifies that a server-hosted payee list is maintained for each payments user. This list contains the payees that a user has paid through the payment system, or has set up to pay. Updates to this list are available through the synchronization mechanism. This insures that multiple clients have access to the full list of payees the user has configured. It is only necessary to enter each payee once.

Some payment systems require a first time setup before using a payee. This can occur externally to the client and server software, for example by filling out a paper form or telephoning the bank. In this case, payee list synchronization provides a way for the payee to become accessible to the client software when the FI completes the setup.

The list can contain payees with or without payee IDs. An important function of the payee list is to communicate payee changes from the server to the client. This includes changes in processing date parameters and conversion of a payee to a standard payee.

Once added to the list, the payment system makes payments by the payee list ID. This makes it clear to a client when the user is adding to a payee list, and when he or she modifies an existing payee on the list.

Although the messages make it clear whether a client is trying to add a new payee, a careful server will check for exact matches on payee adds and not create new payee list entries unnecessarily.

“Pay-any” systems can perform background processing that matches billing addresses with standard payees. When this occurs, the server can update the relevant payee lists and update the clients when they synchronize with the modified list data.

Each payee entry in the list can also include a list of the user’s accounts with that payee. This further reduces the data entry required by a user to make a payment, and facilitates the implementation of lightweight clients.

For a single account, it is important that references to a payee by <PAYEELSTID> do not resolve to different physical payees if the account is being used by more than one user. The same <PAYEELSTID> must map to the same corresponding payee billing description and payee name <NAME>. For example, <PAYEELSTID>12345 may be used for ABC Rentals for one user of a single account. If the same account is used for a second user, <PAYEELSTID>12345 cannot be used for XYZ Supplies. Conversely, if two or more users share an account, any added payees in each payee list must map to the same <PAYEELSTID>. Servers must ensure the integrity of the payee list for shared accounts, either by issuing a different userid for such accounts or by issuing payee and payment deletes followed by payee and payment adds to correct the payee list and update the payment history when two payee lists are merged. In the latter case, payees added by each user must be reconciled with the shared payee list on an ongoing basis.

12.2.3 Standard Payee Lists

Many payment systems maintain a list of payees that receive payments from a large number of users. Payments to these payees are usually consolidated for low electronic funds transfers or are mailed in large batches to the payee. Payees that receive this special processing are generally referred to as standard payees. In a “pay-some” system, all the approved payees can be considered to be standard payees. When a user pays a standard payee, there might be different processing lead-times used to calculate the payment and/or processing date of a payment.

When a payment system includes a standard payee list, it might be desirable to present the list to the user, who can then select payees he or she wants to pay. Unfortunately, it is cumbersome to provide this functionality in the client software due to the potential size of this list, which makes it problematic to keep updated and to present to the user. While the list can contain thousands of payee entries, a user will typically need less than ten or twenty entries from the list. It can also be difficult for a user to choose the correct payee entry when the list contains a number of similarly named payees.

Therefore, OFX does not provide a mechanism for delivering these lists to the client. However, there are several ways that an external presentation of such a list can be integrated into the client or server. For example, a payment provider’s Web site could present a search engine that assists the user to locate the correct payee. Once identified, the payee can either be imported into the client, or inserted into the user’s payee list on the server. In the latter case, synchronizing the payee list will make the newly added payees visible to the client.

12.2.4 Identifying Payees

Payees can be identified in several ways:

- Name and address, by means of <PAYEE>, must be identified only once for each payee. Thereafter, clients must use the assigned <PAYEELSTID> and, if assigned, <PAYEEID>. If the clients send both <PAYEE> and <PAYEELSTID> in a <PMTRQ>, <PMTMODRQ>, <RECPMTRQ>, or <RECPMTMODRQ>, the client is making an implicit payee modification request.
  In <BILLPAYMSGSRSV1>, the server must return a <PAYEEMODRS> in a subsequent <PAYEESYNCRS> for all actual changes. This is not necessary (though still allowed) if no change were made.

  If a client sends just <PAYEE> in a <PMTRQ> or <RECPMTRQ>, the client is making an implicit payee add request. (Clients must include the known <PAYEELSTID> in a <PMTMODRQ> or <RECPMTMODRQ>.) For more information about implicit payee adds and modifications, see section 12.2.5.
- Destination bank account <BANKACCTTO> should be done only once for each payee. Thereafter, clients should use the assigned <PAYEELSTID> or <PAYEEID>, as with name and address payees. The <PAYEE> aggregate is required to provide name and address information as a backup to account transfers.
- Payee list ID <PAYEELSTID> after a payee has been added to the list.

  Note: Duplicate payee list entries can occur if clients are not careful to send the payee list ID in subsequent requests.

- Standard payee ID <PAYEEID> for any payee that has been assigned a standard payee ID. This could happen before a closed system makes any payments, or anytime after the server has notified the client that a payee has a standard payee ID. If a <PAYEELSTID> also exists for the payee, it is required in the request and response, in addition to the <PAYEEID>.

  Note: Servers must always assign <PAYEELSTID>s to payees. Once <PAYEELSTID>s have been assigned, clients must always send the <PAYEELSTID>, even if a payee has both a <PAYEEID> and a <PAYEELSTID>.

12.2.5 Side Effects of Payee Adds and Modifications

Payees are added either implicitly or explicitly. Explicit adds occur by executing a <PAYEERQ>. Implicit payee adds occur with the execution of a <PMTRQ> or <RECPMTRQ> where a payee list ID is not sent with the request. (Thus, duplicate payee list entries can occur if clients are not careful to send the payee list ID if it is known.) In the case of an implicit payee add, a server must create and store a <PAYEERS> to be returned to the client in subsequent payee synchronization responses. Since the change was not generated by an explicit request, the <TRNUID> in this response would be zero.

Payees are modified either implicitly or explicitly. Explicit changes occur by executing a <PAYEEMODRQ>. Implicit payee changes occur with the execution of a <PMTRQ>, <PMTMODRQ>, <RECPMTRQ>, or <RECPMTMODRQ>, if the payee list ID is sent along with the payee aggregate. In <BILLPAYMSGSETV1>, a <PAYEE> aggregate must accompany these requests. In such cases, since the <PAYEELSTID> is present, a server may check to see if the payee information has changed, and only if so, process an implicit payee modification. An implicit payee change must cause the server to create and store a <PAYEEMODRS>, to be returned to the client in subsequent payee synchronization responses. Since the change was not generated by an explicit request, the <TRNUID> in these responses will be zero.

In addition to the above, a payee change (implicit or otherwise) may also affect models and their future (though not pending) payments. Thus, for any model that is affected by an explicit or implicit payee modification, the server must create and store a <RECPMTMODRS>, to be returned to the client in subsequent recurring payment synchronization responses. The <TRNUID> in this response will be zero.

12.3 Identifiers Used in Payment Transactions

Payment transactions use four types of identifiers. It is important to understand the purpose, scope, and life span of these identifiers.

The client-to-request messages assign the Transaction Universal Identifier <TRNUID>. Its purpose is to allow the client to easily match responses from the server to their corresponding requests. A given transaction ID is used only for a client request and the corresponding server response.

The Server Identifier, <SRVRTID> or <RECSRVRTID>, is assigned by the server to a payment “object,” which can either be a payment or a recurring payment model (in which case it is named <RECSRVRTID>). Both the client and server use the ID thereafter to refer to the payment or model in any transactions that operate on them. For example, the <SRVRTID> is used to identify a payment in a request to modify or cancel it. The <SRVRTID> is valid for the life span of the payment within the payment system. Similarly the <RECSRVRTID> is valid as long as the associated model exists, that is until the model generates all payments, or the model is canceled. Once a server processes a payment or a model generates all its required payments, the associated <SRVRTID> (or <RECSRVRTID>) is no longer known to the server. Note that the payment system might continue to maintain knowledge of a payment <SRVRTID> or model <RECSRVRTID> for some specified period after it completes processing. This allows clients to access the “completed” status of these operations.

12.4 The Payment Life Cycle

12.4.1 Payment Creation

The client formulates a <PMTRQ> that includes the payee, the date, the amount of the payment, the funding account, and the user's account number with the payee. If supported by the user's payment system, the billing address can specify the payee.

The server will look up the payee in the user's payee list. If it is not already in the table, the server will add it and issue a payee list identifier <PAYEELSTID>. This form of payment request performs an implicit Payee Request <PAYEERQ>, which is equivalent to explicitly adding the payee (by means of a <PAYEERQ>), prior to issuing the <PMTRQ>. It has the advantage of being atomic. If the payment request fails, the payee is not added to the user's payee list. Conversely the payment request will fail if the payee information is invalid.

The server responds to the <PMTRQ> with a Payment Response <PMTRS>. Some servers will not be able to immediately return a payee ID at this point, or might not issue payee IDs for all payees. Therefore the <PAYEELSTID> contained in the response functions as the linkage between the payee and the payment. Payment systems use the <SRVRTID> returned in the <PMTRS> to identify the payment for the length of its instantiation on the payment system.

> **Note:** Servers should generate explicit responses to implicit requests. In other words, implicit payee additions or modifications resulting from a new or changed payment should generate explicit payee add or payee change responses from the server. Such explicit responses are only returned to the client in a SYNC response. If the payment transactions containing implicit payee additions or modifications fail, then the payee actions are not executed, since such a compound payment transaction represents a single unit of work (comprised of both payee and payment actions).

12.4.2 Payment Modification

Between the time the client schedules a payment and the time the server processes the payment, the client can request changes to the parameters of that payment. For example, the amount or date of the payment can be modified. The system uses the Payment Modification Request <PMTMODRQ> for this purpose, where the <SRVRTID> from the <PMTRS> identifies the targeted payment. The user request must specify the full contents of the payment request, including both modified and unmodified data.

Full-featured servers will use <PMTMODRS> messages, conveyed to the client during synchronization <PMTSYNCRS>, to inform the client about changes in the state of the client that occur due to server processing. This would include reporting the date the server actually processed a payment, or it failed due to insufficient funds. Servers that are unable to generate <PMTMODRS> responses for this purpose must support the <PMTINQRQ> message described below.

12.4.3 Payment Status Inquiry

As a scheduled payment progresses through its “life-cycle” on the server, the processing status changes accordingly from “scheduled to be processed” to “was processed” or “failed processing.” A processing date is associated with these states. The preferred method for providing updated processing status to a client is by use of server-generated Payment Modification messages <PMTMODRS>, as discussed above. However it is possible that less full-featured servers might have difficulty in implementing this form of notification. In this case, OFX requires such servers to implement the Payment Status Inquiry message <PMTINQRQ>, which provides an interface for the client to explicitly request the processing status of individual payments.

12.4.4 Payment Cancellation

In the interval between successful processing of a <PMTRQ> and the actual processing of the payment, the client can cancel the payment by issuing a Payment Cancellation Request <PMTCANCRQ>. The <SRVRTID> value returned in <PMTRS> identifies the payment.

When a payment system cancels a payment, servers can generate a <PMTCANCRS>. This might occur if the user requests payment cancellation by way of a telephone call to customer support or through an e-mail message. The client will receive this response when performing a payment synchronization <PMTSYNCRQ>/<PMTSYNCRS>.

12.4.5 Delayed Payee Matching

Payment systems that allow payment by payee billing address often perform a matching operation to determine if the payee is a standard payee. If this matching occurs in the processing of a <PMTRQ>, and the server recognizes the payee as a standard one, then the server returns the payee ID and payment parameters in the <EXTDPAYEE> aggregate of the <PMTRS>. However some payment systems will not be able to perform “payee matching” at this point in processing. In this case, the server sends updated payee information to the client by using <PAYEESYNCRS> to synchronize the payee list. The client can link payee information in the <PAYEESYNCRS> messages to payments with matching <PAYEELSTID> identifiers.

12.5 Common Payments Aggregates

This section documents several aggregates used throughout the Payments portion of the OFX specification.

12.5.1 Payments Account Information <BPACCTINFO>

OFX uses the payments account information aggregate to download account information from an FI. It includes account number specification in <BANKACCTFROM> as well as the status of the service. In OFX, Banking and Payments share the <BANKACCTFROM> aggregate to identify a specific account. For more information, see section 11.3.1.

	Description
<BPACCTINFO>	Payments-account-information aggregate
<BANKACCTFROM>	Bank-account-from aggregate, see section 11.3.1
</BANKACCTFROM>	
<SVCSTATUS>	Status of the account AVAIL = Available, but not yet requested PEND = Requested, but not yet available ACTIVE = In use
</BPACCTINFO>	

12.5.2 Payment Information <PMTINFO>

The Payment Information aggregate is used to specify detailed payment information. It is used for both single payments and recurring payments. Clients must send the <PAYEELSTID> and <PAYEEID> if known. See section 12.2.4 on identifying payees, above. The <EXTDPMT> aggregate (see section 12.5.2.2) allows the inclusion of disbursement instructions to be printed with the payment. This aggregate is optional.

In the case of an implicit add, the returned <PMTINFO> aggregate found in <PMTRS> and <RECPMTRS> must include the generated <PAYEELSTID>. This aggregate may also include <EXTDPMT> information.

The <DTDUE> in a response may have been adjusted by a server. For example, the server may adjust <DTDUE> to comply with non-processing days. If a client sends a request to make a transfer on July 4 and July 4 happens to be a non-processing day, the <DTDUE> in the response may be July 4 (because the server hasn't adjusted it yet), July 5 (because the server rolls dates forward), or some other date.

Tag	Description
<PMTINFO>	
<BANKACCTFROM>	Account-from aggregate, see section 11.3.1
</BANKACCTFROM>	
<TRNAMT>	Payment amount, amount This amount should be specified as a positive number
Specify payee; either <PAYEEID> or <PAYEE>.	
<PAYEEID>	Server payee identifier (required if assigned). Either <PAYEEID> or <PAYEE> can be sent, but not both. A-12
<PAYEE>	Complete payee billing information, see section 12.5.2.1 Either <PAYEEID> or <PAYEE> can be sent, but not both.
</PAYEE>	
<PAYEELSTID>	Payee list ID (required if assigned), A-12
<BANKACCTTO>	Destination account (see section 11.3.1) information for systems that pay by transfers (<PAYEE> also required)
</BANKACCTTO>	
<EXTDPMT>	Zero or more extended Payment aggregates, see section 12.5.2.2 Note: Although PMTINFO allows multiple occurrences of EXTDPMT, it is recommended that multiple invoices be expressed using multiple occurrences of the INVOICE aggregate.

</EXTDPMT>

<PAYACCT>	User account number with the payee.	An32
<DTDUE>	Payment due date or the date by which payment must be received by payee.	datetime
<MEMO>	Memo from user to payee.	memo
<BILLREFINFO>	Biller-supplied reference information when paying a bill, if available.	An80
	Note: If the client user interface has a single field that can contain either free-	

12.5.2.1 Payee <PAYEE>,

<PAYEE> specifies a complete billing address for a payee.

	Description
<PAYEE>	
<NAME>	Name of payee. *A-32*
<ADDR1>	Payee's address line 1, *A-32*
<ADDR2>	Payee's address line 2, *A-32*
<ADDR3>	Payee's address line 3. Use of <ADDR3> requires the presence of <ADDR2>, *A-32*
<CITY>	Payee's city, *A-32*
<STATE>	Payee's state, *A-5*
<POSTALCODE>	Payee's postal code, *A-11*
<COUNTRY>	Payee's country; 3-letter country code from ISO/DIS-3166, *A-3*
<PHONE>	Payee's telephone number, *A-32*
</PAYEE>	

12.5.2.2 Extended Payment <EXTDPMT>

The Extended Payment aggregate provides the payee with information for applying a payment across multiple invoices. It is structured to allow for electronic processing of the invoice data, and allows multiple invoices, as well as multiple line items per invoice, to be specified.

In this case, <EXTDPMT> can specify a block of free text to be transmitted with the payment, by using the <EXTDPMTDSC> instead of the <EXTDPMTINV> element.

Tag	*Description*
<EXTDPMT>	Extended Payment aggregate
<EXTDPMTFOR>	INDIVIDUAL or BUSINESS. Indicates whether the payment is for an individual or business account. This allows the payment processor to remit payments to the appropriate address for consumers or businesses.
<EXTDPMTCHK>	Check number to use for this payment. Overrides "next check in range." *N-10*
Payment description. At least one of the following: <EXTDPMTDSC>, or <EXTDPMTINV>.	
<EXTDPMTDSC>	Free text to communicate with the payment, *A-255*

Tag	Description
<EXTDPMTINV> **<INVOICE>** **</INVOICE>** **</EXTDPMTINV>**	One or more invoice aggregates. See section 12.5.2.3
</EXTDPMT>	

12.5.2.3 Invoice Description <INVOICE>

	Description
<INVOICE>	Start tag for the invoice aggregate. There can be one or more invoices per payment request.
<INVNO>	Invoice number associated with the payment. *A-32*
<INVTOTALAMT>	This value represents the total invoice amount, *amount* This amount should be specified as a positive number
<INVPAIDAMT>	This value represents the amount of the invoice being paid, *amount* This amount should be specified as a positive number
<INVDATE>	Date to apply the invoice, *datetime*
<INVDESC>	Invoice description, *A-80*
<DISCOUNT>	Discount aggregate; only one discount aggregate per invoice
<DSCRATE>	Discount rate, *rate*
<DSCAMT>	Discount amount, *amount* This amount should be specified as a positive number
<DSCDATE>	Date to apply the discount, *datetime*
<DSCDESC>	Discount description, *A-80*
</DISCOUNT>	
<ADJUSTMENT>	Adjustment aggregate; only one adjustment aggregate per invoice, see 12.5.2.4
</ADJUSTMENT>	
<LINEITEM>	Line item aggregate; there can be multiple line items per invoice, see 12.5.2.5
</LINEITEM>	
</INVOICE>	

12.5.2.4 <ADJUSTMENT>

	Description
<ADJUSTMENT>	
<ADJNO>	Adjustment number associated with the payment, *A-32*
<ADJDESC>	Adjustment description, *A-80*
<ADJAMT>	Amount of the adjustment, *amount* This amount should be signed + or -, as appropriate. A positive adjustment means that the payment amount has been reduced.
<ADJDATE>	Date of adjustment, *datetime*
</ADJUSTMENT>	

12.5.2.5 <LINEITEM>

	Description
<LINEITEM>	Line item aggregate; there can be multiple line items per invoice
<LITMAMT>	Amount of the line item, *amount* This amount should be signed + or -, as appropriate. A positive line item amount is an addition to the payment amount, and a negative line item is a discount or reduction in the payment amount.
<LITMDESC>	Line item description, *A-80*
</LINEITEM>	

12.5.2.6 Extended Payee <EXTDPAYEE>

The Extended Payee aggregate communicates a payee identifier to the client. It also contains the processing day parameters for a payee. It can be sent to the client for any payee whose processing day parameters are different from the processor's default values, even for payees with no <PAYEEID>.

	Description
<EXTDPAYEE>	Extended-payee aggregate
<PAYEEID>	Server-assigned payee ID, *A-12* If <PAYEEID> is present, <IDSCOPE> and <NAME> are required. Should not be included unless the payee is a standard payee.
<IDSCOPE>	Scope of the payee ID; one of (GLOBAL, USER), where GLOBAL = payee ID valid across the entire payment system USER = payee ID valid with all FI accounts set up for the user's payments account Required if <PAYEEID> is present.
<NAME>	Standard payee name, *A-32* Required if <PAYEEID> is present.
<DAYSTOPAY>	Minimum number of business days needed to process, *N-3*
</EXTDPAYEE>	

12.5.2.7 Payment Processing Status <PMTPRCSTS>

The Payment Processing Status aggregate contains the current processing status for a payment. This aggregate is intended to describe status changes to the associated payment after creation. The interpretation of the date value depends on the value of <PMTPRCCODE>.

	Description
<PMTPRCSTS>	
<PMTPRCCODE>	See table 12.6.2.1
<DTPMTPRC>	Payment processing date; interpretation depends on <PMTPRCCODE>, *datetime*
</PMTPRCSTS>	Ending tag for payment processing status

12.6 Payments Functions

Payments functions allow a client to create a Payment Request to pay a bill on a specified date. The Payment Request identifies the payee and the amount to pay. Because the flow of money is unambiguous, bill payment amounts are usually specified as positive numbers. See tables for details.

	Server Responds
Account information Payment date Amount Payee address, list ID, transfer acct, or standard ID	
	Payment status Check number Server-assigned ID

From the time the client issues a Payment Request until it is paid, the client can modify the transaction through the Payment Modification Request, <PMTMODRQ>; see section 12.4.2. This request allows payment parameters such as the payment date and payment amount to be changed.

Client Sends	***Server Responds***
Account information Server-assigned ID Information to change: Payment date, Amount,...	
	Acknowledgment or Error

The client can cancel a Payment Request with a Payment Cancellation Request, <PMTCANCRQ>; see section 12.4.4.

Client Sends	***Server Responds***
Account information Server-assigned ID	
	Acknowledgment or Error

12.6.1 Payment Creation

A Payment Request is used to schedule an electronic payment. The server responds with a Payment Response. Separate transactions are provided for modifying and canceling a Payment Request.

12.6.1.1 Payment Request <PMTRQ>

The <PMTRQ> request must appear within a <PMTTRNRQ> transaction wrapper.

	Description
<PMTRQ>	Payment-request aggregate
<PMTINFO>	Payment Information aggregate, see section 12.5.2
</PMTINFO>	
</PMTRQ>	

Note: If the <PMTRQ> created a new payee or modified an existing one, the server must create and store a payee response that would be available for subsequent payee synchronization requests. In addition, the server should be aware of the fact that implicit payee modifications may affect models. Such changes to models must also appear in subsequent recurring synchronization responses. In all cases, the server need only send the <PMTRS> as a response to the <PMTRQ>, but any implicit payee and recurring changes must be made by the server, and be returned in later synchronization responses. See section 12.2.5 for further discussion of implicit payee adds and modifications.

12.6.1.2 Payment Response <PMTRS>

The server sends a Payment Response in response to a Payment Request. The processing status code for a new payment is normally WILLPROCESSON, but in the case of synchronization it can return other status codes. Servers should inform clients of any errors found while processing this transaction using the <STATUS> aggregate. A response containing <STATUS><CODE>0 and <PMTPRCSTS><PMTPRCCODE>FAILEDON should be avoided for problems such as an invalid account or amount.

The <PMTRS> response must appear within a <PMTTRNRS> transaction wrapper.

Note: When processing a <PMTRQ> request that does not contain a <PAYEEID> or <PAYEELSTID>, a server may check the payee against the user's current payee list and return the found <PAYEELSTID> and <PAYEEID> (if any). If the server does this, it should only find a match when all <PAYEE> data, including all <PAYACCT> elements, match exactly. If the <PAYACCT> or any other element is different, the server must perform an implicit payee addition. If a server doesn't check for duplicate payees, a client could show duplicate entries in the payee list.

Note: Servers matching well-known payees against entries in a user's payee list must ignore the <PAYACCT> information in the user's list. No corresponding information appears in the list of well-known payees.

	Description
<PMTRS>	Payment-response aggregate
<SRVRTID>	ID assigned by the server to the payment being created, *SRVRTID*
<PAYEELSTID>	Server-assigned payee list record ID for this payee, *A-12* **Note:** This identifier must match that found (and required) in the returned <PMTINFO>.
<CURDEF>	Default currency for the Recurring Payment Response, *currsymbol*
<PMTINFO>	Payment Information aggregate, see section 12.5.2
</PMTINFO>	
<EXTDPAYEE>	Standard payee information if payee is a standard payee, or payee has non-default processing day parameters; see section 12.5.2.6
</EXTDPAYEE>	
<CHECKNUM>	Check number, *A-12*
<PMTPRCSTS>	Payment processing status
</PMTPRCSTS>	
<RECSRVRTID>	References the payment if it was generated by a recurring payment, *SRVRTID*
</PMTRS>	

12.6.1.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2006	Source account not found (ERROR)
2007	Source account closed (ERROR)
2008	Source account not authorized (ERROR)
2009	Destination account not found (ERROR)
2010	Destination account closed (ERROR)
2011	Destination account not authorized (ERROR)
2012	Invalid amount (ERROR)
2014	Date too soon (ERROR)
2015	Date too far in future (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10501	Invalid payee (ERROR)
10502	Invalid payee address (ERROR)
10503	Invalid payee account number (ERROR)
10510	Invalid payee ID (ERROR)
10511	Invalid payee city (ERROR)
10512	Invalid payee state (ERROR)
10513	Invalid payee postal code (ERROR)
10517	Invalid payee name (ERROR)
10519	Invalid payee list ID (ERROR)

12.6.1.4 Discussion

Once the server has assigned a payee identifier <PAYEEID> to the payee, it includes the <EXTDPAYEE> in any <PMTRS> for that transaction. If the <EXTDPAYEE> aggregate is present in the Payment Response <PMTRS>, the client records the standard payee information for use in future payments to the same payee.

When a payment is made using the <PAYEE> aggregate, and no <PAYEELSTID> is present, the payee is implicitly added to the payee list. This is therefore equivalent to first transmitting a <PAYEERQ> for the payee. For payment systems that can immediately return payee IDs, it is preferable to use the single <PMTRQ> message to both add the payee and create the payment. If either operation fails, the server will not complete the other.

If <PAYEELSTID> and <PAYEE> are both included, it is equivalent to sending a <PAYEEMODRQ>. In <BILLPAYMSGSRSV1>, the server must return a <PAYEEMODRS> in a subsequent <PAYEESYNCRS> for all actual changes. This is not necessary (though still allowed) if no change were made.

The <PMTRS> response will include the <EXTDPAYEE> aggregate if the processor has assigned a payee ID to the payee specified in the payment. It will also appear in the response when the payee has no assigned ID, but has processing day parameters that different from the processor's defaults for these values. This might occur, for instance, if the processor notes that the postal code of the payee indicates a certain proximity to the payer, and therefore wishes to offer a shorter <DAYSTOPAY> value.

12.6.2 Payment Modification

The Payment Modification Request allows a client to modify a previously scheduled payment. Once created and retrieved by the customer, spawned payments are almost identical to customer-created payments. (The exception is when a spawned payment is modified or cancelled due to a recurring modification or cancellation request.) As with ordinary payments, you can cancel or modify transactions individually. When modifying a payment, the client must specify all of the elements and aggregates within the <PMTINFO> aggregate that were specified during the payment creation or previous modification, not just the elements and aggregates that it wants to modify. Some servers cannot support the modifications of certain values. Servers must indicate this by returning status code 10505 when the client requests an unsupported modification.

12.6.2.1 Payment Processing Status Values <PMTPRCCODE>

	Description
WILLPROCESSON	Will be processed on <DTPMTPRC>
PROCESSEDON	Was processed for payment on <DTPMTPRC>
NOFUNDSON	Funds not available to make payment on <DTPMTPRC>
FAILEDON	Unable to make payment for unspecified reasons on <DTPMTPRC>
CANCELEDON	User canceled payment on <DTPMTPRC>

12.6.2.2 Payment Modification Request <PMTMODRQ>

The client sends a Payment Modification Request to request modification of a payment. The client must provide the full <PMTINFO> including both changed and unchanged values.

The client may modify any data in <PMTINFO> except the recipient or funding account. In particular, payee list ID <PAYEELSTID>, payee ID <PAYEEID>, funding bank account <BANKACCTFROM>, and the <NAME> element of <PAYEE> must match that returned in the original <PMTRS>. Implicit payee modifications (changes in address information for example) are allowed.

The <PMTMODRQ> request must appear within a <PMTTRNRQ> transaction wrapper.

	Description
<PMTMODRQ>	Modification-request this references
<SRVRTID>	ID assigned by the server to the payment being modified, *SRVRTID*
<PMTINFO>	Payment Information aggregate, see section 12.5.2
</PMTINFO>	
</PMTMODRQ>	

12.6.2.3 Payment Modification Response <PMTMODRS>

The server sends a Payment Modification Response in response to a Payment Modification Request.

The <PMTMODRS> response must appear within a <PMTTRNRS> transaction wrapper.

	Description
<PMTMODRS>	Payment-modification-response this references
<SRVRTID>	ID assigned by the server to the payment being modified, *SRVRTID*
<PMTINFO>	Payment Information aggregate, see section 12.5.2
</PMTINFO>	
<PMTPRCSTS>	Payment processing status, see section 12.5.2.7
</PMTPRCSTS>	
</PMTMODRS>	

12.6.2.4 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2006	Source account not found (ERROR)
2007	Source account closed (ERROR)
2008	Source account not authorized (ERROR)
2009	Destination account not found (ERROR)
2010	Destination account closed (ERROR)
2011	Destination account not authorized (ERROR)
2012	Invalid amount (ERROR)
2014	Date too soon (ERROR)
2015	Date too far in future (ERROR)
2016	Transaction already committed (ERROR)
2017	Already canceled (ERROR)
2018	Unknown server ID (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10501	Invalid payee (ERROR)
10502	Invalid payee address (ERROR)
10503	Invalid payee account number (ERROR)
10505	Cannot modify element (ERROR)
10510	Invalid payee ID (ERROR)
10511	Invalid payee city (ERROR)
10512	Invalid payee state (ERROR)
10513	Invalid payee postal code (ERROR)
10514	Transaction already processed (ERROR)
10517	Invalid payee name (ERROR)
10519	Invalid payee list ID (ERROR)

12.6.2.5 Discussion

Servers can initiate <PMTMODRS> messages to communicate changes in the processing status of a payment as it moves through the payment system. This mechanism allows a client to capture the updated status of payments every time it synchronizes.

Implicit payee changes contained in a payment modification transaction do not affect any other existing pending payments. The changes are propagated to the server's payee list and affect payments to that payee as subsequently initiated by the client after the change, or as subsequently spawned from a recurring model. Explicit payee changes are not propagated to payments pending for the changed payee at the time of the change.

12.6.3 Payment Cancellation

The Payment Cancellation Request allows a client to cancel a previously scheduled payment created with a Payment Request (<PMTRQ> in section 12.6.1.1).

Servers cannot initiate <PMTCANCRS> when communicating status changes. This response should be used only when a payment was actually cancelled (by an OFX client or at users request via the phone). When conveying information about a failure in payment processing (such as insufficient funds), a <PMTMODRS> (with the updated <PMTPRCSTS>) should be added to the next <PMTSYNCRS> download.

12.6.3.1 Request <PMTCANCRQ>

The client sends a Payment Cancellation to cancel a scheduled payment request.

The <PMTCANCRQ> request must appear within a <PMTTRNRQ> transaction wrapper.

	Description
<PMTCANCRQ>	Cancellation-request this references
<SRVRTID>	ID assigned by the server to the payment being cancelled, *SRVRTID*
</PMTCANCRQ>	

12.6.3.2 Response <PMTCANCRS>

The server sends a Payment Cancellation Response in response to a Payment Cancellation Request.

The <PMTCANCRS> response must appear within a <PMTTRNRS> transaction wrapper.

	Description
<PMTCANCRS>	Cancellation-response this references
<SRVRTID>	ID assigned by the server to the payment being canceled, *SRVRTID*
</PMTCANCRS>	

12.6.3.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2016	Transaction already committed (ERROR)
2017	Already canceled (ERROR)
2018	Unknown server ID (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10514	Transaction already processed (ERROR)

12.6.4 Payment Status Inquiry

The Payment Status Inquiry Request allows a client to obtain the current processing status of a payment from the server.

12.6.4.1 Request <PMTINQRQ>

The client sends a Payment Status Inquiry Request to obtain the current processing status of a payment.

The <PMTINQRQ> request must appear within a <PMTINQTRNRQ> transaction wrapper.

	Description
<PMTINQRQ>	Payment-status-inquiry-request aggregate
<SRVRTID>	ID assigned by the server to the payment being queried, *SRVRTID*
</PMTINQRQ>	

12.6.4.2 Response <PMTINQRS>

The server sends a Payment Status Inquiry Response in response to a Payment Status Inquiry Request.

The <PMTINQRS> response must appear within a <PMTINQTRNRS> transaction wrapper.

	Description
<PMTINQRS>	Payment-status-inquiry-response aggregate
<SRVRTID>	ID assigned by the server to the payment being queried, *SRVRTID*
<PMTPRCSTS>	Payment processing status
</PMTPRCSTS>	
<CHECKNUM>	Check number assigned by the server to this payment, *A-12*
</PMTINQRS>	

12.6.4.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2018	Unknown server ID (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)

12.7 Recurring Payments

Recurring payments are used when a payment is to be made repeatedly at some known interval. Setting up a recurring payment is similar to creating an individual payment, but with additional information about the frequency and number of payments. After a recurring payment is created, the server will generate payments transactions when there are a specified number of days remaining until the next projected payment is due (usually 30 days). The client will be made aware of any generated payment transactions through the synchronization process. Chapter 10, "Recurring Transactions" and Chapter 11, "Banking," provide additional details on models and recurring transactions, and define the recurring transaction aggregates.

Note: As with individual payments, if the recurring payment request adds a payee or changes payee information, the server must create and store a payee response, to be returned in subsequent payee synchronization responses. Furthermore, implicit payee modifications may affect other models (but not their pending payments). The server must also create and store recurring modification responses for these models, to be returned in subsequent recurring synchronization responses. See section 12.2.5 for further discussion of implicit payee adds and changes.

Note: The <MODELWND> profile value indicates when the server spawns a payment. If <MODELWND>0 is specified, the server only spawns one payment at a time for each model. In other words, there is always one pending payment per model, unless the model has expired. If <MODELWND> is greater than 0, its value is the number of days before a payment is due to be paid that it is spawned from the model. In this case, it is possible to have zero or more pending payments instantiated at a time.

The table below lists the functional elements for creating a recurring payment:

	Server Responds
Account information	
Payment frequency	
Number of payments	
Payment date	
Amount	
Payee address, list ID or payee ID	
	Standard payee information
	Server-assigned ID

The table below lists the functional elements for modifying a recurring payment:

Client Sends	***Server Responds***
Account information	
Server-assigned ID	
Information to change:	
Payment frequency,	
Number of payments,	
Payment date,	
Amount,...	
	Acknowledgment or Error

The table below lists the functional elements for canceling a recurring payment:

Client Sends	***Server Responds***
Account information	
Server-assigned ID	
	Acknowledgment or Error

12.7.1 Creating a Recurring Payment

Use a Recurring Payment Request to set up a recurring electronic payment. The user can specify the frequency and duration of the payments using the Recurring Instructions aggregate <RECURRINST>. The <PMTINFO> aggregate (see section 12.5.2) specifies the payment information for the model, as well as the initial and final amounts (if present and where applicable).

12.7.1.1 Request <RECPMTRQ>

The <RECPMTRQ> request must appear within a <RECPMTTRNRQ> transaction wrapper.

	Description
<RECPMTRQ>	Recurring-payment-request aggregate
<RECURRINST>	Recurring Instructions aggregate, see section 10.2.
</RECURRINST>	
<PMTINFO>	Payment-information aggregate, see section 12.5.2
</PMTINFO>	
<INITIALAMT>	Amount of the initial payment, if different than the following payments, *amount* This amount should be specified as a positive number
<FINALAMT>	Amount of the final payment, if different than the preceding payments, *amount* This amount should be specified as a positive number
</RECPMTRQ>	

12.7.1.2 Response <RECPMTRS>

The server sends a Recurring Payment Response upon receipt of a Recurring Payment Request.

The <RECPMTRS> response must appear within a <RECPMTTRNRS> transaction wrapper.

	Description
<RECPMTRS>	Recurring-payment-response aggregate
<RECSRVRTID>	Server-assigned ID for this transaction, *SRVRTID*
<PAYEELSTID>	Server-assigned record ID for this payee record, *A-12* **Note:** This identifier must match that found (and required) in the returned <PMTINFO>.
<CURDEF>	Default currency for the Recurring Payment Response, *currsymbol*
<RECURRINST>	Recurring-instructions aggregate, see section 10.2.
</RECURRINST>	
<PMTINFO>	Payment-information aggregate, see section 12.5.2
</PMTINFO>	
<INITIALAMT>	Amount of the initial payment, if different than the following payments, *amount* This amount should be specified as a positive number
<FINALAMT>	Amount of the final payment, if different than the preceding payments, *amount* This amount should be specified as a positive number
<EXTDPAYEE>	Extended payee information, see section 12.5.2.6.
</EXTDPAYEE>	
</RECPMTRS>	

12.7.1.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2006	Source account not found (ERROR)
2007	Source account closed (ERROR)
2008	Source account not authorized (ERROR)
2009	Destination account not found (ERROR)
2010	Destination account closed (ERROR)
2011	Destination account not authorized (ERROR)
2012	Invalid amount (ERROR)
2014	Date too soon (ERROR)
2015	Date too far in future (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10501	Invalid payee (ERROR)
10502	Invalid payee address (ERROR)
10503	Invalid payee account number (ERROR)
10508	Invalid frequency (ERROR)
10510	Invalid payee ID (ERROR)
10511	Invalid payee city (ERROR)
10512	Invalid payee state (ERROR)
10513	Invalid payee postal code (ERROR)
10517	Invalid payee name (ERROR)
10519	Invalid payee list ID (ERROR)

12.7.1.4 Discussion

The <DTDUE> element of <PMTINFO> specifies payment due date or the date by which the first payment must be received by payee (see section 12.5.2).

The <DTDUE> in a response may have been adjusted by a server. For example, the server may adjust <DTDUE> to comply with non-processing days. If a client sends a request to make a transfer on July 4 and July 4 happens to be a non-processing day, the <DTDUE> in the response may be July 4 (because the server hasn't adjusted it yet), July 5 (because this server rolls dates forward), or some other date. For this reason, a client should pay attention to the <DTDUE> in the response.

12.7.2 Recurring Payment Modification

The client sends a Recurring Payment Modification Request to request modifications to a recurring payment previously created with a Recurring Payment Request. The payment frequency <RECURRINST>, the payment parameters <PMTINFO>, or both, can be changed.

12.7.2.1 Request <RECPMTMODRQ>

The client sends a Recurring Payment Modification Request to request changes to a recurring payment model.

The client may modify any data in <PMTINFO> except the recipient or funding account. In particular, payee list ID <PAYEELSTID>, payee ID <PAYEEID>, funding bank account <BANKACCTFROM>, and the <NAME> element of <PAYEE> must match that returned in the original <RECPMTRS>. Implicit payee modifications (changes in address information for example) are allowed. Clients can modify both elements in the <RECURRINST> aggregate (i.e. <NINSTS> and <FREQ>). Client should send the original number of payments scheduled if there is no change. If there is a change in the number of payments scheduled, clients should send the new number of payments.

A <RECPMTMODRQ> that modifies pending payments via the <MODPENDING> flag is a compound transaction and the server should create and store <PMTMODRS>s, which are returned to the client in subsequent payment synchronization responses. For example, a change to the <TRNAMT> element would cause the server to create and store a <PMTMODRS> for each pending payment, to be returned in a subsequent payment synchronization response. Changes to payment information apply to all future payments.

> **Note:** The <RECPMTMODRQ> element may implicitly modify a payee. A payee modification can, in turn, modify other existing models (though not their pending payments). In such cases, the server must create and store the appropriate responses (<PAYEEMODRS> and, possibly, additional, <RECPMTMODRS>), to be returned to the client in subsequent synchronization responses. See section 12.2.5 for further discussion of implicit payee adds and changes. If the <RECPMTMODRQ> created a new payee (this is only possible if the payment were originally created outside of the OFX protocol), the server must create and store a <PAYEERS> that would be available for a payee synchronization request. In all cases, the server need only send the <RECPMTMODRS> as a response to the <RECPMTMODRQ>, but

any implicit payee and recurring changes must be made by the server, and be returned in later synchronization responses.

The <RECPMTMODRQ> request must appear within a <RECPMTTRNRQ> transaction wrapper.

	Description
<RECPMTMODRQ>	Modification-request aggregate
<RECSRVRTID>	ID assigned by the server to the payment being modified, *SRVRTID*
<RECURRINST>	Recurring Instructions aggregate, see section 10.2
</RECURRINST>	
<PMTINFO>	Payment-Information aggregate, see section 12.5.2
</PMTINFO>	
<INITIALAMT>	Amount of the initial payment, if different than the following payments, *amount* This amount should be specified as a positive number
<FINALAMT>	Amount of the final payment, if different than the preceding payments, *amount* This amount should be specified as a positive number
<MODPENDING>	Modify pending flag If the client sets this flag, the server must modify pending and future payments, *Boolean*
</RECPMTMODRQ>	

12.7.2.2 Response <RECPMTMODRS>

The server sends a Recurring Payment Modification Response in response to a Recurring Payment Modification Request.

The <RECPMTMODRS> response must appear within a <RECPMTTRNRS> transaction wrapper.

	Description
<RECPMTMODRS>	Modification-response aggregate
<RECSRVRTID>	ID assigned by the server to the payment being modified, *SRVRTID*
<RECURRINST>	Recurring-Instructions aggregate, see section 10.2
</RECURRINST>	
<PMTINFO>	Payment-Information aggregate, see section 12.5.2
</PMTINFO>	
<INITIALAMT>	Amount of the initial payment, if different than the following payments, *amount* This amount should be specified as a positive number
<FINALAMT>	Amount of the final payment, if different than the preceding payments, *amount* This amount should be specified as a positive number
<MODPENDING>	Y if the client requested that the server modify pending and future payments. N if the client did not request that the server modify pending and future payments., *Boolean*
</RECPMTMODRS>	

12.7.2.3 Status Codes

Code	*Meaning*
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2006	Source account not found (ERROR)
2007	Source account closed (ERROR)
2008	Source account not authorized (ERROR)
2009	Destination account not found (ERROR)
2010	Destination account closed (ERROR)
2011	Destination account not authorized (ERROR)
2012	Invalid amount (ERROR)
2014	Date too soon (ERROR)

Code	Meaning
2015	Date too far in future (ERROR)
2016	Transaction already committed (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10501	Invalid payee (ERROR)
10502	Invalid payee address (ERROR)
10503	Invalid payee account number (ERROR)
10505	Cannot modify element (ERROR)
10508	Invalid frequency (ERROR)
10511	Invalid payee city (ERROR)
10512	Invalid payee state (ERROR)
10513	Invalid payee postal code (ERROR)
10514	Transaction already processed (ERROR)
10517	Invalid payee name (ERROR)
10518	Unknown model ID (ERROR)
10519	Invalid payee list ID (ERROR)

12.7.3 Recurring Payment Cancellation

The client sends a Recurring Payment Cancellation Request to cancel a recurring payment previously created with a Recurring Payment Request.

12.7.3.1 Request <RECPMTCANCRQ>

The <RECPMTCANCRQ> request must appear within a <RECPMTTRNRQ> transaction wrapper.

Note: A <RECPMTCANCRQ> that cancels pending payments via the <CANPENDING> flag is a compound transaction, and generates the appropriate explicit payment responses that reflect such cancellations, which are returned to the client via synchronization.

	Description
<RECPMTCANCRQ>	Cancellation-request aggregate
<RECSRVRTID>	ID assigned by the server to the payment being canceled, *SRVRTID*
<CANPENDING>	Cancel pending flag, *Boolean* If Y, server should cancel all pending and unspawned payments. If N, server should cancel only the model (and unspawned payments).
</RECPMTCANCRQ>	

12.7.3.2 Response <RECPMTCANCRS>

The server sends a Recurring Payment Cancellation Response in response to a Recurring Payment Cancellation Request.

The <RECPMTCANCRS> response must appear within a <RECPMTTRNRS> transaction wrapper.

	Description
<RECPMTCANCRS>	Modification-request aggregate
<RECSRVRTID>	ID assigned by the server to the payment being modified, *SRVRTID*
<CANPENDING>	Cancel pending flag, *Boolean* Y if the client requested that the server cancel all pending and unspawned payments. N if the client requested that the server cancel only unspawned payments.
</RECPMTCANCRS>	

12.7.3.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2016	Transaction already committed (ERROR)
2017	Already canceled (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10514	Transaction already processed (ERROR)
10518	Unknown model ID (ERROR)

12.8 Payment Mail

Users can correspond by way of e-mail to resolve problems or ask questions about their payments accounts. This function makes use of the general OFX e-mail facility, which is described in Chapter 9, "Customer to FI Communication."

Note: There is no way to indicate non-support of payment e-mail in the profile. A server that doesn't support <PMTMAILSYNCRQ> should return an "empty" payment mail sync response rather than just ignore the request. This empty sync response includes <TOKEN>0 and no history.

12.8.1 Payment Mail Request and Response

12.8.1.1 Request <PMTMAILRQ>

The <PMTMAILRQ> allows a client to issue an e-mail to the payments processor. If the message refers to a specific payment, then both <SRVRTID> and <PMTINFO> are required to identify the payment to the processor.

The <PMTMAILRQ> request must appear within a <PMTMAILTRNRQ> transaction wrapper.

	Description
<PMTMAILRQ>	Payment e-mail-request aggregate
<MAIL>	General e-mail aggregate
</MAIL>	
<SRVRTID>	Transaction ID of the payment that is the subject of the correspondence, *SRVRTID*
<PMTINFO>	Payment Information aggregate, see section 12.5.2
</PMTINFO>	
</PMTMAILRQ>	

12.8.1.2 Response <PMTMAILRS>

The server sends <PMTMAILRS> in response to a Payment E-mail request.

The <PMTMAILRS> response must appear within a <PMTMAILTRNRS> transaction wrapper.

	Description
<PMTMAILRS>	Payment e-mail-response aggregate
<MAIL>	General e-mail aggregate, see Chapter 9, "Customer to FI Communication."
</MAIL>	
<SRVRTID>	Transaction ID of the payment that is the subject of the correspondence, *SRVRTID*
<PMTINFO>	Payment Information aggregate, see section 12.5.2
</PMTINFO>	
</PMTMAILRS>	

12.8.1.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2003	Account not found (ERROR)
2004	Account closed (ERROR)
2005	Account not authorized (ERROR)
2018	Unknown server ID (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
15508	Transaction not authorized (ERROR)
16500	HTML not allowed (ERROR)
16501	Unknown mail To: (ERROR)

12.8.2 Payment Mail Synchronization

Payment mail is subject to synchronization. The scope of the synchronization request is all of the accounts for which the user might have sent mail, not a specific account.

12.8.2.1 Request <PMTMAILSYNCRQ>

	Description
<PMTMAILSYNCRQ> *Client synchronization option; <TOKEN>, <TOKENONLY>, or <REFRESH>*	Synchronization-request aggregate
<TOKEN>	Previous value of <TOKEN> received for this type of synchronization request from server; 0 for first-time requests; *token*
<TOKENONLY>	Request for just the current <TOKEN> without the history, *Boolean*
<REFRESH>	Request for refresh of current state, *Boolean*
<REJECTIFMISSING>	If Y, do not process requests if client <TOKEN> is out of date, *Boolean*
<INCIMAGES>	Y if the client accepts mail with images in the message body. N if the client does not accept mail with images in the message body. *Boolean*
<USEHTML>	Y if client wants an HTML response, N if client wants plain text, *Boolean*
<PMTMAILTRNRQ>	Payment-mail transactions (0 or more)
</PMTMAILTRNRQ>	
</PMTMAILSYNCRQ>	

12.8.2.2 Response <PMTMAILSYNCRS>

	Description
<PMTMAILSYNCRS>	Synchronization-response aggregate
<TOKEN>	New synchronization token, *token*
<LOSTSYNC>	Y if the token in the synchronization request is older than the earliest entry in the server's history table. In this case, some responses have been lost. N if the token in the synchronization request is newer than or matches a token in the server's history table. *Boolean*
<PMTMAILTRNRS>	Payment-mail transactions (0 or more)
</PMTMAILTRNRS>	
</PMTMAILSYNCRS>	

12.9 Payee Lists

Payments-system servers store lists of payees set up for payment by each user. Some systems require this before the user can issue a payment to a payee. In other payment systems, this feature enables the sharing of payee entry among multiple clients, and simplifies server payee maintenance.

A server-assigned payee list-entry ID identifies entries in the payee list. The following set of messages allows clients to obtain this list of payees. Users can add, modify, and delete individual entries in the list. The user-defined Payee list is subject to synchronization, so that multiple clients can use the list.

Creating a payee:

	Server Responds
Server-assigned payee identifier, or payee billing address User's account number with the payee	 Payee address Standard payee information

Modifying a payee:

	Server Responds
Server-assigned payee identifier User's account number with the payee Information to change: payee name address city state postal code phone number new payee account #	
	Extended payee information if payee is a standard payee or has non-default processing lead times Acknowledgment or Error

Deleting a payee:

Client Sends	*Server Responds*
Server-assigned payee identifier User's account number with the payee	
	Acknowledgment or Error

12.9.1 Adding a Payee to the Payee List

The user can use the Payee Request to add a payee to the server payee list. The server responds with a Payee Response, which can contain a complete billing address for the payee, or if the payee is a standard payee, the lead-time and payee name values.

12.9.1.1 Payee Request <PAYEERQ>

The <PAYEERQ> requests the addition of a payee entry to the server's payee list. Note that the user can use a <PMTRQ> to simultaneously set up a payee. OFX does not require the client to send a <PAYEERQ> before making an initial <PMTRQ> to a payee.

The <PAYEERQ> request must appear within a <PAYEETRNRQ> transaction wrapper.

	Description
<PAYEERQ> *Specify payee; either <PAYEEID> or <PAYEE>.*	Payee-request aggregate
<PAYEEID>	Server-assigned payee identifier, *A-12*
<PAYEE> **</PAYEE>**	Complete payee billing information, see section 12.5.2.1.
<BANKACCTTO> </BANKACCTTO>	The destination bank account (see section 11.3.1), specified in countries that pay using transfers. The <PAYEE> (above) must also be specified.
<PAYACCT>	User's account number(s) with the payee (0 or more), *A-32*
</PAYEERQ>	

12.9.1.2 Payee Response <PAYEERS>

The server sends the Payee Response in response to a Payee Request. It contains the full billing information for the payee if it is not a standard payee. Otherwise, it contains the standard payee information, including lead time and payee name. If the server identifies the payee as having an assigned payee ID, then the server will include the <EXTDPAYEE> aggregate in the response.

If the response indicates that the payee does not have an assigned <PAYEEID>, the client should specify the full billing address <PAYEE> information in subsequent payment requests to that payee or when that payee is being modified. Otherwise the <PAYEEID> is used in lieu of the <PAYEE> aggregate.

If the response indicates that the payee does have a <PAYEEID>, then the client should use the <PAYEEID> for making payments to that payee.

The <PAYEERS> response must appear within a <PAYEETRNRS> transaction wrapper.

	Description
<PAYEERS>	Payee-response aggregate
<PAYEELSTID>	Server-assigned record ID for this payee record, *A-12*
<PAYEE>	Complete payee billing information, see section 12.5.2.1.
</PAYEE>	
<BANKACCTTO>	The destination bank account (see section 11.3.1), specified in countries that pay using transfers. The <PAYEE> (above) must also be specified.
</BANKACCTTO>	
<EXTDPAYEE>	Extended payee information, see section 12.5.2.6
</EXTDPAYEE>	
<PAYACCT>	User's account number(s) with the payee (0 or more), *A-32*
</PAYEERS>	

12.9.1.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2001	Invalid account (ERROR)
2002	General account error (ERROR)
2009	Destination account not found (ERROR)
2010	Destination account closed (ERROR)
2011	Destination account not authorized (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10501	Invalid payee (ERROR)
10502	Invalid payee address (ERROR)
10503	Invalid payee account number (ERROR)
10511	Invalid payee city (ERROR)
10512	Invalid payee state (ERROR)
10513	Invalid payee postal code (ERROR)

12.9.2 Payee Modification

The Payee Modification Request allows the client to make changes to payee entries in the server's payee list. Payments spawned from a model after the payee modification will use the updated information from the server's payee list as modified by the Payee Modification request.

12.9.2.1 Request <PAYEEMODRQ>

The client sends the Payee Modification Request to request changes to an existing payee list entry. The <PAYEE> aggregate must specify the changed and unchanged payee information. Absence of a <PAYACCT> in a <PAYEEMODRQ> could be interpreted as an implicit disassociation of the <PAYACCT> with the payee. Presence or absence of a <PAYACCT> does not imply selective <PAYEE> aggregate changes for the same <PAYEELSTID> as referenced by more than one <PAYACCT>. The <PAYEEMODRQ> request must appear within a <PAYEETRNRQ> transaction wrapper.

A payee modification may also affect models (though not their pending payments). In this case, a server must create and store <RECPMTMODRS> responses, to be returned to the client in subsequent recurring synchronization responses. See section 12.2.5 for further discussion of implicit payee changes.

	Description
<PAYEEMODRQ>	Modification-request aggregate
<PAYEELSTID>	Server-assigned record ID for this payee record, *A-12*
<PAYEE>	Payee information to modify
</PAYEE>	
<BANKACCTTO>	Destination account (see section 11.3.1) for countries that pay using transfers (<PAYEE> required)
</BANKACCTTO>	
<PAYACCT>	Payer account number(s) with the payee (0 or more), *A-32*
</PAYEEMODRQ>	

12.9.2.2 Response <PAYEEMODRS>

The server returns a Payee Modification Response in reply to a Payee Modification Request.

When a server-initiated change occurs to the extended payee information for a payee (for example a change in the payee's lead-time), the server can include this information in the <EXTDPAYEE> of the response.

If a server-initiated response indicates either that a payee now has a payee ID, or no longer has one, then the client should use the appropriate form of designating the payee in any future payment requests <PMTRQ> to that payee.

The <PAYEEMODRS> response must appear within a <PAYEETRNRS> transaction wrapper.

	Description
<PAYEEMODRS>	Modification-response aggregate
<PAYEELSTID>	Server-assigned record ID for this payee record, *A-12*
<PAYEE>	Payee information that was modified, see section 12.5.2.1
</PAYEE>	
<BANKACCTTO>	Destination account (see section 11.3.1) for countries that pay bills using transfers (<PAYEE> required as well)
</BANKACCTTO>	
<PAYACCT>	Payer's account number(s) with the payee (0 or more), *A-32*
<EXTDPAYEE>	Extended payee information, see section 12.5.2.6
</EXTDPAYEE>	
</PAYEEMODRS>	

12.9.2.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2001	Invalid account (ERROR)
2002	General account error (ERROR)
2009	Destination account not found (ERROR)
2010	Destination account closed (ERROR)
2011	Destination account not authorized (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10501	Invalid payee (ERROR)
10502	Invalid payee address (ERROR)
10503	Invalid payee account number (ERROR)
10510	Invalid payee ID (ERROR)
10511	Invalid payee city (ERROR)
10512	Invalid payee state (ERROR)
10513	Invalid payee postal code (ERROR)
10515	Payee not modifiable by client (ERROR)

12.9.3 Payee Deletion

The Payee Deletion Request allows a client to delete a payee entry from the server's list of the user's payees. To delete specific <PAYACCT> associations with a payee, clients should use the <PAYEELSTID> combined with absent <PAYACCT>s via a <PAYEEMODRQ>.

The Payee delete request does not cancel payments that are pending at the time of the payee's deletion. References to pending payments subsequent to a payee's deletion pose issues regarding <PAYEELSTID> assignment at both the client and server levels. Therefore, it is suggested that the client disallow payee deletes if there are pending payments/models.

12.9.3.1 Request <PAYEEDELRQ>

The <PAYEEDELRQ> requests the deletion of a payee entry.

The <PAYEEDELRQ> request must appear within a <PAYEETRNRQ> transaction wrapper.

	Description
<PAYEEDELRQ>	Deletion-request aggregate
<PAYEELSTID>	Server-assigned record ID for this payee record, *A-12*
</PAYEEDELRQ>	

12.9.3.2 Response <PAYEEDELRS>

The server sends the Payee Deletion Response <PAYEEDELRS> in response to a Payee Deletion Request <PAYEEDELRQ>.

The <PAYEEDELRS> response must appear within a <PAYEETRNRS> transaction wrapper.

	Description
<PAYEEDELRS>	Deletion-response aggregate
<PAYEELSTID>	Server-assigned record ID for this payee record, *A-12*
</PAYEEDELRS>	

12.9.3.3 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2019	Duplicate request (ERROR)
6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)
10519	Invalid payee list ID (ERROR)

12.9.4 Payee List Synchronization

This message allows clients to obtain a list of payees stored on the server that it has configured for use in payments. In a “pay some” system, users are sometimes required to explicitly configure the payees to whom the system will make payments. This can be done by means of a telephone call to the payments provider or through some other interface. The client can then use this message to obtain the user’s list of configured payees. In other systems, the payments provider can elect to store a list of all payees that the user has paid. This is a convenience to the client. It allows payees set up on one client to be accessible from a user’s other clients and ensures each client has the latest version of this list.

12.9.4.1 Request <PAYEESYNCRQ>

	Description
<PAYEESYNCRQ> *Client synchronization option; <TOKEN>, <TOKENONLY>, or <REFRESH>*	Payee-list request aggregate
<TOKEN>	Previous value of <TOKEN> received for this type of synchronization request from server; 0 for first-time requests; *token*
<TOKENONLY>	Request for just the current <TOKEN> without the history, *Boolean*
<REFRESH>	Request for refresh of current state, *Boolean*
<REJECTIFMISSING>	If Y, do not process requests if client <TOKEN> is out of date, *Boolean*
<PAYEETRNRQ> </PAYEETRNRQ>	Payee transactions (0 or more)
</PAYEESYNCRQ>	

12.9.4.2 Response <PAYEESYNCRS>

	Description
<PAYEESYNCRS>	Payee-list response aggregate
<TOKEN>	New synchronization token, *token*
<LOSTSYNC>	Y if the token in the synchronization request is older than the earliest entry in the server's history table. In this case, some responses have been lost. N if the token in the synchronization request is newer than or matches a token in the server's history table. *Boolean*
<PAYEETRNRS>	Payee transactions (0 or more)
</PAYEETRNRS>	
</PAYEESYNCRS>	

12.10 Data Synchronization for Payments

Users of OFX Payments need to be able to obtain the current status of transactions previously sent to the server for processing. For example, once the user schedules a payment and the payment date has passed, the user might want to verify that the server made the payment as directed. Additionally, OFX allows for interactions with the server through multiple clients. This means, for example, that the user can perform some transactions from a home PC and others from an office computer with each session incorporating the activities performed on the other.

In order to accomplish these tasks, the client uses a synchronization scheme to insure that it has an accurate copy of the server data that is relevant to the client application. The intent of this scheme is to address three scenarios in which the client might lose synchronization with the server:

- A transaction has changed its state based on processing actions on the server. For example, a scheduled payment has passed its due date and has been paid or rejected.
- Transactions relevant to the client's application state have been added, deleted, or modified by a second client. For example, a user might enter or change transactions from more than one PC or application.
- A communications session between the client and server was interrupted or completed abnormally. As a result the client does not have responses from the server indicating that all the transactions were received and processed.

  Note: Except for the <REFRESH>Y sync response, no payee information in any particular response in a sync should have changed from that in the response when it was originally sent. In other words, if a <PMTMODRS> caused a change to that payment's payee address, the original <PMTRS> in the sync should have the old address in it. The <PMTMODRS>, appearing later in the sync, would cause the client to update the payment appropriately.

12.10.1 Payment Synchronization

12.10.1.1 Request <PMTSYNCRQ>

	Description
<PMTSYNCRQ> *Client synchronization option; <TOKEN>, <TOKENONLY>, or <REFRESH>*	Synchronization-request aggregate
<TOKEN>	Previous value of <TOKEN> received for this type of synchronization request from server; 0 for first-time requests; *token*
<TOKENONLY>	Request for just the current <TOKEN> without the history, *Boolean*
<REFRESH>	Request for refresh of current state, *Boolean*
<REJECTIFMISSING>	If Y, do not process requests if client <TOKEN> is out of date, *Boolean*
<BANKACCTFROM>	Opening tag for account from aggregate, see section 11.3.1
</BANKACCTFROM>	
<PMTTRNRQ>	Payment transactions (0 or more)
</PMTTRNRQ>	
</PMTSYNCRQ>	

12.10.1.2 Response <PMTSYNCRS>

	Description
<PMTSYNCRS>	Synchronization-response aggregate
<TOKEN>	New synchronization token, *token*
<LOSTSYNC>	Y if the token in the synchronization request is older than the earliest entry in the server's history table. In this case, some responses have been lost. N if the token in the synchronization request is newer than or matches a token in the server's history table. *Boolean*
<BANKACCTFROM>	Opening tag for account from aggregate, see section 11.3.1
</BANKACCTFROM>	
<PMTTRNRS>	Payment transactions (0 or more)
</PMTTRNRS>	
</PMTSYNCRS>	

12.10.2 Recurring Payment Synchronization

12.10.2.1 Request <RECPMTSYNCRQ>

	Description
<RECPMTSYNCRQ> *Client synchronization option; <TOKEN>, <TOKENONLY>, or <REFRESH>*	Synchronization-request aggregate
<TOKEN>	Previous value of <TOKEN> received for this type of synchronization request from server; 0 for first-time requests; *token*
<TOKENONLY>	Request for just the current <TOKEN> without the history, *Boolean*
<REFRESH>	Request for refresh of current state, *Boolean*
<REJECTIFMISSING>	If Y, do not process requests if client <TOKEN> is out of date, *Boolean*
<BANKACCTFROM>	Opening tag for account from aggregate, see section 11.3.1
</BANKACCTFROM>	
<RECPMTTRNRQ>	Recurring-payment transactions (0 or more)
</RECPMTTRNRQ>	
</RECPMTSYNCRQ>	

12.10.2.2 Response <RECPMTSYNCRS>

	Description
<RECPMTSYNCRS>	Synchronization-response aggregate
<TOKEN>	New synchronization token, *token*
<LOSTSYNC>	Y if the token in the synchronization request is older than the earliest entry in the server's history table. In this case, some responses have been lost. N if the token in the synchronization request is newer than or matches a token in the server's history table. *Boolean*
<BANKACCTFROM>	Opening tag for account from aggregate, see section 11.3.1
</BANKACCTFROM>	
<RECPMTTRNRS>	Recurring-payment transactions (0 or more)
</RECPMTTRNRS>	
</RECPMTSYNCRS>	

12.10.3 Discussion

This section describes specific synchronization processing for the OFX Payments functions. Chapter 6, "Data Synchronization," provides a more extensive discussion of the OFX synchronization mechanism.

The client follows the steps below to synchronize:

1. The client sends a <PMTSYNCRQ> and/or <RECPMTSYNCRQ> containing the token it has stored from its last successful synchronization (or the special initial token value).
2. The client processes the <PMTSYNCRS> and/or <RECPMTSYNCRS> response from the server.

When the client has requested the server to add a transaction, a response that contains a <TRNUID> matching a transaction originally sent by the client—for which the client has not recorded an associated <SRVRTID>—is the normal scenario. This scenario could also occur if the server response did not reach the client in the previous session. In either case, the client should add these server IDs to their associated transactions at this point.

If the client previously recorded the <SRVRTID>, this response is a change in status or in the contents of the transaction. The request might have originated from this client, another client, or might be the result of server processing.

If the <TRNUID> does not match any transaction known to the client, a second client initiated this transaction. In rarer cases the response might be a transaction initially requested by this client, for which the client has lost its record; for example, the client has reverted to a backup.

12.11.2 Bill Pay Message Set Profile <BILLPAYMSGSET>

Tag	*Description*

Tag	Description
<PMTBYXFER>	The payment provider supports payments to payees identified by destination account, *Boolean*
<PMTBYPAYEEID>	The payment provider supports payments to payees identified by a server-supplied payee ID, *Boolean*
<CANADDPAYEE>	User can add payees. if no, the user is restricted to payees added to the user's payee list by the payment system, *Boolean*
<HASEXTDPMT>	Supports the EXTDPMT business payment aggregate, *Boolean*
<CANMODPMTS>	Permits modifications to payments, that is PMTMODRQ, *Boolean*
<CANMODMDLS>	Permits modifications to models, that is REQPMTMODRQ, *Boolean*
<DIFFFIRSTPMT>	Support for specifying a different amount for the first payment generated by a model, *Boolean*
<DIFFLASTPMT>	Support for specifying a different amount for the last payment generated by a model, *Boolean*
<BILLPUBCONTEXT>	The payment provider supports bill presentment context information in payment operations, *Boolean*
</BILLPAYMSGSETV1>	
</BILLPAYMSGSET>	

12.12 Examples

12.12.1 Scheduling a Payment

Create a payment to “J.C. Counts” for $123.45 to be paid on October 1, 2005 using funds in a checking account:

```
<!-- payment example 1 -->
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>                    <!-- ...Sign on request. For a
                               complete example, see section
                               11.14.1-->
    </SONRQ>
  </SIGNONMSGSRQV1>
  <BILLPAYMSGSRQV1>
    <PMTTRNRQ>
      <TRNUID>1001</TRNUID>
      <PMTRQ>
        <PMTINFO>
          <BANKACCTFROM>
            <BANKID>123432123</BANKID>
            <ACCTID>516273</ACCTID>
            <ACCTTYPE>CHECKING</ACCTTYPE>
          </BANKACCTFROM>
          <TRNAMT>123.45</TRNAMT>
          <PAYEE>
            <NAME>J. C. Counts</NAME>
            <ADDR1>100 Main St.</ADDR1>
            <CITY>Turlock</CITY>
            <STATE>CA</STATE>
            <POSTALCODE>90101</POSTALCODE>
            <PHONE>415.987.6543</PHONE>
          </PAYEE>
          <PAYACCT>10101</PAYACCT>
          <DTDUE>20051001</DTDUE>
          <MEMO>payment #3</MEMO>
        </PMTINFO>
      </PMTRQ>
    </PMTTRNRQ>
  </BILLPAYMSGSRQV1>
</OFX>
```

The server responds, indicating that it will make the payment on the date requested and that the payee is a standard payee:

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>                      <!-- ...Sign on response. For a
                                 complete example, see section
                                 11.14.1-->
    </SONRS>
  </SIGNONMSGSRSV1>
  <BILLPAYMSGSRSV1>
    <PMTTRNRS>
      <TRNUID>1001</TRNUID>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <PMTRS>
        <SRVRTID>1030155</SRVRTID>
        <PAYEELSTID>123214<PAYEELSTID>
        <CURDEF>USD</CURDEF>
        <PMTINFO>
          <BANKACCTFROM>
            <BANKID>123432123</BANKID>
            <ACCTID>516273</ACCTID>
            <ACCTTYPE>CHECKING</ACCTTYPE>
          </BANKACCTFROM>
          <TRNAMT>123.45</TRNAMT>
          <PAYEE>
            <NAME>J. C. Counts</NAME>
            <ADDR1>100 Main St.</ADDR1>
            <CITY>Turlock</CITY>
            <STATE>CA</STATE>
            <POSTALCODE>90101</POSTALCODE>
            <PHONE>415.987.6543</PHONE>
          </PAYEE>
          <PAYEELSTID>123214</PAYEELSTID>
          <PAYACCT>10101</PAYACCT>
          <DTDUE>20051001</DTDUE>
          <MEMO>payment #3</MEMO>
        </PMTINFO>
        <EXTDPAYEE>
          <PAYEEID>9076</PAYEEID>
```

```
            <IDSCOPE>USER</IDSCOPE>
            <NAME>J. C. Counts</NAME>
            <DAYSTOPAY>3</DAYSTOPAY>
          </EXTDPAYEE>
          <PMTPRCSTS>
            <PMTPRCCODE>WILLPROCESSON</PMTPRCCODE>
            <DTPMTPRC>20050928</DTPMTPRC>
          </PMTPRCSTS>
        </PMTRS>
      </PMTTRNRS>
    </BILLPAYMSGSRSV1>
</OFX>
```

Create a second payment to the payee, using the payee ID returned in the previous example:

```
<!-- payment example 2 -->


<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>                     <!-- ...Sign on request. For a
                                complete example, see section
                                11.14.1-->
    </SONRQ>
  </SIGNONMSGSRQV1>
  <BILLPAYMSGSRQV1>
    <PMTTRNRQ>
      <TRNUID>1002</TRNUID>
      <PMTRQ>
        <PMTINFO>
          <BANKACCTFROM>
            <BANKID>123432123</BANKID>
            <ACCTID>516273</ACCTID>
            <ACCTTYPE>CHECKING</ACCTTYPE>
          </BANKACCTFROM>
          <TRNAMT>123.45</TRNAMT>
          <PAYEEID>9076</PAYEEID>
          <PAYEELSTID>123214</PAYEELSTID>
          <PAYACCT>10101</PAYACCT>
          <DTDUE>20051101</DTDUE>
          <MEMO>Payment #4</MEMO>
        </PMTINFO>
      </PMTRQ>
    </PMTTRNRQ>
```

```
  </BILLPAYMSGSRQV1>
</OFX>
```

The server responds, indicating that it will make the payment on the date requested:

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>                    <!-- ...Sign on response. For a
                               complete example, see section
                               11.14.1-->
    </SONRS>
  </SIGNONMSGSRSV1>
  <BILLPAYMSGSRSV1>
    <PMTTRNRS>
      <TRNUID>1002</TRNUID>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <PMTRS>
        <SRVRTID>1068405<SRVRTID>
        <PAYEELSTID>123214</PAYEELSTID>
        <CURDEF>USD</CURDEF>
        <PMTINFO>
          <BANKACCTFROM>
             <BANKID>123432123</BANKID>
             <ACCTID>516273</ACCTID>
             <ACCTTYPE>CHECKING</ACCTTYPE>
          </BANKACCTFROM>
          <TRNAMT>123.45</TRNAMT>
          <PAYEEID>9076</PAYEEID>
          <PAYEELSTID>123214</PAYEELSTID>
          <PAYACCT>10101</PAYACCT>
          <DTDUE>20051101</DTDUE>
          <MEMO>payment #4</MEMO>
        </PMTINFO>
        <EXTDPAYEE>
          <PAYEEID>9076</PAYEEID>
          <IDSCOPE>USER</IDSCOPE>
          <NAME>J. C. Counts</NAME>
          <DAYSTOPAY>3</DAYSTOPAY>
        </EXTDPAYEE>
        <PMTPRCSTS>
```

```
            <PMTPRCCODE>WILLPROCESSON</PMTPRCCODE>
            <DTPMTPRC>20051029</DTPMTPRC>
          </PMTPRCSTS>
        </PMTRS>
      </PMTTRNRS>
    </BILLPAYMSGSRSV1>
</OFX>
```

12.12.2 Modifying a Payment

Change the amount of the first payment to $125.99

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>                    <!-- ...Sign on request. For a
                               complete example, see section
                               11.14.1-->
    </SONRQ>
  </SIGNONMSGSRQV1>
  <BILLPAYMSGSRQV1>
    <PMTTRNRQ>
      <TRNUID>1021</TRNUID>
      <PMTMODRQ>
        <SRVRTID>1030155</SRVRTID>
        <PMTINFO>
          <BANKACCTFROM>
            <BANKID>123432123</BANKID>
            <ACCTID>516273</ACCTID>
            <ACCTTYPE>CHECKING</ACCTTYPE>
          </BANKACCTFROM>
          <TRNAMT>125.99</TRNAMT><!-- changed amount -->
          <PAYEEID>9076</PAYEEID>
          <PAYEELSTID>123214</PAYEELSTID>
          <PAYACCT>10101</PAYACCT>
          <DTDUE>20051001</DTDUE>
          <MEMO>payment #3</MEMO>
        </PMTINFO>
      </PMTMODRQ>
    </PMTTRNRQ>
  </BILLPAYMSGSRQV1>
</OFX>
```

The server responds:

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>                     <!-- ...Sign on response. For a
                                complete example, see section
                                11.14.1-->
    </SONRS>
  </SIGNONMSGSRSV1>
  <BILLPAYMSGSRSV1>
    <PMTTRNRS>
      <TRNUID>1021</TRNUID>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <PMTMODRS>
        <SRVRTID>1030155</SRVRTID>
        <PMTINFO>
          <BANKACCTFROM>
            <BANKID>123432123</BANKID>
            <ACCTID>516273</ACCTID>
            <ACCTTYPE>CHECKING</ACCTTYPE>
          </BANKACCTFROM>
          <TRNAMT>125.99</TRNAMT><!-- changed amount -->
          <PAYEEID>9076</PAYEEID>
          <PAYEELSTID>123214</PAYEELSTID>
          <PAYACCT>10101</PAYACCT>
          <DTDUE>20051001</DTDUE>
          <MEMO>payment #3</MEMO>
        </PMTINFO>
        <PMTPRCSTS>
          <PMTPRCCODE>WILLPROCESSON</PMTPRCCODE>
          <DTPMTPRC>20050928</DTPMTPRC>
        </PMTPRCSTS>
      </PMTMODRS>
    </PMTTRNRS>
  </BILLPAYMSGSRSV1>
</OFX>
```

Change the date of the same payment to December 12, 2005.

```
<OFX>
```

```
  <SIGNONMSGSRQV1>
    <SONRQ>                     <!-- ...Sign on request. For a
                                complete example, see section
                                11.14.1-->
    </SONRQ>
  </SIGNONMSGSRQV1>
  <BILLPAYMSGSRQV1>
    <PMTTRNRQ>
      <TRNUID>32456</TRNUID>
      <PMTMODRQ>
        <SRVRTID>1030155</SRVRTID>
        <PMTINFO>
          <BANKACCTFROM>
             <BANKID>123432123</BANKID>
             <ACCTID>516273</ACCTID>
             <ACCTTYPE>CHECKING</ACCTTYPE>
          </BANKACCTFROM>
          <TRNAMT>125.99</TRNAMT>
          <PAYEEID>9076</PAYEEID>
          <PAYEELSTID>123214</PAYEELSTID>
          <PAYACCT>10101</PAYACCT>
          <DTDUE>20051212</DTDUE><!-- changed date -->
          <MEMO>payment #3</MEMO>
        </PMTINFO>
      </PMTMODRQ>
    </PMTTRNRQ>
  </BILLPAYMSGSRQV1>
</OFX>
```

The server responds:

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>                     <!-- ...Sign on response. For a
                                complete example, see section
                                11.14.1-->
    </SONRS>
  </SIGNONMSGSRSV1>
  <BILLPAYMSGSRSV1>
    <PMTTRNRS>
      <TRNUID>32456</TRNUID>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</INFO>
```

```
    </BILLPAYMSGSRQV1>
</OFX>
```

The server responds:

```
<OFX>
    <SIGNONMSGSRSV1>
        <SONRS>                          <!-- ...Sign on response. For a
                                         complete example, see section
                                         11.14.1-->
        </SONRS>
    </SIGNONMSGSRSV1>
    <BILLPAYMSGSRSV1>
        <PMTTRNRS>
            <TRNUID>54601</TRNUID>
            <STATUS>
                <CODE>0</CODE>
                <SEVERITY>INFO</SEVERITY>
            </STATUS>
            <PMTCANCRS>
                <SRVRTID>1030155</SRVRTID>
            </PMTCANCRS>
        </PMTTRNRS>
    </BILLPAYMSGSRSV1>
</OFX>
```

12.12.4 Updating Payment Status

Update payment status:

```
<OFX>
    <SIGNONMSGSRQV1>
        <SONRQ>                          <!-- ...Sign on request. For a
                                         complete example, see section
                                         11.14.1-->
        </SONRQ>
    </SIGNONMSGSRQV1>
    <BILLPAYMSGSRQV1>
        <PMTINQTRNRQ>
            <TRNUID>7865</TRNUID>
            <PMTINQRQ>
                <SRVRTID>565321</SRVRTID>
            </PMTINQRQ>
        </PMTINQTRNRQ>
```

```
  </BILLPAYMSGSRQV1>
</OFX>
```

The server responds with updated status and check number:

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>                     <!-- ...Sign on response. For a
                                complete example, see section
                                11.14.1-->
    </SONRS>
  </SIGNONMSGSRSV1>
  <BILLPAYMSGSRSV1>
    <PMTINQTRNRS>
      <TRNUID>7865</TRNUID>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <PMTINQRS>
        <SRVRTID>565321</SRVRTID>
        <PMTPRCSTS>
          <PMTPRCCODE>PROCESSEDON</PMTPRCCODE>
          <DTPMTPRC>20050201</DTPMTPRC>
        </PMTPRCSTS>
        <CHECKNUM>6017</CHECKNUM>
      </PMTINQRS>
    </PMTINQTRNRS>
  </BILLPAYMSGSRSV1>
</OFX>
```

12.12.5 Scheduling a Recurring Payment

Create a recurring payment of 36 monthly payments of $395 to a (previously known) standard payee. The first payment will be on November 15, 2005:

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>                     <!-- ...Sign on request. For a
                                complete example, see section
                                11.14.1-->
    </SONRQ>
  </SIGNONMSGSRQV1>
  <BILLPAYMSGSRQV1>
```

```
    <RECPMTTRNRQ>
      <TRNUID>12354</TRNUID>
      <RECPMTRQ>
        <RECURRINST>
          <NINSTS>36</NINSTS/
          <FREQ>MONTHLY</FREQ>
        </RECURRINST>
        <PMTINFO>
          <BANKACCTFROM>
            <BANKID>555432180</BANKID>
            <ACCTID>763984</ACCTID>
            <ACCTTYPE>CHECKING</ACCTTYPE>
          </BANKACCTFROM>
          <TRNAMT>395.00</TRNAMT>
          <PAYEEID>77810</PAYEEID>
          <PAYEELSTID>27983</PAYEELSTID>
          <PAYACCT>444-78-97572</PAYACCT>
          <DTDUE>20051115</DTDUE>
          <MEMO>Auto loan payment</MEMO>
        </PMTINFO>
      </RECPMTRQ>
    </RECPMTTRNRQ>
  </BILLPAYMSGSRQV1>
</OFX>
```

The server responds with the assigned server transaction ID:

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>                    <!-- ...Sign on response. For a
                               complete example, see section
                               11.14.1-->
    </SONRS>
  </SIGNONMSGSRSV1>
  <BILLPAYMSGSRSV1>
    <RECPMTTRNRS>
      <TRNUID>12345</TRNUID>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</INFO>
      </STATUS>
      <RECPMTRS>
        <RECSRVRTID>387687138</RECSRVRTID>
```

```
    <RECSRVRTID>387687138</RECSRVRTID>
    <RECURRINST>
      <NINSTS>36</NINSTS>
      <FREQ>MONTHLY</FREQ>
    </RECURRINST>
    <PMTINFO>
      <BANKACCTFROM>
        <BANKID>555432180</BANKID>
        <ACCTID>763984</ACCTID>
        <ACCTTYPE>CHECKING</ACCTTYPE>
      </BANKACCTFROM>
      <TRNAMT>399.95</TRNAMT><!-- changing amount -->
      <PAYEEID>77810</PAYEEID>
      <PAYEELSTID>27983</PAYEELSTID>
      <PAYACCT>444-78-97572</PAYACCT>
      <DTDUE>20051115</DTDUE>
      <MEMO>Auto loan payment</MEMO>
    </PMTINFO>
    <MODPENDING>N</MODPENDING>
  </RECPMTMODRS>
```

```
      </RECPMTTRNRS>
   </BILLPAYMSGSRSV1>
</OFX>
```

12.12.7 Canceling a Recurring Payment

Cancel a recurring payment:

```
<OFX>
   <SIGNONMSGSRQV1>
      <SONRQ>                          <!-- ...Sign on request. For a
                                       complete example, see section
                                       11.14.1-->
      </SONRQ>
   </SIGNONMSGSRQV1>
   <BILLPAYMSGSRQV1>
      <RECPMTTRNRQ>
         <TRNUID>11122</TRNUID>
         <RECPMTCANCRQ>
            <RECSRVRTID>387687138</RECSRVRTID>
            <CANPENDING>Y</CANPENDING>
         </RECPMTCANCRQ>
      </RECPMTTRNRQ>
   </BILLPAYMSGSRQV1>
</OFX>
```

The server responds:

```
<OFX>
   <SIGNONMSGSRSV1>
      <SONRS>                          <!-- ...Sign on response. For a
                                       complete example, see section
                                       11.14.1-->
      </SONRS>
   </SIGNONMSGSRSV1>
   <BILLPAYMSGSRSV1>
      <RECPMTTRNRS>
         <TRNUID>11122</TRNUID>
         <STATUS>
            <CODE>0</CODE>
            <SEVERITY>INFO</SEVERITY>
         </STATUS>
         <RECPMTCANCRS>
            <RECSRVRTID>387687138</RECSRVRTID>
```

```
          <CANPENDING>Y</CANPENDING>
        </RECPMTCANCRS>
      </RECPMTTRNRS>
    </BILLPAYMSGSRSV1>
</OFX>
```

12.12.8 Adding a Payee to the Payee List

The user sends a request to add a payee to the user's payee list:

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>                     <!-- ...Sign on request. For a
                                complete example, see section
                                11.14.1-->
    </SONRQ>
  </SIGNONMSGSRQV1>
  <BILLPAYMSGSRQV1>
    <PAYEETRNRQ>
      <TRNUID>127677</TRNUID>
      <PAYEERQ>
        <PAYEE>
          <NAME>ACME Rocket Works</NAME>
          <ADDR1>101 Spring St.</ADDR1>
          <ADDR2>Suite 503</ADDR2>
          <CITY>Watkins Glen</CITY>
          <STATE>NY</STATE>
          <POSTALCODE>12345-6789</POSTALCODE>
          <PHONE>888.555.1212</PHONE>
        </PAYEE>
        <PAYACCT>1001-99-8876</PAYACCT>
      </PAYEERQ>
    </PAYEETRNRQ>
  </BILLPAYMSGSRQV1>
</OFX>
```

The server responds:

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>                     <!-- ...Sign on response. For a
                                complete example, see section
                                11.14.1-->
    </SONRS>
  </SIGNONMSGSRSV1>
  <BILLPAYMSGSRSV1>
    <PAYEETRNRS>
      <TRNUID>127677</TRNUID>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <PAYEERS>
        <PAYEELSTID>78096786</PAYEELSTID>
        <PAYEE>
          <NAME>ACME Rocket Works</NAME>
          <ADDR1>101 Spring St.</ADDR1>
          <ADDR2>Suite 503</ADDR2>
          <CITY>Watkins Glen</CITY>
          <STATE>NY</STATE>
          <POSTALCODE>12345-6789</POSTALCODE>
          <PHONE>888.555.1212</PHONE>
        </PAYEE>
        <EXTDPAYEE>
          <PAYEEID>88878</PAYEEID>
          <IDSCOPE>GLOBAL</IDSCOPE>
          <NAME>ACME Rocket Works, Inc.</NAME>
          <DAYSTOPAY>2</DAYSTOPAY>
        </EXTDPAYEE>
        <PAYACCT>1001-99-8876</PAYACCT>
      </PAYEERS>
    </PAYEETRNRS>
  </BILLPAYMSGSRSV1>
</OFX>
```

12.12.9 Synchronizing Scheduled Payments

A client wishes to obtain all Payments active on the server for a particular account:

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>                    <!-- ...Sign on request. For a
                               complete example, see section
                               11.14.1-->
    </SONRQ>
  </SIGNONMSGSRQV1>
  <BILLPAYMSGSRQV1>
    <PMTSYNCRQ>
      <REFRESH>Y</REFRESH>
      <REJECTIFMISSING>N</REJECTIFMISSING>
      <BANKACCTFROM>
        <BANKID>123432123</BANKID>
        <ACCTID>516273</ACCTID>
        <ACCTTYPE>CHECKING</ACCTTYPE>
      </BANKACCTFROM>
    </PMTSYNCRQ>
  </BILLPAYMSGSRQV1>
</OFX>
```

Assuming the only activity on this account has been the two payments created above, the server responds with one payment since the other payment was cancelled. The server also includes the current <TOKEN> value.

> **Note:** If the one outstanding payment had a modification to it, the modification should have been integrated into the one <PMTRS> since this is a refresh, not a sync of all history. In that case, <TRNUID>0 must be returned in the response transaction (no client initiated an exact matching transaction).

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>                    <!-- ...Sign on response. For a
                               complete example, see section
                               11.14.1-->
    </SONRS>
  </SIGNONMSGSRSV1>
  <BILLPAYMSGSRSV1>
    <PMTSYNCRS>
      <TOKEN>3247989384</TOKEN>
      <BANKACCTFROM>
        <BANKID>123432123</BANKID>
```

```
          <ACCTID>516273</ACCTID>
          <ACCTTYPE>CHECKING</ACCTTYPE>
        </BANKACCTFROM>
        <PMTTRNRS>
          <TRNUID>0</TRNUID>
          <STATUS>
            <CODE>0</CODE>
            <SEVERITY>INFO</SEVERITY>
          </STATUS>
          <PMTRS>
            <SRVRTID>1068405</SRVRTID>
            <PAYEELSTID>123214</PAYEELSTID>
            <CURDEF>USD</CURDEF>
            <PMTINFO>
               <BANKACCTFROM>
                  <BANKID>123432123</BANKID>
                  <ACCTID>516273</ACCTID>
                  <ACCTTYPE>CHECKING</ACCTTYPE>
               </BANKACCTFROM>
               <TRNAMT>123.45</TRNAMT>
               <PAYEEID>9076</PAYEEID>
               <PAYEELSTID>123214</PAYEELSTID>
               <PAYACCT>10101</PAYACCT>
               <DTDUE>20051001</DTDUE>
               <MEMO>payment #4</MEMO>
            </PMTINFO>
            <EXTDPAYEE>
               <PAYEEID>9076</PAYEEID>
               <IDSCOPE>USER</IDSCOPE>
               <NAME>J. C. Counts</NAME>
               <DAYSTOPAY>3</DAYSTOPAY>
            </EXTDPAYEE>
               <PMTPRCSTS>
                 <PMTPRCCODE>WILLPROCESSON</PMTPRCCODE>
                 <DTPMTPRC>20051001</DTPMTPRC>
               </PMTPRCSTS>
          </PMTRS>
        </PMTTRNRS>
      </PMTSYNCRS>
    </BILLPAYMSGSRSV1>
</OFX>
```

CHAPTER 13 INVESTMENTS

OFX supports download of security information and detailed investment account statements including transactions, open orders, balances, and positions.

	Server Responds
Account identifier	
Whether to download open orders	
Whether to download transactions	
Date range if transactions should be downloaded	
Whether to download positions	
Whether to download balances	
Additional securities to send information about	
	Date and time for statement
	Default currency for statement
	Account identifier
	Investment transactions
	Banking transactions
	Open orders
	Positions
	Account balances
	Available Cash Balance
	Short Balance
	Margin Balance
	Buying power
	Marketing message
	List of securities

Note: This release of OFX does not support trading or tax lots.

13.1 Types of Response Information

The response consists of five types of information:

- Transactions – a combination of bank transaction detail records and investment transaction detail records. Transactions only within the specified start and stop dates are sent.
- Positions – positions a user has at a brokerage. Each statement response must contain a complete set of position records, even if no transactions occurred in the requested statement period for a particular holding.
- Balances – current balances typically reported on an FI statement, such as cash balance or buying power. They can also convey other numbers of interest, such as current interest rates.
- Open Orders – current open trading orders that a user has at a brokerage.
- Securities – any security referenced in either transactions, positions, open orders or explicitly requested.
- Images - images for investment banking transactions
- Closing statements

13.2 Sub-Accounts

Many FIs distinguish between activity and positions in cash, margin, and short accounts, with some FIs having many other types of “sub-accounts.” OFX defines four standard types of sub-accounts: Cash, Margin, Short, Other. Position, Transaction, and Open Order records identify the sub-account.

13.3 Units, Precision, and Signs

This section provides information about numerical values for investment transactions. For more information about common data types used within OFX, refer to Chapter 3, "Common Aggregates, Elements, and Data Types."

13.3.1 Units

The units for security units and unit price are those commonly used on brokerage statements, and differ for each type of security.

- Stocks and Other – use number of shares for units and dollar value for unit price.
- Mutual Funds – in most cases shares are used, but in some cases the dollar value is used. The unit type is specified in cases in which it can be either.
- Bonds – use face value for units and percentage of par for unit price. For example, a $25,000 bond trading at $88 would use 25000 as the units and 88 as the unit price.

- Options – use number of contracts (not shares) for units, and price per share (not contract) for unit price.

13.3.2 Precision

OFX does not specify the precision of fields since the precision is client-dependent. However, it is recommended that clients and servers follow these rules:

- Clients and servers should send as much precision as they have
- Clients and servers should use a precision equal to or better than 1/256 of a share

13.3.3 Signs

Chapter 3, "Common Aggregates, Elements, and Data Types," describes how to use positive and negative numbers. Briefly, quantities and total values should be signed from the perspective of the user. In a stock buy, the total value is negative, the unit price is always positive, and the number of units is positive.

In a transaction, the signage of UNITS and TOTALs is determined by the transaction's effect on the underlying cash balance (an investment sell would contain a positive TOTAL and negative UNITS whereas an investment buy would contain a negative TOTAL and positive UNITS). All other Investment transaction amounts are always positively signed. In other words UNITPRICE, COMMISSION, FEES, TAXES, PENALTY, WITHHOLDING, STATEWITHHOLDING, LOAD, MARKUP and MARKDOWN are always positive numbers.

A positive COMMISSION, TAXES, LOAD, PENALTY, WITHHOLDING, STATEWITHHOLDING or FEES increases the negatively signed TOTAL on a BUYSTOCK and decreases the positively signed TOTAL on a SELLSTOCK.

MARKUP and MARKDOWN increase or decrease, respectively, the UNITPRICE.

13.4 Bank and Investment Transactions

Many FIs provide investment accounts that allow users to write checks and perform other traditional banking transactions, as well as investment transactions. OFX requires FIs to indicate in the download whether check-writing privileges exist for a given account.

FIs need to use the correct transaction record, bank or investment, for each real-world transaction. Use the following guidelines:

- Checks, electronic funds transfers, and ATM transactions associated with CMA or money market sweep accounts are always represented with a bank transaction record.
- Investment actions that involve securities (buy, sell, stock split, reinvest, etc.) are always represented with an investment record. Actions that are cash-only but are directly associated with a security are also investment actions (for example, dividends).
- Other cash-only actions require careful analysis by the FI. Those that affect investment performance analysis should be sent using the appropriate investment action (investment income - miscellaneous, investment expense). Those that are completely unrelated to investment should be sent as a bank record.

13.5 Money Market Funds

Money market funds can be handled in one of three different ways depending on how the fund is modeled at the financial institution

- Separate account at the financial institution
- Sweep account within an investment account
- Position within an investment account

13.5.1 Separate Account at the Financial Institution

In this case, the money market fund is in its own account with its own account number, distinct from the investment account. In OFX, you should model the money market fund as a separate money market bank account; see Chapter 11, "Banking." The banking <STMTRQ> request aggregate and <STMTRS> response aggregate will be used to download transactions.

13.5.2 Sweep Account Within an Investment Account

OFX uses the money market as a “sweep” account, where cash is “swept” as needed when buying and selling securities. The money market fund does not have its own account number. The customer sees the money market fund as an investment-account cash balance. In OFX, checks, ATMs, electronic fund transfer, deposit, and withdrawal transactions should be downloaded using banking transactions within the investment account. However, the sweep transactions in and out of the money market fund should not be downloaded to the client.

13.5.3 Position Within an Investment Account

The customer purchases the money market fund and is held in the account as a position. The money market fund does not have its own account number. In OFX, the money market fund should be returned as a <POSOTHER> position in the <INVPOSLIST>, with a <UNITPRICE> of 1.00 and <UNITS> as the current value of the position. Purchases and redemptions should be modeled as <BUYOTHER> and <SELLOTHER> transactions with a <UNITPRICE> of 1.00 and <UNITS> as the transaction amount.

13.6 Investment Accounts

Investment account information is downloaded using the account information response aggregate <ACCTINFORS>. For more information, refer to Chapter 8, "Activation & Account Information." <INVACCTFROM> specifies the account. The <INVACCTINFO> aggregate specifies the investment-specific information.

13.6.1 Specifying the Investment Account <INVACCTFROM>

	Description
<INVACCTFROM>	Account-from aggregate
<BROKERID>	Unique identifier for the FI, *A-22*
<ACCTID>	Account number at FI, *A-22*
</INVACCTFROM>	

Brokers should use the domain name of their company’s URL as the BROKERID, e.g.,

```
If URL=www.broker.com
then BROKERID=broker.com
```

The <INVACCTTO> aggregate contains the same elements.

13.6.2 Investment Account Information <INVACCTINFO>

The <INVACCTINFO> aggregate should appear in the <ACCTINFO> aggregate for accounts that support investment statement download. For more information about the <ACCTINFO> aggregate, refer to Chapter 8, "Activation & Account Information."

	Description
<INVACCTINFO>	Investment-account-information-record aggregate
<INVACCTFROM>	Account at FI, see 13.6.1
</INVACCTFROM>	
<USPRODUCTTYPE>	Classification of account. See section 13.6.2.1 for values
<CHECKING>	Whether the account has check writing privileges, *Boolean*
<SVCSTATUS>	Activation status for investment statement download for the account. ACTIVE (signed up), PEND (in the process of signing up), AVAIL (have not signed up).
<INVACCTTYPE>	Type of account. INDIVIDUAL, JOINT, TRUST, CORPORATE
<OPTIONLEVEL>	Text description of option trading privileges, *A-40*
</INVACCTINFO>	

If an investment account has payments functionality, the analogous PMTINFO aggregate (see 12.5.2) should also be sent in the ACCTINFO for the account. Payment information will be sent using the message sets described in the 12.5.2 , "Payment Information <PMTINFO>."

13.6.2.1 Values for <USPRODUCTTYPE>

<USPRODUCTTYPE> classifies accounts according to their account type. Valid values are:

	Description
401K	A 401(K) account
403B	A 403(B) account
IRA	An IRA account
KEOGH	Keogh (Money Purchase/Profit Sharing)
OTHER	Other account type
SARSEP	Salary Reduction Simplified Employer Pension plan
SIMPLE	Savings Incentive Match Plan for employees
NORMAL	Regular account
TDA	Tax Deferred Annuity
TRUST	Trust (including UTMA)
UGMA	Custodial account

Note: Server should return 401K as the value for <USPRODUCTTYPE> in the <INVACCTINFO> aggregate for 401(k) accounts.

13.6.2.2 International Note

The <USPRODUCTTYPE> element is intended for use by FIs in the United States. OFX will be expanded to provide equivalent elements to support the needs for other countries.

13.6.3 Brokerage, Mutual Fund, and 401K Accounts

Investment accounts include brokerage accounts, mutual fund accounts, 401(k) accounts, and other retirement accounts. OFX supports transactions, positions, balances, and open orders for all of these account types.

13.6.3.1 401(k) Accounts

401(k) accounts have the following additional characteristics:

- Funds can be provided which are to be considered “before tax” or “after tax”. In addition, funds can be provided which are not immediately available to the user (vesting of employer contributed funds). The separate sources of funds must be tracked separately in order to properly report the user’s account.

13.7.1 Investment Statement Download

13.7.1.1 Investment Message Set Profile <INVSTMTMSGSET>

The investment statement message set profile aggregate <INVSTMTMSGSET> is used in the response to a financial institution profile request (see Chapter 7, "FI Profile") to specify which activities it supports.

	Description
<INVSTMTMSGSET>	Investment-statement-message-set-profile aggregate
<INVSTMTMSGSETV1>	Version 1 message set
<MSGSETCORE>	Common message set information, see Chapter 7, "FI Profile"
</MSGSETCORE>	
<TRANDNLD>	Whether the FI server downloads investment statement transactions, *Boolean*
<OODNLD>	Whether the FI server downloads investment open orders, *Boolean*
<POSDNLD>	Whether the FI server downloads investment statement positions, *Boolean*
<BALDNLD>	Whether the FI server downloads investment balances, *Boolean*
<CANEMAIL>	Whether the FI supports investment e-mail. Use generic e-mail profile to specify whether generic e-mail is supported; see Chapter 9, "Customer to FI Communication." *Boolean*
<INV401KDNLD>	Whether the FI server downloads 401(k) account information, *Boolean*
<CLOSINGAVAIL>	Closing statement information available. If 'N' or not present, do not sent <INVSTMTENDRQ> to server, *Boolean*
<IMAGEPROF>	Image profile (if supported), see section 3.1.6.2
</IMAGEPROF>	
</INVSTMTMSGSETV1>	
</INVSTMTMSGSET>	

13.7.1.2 Investment Statement Message Set and Messages

The following tables show the OFX transactions and their corresponding transaction wrappers for requests and responses.

13.7.1.2.1 Investment Statement Message Set Request Messages

	Description
<INVSTMTMSGSRQV1>	INVSTMTTRNRQ
	INVSTMTTRQ
	INVMAILTRNRQ
	INVMAILRQ
	INVMAILSYNCRQ
	INVSTMTENDTRNRQ
	INVSTMTENDRQ
</INVSTMTMSGSRQV1>	

13.7.1.2.2 Investment Statement Message Set Response Messages

	Description
<INVSTMTMSGSRSV1>	INVSTMTTRNRS
	INVSTMTRS
	INVMAILTRNRS
	INVMAILRS
	INVMAILSYNCRS
	INVSTMTENDTRNRS
	INVSTMTENDRS
</INVSTMTMSGSRSV1>	

13.7.2 Security Information

13.7.2.1 Security List Message Set Profile <SECLISTMSGSET>

The security list message set profile aggregate <SECLISTMSGSET> is used in the response to an FI profile request (see Chapter 7, “FI Profile”) to specify which activities it supports.

	Description
<SECLISTMSGSET>	Security-information-message-set-profile aggregate
<SECLISTMSGSETV1>	Version 1 message set
<MSGSETCORE>	Common message set information, see Chapter 7, "FI Profile"
</MSGSETCORE>	
<SECLISTRQDNLD>	Whether the FI server responds to security list requests, *Boolean*
</SECLISTMSGSETV1>	
</SECLISTMSGSET>	

13.7.2.2 Security List Message Set and Messages

13.7.2.2.1 Security List Message Set Request Messages

	Description
<SECLISTMSGSRQV1>	SECLISTTRNRQ SECLISTRQ
</SECLISTMSGSRQV1>	

13.7.2.2.2 Security List Message Set Response Messages

	Description
<SECLISTMSGSRSV1>	SECLISTTRNRS
	SECLISTRS
	SECLIST
</SECLISTMSGSRSV1>	

Note: The <SECLISTMSGSRS> aggregate may appear in a response file when no corresponding <SECLISTMSGSRQ> aggregate appears in the request file. Servers should add the message set response wrapper only when downloading a statement and providing the <SECLIST>.

13.8 Investment Securities

13.8.1 Security Identification <SECID>

Securities must be consistently identified to allow client applications to prepare accurate investment reports across all user investment accounts, even at multiple FIs. At this time, neither a security name nor its symbol is standardized. Therefore, OFX uses CUSIP numbers (a unique 9-digit alphanumeric identifier) to identify securities. CUSIP numbers are available for the vast majority of securities traded today, including those without symbols such as bonds. For a security that does not have a CUSIP, a financial institution must follow the standard procedure of assigning a CUSIP by using itself as the issuer to avoid conflict with any other CUSIP.

	Description
<SECID>	Security-identifier aggregate
<UNIQUEID>	Unique identifier for the security. CUSIP for US FIs. *A-32*
<UNIQUEIDTYPE>	Name of standard used to identify the security i.e., “CUSIP” for FIs in the United States, *A-10*
</SECID>	

13.8.1.1 International Note

Non-US financial institutions that do not have access to CUSIP numbers must supply a unique identifier for each security in the UNIQUEID field of this aggregate. OFX will be expanded to include other security identifying standards.

13.8.2 Security List Request

The user can use the SECLISTTRNRQ and SECLISTRQ aggregates to request information about specific securities. The SECLISTTRNRQ is the transaction-level aggregate that contains the SECLISTRQ. The SECLISTRQ aggregate specifies for which securities information is being requested.

13.8.2.1 Security List Transaction Request <SECLISTTRNRQ>

	Description
<SECLISTTRNRQ>	Transaction-request aggregate
<TRNUID>	Client-assigned globally unique ID for this transaction *trnuid*
<CLTCOOKIE>	Data to be echoed in the transaction response, *A-32*
<TAN>	Transaction authorization number; used in some countries with some types of transactions. Country-specific documentation will define messages that require a <TAN>, *A-80*
<SECLISTRQ>	Aggregate for the security list request (see section 13.8.2.2)
</SECLISTRQ>	
</SECLISTTRNRQ>	

13.8.2.2 Security List Request <SECLISTRQ>

For the security list request, securities must be specified with either a SECID aggregate, a ticker symbol, or an FI assigned identifier.

	Description
<SECLISTRQ>	Security-list-request aggregate
<SECRQ>	Security request (one or more)
Security identification. Specify either <SECID>, <TICKER>, or <FIID>.	
<SECID>	Security identifier aggregate
</SECID>	
-or-	
<TICKER>	Ticker symbol, *A-32*
-or-	
<FIID>	FI specific ID for the security, *A-32*
</SECRQ>	
</SECLISTRQ>	

13.8.3 Security List Response

If the client sends a security list request to an FI, then the server must send back a security list response to the client application. The security list response is used primarily to report the status of the security list request. The actual security information should be sent in the security list SECLIST aggregate described in section 13.8.4.

13.8.3.1 Security List Transaction Response <SECLISTTRNRS>

	Description
<SECLISTTRNRS>	Transaction-response aggregate
<TRNUID>	Client-assigned globally unique ID for this transaction *trnuid*
<STATUS>	Status aggregate
</STATUS>	
<CLTCOOKIE>	Client-provided data, **REQUIRED** if provided in request, *A-32*
<SECLISTRS>	Aggregate for the security list response, see 13.8.3.3
</SECLISTRS>	
</SECLISTTRNRS>	

13.8.3.2 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2019	Duplicate request (ERROR)
12500	One or more securities not found (ERROR)

13.8.3.3 Security List Response <SECLISTRS>

The security list response aggregate, the only empty aggregate in OFX, is used to respond to the <SECLISTRQ>. It is used to signify that the security list is generated as a result of a security list request. The actual security information should be included in the <SECLIST> aggregate.

Tag	*Description*
<SECLISTRS>	Security-list-response (the only empty aggregate in OFX).
</SECLISTRS>	

13.8.4 Security List <SECLIST>

The SECLIST should be sent in the following two cases.

- In response to a <SECLISTRQ>.

  Note: An empty <SECLISTRS> is sent in response to the <SECLISTRQ>. The <SECLIST> aggregate is sent after the <SECLISTTRNRS> aggregate that wraps the <SECLISTRS>.

- When the response file contains an investment statement download that has positions, transactions, or open orders. The <SECLIST> should contain information about each security referenced in the investment statement download. Clients are completely dependent on the security list to provide descriptive information for the securities referenced in positions, transactions, and open orders.

  Note: The <SECLISTMSGSRSV1> aggregate may appear in a response file when no corresponding <SECLISTMSGSRQV1> aggregate appears in the request file. Servers should add the message set response wrapper only when

13.8.5.2 Debt Information <DEBTINFO>

	Description
<DEBTINFO>	Opening tag for debt information aggregate
<SECINFO>	Security information aggregate
</SECINFO>	
<PARVALUE>	Par value, *amount*
<DEBTTYPE>	Debt type (at most one) COUPON = coupon ZERO = zero coupon
<DEBTCLASS>	Classification of debt. TREASURY, MUNICIPAL, CORPORATE, OTHER.
<COUPONRT>	Bond coupon rate for next closest call date (at most one), *rate*
<DTCOUPON>	Maturity date for next coupon, *date*
<COUPONFREQ>	When coupons mature. One of the following values: MONTHLY, QUARTERLY, SEMIANNUAL, ANNUAL, or OTHER.
<CALLPRICE>	Bond call price (at most one), *unitprice*
<YIELDTOCALL>	Yield to next call, *rate*
<DTCALL>	Next call date (at most one), *date*
<CALLTYPE>	Type of next call. CALL, PUT, PREFUND, MATURITY
<YIELDTOMAT>	Yield to maturity, *rate*
<DTMAT>	Debt maturity date (at most one), *date*
<ASSETCLASS>	Asset Class (at most one), DOMESTICBOND, INTLBOND, LARGESTOCK SMALLSTOCK, INTLSTOCK, MONEYMRKT, OTHER
<FIASSETCLASS>	Text string containing an FI defined asset class, *A-32*
</DEBTINFO>	

13.8.5.3 Mutual Fund Information <MFINFO>

	Description
<MFINFO>	Mutual-fund-information aggregate
<SECINFO>	Security-information aggregate
</SECINFO>	
<MFTYPE>	Mutual fund type. OPENEND, CLOSEEND, OTHER
<YIELD>	Current yield reported as portion of the fund's assets (at most one), *rate*
<DTYIELDASOF>	As-of date for yield value, *datetime*
<MFASSETCLASS>	Asset class breakdown for the mutual fund
<PORTION>	Portion of the mutual fund with a specific asset classification (one or more)
<ASSETCLASS>	Asset Class, DOMESTICBOND, INTLBOND, LARGESTOCK SMALLSTOCK, INTLSTOCK, MONEYMRKT, OTHER
<PERCENT>	Percentage of the fund that falls under this asset class, *rate*
</PORTION>	
</MFASSETCLASS>	
<FIMFASSETCLASS>	FI defined asset class breakdown for the mutual fund
<FIPORTION>	Portion of the mutual fund with a specific asset classification (one or more)
<FIASSETCLASS>	Text string containing an FI defined asset class, *A-32*
<PERCENT>	Percentage of the fund that falls under this asset class, *rate*
</FIPORTION>	
</FIMFASSETCLASS>	
</MFINFO>	

13.8.5.6 Stock Information <STOCKINFO>

	Description
<STOCKINFO>	Stock-information aggregate
<SECINFO>	Security-information aggregate
</SECINFO>	
<STOCKTYPE>	Stock type: COMMON, PREFERRED, CONVERTIBLE, OTHER
<YIELD>	Current yield reported as the dividend expressed as a portion of the current stock price (at most one), *rate*
<DTYIELDASOF>	As-of date for yield value, *datetime*
<ASSETCLASS>	Asset Class (at most one): DOMESTICBOND, INTLBOND, LARGESTOCK SMALLSTOCK, INTLSTOCK, MONEYMRKT, OTHER
<FIASSETCLASS>	Text string containing an FI defined asset class, *A-32*
</STOCKINFO>	

13.8.5.7 Asset Class Descriptions

	Description
DOMESTICBOND	The Domestic Bonds asset class consists of government or corporate bonds issued in the United States.
INTLBOND	The International Bonds asset class consists of government or corporate bonds issued in foreign countries or the United States.
LARGESTOCK	The Large Cap Stocks asset class consists of stocks for U.S. companies with market capitalizations of $2 billion or more.
SMALLSTOCK	The Small Cap Stocks asset class consists of stocks for U.S. companies with market capitalizations of approximately $100 million to $2 billion.
INTLSTOCK	The International Stocks asset class consists of publicly-traded stocks for companies based in foreign countries.
MONEYMRKT	The Money Market asset class consists of stable, short-term investments which provide income that rises and falls with short-term interest rates.
OTHER	The Other asset class consists of investments which do not fit in any of the other asset classes.

13.9 Investment Statement Download

Investment statement download allows a customer to receive transactions, positions, open orders, and balances that are typically part of a regular paper statement.

Clients usually allow customers to view investment transactions and guide customers through a process of updating their account registers based on the downloaded transactions. By using <FITID> values supplied by FIs, OFX makes it possible for clients to insure that each transaction is downloaded only once. The request also contains starting and ending dates to limit the amount of downloaded data. Clients can remember the last date they receive a download, and use that date as the starting date in the next request.

Investment statement download requires the client to designate an account for the download, and to indicate what type of data should be downloaded. If the client wishes to download transactions, it can specify a date range that the transactions fall within. The server returns transactions that match the date range, if one is specified. If a date range is not specified, the server returns all available transactions for the account.

13.9.1 Investment Statement Request

Investment statement download can be requested using the INVSTMTTRNRQ and INVSTMTRQ aggregates. The INVSTMTTRNRQ is the transaction level aggregate that contains the INVSTMTRQ. The INVSTMTRQ aggregate specifies what types of information to include in the statement download and from which account to download the information.

Image transaction information can be requested by including <INCTRANIMG>Y in a <INVSTMTRQ>, requesting the server to return image data corresponding to transaction images, if available. The image identification data returned in the <STMTTRN> aggregate (see section 11.4.3) is not the actual image, but will be used to access the image through subsequent client sessions. See Chapter 15, “Image Download”, for more information about retrieving images from the server.

13.9.1.1 Investment Statement Transaction Request <INVSTMTTRNRQ>

	Description
<INVSTMTTRNRQ>	Transaction-request aggregate
<TRNUID>	Client-assigned globally unique ID for this transaction, *trnuid*
<CLTCOOKIE>	Data to be echoed in the transaction response, *A-32*
<TAN>	Transaction authorization number; used in some countries with some types of transactions. Country-specific documentation will define messages that require a <TAN>, *A-80*
<INVSTMTRQ>	Aggregate for the investment statement download request (see section 13.9.1.2)
</INVSTMTRQ>	
</INVSTMTTRNRQ>	

13.9.1.2 Investment Statement Request <INVSTMTRQ>

The following table shows the Investment Statement Request record. It is similar to a bank statement request, except that there are extra elements to indicate which pieces the user desires. Note that because transaction and position requests require date information, they use aggregates, whereas the other requests are elemental of type Boolean.

Clients and servers should interpret <DTSTART> and <DTEND> as described in Chapter

Tag	Description and Type
<INCTRAN>	Include-transactions aggregate (at most one)
<DTSTART>	Start date of request, *datetime*
<DTEND>	Ending date of request (at most one), *datetime*
<INCLUDE>	Whether to include transactions in the statement download, *Boolean*
</INCTRAN>	
<INCOO>	Include investment open orders in response, *Boolean*
<INCPOS>	Include investment positions in response
<DTASOF>	Date that positions should be sent down for, *datetime*
<INCLUDE>	Whether to include positions in the statement download, *Boolean*
</INCPOS>	
<INCBAL>	Include investment balance in response, *Boolean*
<INC401K>	Include 401(k) information in response, *Boolean*
<INC401KBAL>	Include 401(k) balance information in response, *Boolean*
<INCTRANIMG>	Include image information for banking transactions, *Boolean*
</INVSTMTRQ>	

13.9.2 Investment Statement Response

13.9.2.1 Investment Statement Transaction Response <INVSTMTTRNRS>

	Description
<INVSTMTTRNRS>	Transaction-response aggregate
<TRNUID>	Client-assigned globally unique ID for this transaction, *trnuid*
<STATUS>	Status aggregate
</STATUS>	
<CLTCOOKIE>	Client-provided data, **REQUIRED** if provided in request, *A-32*
<INVSTMTRS>	Aggregate for the investment statement download response (see section 13.9.2.2)
</INVSTMTRS>	
</INVSTMTTRNRS>	

13.9.2.2 Investment Statement Response <INVSTMTRS>

The response can contain transaction, position, open order, and/or balance detail records; each in its own aggregate. The transaction list aggregate can contain a mixture of bank statement records and investment transactions, as specified below.

For 401(k) accounts, both the <INV401KBAL> and the <INVBAL> aggregates can be returned if asked for specifically. In other words, for 401(k) accounts the <INV401KBAL> aggregate is returned if asked for by the client with the <INC401KBAL> flag, and the <INVBAL> aggregate is returned if <INCBAL> is Y.

Tag	*Description*
<INVSTMTRS>	Investment-response aggregate
<DTASOF>	As of date & time for the statement download, *datetime*
<CURDEF>	Default currency for the statement, *currsymbol*
<INVACCTFROM>	Which account at FI, see 13.6.1
</INVACCTFROM>	
<INVTRANLIST>	Begin transaction list (at most one)
<DTSTART>	Start date for transaction data, *datetime*
<DTEND>	This is the value that should be sent in the next <DTSTART> request to insure that no transactions are missed, *datetime*
(*investment transaction aggregates*)	Investment statement transaction aggregates (zero or more); see section 13.9.2.4.4.

Tag	Description
<INVBANKTRAN>	Banking-related transactions for the investment account (zero or more)
</INVBANKTRAN>	(See section 13.9.2.3)
</INVTRANLIST>	End of investment transaction list
<INVPOSLIST>	Beginning of investment position list Though the DTD allows an empty <INVPOSLIST> in the response, servers should instead leave out the optional list aggregate.
<POS*xxxxx*>	Security type specific position aggregates (zero or more): POSMF, POSSTOCK, POSDEBT, POSOPT, POSOTHER
</POS*xxxxx*>	
</INVPOSLIST>	End of investment position list
<INVBAL>	Balances aggregate, see section 13.9.2.7
</INVBAL>	
<INVOOLIST>	Beginning of investment open order list Though the DTD allows an empty <INVOOLIST> in the response, servers should instead leave out the optional list aggregate.
<OO*xxxxx*>	Action and security type specific open order aggregates (zero or more): see section 13.9.2.5.2
</OO*xxxxx*>	
</INVOOLIST>	End of investment open order list
<MKTGINFO>	Marketing information (at most one), *A-360*.
<INV401K>	401(k) information aggregate (at most one) (See section 13.9.3).
</INV401K>	End of 401(k) information.
<INV401KBAL>	401(k) balance information aggregate (see section 13.9.2.9).
</INV401KBAL>	End of 401(k) balance information.
</INVSTMTRS>	

The various sections of the investment statement download are returned only if requested.

13.9.2.2.1 Note on Margin Calls

For investment statement download, margin call information should be included in the balances section. Margin call information should be contained in a <BAL> aggregate and included in the balance list <BALLIST>.

13.9.2.3 Bank Transactions <INVBANKTRAN>

Use the INVBANKTRAN aggregate to download bank transactions in an investment statement download.

	Description
<INVBANKTRAN>	Banking related transactions for the investment account
<STMTTRN>	Bank (cash) transaction aggregates
</STMTTRN>	(See Chapter 11, "Banking")
<SUBACCTFUND>	The sub-account associated with the funds for the transaction; see section 13.9.2.4.2
</INVBANKTRAN>	

13.9.2.4 Investment Transactions

Note that the following types of investment actions found on statements should ***not*** be sent in OFX:

- Transaction-specific miscellaneous/fees—fees or other amounts that affect the basis of the transaction should be incorporated into the <COMMISSION>, <FEES>, <LOAD>, <PENALTY>, <WITHHOLDING>, <STATEWITHHOLDING> or <TAXES> amounts.
- Settlement actions.
- Sweeps, unless handled as any other investment position.

For transactions that involve securities, the client can create transactions based on the formula

total = (units * (unitprice +/- markup/markdown)) +/- (commission + fees + load + taxes + penalty + withholding + statewithholding)
(after adjusting quantity and unitprice to standard units based on the type of security.)

Thus, it is important the FIs incorporate all other transactional fees into the commission field. Clients can account for bond accrued interest and withholding using separate client transactions.

13.9.2.4.1 General Transaction Aggregate <INVTRAN>

The <INVTRAN> aggregate contains fields common to many of the investment transactions. It is referenced within the transaction aggregates in the following sections.

Each <INVTRAN> contains an <FITID> that the client uses to detect whether the server previously downloaded the transaction.

	Description
<INVTRAN>	Investment-transaction-response aggregate
<FITID>	Unique FI-assigned transaction ID. This ID is used to detect duplicate downloads. *FITID*
<SRVRTID>	Server assigned transaction ID, *SRVRTID*
<DTTRADE>	Trade date; for stock splits, day of record, *datetime*
<DTSETTLE>	Settlement date; for stock splits, execution date, *datetime*
<REVERSALFITID>	For a reversal transaction, the FITID of the transaction that is being reversed. See section 13.9.2.4.1.1 for details, *FITID*
<MEMO>	Other information about transaction (at most one), *memo*
</INVTRAN>	

13.9.2.4.1.1 Transaction Reversals

Reversals may be accomplished using two different mechanisms, depending on the support for <REVERSALFITID> at the client.

Older clients that do not support <REVERSALFITID> will have transactions reversed using the opposite transaction type when appropriate, e.g., a correction to a buy is returned as a sell. Because investment transactions cannot include negative amounts, certain transaction components such as commissions and fees can be reversed by sending an <INVBANKTRAN>.

Newer clients that do support <REVERSALFITID> will have transactions reversed using the same transaction type as was originally sent. The <REVERSALFITID> must contain the <FITID> of the transaction that is being reversed. The client will reverse all components of the original transaction including commissions and fees. Partial corrections are not allowed when transactions are reversed by use of the <REVERSALFITID> tag. The <FITID> of the new reversal transaction shall be new and unique (not the same as the <REVERSALFITID>).

13.9.2.4.2 Transaction Aggregate Elements

The following elements are referenced within of the following investment transaction aggregates.

Tag	Description
<ACCRDINT>	For debt purchases, accrued interest, *amount*
<AVGCOSTBASIS>	Average cost basis, *amount*
<BUYTYPE>	Type of purchase: BUY, BUYTOCOVER
<COMMISSION>	Transaction commission. *amount*
<DENOMINATOR>	For stock splits, split ratio denominator, *quantity*
<DTPAYROLL>	For 401(k)accounts, date the funds for this transaction was obtained via payroll deduction, *datetime*
<DTPURCHASE>	The security's original purchase date, *date*
<GAIN>	For sales, total gain, *amount*
<FEES>	Fees applied to trade, *amount*
<FRACCASH>	Cash for fractional units., (used for stock splits), *amount*
<INCOMETYPE>	Type of investment income: CGLONG (capital gains-long term), CGSHORT (capital gains-short term), DIV (dividend), INTEREST, MISC
<INV401KSOURCE>	For 401(k) accounts, source of money used for this security. Must be one of the following: PRETAX AFTERTAX MATCH PROFITSHARING ROLLOVER OTHERVEST OTHERNONVEST Default if not present is OTHERNONVEST. The following cash source types are subject to vesting: MATCH, PROFITSHARING, and OTHERVEST.
<LOAD>	Load on the transaction, *amount*
<LOANID>	For 401(k) accounts only. Indicates that the transaction was due to a loan or a loan repayment, and which loan it was. *A-32*
<LOANINTEREST>	For 401(k) accounts only. Indicates how much of the loan repayment was interest. *Amount*
<LOANPRINCIPAL>	For 401(k) accounts only. Indicates how much of the loan repayment was principal. *Amount*
<MARKDOWN>	Portion of the unit price that is attributed to the dealer markdown, *unitprice*

<MARKUP>	Portion of the unit price that is attributed to the dealer markup, *unitprice*
<NEWUNITS>	

Tag	Description
<TFERACTION>	Action for transfers: IN, OUT
<UNITPRICE>	Price per commonly-quoted unit. Does not include markup/markdown, *unitprice*. Share price for stocks, mutual funds, and others Percentage of par for bonds Per share (not contract) for options
<UNITS>	For security-based actions other than stock splits, *quantity* Shares for stocks, mutual funds, and others. Face value for bonds. Contracts for options.
<UNITTYPE>	Type of the units value: SHARES, CURRENCY
<WITHHOLDING>	Federal Tax withholdings, *amount*

13.9.2.4.3 Investment Buy/Sell Aggregates <INVBUY>/<INVSELL>

These aggregates are referenced within investment transaction aggregates

Note: For 401(k) accounts, securities can be sold to fund loans or other withdrawals. Loans and loan repayments, and penalties are shown as part of the Buy and Sell transactions, rather than directly on any associated Deposit or Withdrawal bank transactions (see section 11.4.3.1) because some 401(k) providers might not report these bank transactions.

Also, for 401(k) accounts, as loans are repaid the funds are disbursed into the 401(k) sources of money from where they were originally withdrawn. To accurately reflect the disbursement of repaid funds into each source, a separate Buy transaction will be issued with the <INV401KSOURCE> tag set to indicate the source of money. Each of these transactions will include the amount of principal and/or interest paid to the source.

Aggregate Name	Elements	Description
INVBUY	**<INVTRAN>** aggregate	
	<SECID> aggregate	
	<UNITS>	
	<UNITPRICE>	
	<MARKUP>	
	<COMMISSION>	
	<TAXES>	
	<FEES>	
	<LOAD>	
	<TOTAL>	
	<CURRENCY> aggregate <ORIGCURRENCY> aggregate	Though the DTD allows <CURRENCY> and <ORIGCURRENCY> together in this aggregate, servers should return neither or one of the two, but not both.
	<SUBACCTSEC>	
	<SUBACCTFUND>	
	<LOANID>	Required if <LOANPRINCIPAL> and <LOANINTEREST> are provided. See section 13.9.2.4.2
	<LOANPRINCIPAL>	<LOANID> and <LOANINTEREST> must be provided if <LOANPRINCIPAL> is provided. See section 13.9.2.4.2
	<LOANINTEREST>	<LOANID> and <LOANPRINCIPAL> must be provided if <LOANINTEREST> is provided. See section 13.9.2.4.2
	<INV401KSOURCE>	Source of money for this transaction. See section 13.9.2.4.2
	<DTPAYROLL>	See section 13.9.2.4.2
	<PRIORYEARCONTRIB>	See section 13.9.2.4.2

Aggregate Name	Elements	Description
INVSELL	**<INVTRAN>** aggregate	
	<SECID> aggregate	
	<UNITS>	
	<UNITPRICE>	
	<MARKDOWN>	
	<COMMISSION>	
	<TAXES>	
	<FEES>	
	<LOAD>	
	<WITHHOLDING>	
	<TAXEXEMPT>	
	<TOTAL>	
	<GAIN>	
	<CURRENCY> aggregate	Though the DTD allows <CURRENCY> and <ORIGCURRENCY> together in this aggregate, servers should return neither or one of the two, but not both.
	<ORIGCURRENCY> aggregate	
	<SUBACCTSEC>	
	<SUBACCTFUND>	
	<LOANID>	See section 13.9.2.4.2
	<STATEWITHHOLDING>	See section 13.9.2.4.2
	<PENALTY>	See section 13.9.2.4.2
	<INV401KSOURCE>	Source of money for this transaction. See section 13.9.2.4.2

13.9.2.4.4 Investment Transaction Aggregates

Aggregate Name	*Elements*	*Description*
<BUYDEBT>	<**INVBUY**>	

Aggregate Name	Elements	Description
<INCOME>	**<INVTRAN>** aggregate	Investment income is realized as cash into the investment account.
	<SECID> aggregate	
	<INCOMETYPE>	
	<TOTAL>	A negative TOTAL is used to denote adjustments to income.
	<SUBACCTSEC>	
	<SUBACCTFUND>	
	<TAXEXEMPT>	
	<WITHHOLDING>	
	<CURRENCY> aggregate	
	<ORIGCURRENCY> aggregate	Though the DTD allows <CURRENCY> and <ORIGCURRENCY> together in this aggregate, servers should return neither or one of the two, but not both.
	<INV401KSOURCE>	Source of money for this transaction. See section 13.9.2.4.2.
<INVEXPENSE>	**<INVTRAN>** aggregate	Misc. investment expense that is associated with a specific security.
	<SECID> aggregate	If the expense is associated with the account then an INVBANKTRAN - DEBIT should be used.
	<TOTAL>	
	<SUBACCTSEC>	
	<SUBACCTFUND>	
	<CURRENCY> aggregate	
	<ORIGCURRENCY> aggregate	Though the DTD allows <CURRENCY> and <ORIGCURRENCY> together in this aggregate, servers should return neither or one of the two, but not both.
	<INV401KSOURCE>	Source of money for this transaction. See section 13.9.2.4.2.
<JRNLFUND>	**<INVTRAN>** aggregate	Journaling cash holdings between sub-accounts within the same investment account.
	<SUBACCTTO>	
	<SUBACCTFROM>	
	<TOTAL>	

Aggregate Name	*Elements*	*Description*
<JRNLSEC>	**<INVTRAN>** aggregate **<SECID>** aggregate **<SUBACCTTO>** **<SUBACCTFROM>** **<UNITS>**	Journaling security holdings between sub-accounts within the same investment account.
<MARGININTEREST>	**<INVTRAN>** aggregate **<TOTAL>** **<SUBACCTFUND>** <CURRENCY> aggregate <ORIGCURRENCY> aggregate	Margin interest expense Though the DTD allows <CURRENCY> and <ORIGCURRENCY> together in this aggregate, servers should return neither or one of the two, but not both.
<REINVEST>	**<INVTRAN>** aggregate **<SECID>** aggregate **<INCOMETYPE>** **<TOTAL>** **<SUBACCTSEC>** **<UNITS>** **<UNITPRICE>** <COMMISSION> <TAXES> <FEES> <LOAD> <TAXEXEMPT> <CURRENCY> aggregate <ORIGCURRENCY> aggregate <INV401KSOURCE>	Reinvestment of income REINVEST is a single transaction that contains both income and an investment transaction. If servers can't track this as a single transaction they should return an INCOME transaction and an INVTRANLIST transaction. TOTAL and UNITS are signed as for an investment buy. Corrections to a REINVEST are signed as for an investment sell. Though the DTD allows <CURRENCY> and <ORIGCURRENCY> together in this aggregate, servers should return neither or one of the two, but not both. Source of money for this transaction. See section 13.9.2.4.2.

Aggregate Name	Elements	Description
<RETOFCAP>	**<INVTRAN>** **<SECID>** **<TOTAL>** **<SUBACCTSEC>** **<SUBACCTFUND>** <CURRENCY> aggregate <ORIGCURRENCY> aggregate <INV401KSOURCE>	Return of capital Though the DTD allows <CURRENCY> and <ORIGCURRENCY> together in this aggregate, servers should return neither or one of the two, but not both. Source of money for this transaction. See section 13.9.2.4.2.
<SELLDEBT>	**<INVSELL>** aggregate **<SELLREASON>** <ACCRDINT>	Sell debt security. Used when debt is sold, called, or reached maturity.
<SELLMF>	**<INVSELL>** aggregate **<SELLTYPE>** <AVGCOSTBASIS> <RELFITID>	Sell mutual fund RELFITID used to relate transactions associated with mutual fund exchanges.
<SELLOPT>	**<INVSELL>** aggregate **<OPTSELLTYPE>** **<SHPERCTRCT>** <RELFITID> <RELTYPE> <SECURED>	Sell option The SELLTOCLOSE action is selling a previously bought option. The SELLTOOPEN action is writing an option
<SELLOTHER>	**<INVSELL>** aggregate	Sell other type of security
<SELLSTOCK>	**<INVSELL>** aggregate **<SELLTYPE>**	Sell stock

13.9.2.4.5 Valid Transactions by Security Type

		Mutual Fund	*Option*	*Other*	*Stock*
BUYDEBT	×				
BUYMF		×			
BUYOPT			×		
BUYOTHER				×	
BUYSTOCK					×
CLOSUREOPT			×		
INCOME	×	×		×	×
INVEXPENSE	×	×	×	×	×
JRNLFUND					
JRNLSEC	×	×	×	×	×
MARGININTEREST					
REINVEST	×	×		×	×
RETOFCAP	×	×	×	×	×
SELLDEBT	×				
SELLMF		×			
SELLOPT			×		
SELLOTHER				×	
SELLSTOCK					×
SPLIT		×			×
TRANSFER	×	×	×	×	×

Since JRNLFUND and MARGININTEREST do not refer to securities, there are no checks in any of the security columns for these transactions.

13.9.2.4.6 Notes on Mutual Fund Exchanges

In investment statement download, two transactions are needed to reflect mutual fund exchanges. A SELLMF should be generated for the mutual fund being switched from and a BUYMF should be generated for the mutual fund being switched to. You can use the RELFITID element to link these two transactions to each other. You should use the MEMO element of the individual transactions to explain that a mutual fund exchange occurred.

13.9.2.4.7 Notes on Corporate Actions

Since corporate actions can often be very complicated, it is difficult to define a single action aggregate that encompasses all possible scenarios. Instead, you should describe corporate actions using one or more of the provided basic action types. You should use the memo field of the individual transactions to link transactions to an encompassing corporate action.

13.9.2.4.8 Notes on Option Splits

When the underlying security for an option splits, a new security is generated for the option since the strike price changes. In investment statement download, you need two transactions to reflect this activity. There should be a TRANSFER transaction to show that the old option security is removed from the account and another TRANSFER transaction to show that the new option security is moved into the account.

13.9.2.4.9 Notes on Optiions

d ofsactiionsties sd fllows:s

13.9.2.5 Open Orders

13.9.2.5.1 General Open Order Aggregate <OO>

The <OO> aggregate contains fields common to all open orders. Use this aggregate to define the open order aggregates as show in the following section.

	Description
<OO>	General-open-order aggregate
<FITID>	Unique FI-assigned transaction ID, *FITID*
<SRVRTID>	Unique server-assigned transaction ID, *SRVRTID*
<SECID>	Security identified aggregate
</SECID>	
<DTPLACED>	Date-time the order was placed, *datetime*
<UNITS>	Quantity of the security the open order is for, *quantity*
<SUBACCT>	Sub-account type. CASH, MARGIN, SHORT, OTHER
<DURATION>	How long the order is good for: DAY, GOODTILCANCEL, IMMEDIATE
<RESTRICTION>	Special restriction on the order: ALLORNONE, MINUNITS, NONE
<MINUNITS>	Minimum number of units that must be filled for the order, *quantity*
<LIMITPRICE>	Limit price, *unitprice*
<STOPPRICE>	Stop price, *unitprice*
<MEMO>	Other information about order (at most one), *memo*
<CURRENCY>	Overriding currency aggregate
</CURRENCY>	
<INV401KSOURCE>	For 401(k) accounts, source of money for this order. See section 13.9.2.4.2.
</OO>	

Note: An open order is assumed to be a market order if no limit price or stop price is specified.

13.9.2.5.2 Investment Open Order Aggregates

	Elements	*Description of Elements*
<OOBUYDEBT>	**<OO>** aggregate	
	<AUCTION>	Whether the debt should be purchased at the auction, *Boolean*
	<DTAUCTION>	Date of the auction, *date*
<OOBUYMF>	**<OO>** aggregate	
	<BUYTYPE>	Type of purchase: BUY, BUYTOCOVER.
	<UNITTYPE>	What the units represent: SHARES, CURRENCY
<OOBUYOPT>	**<OO>** aggregate	
	<OPTBUYTYPE>	Type of purchase: BUYTOOPEN, BUYTOCLOSE
<OOBUYOTHER>	**<OO>** aggregate	
	<UNITTYPE>	What the units represent: SHARES, CURRENCY
<OOBUYSTOCK>	**<OO>** aggregate	
	<BUYTYPE>	Type of purchase: BUY, BUYTOCOVER
<OOSELLDEBT>	**<OO>** aggregate	
<OOSELLMF>	**<OO>** aggregate	
	<SELLTYPE>	Type of sale: SELL, SELLSHORT
	<UNITTYPE>	What the units represent: SHARES, CURRENCY.
	<SELLALL>	Sell entire holding, *Boolean*
<OOSELLOPT>	**<OO>** aggregate	
	<OPTSELLTYPE>	Type of sale: SELLTOOPEN, SELLTOCLOSE
<OOSELLOTHER>	**<OO>** aggregate	
	<UNITTYPE>	What the units represent: SHARES, CURRENCY
<OOSELLSTOCK>	**<OO>** aggregate	
	<SELLTYPE>	Type of sale: SELL, SELLSHORT
<SWITCHMF>	**<OO>** aggregate	
	<SECID> aggregate	Security ID of the mutual fund to switch to or purchase
	<UNITTYPE>	What the units represent: SHARES, CURRENCY
	<SWITCHALL>	Switch entire holding, *Boolean*

13.9.2.6 Investment Positions

Position records represent a user's current positions, regardless of the transactional history. Prices and values should be the most recent available, even if different from a transaction price on the same day.

In position records, securities are identified as being either short or long. Because each FI has different rules regarding which sub-accounts can be used for short compared to long activity, FIs must explicitly indicate the type of position in addition to specifying the sub-account where the position takes place.

For options, position type SHORT is equivalent to WRITING an option, and position type LONG is equivalent to HOLDING an option. For security types where there is only one type (for example, bonds), use LONG.

For 401(k) accounts, securities can be purchased with any of the 401(k) cash sources. Any securities purchased from more than one cash source will appear as a separate position for each.

13.9.2.6.2 Investment Positions

	Elements	*Description of Elements*
<POSDEBT>	**<INVPOS>** aggregate	
<POSMF>	**<INVPOS>** aggregate	
	<UNITSSTREET>	Units in the FI's street name, *positive quantity*
	<UNITSUSER>	Units in the user's name directly, *positive quantity*
	<REINVDIV>	Reinvest dividends, *Boolean*
	<REINVCG>	Reinvest capital gains, *Boolean*
<POSOPT>	**<INVPOS>** aggregate	
	<SECURED>	How the option is secured. NAKED, COVERED.
<POSOTHER>	**<INVPOS>** aggregate	
<POSSTOCK>	**<INVPOS>** aggregate	
	<UNITSSTREET>	Units in the FI's street name, *positive quantity*
	<UNITSUSER>	Units in the user's name directly, *positive quantity*
	<REINVDIV>	Reinvest dividends, *Boolean*

13.9.2.7 Investment Balances <INVBAL>

The <INVBAL> aggregate contains five specified balances. It can also contain a <BALLIST> aggregate that contains one or more <BAL> aggregates. The <BAL> aggregate (see Chapter 3, “Common Aggregates, Elements, and Data Types”) allows an FI to send any number of balances to the user, complete with description and Help text. The intent is to capture the same type of balance information present on the first page of many FI brokerage statements. You can also use the <BAL> aggregate to send margin call information.

	Description
<INVBAL>	Balances aggregate
<AVAILCASH>	Cash balance across all sub-accounts. Should include sweep funds. *Amount*
<MARGINBALANCE>	Margin balance. A positive balance indicates a positive cash balance, while a negative balance indicates the customer has borrowed funds. *Amount*
<SHORTBALANCE>	Market value of all short positions. This is a positive balance, *Amount*
<BUYPOWER>	Buying power, *amount*
<BALLIST>	Beginning of Investment balance list (at most one)
<BAL>	Balance aggregates (0 or more)
</BAL>	See Chapter 3, "Common Aggregates, Elements, and Data Types"
</BALLIST>	
</INVBAL>	

13.9.2.8 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2003	Account not found (ERROR)
2004	Account closed (ERROR)
2005	Account not authorized (ERROR)
2019	Duplicate request (ERROR)
2020	Invalid date (ERROR)
2027	Invalid date range (ERROR)
12250	Investment transaction download not supported (WARN)
12251	Investment position download not supported (WARN)
12252	Investment positions for specified date not available (WARN)
12253	Investment open order download not supported (WARN)
12254	Investment balances download not supported (WARN)

13.9.2.9 401(k) Balances <INV401KBAL>

The <INV401KBAL> aggregate contains an optional cash balance. It also contains the balances of the standard 401(k) sub-accounts. The date of these balances is taken from the <DTASOF> element of the <INVSTMTRS>aggregate.

	Description
<INV401KBAL>	Beginning of 401(k) balances list (at most one)
<CASHBAL>	Cash balance available for the 401(k) account
<PRETAX>	Current value of all securities purchased with Before Tax Employee contributions, *amount*
<AFTERTAX>	Current value of all securities purchased with After Tax Employee contributions, *amount*
<MATCH>	Current value of all securities purchased with Employer Match contributions, *amount*
<PROFITSHARING>	Current value of all securities purchased with Employer Profit Sharing contributions, *amount*
<ROLLOVER>	Current value of all securities purchased with Rollover contributions, *amount*
<OTHERVEST>	Current value of all securities purchased with Other (vesting) Employer contributions, *amount*
<OTHERNONVEST>	Current value of all securities purchased with Other (non-vesting) Employer contributions, *amount*
<TOTAL>	Current value of all securities purchased with all contributions, *amount*
<BALLIST>	Beginning of generic balance list (at most one)
<BAL>	Balance aggregates (zero or more), see Chapter 3, "Common Aggregates, Elements, and Data Types"
</BAL>	
</BALLIST>	
</INV401KBAL>	

13.9.2.10 Status Codes

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2003	Account not found (ERROR)
2004	Account closed (ERROR)
2005	Account not authorized (ERROR)
2019	Duplicate request (ERROR)
2020	Invalid date (ERROR)
2027	Invalid date range (ERROR)
12250	Investment transaction download not supported (WARN)
12251	Investment position download not supported (WARN)
12252	Investment positions for specified date not available (WARN)
12253	Investment open order download not supported (WARN)
12254	Investment balances download not supported (WARN)
12255	401(k) information requested from a non-401(k) account (ERROR)

13.9.3 401(k) Account Information

The following is included in the investment statement response for 401(k) accounts to provide a summary of the user's 401(k) plan information. Note that some of this information may not be available at the server and may be omitted.

Tag	*Description*
<INV401K>	401(k) Summary aggregate
<EMPLOYERNAME>	Name of the employer, *A-32*
<PLANID>	Plan number, *A-32*
<PLANJOINDATE>	Date the employee joined the plan, *date*
<EMPLOYERCONTACTINFO>	Name of contact person at employer, plus any available contact information, such as phone number, *A-255*
<BROKERCONTACTINFO>	Name of contact person at broker, plus any available contact information, such as phone number, A-255

Tag	Description
<DEFERPCTPRETAX>	Percent of employee salary deferred before tax, *rate*
<DEFERPCTAFTERTAX>	Percent of employee salary deferred after tax, *rate*
<MATCHINFO>	Aggregate containing employer match information. Absent if employer does not contribute matching funds.
<MATCHPCT>	Percent of employee contribution matched, e.g., 75% if contribution rate is $0.75/$1.00, *rate*
<MAXMATCHAMT>	Maximum employer contribution amount in any year, *amount.*
<MAXMATCHPCT>	Current maximum employer contribution percentage. Maximum match in a year is MAXMATCHPCT up to the MAXMATCHAMT, if provided. *rate*
<STARTOFYEAR>	Specifies when the employer contribution max is reset. Some plans have a maximum based on the company fiscal year rather than calendar year. Assume calendar year if omitted. Only the month and day (MMDD) are used; year (YYYY) and time are ignored. *date*
<BASEMATCHAMT>	Specifies a fixed dollar amount contributed by the employer if the employee participates in the plan at all. This may be present in addition to the <MATCHPCT>. $0 if omitted. *amount*
<BASEMATCHPCT>	Specifies a fixed percent of employee salary matched if the employee participates in the plan at all. This may be present in addition to the MATCHPCT>. 0% if omitted. Base match in a year is BASEMATCHPCT up to the BASEMATCHAMT,if provided. *rate*
</MATCHINFO>	
<CONTRIBINFO>	Aggregate to describe how new contributions are distributed among the available securities.
<CONTRIBSECURITY>	Identifies current contribution allocation for a security (1 or more)
<SECID>	Security identifier. See section 13.8.1.
</SECID>	
Current contribution allocation. Specify either <xxxPCT> or <xxxAMT>. The new contributions to each security are either all specified by a percentage of contributions or by a fixed dollar amount, but not both. At least one source must be provided.	

<PRETAXCONTRIBPCT>	Percentage of each new employee pretax contribution allocated to this security, *rate.*
-or-	
<PRETAXCONTRIBAMT>	Fixed amount of each new em

Tag	Description
</VESTINFO>	
<LOANINFO>	List of loans outstanding against this account. 0 or more.
<LOANID>	Identifier of this loan, *A-32*
<LOANDESC>	Loan description, *A-32*
<INITIALLOANBAL>	Initial loan balance. *amount*
<LOANSTARTDATE>	Start date of loan. *date*
<CURRENTLOANBAL>	Current loan principal balance. *amount*
<DTASOF>	Date and time of the current loan balance. *datetime*
<LOANRATE>	Loan annual interest rate. *rate*
<LOANPMTAMT>	Loan payment amount. *amount*
<LOANPMTFREQ>	Frequency of loan repayments: WEEKLY, BIWEEKLY, TWICEMONTHLY, MONTHLY, FOURWEEKS, BIMONTHLY, QUARTERLY, SEMIANNUALLY, ANNUALLY, OTHER. See section 10.2.1 for calculation rules.
<LOANPMTSINITIAL>	Initial number of loan payments.
<LOANPMTSREMAINING>	Remaining number of loan payments. *N-5*
<LOANMATURITYDATE>	Expected loan end date. *date*
<LOANTOTALPROJINTEREST>	Total projected interest to be paid on this loan. *amount*
<LOANINTERESTTODATE>	Total interested paid to date on this loan. *amount*
<LOANNEXTPMTDATE>	Next payment due date. *date*
</LOANINFO>	
<INV401KSUMMARY>	List of contributions to 401(k) account.
<YEARTODATE>	Contributions to date for this calendar year.
<DTSTART>	Start date for this calendar year. *date*
<DTEND>	End date for this year-to-date information. *date*
<CONTRIBUTIONS>	401 (k) contribution aggregate (at most one) (See section 13.9.3.1)
</CONTRIBUTIONS>	

Tag	Description
<WITHDRAWALS>	401(k) withdrawals aggregate (at most one) (See section 13.9.3.2)
</WITHDRAWALS>	
<EARNINGS>	401(k) earnings aggregate (at most one) (See section 13.9.3.3)
</EARNINGS>	
</YEARTODATE>	
<INCEPTODATE>	Total contributions to date (since inception)
<DTSTART>	Start date for the inception of this account. *date*
<DTEND>	End date for the inception-to-date information. *date*
<CONTRIBUTIONS>	401(k) contribution aggregate (at most one) (See section 13.9.3.1)
</CONTRIBUTIONS>	
<WITHDRAWALS>	401(k) withdrawals aggregate (at most one) (See section 13.9.3.2)
</WITHDRAWALS>	
<EARNINGS>	401(k) earnings aggregate (at most one) (See section 13.9.3.3)
</EARNINGS>	
</INCEPTODATE>	
<PERIODTODATE>	
<DTSTART>	Start date for the current period. *date*
<DTEND>	End date for the period-to-date information. *date*
<CONTRIBUTIONS>	401(k) contribution aggregate (at most one) (See section 13.9.3.1)
</CONTRIBUTIONS>	
<WITHDRAWALS>	401(k) withdrawals aggregate (at most one) (See section 13.9.3.2)
</WITHDRAWALS>	
<EARNINGS>	401(k) earnings aggregate (at most one) (See section 13.9.3.3)
</EARNINGS>	
</PERIODTODATE>	
</INV401KSUMMARY>	
</INV401K>	

13.9.3.1 401(k) Contribution Aggregate <CONTRIBUTIONS>

The following table shows the new 401(k) contribution aggregate and its tags.

	Description
<CONTRIBUTIONS>	401(k) contribution aggregate. Note: this includes loan payments.
<PRETAX>	Pretax contribution. *amount*
<AFTERTAX>	After tax contribution. *amount*
<MATCH>	Employer matching contribution. *amount*
<PROFITSHARING>	Profit sharing contribution. *amount*
<ROLLOVER>	Rollover contribution. *amount*
<OTHERVEST>	Other vesting contributions. *amount*
<OTHERNONVEST>	Other non-vesting contributions. *amount*
<TOTAL>	Sum of contributions from all fund sources. *amount*
</CONTRIBUTIONS>	

13.9.3.2 401(k) Withdrawals Aggregate <WITHDRAWALS>

	Description
<WITHDRAWALS>	401(k) withdrawals aggregate. Note: this includes loan withdrawals.
<PRETAX>	Pretax withdrawals. *amount*
<AFTERTAX>	After tax withdrawals. *amount*
<MATCH>	Employer matching withdrawals. *amount*
<PROFITSHARING>	Profit sharing withdrawals. *amount*
<ROLLOVER>	Rollover withdrawals. *a580.320011,VER≈PROFITS2(<PRETAX>1JETEMC /Span <</MCID 13*

13.10 Investment Statement Closing Information

OFX provides a way for customers to receive closing statement information that typically appears as part of a paper statement. Image download information can be requested in an investment closing request by including <INCSTMTIMG>Y, which requests that the server return image data corresponding to closing statements, if available. The image identification data returned in the <INVCLOSING> aggregate will not be the actual image, but will be used to access the image through subsequent client sessions. See Chapter 15, “Image Download”, for more information on retrieving images from the server.

13.10.1 Request <INVSTMTENDRQ>

The <INVSTMTENDRQ> request must appear within an <INVSTMTENDTRNRQ> transaction wrapper.

	Description
<INVSTMTENDRQ>	Closing-statement-request aggregate
<INVACCTFROM>	Investment-account-from aggregate
</INVACCTFROM>	
<DTSTART>	Start date for statement closing information, *datetime*
<DTEND>	End date of statement closing information, *datetime*
<INCSTMTIMG>	Include data for closing statement images, *Boolean*
</INVSTMTENDRQ>	

13.10.2 Response <INVSTMTENDRS>

The <INVSTMTENDRS> response must appear within a <INVSTMTENDTRNRS> transaction wrapper.

	Description
<INVSTMTENDRS>	Closing-statement-response aggregate
<CURDEF>	Default currency used for closing information, *currsymbol*
<INVACCTFROM>	Account from aggregate
</INVACCTFROM>	
<INVCLOSING>	Statement information (0 or more), see section 13.10.3
</INVCLOSING>	
</INVSTMTENDRS>	

13.10.2.1 Status Codes

	Meaning
0	Success
2000	General error (ERROR)
2002	General account error (ERROR)
2003	Account not found (ERROR)
2004	Account closed (ERROR)
2005	Account not authorized (ERROR)
2019	Duplicate request (ERROR)
2020	Invalid date (ERROR)
2027	Invalid date range (ERROR)

13.10.3 Investment Statement <INVCLOSING>

The <INVCLOSING> aggregate provides image information for client retrieval of investment statement images. The <FITID> provides a way for the client to distinguish one closng statement from another.

Tag	*Description*
<INVCLOSING>	Closing-statement-response aggregate
<FITID>	Unique identifier for this statement, *FITID*
<IMAGEDATA>	Image data aggregate, see section 3.1.6.1
</IMAGEDATA>	
</INVCLOSING>	

13.11 Investment E-Mail

OFX currently defines one investment e-mail message that clients can send to an FI. With this message, the user can prepare a message to the FI regarding one of their accounts. The server acknowledges receipt of the message. The FI prepares the response that the client picks up when it synchronizes with the server. E-mail is subject to synchronization, using <INVMAILSYNCRQ> / <INVMAILSYNCRS>.

	Server Responds
Addressed message	
Inv. account information	
	Acknowledgment
.	
.	
.	
Synchronization request	
	Response to customer

13.11.1 Investment E-Mail Request and Response

13.11.1.1 Request <INVMAILRQ>

The client must identify to which investment account the customer query is related.

Tag	*Description*
<INVMAILRQ>	Investment-e-mail-request aggregate
<INVACCTFROM>	Account-from aggregate, see 13.6.1
</INVACCTFROM>	
<MAIL>	To, from, message information, see section 9.2.2
</MAIL>	
</INVMAILRQ>	

13.11.2 Investment E-Mail Synchronization

13.11.2.1 Request <INVMAILSYNCRQ>

	Description
<INVMAILSYNCRQ> *Client synchronization option; <TOKEN>, <TOKENONLY>, or <REFRESH>*	Synchronization-request aggregate
<TOKEN>	Previous value of <TOKEN> received for this type of synchronization request from server; 0 for first-time requests; *token*
<TOKENONLY>	Request for just the current <TOKEN> without the history, *Boolean*
<REFRESH>	Request for refresh of current state, *Boolean*
<REJECTIFMISSING>	If Y, do not process requests if client <TOKEN> is out of date, *Boolean*
<INCIMAGES>	Y if the client accepts mail with images in the message body. N if the client does not accept mail with images in the message body. *Boolean*
<USEHTML>	Y if client wants an HTML response, N if client wants plain text, *Boolean*
<INVACCTFROM>	Investment account of interest; token must be interpreted in terms of this account, see 13.6.1
</INVACCTFROM>	
<INVMAILTRNRQ>	Investment-mail transactions (0 or more)
</INVMAILTRNRQ>	
</INVMAILSYNCRQ>	

13.11.2.2 Response <INVMAILSYNCRS>

	Description
<INVMAILSYNCRS>	Synchronization-response aggregate
<TOKEN>	New synchronization token, *token*
<LOSTSYNC>	Y if the token in the synchronization request is older than the earliest entry in the server's history table. In this case, some responses have been lost. N if the token in the synchronization request is newer than or matches a token in the server's history table. *Boolean*
<INVACCTFROM>	Investment account of interest; token must be interpreted in terms of this account, see 13.6.1
</INVACCTFROM>	
<INVMAILTRNRS>	Investment-mail transactions (0 or more)
</INVMAILTRNRS>	
</INVMAILSYNCRS>	

13.12 Complete Example

This example is for a user who requests an investment statement download for a single account.

The request file:

```
<OFX>                          <!--Beginning of request data-->
 <SIGNONMSGSRQV1>
  <SONRQ>                      <!-- ...Sign on request. For a
                               complete example, see section
                               11.14.1-->
  </SONRQ>                     <!--End of signon-->
 </SIGNONMSGSRQV1>
 <INVSTMTMSGSRQV1>
  <INVSTMTTRNRQ>               <!--First request in file-->
    <TRNUID>1001</TRNUID>      <!--Unique ID for this request-->
    <INVSTMTRQ>                <!--Beginning of statement download-->
      <INVACCTFROM>            <!--Identify the account: -->
        <BROKERID>121099999</BROKERID><!--FI ID-->
        <ACCTID>999988</ACCTID><!--Account number-->
      </INVACCTFROM>           <!--End of account ID-->
      <INCTRAN>                <!--Request transactions-->
        <DTSTART>20050824130105</DTSTART><!--Send transactions posted
after-->
                               <!--Aug 24, 2005 1:01:05pm-->

        <INCLUDE>Y</INCLUDE>   <!--Include transactions -->
      </INCTRAN>
      <INCOO>Y</INCOO>         <!--Include open orders in response-->
      <INCPOS>                 <!--Request positions -->
         <INCLUDE>Y</INCLUDE>  <!--Include current positions -->
      </INCPOS>
      <INCBAL>Y</INCBAL>       <!--Include balances in request-->
    </INVSTMTRQ>
  </INVSTMTTRNRQ>              <!--End of first request-->
 </INVSTMTMSGSRQV1>
</OFX>                         <!--End of OFX request data-->
```

A typical server response:

This user has one investment transaction, one bank transaction, one open order, two position entries, and one balance entry. The user deposits some money and buys shares in Acme. The user has an open limit order to buy 100 shares of Hackson Unlimited at $50/share. The holdings show the user already had 100 shares of Acme and now has 200 shares. The user also has one option contract to sell Lucky Airlines shares, bought before this download.

```
<OFX>                              <!--Beginning of request data-->
 <SIGNONMSGSRSV1>
  <SONRS>                          <!-- ...Sign on response. For a
                                   complete example, see section
                                   11.14.1-->
  </SONRS>                         <!--End of signon-->
</SIGNONMSGSRSV1>
<INVSTMTMSGSRSV1>
  <INVSTMTTRNRS>                   <!--First request in file-->
    <TRNUID>1001</TRNUID>          <!--Client ID for this request-->
    <STATUS>
      <CODE>0</CODE>               <!--0 = accepted, good data follows-->
      <SEVERITY>INFO</SEVERITY>
    </STATUS>
    <INVSTMTRS>                    <!--Beginning of statement download-->
      <DTASOF>20050827010000</DTASOF> <!--Statement as of Aug 27, 2005
1am-->
      <CURDEF>USD</CURDEF>         <!--Default currency is US Dollar-->
      <INVACCTFROM>                <!--Beginning of account information-->
        <BROKERID>121099999</BROKERID><!--FI ID-->
.5(!--)5..ITY>    <ACCTID>999988</ACCTID><!--Account number-->
      </INVACCTFROM>               <!--End of account information-->
      <INVTRANLIST>                <!--Beginning of transactions-->
        <DTSTART>20050824130105</DTSTART><!--Send transactions posted
after-->
                                   <!--Aug 24, 2005 1:01:05pm-->
        <DTEND>20050828101000</DTEND><!--End timestamp (now) -->
        <I217ySTOCK5(T>7676.5(<!--)5Buy ofctio)5ck.5( tr tr)5..5(ct ID--)5.5(>)]T
```

```
              <SECID>           <!--Security ID-->
                 <UNIQUEID>123456789</UNIQUEID><!--CUSIP for ACME -->
                 <UNIQUEIDTYPE>CUSIP</UNIQUEIDTYPE>
              </SECID>
              <UNITS>100</UNITS><!--100 shares-->
              <UNITPRICE>50.00</UNITPRICE><!--$50/share-->
              <COMMISSION>25.00</COMMISSION><!--$25 commission -->
              <TOTAL>-5025.00</TOTAL><!--Total amount $5025.00-->
              <SUBACCTSEC>CASH</SUBACCTSEC><!--Holding resides in cash
account-->
              <SUBACCTFUND>CASH</SUBACCTFUND><!--Bought in cash
account-->
           </INVBUY>
           <BUYTYPE>BUY</BUYTYPE><!--Normal buy-->
         </BUYSTOCK>            <!--End of buy stock transaction-->
         <INVBANKTRAN>          <!--Investment acct bank transaction-->
           <STMTTRN>            <!--Beginning of a bank transaction-->
              <TRNTYPE>CREDIT</TRNTYPE><!--Generic credit-->
              <DTPOSTED>20050825</DTPOSTED><!--Aug 25, 2005-->
              <DTUSER>20050825</DTUSER><!--Aug 25, 2005-->
              <TRNAMT>1000.00</TRNAMT><!--$1,000.00-->
              <FITID>12345</FITID><!--FI transaction ID 12345-->
              <NAME>Customer deposit</NAME><!--Description of
transaction-->
```

```
              <MKTVAL>9900.00</MKTVAL><!--Current market value
$9900.00-->
              <DTPRICEASOF>20050827010000</DTPRICEASOF> <!--Prices as
of Aug27,2005
                                          1am-->
              <MEMO>Next dividend payable Sept 1</MEMO>
            </INVPOS>
          </POSSTOCK>            <!--End of position-->
          <POSOPT>               <!--Beginning of position-->
            <INVPOS>
              <SECID>            <!--Security ID-->
                 <UNIQUEID>000342222</UNIQUEID><!--CUSIP for the option
-->
                 <UNIQUEIDTYPE>CUSIP</UNIQUEIDTYPE>
              </SECID>
              <HELDINACCT>CASH</HELDINACCT><!--Cash account-->
              <POSTYPE>LONG</POSTYPE><!--Long position-->
              <UNITS>1</UNITS>  <!--100 shares-->
              <UNITPRICE>5</UNITPRICE><!--Latest price-->
              <MKTVAL>500</MKTVAL><!--Current market value $500.00-->
              <DTPRICEASOF>20050827010000</DTPRICEASOF> <!--Prices as
of Aug27,2005 1am-->
              <MEMO> Option is in the money</MEMO>
            </INVPOS>
          </POSOPT>              <!--End of option position -->
        </INVPOSLIST>            <!--End of position -->
        <INVBAL>
          <AVAILCASH>200.00</AVAILCASH><!--$200.00 cash balance-->
          <MARGINBALANCE>-50.00</MARGINBALANCE><!--$50.00 owed on margin
balance-->
          <SHORTBALANCE>0</SHORTBALANCE><!--$0 short balance-->

          <BALLIST>              <!--Beginning of FI-defined balances-->
            <BAL>                <!--Beginning of a balance-->
              <NAME>Margin Interest Rate</NAME> <!--Name of balance
entry-->
              <DESC>Current interest rate on margin balances</DESC>
                                   <!--Help text for this balance-->
              <BALTYPE>PERCENT</BALTYPE><!--Format as percent-->
              <VALUE>7.85</VALUE><!--Will be formatted 7.85%-->
              <DTASOF>20050827010000</DTASOF> <!--Rate as of Aug 27,
2005 1am-->
            </BAL>               <!--End of balance entry-->
          </BALLIST>             <!--End of balances-->
```

```
      </INVBAL>
      <INVOOLIST>
       <OOBUYSTOCK>
          <OO>
             <FITID>23321</FITID><!--FI transaction ID-->
             <SECID> <!--Security ID-->
                <UNIQUEID>666678578</UNIQUEID><!--CUSIP for Hackson
Unlimited-->
                <UNIQUEIDTYPE>CUSIP</UNIQUEIDTYPE>
             </SECID>
             <DTPLACED>20050624031505</DTPLACED> <!--Order placed 6/
24/96 3:15:05pm-->
             <UNITS>100</UNITS><!--100 shares-->
             <SUBACCT>CASH</SUBACCT><!--Purchase with cash-->
             <DURATION>GOODTILCANCEL</DURATION><!--GOODTILCANCEL-->
             <RESTRICTION>NONE</RESTRICTION><!--No special
restrictions-->
          <LIMITPRICE>50.00</LIMITPRICE><!--Limit price $50/share-->
          </OO>
          <BUYTYPE>BUY</BUYTYPE><!--Normal buy-->
        </OOBUYSTOCK>
      </INVOOLIST>
    </INVSTMTRS>
  </INVSTMTTRNRS>              <!--End of first response-->
 </INVSTMTMSGSRSV1>
 <SECLISTMSGSRSV1>
  <SECLIST>                    <!--Beginning of securities list-->
    <STOCKINFO>                <!--Beginning of 1st security ID-->
      <SECINFO>
        <SECID>                <!--Security ID-->
          <UNIQUEID>123456789</UNIQUEID><!--CUSIP for the stock -->
          <UNIQUEIDTYPE>CUSIP</UNIQUEIDTYPE>
        </SECID>
        <SECNAME>Acme Development, Inc.</SECNAME>
        <TICKER>ACME</TICKER>  <!--Ticker symbol-->
        <FIID>1024</FIID>      <!--FI internal security identifier-->
      </SECINFO>
      <YIELD>10</YIELD>        <!--10% yield-->
      <ASSETCLASS>SMALLSTOCK</ASSETCLASS><!--Small Capital Stock asset
class-->
    </STOCKINFO>               <!--End of security ID-->
    <STOCKINFO>
      <SECINFO>
```

```
        <SECID>                 <!--Security ID-->
          <UNIQUEID>666678578</UNIQUEID><!--CUSIP for the stock -->
          <UNIQUEIDTYPE>CUSIP</UNIQUEIDTYPE>
        </SECID>
        <SECNAME>Hackson Unlimited, Inc.</SECNAME>
        <TICKER>HACK</TICKER>  <!--Ticker symbol-->
        <FIID>1027</FIID>      <!--FI internal security identifier-->
      </SECINFO>
      <YIELD>17</YIELD>        <!--17% yield-->
      <ASSETCLASS>SMALLSTOCK</ASSETCLASS><!--Small Capital Stock asset
class-->
    </STOCKINFO>
    <OPTINFO>                  <!--End of security ID-->
      <SECINFO>
        <SECID>                <!--Security ID-->
          <UNIQUEID>000342222</UNIQUEID><!--CUSIP for the option -->
          <UNIQUEIDTYPE>CUSIP</UNIQUEIDTYPE>
        </SECID>
        <SECNAME>Lucky Airlines Jan 97 Put</SECNAME>
        <TICKER>LUAXX</TICKER> <!--Ticker symbol-->
        <FIID>0013</FIID>      <!--FI internal security identifier-->
      </SECINFO>
      <OPTTYPE>PUT</OPTTYPE>
      <STRIKEPRICE>35.00</STRIKEPRICE><!--Strike price $35/share-->
      <DTEXPIRE>20050121</DTEXPIRE><!--Option expires Jan 21, 2005-->
      <SHPERCTRCT>100</SHPERCTRCT> <!--100 shares per contract-->
      <SECID>                  <!--Security ID-->
        <UNIQUEID>000342200</UNIQUEID><!--CUSIP for the underlying
stock -->
        <UNIQUEIDTYPE>CUSIP</UNIQUEIDTYPE>
      </SECID>
      <ASSETCLASS>LARGESTOCK</ASSETCLASS><!--Large Capital Stock asset
class-->
    </OPTINFO>                 <!--End of option information-->
  </SECLIST>                   <!--End of securities list-->
 </SECLISTMSGSRSV1>
</OFX>                         <!--End of OFX request data-->
```

13.13 Complete 401(k) Example

This example is for a user who requests a 401(k) investment statement download for a single account.

The request file:

```
<OFX>                                   <!--Beginning of request data-->
 <SIGNONMSGSRQV1>
   <SONRQ>                               <!-- ...Sign on request. For a
                                         complete example, see section
                                         11.14.1-->
   </SONRQ>                              <!--End of signon-->
 </SIGNONMSGSRQV1>
 <INVSTMTMSGSRQV1>
   <INVSTMTTRNRQ>                        <!--First request in file-->
      <TRNUID>1002</TRNUID>              <!--Unique ID for this request-->
      <INVSTMTRQ>                        <!--Beginning of statement download-->
         <INVACCTFROM>                   <!--Identify the account: -->
            <BROKERID>121099999</BROKERID><!--FI ID-->
            <ACCTID>999988</ACCTID><!--Account number-->
         </INVACCTFROM>                  <!--End of account ID-->
         <INCTRAN>                       <!--Request transactions-->
            <DTSTART>20060101120000</DTSTART><!--Send transactions posted
after Jan 1, 2007 12pm-->
            <INCLUDE>Y</INCLUDE>         <!--Include transactions -->
         </INCTRAN>
         <INCPOS>                        <!--Request positions -->
            <INCLUDE>Y</INCLUDE>         <!--Include current positions -->
         </INCPOS>
         <INC401K>Y</INC401K>            <!--Include 401(k) account info -->
         <INC401KBAL>Y</INC401KBAL><!--Include 401(k) balances -->
      </INVSTMTRQ>
   </INVSTMTTRNRQ>                       <!--End of first request-->
 </INVSTMTMSGSRQV1>
</OFX>                                   <!--End of OFX request data-->
```

A typical server response:

This user is paying back a loan from the 401(k) account and then contributing pretax dollars. A transaction is shown paying principal to Rollover and another paying interest to Rollover. Following that is a pretax contribution transaction. This is then followed by the list of the 401(k) balances and finally the 401(k) account information including a year-to-date summary.

```
<OFX>                          <!--Beginning of request data-->
  <SIGNONMSGSRSV1>
    <SONRS>                    <!-- ...Sign on response. For a
                               complete example, see section
                               11.14.1-->
    </SONRS>                   <!--End of signon-->
  </SIGNONMSGSRSV1>
  <INVSTMTMSGSRSV1>
    <INVSTMTTRNRS>     <!--First request in file-->
      <TRNUID>1002</TRNUID><!--Client ID for this request-->
      <STATUS>
        <CODE>0</CODE>  <!--0 = accepted, good data follows-->
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <INVSTMTRS>      <!--Beginning of statement download-->
        <DTASOF>>20060131172605.000[-4:EST]</DTASOF> <!--Statement as
of Jan 31, 2006 5:26pm-->
        <CURDEF>USD</CURDEF><!--Default currency is US Dollar-->
        <INVACCTFROM><!--Beginning of account information-->
          <BROKERID>121099999</BROKERID><!--FI ID-->
          <ACCTID>999988</ACCTID><!--Account number-->
        </INVACCTFROM><!--End of account information-->
        <INVTRANLIST>
          <DTSTART>20060105172532.000[-5:EST]</DTSTART>
          <DTEND>20060131172532.000[-4:EST]</DTEND>
             <BUYMF>
                <INVBUY>
                   <INVTRAN>
                      <FITID>21283906282029531072 3</FITID>
                      <DTTRADE>20060119000000.000[-5:EST]</DTTRADE>
                   </INVTRAN>
                   <SECID>
                      <UNIQUEID>744316100</UNIQUEID>
                      <UNIQUEIDTYPE>CUSIP</UNIQUEIDTYPE>
                   </SECID>
                   <UNITS>14.6860</UNITS>
```

```
            <UNITPRICE>18.9000</UNITPRICE>
            <TOTAL>-277.5700</TOTAL>
            <CURRENCY>
               <CURRATE>1.0000</CURRATE>
               <CURSYM>USD</CURSYM>
            </CURRENCY>
            <SUBACCTSEC>OTHER</SUBACCTSEC>
            <SUBACCTFUND>OTHER</SUBACCTFUND>
            <LOANID>2</LOANID>
            <LOANPRINCIPAL>277.5700</LOANPRINCIPAL>
            <LOANINTEREST>0.0000</LOANINTEREST>
            <INV401KSOURCE>ROLLOVER</INV401KSOURCE>
            <DTPAYROLL>20060114000000.000[-5:EST]</DTPAYROLL>
            <PRIORYEARCONTRIB>N</PRIORYEARCONTRIB>
         </INVBUY>
         <BUYTYPE>BUY
      </BUYMF>
      <BUYMF>
         <INVBUY>
            <INVTRAN>
               <FITID>212839062820510822977</FITID>
               <DTTRADE>20060119000000.000[-5:EST]</DTTRADE>
            </INVTRAN>
            <SECID>
               <UNIQUEID>744316100</UNIQUEID>
               <UNIQUEIDTYPE>CUSIP</UNIQUEIDTYPE>
            </SECID>
            <UNITS>2.0220</UNITS>
            <UNITPRICE>18.9000</UNITPRICE>
            <TOTAL>-38.2200</TOTAL>
            <CURRENCY>
               <CURRATE>1.0000</CURRATE>
               <CURSYM>USD</CURSYM>
            </CURRENCY>
            <SUBACCTSEC>OTHER</SUBACCTSEC>
            <SUBACCTFUND>OTHER</SUBACCTFUND>
            <LOANID>2</LOANID>
            <LOANPRINCIPAL>0.0000</LOANPRINCIPAL>
            <LOANINTEREST>38.2200</LOANINTEREST>
            <INV401KSOURCE>ROLLOVER</INV401KSOURCE>
            <DTPAYROLL>20060114000000.000[-5:EST]</DTPAYROLL>
            <PRIORYEARCONTRIB>N</PRIORYEARCONTRIB>
```

```
            </INVBUY>
            <BUYTYPE>BUY
         </BUYMF>
         <BUYMF>
            <INVBUY>
               <INVTRAN>
                  <FITID>212849815151950488609</FITID>
                  <DTTRADE>20060106000000.000[-5:EST]</DTTRADE>
               </INVTRAN>
               <SECID>
                  <UNIQUEID>744316100</UNIQUEID>
                  <UNIQUEIDTYPE>CUSIP</UNIQUEIDTYPE>
               </SECID>
               <UNITS>4.9010</UNITS>
               <UNITPRICE>18.7900</UNITPRICE>
               <TOTAL>-92.0900</TOTAL>
               <CURRENCY>
                  <CURRATE>1.0000</CURRATE>
                  <CURSYM>USD</CURSYM>
               </CURRENCY>
               <SUBACCTSEC>OTHER</SUBACCTSEC>
               <SUBACCTFUND>OTHER</SUBACCTFUND>
               <INV401KSOURCE>PRETAX</INV401KSOURCE>
               <DTPAYROLL>20051231000000.000[-5:EST]</DTPAYROLL>
               <PRIORYEARCONTRIB>Y</PRIORYEARCONTRIB>
            </INVBUY>
         <BUYTYPE>BUY</BUYTYPE>
      </BUYMF>
   </INVTRANLIST>
   <INV401KBAL>
      <PRETAX>31690.340000</PRETAX>
      <PROFITSHARING>10725.640000</PROFITSHARING>
      <ROLLOVER>15945.750000</ROLLOVER>
      <OTHERVEST>108.800000</OTHERVEST>
      <TOTAL>58470.530000</TOTAL>
   </INV401KBAL>
   <INV401K>
      <EMPLOYERNAME>ELGIN NATIONAL INDUSTRIES INC</EMPLOYERNAME>
      <PLANID>4343</PLANID>
      <PLANJOINDATE>19940101000000.000[-5:EST]</PLANJOINDATE>
      <MATCHINFO>
         <MATCHPCT>0.00</MATCHPCT>
```

```
            </MATCHINFO>
            <CONTRIBINFO>
               <CONTRIBSECURITY>
                  <SECID>
                     <UNIQUEID>744316100</UNIQUEID>
                     <UNIQUEIDTYPE>CUSIP</UNIQUEIDTYPE>
                  </SECID>
               <PRETAXCONTRIBPCT>50.0000</PRETAXCONTRIBPCT>
               <PROFITSHARINGCONTRIBPCT>100.0000
</PROFITSHARINGCONTRIBPCT>
               <ROLLOVERCONTRIBPCT>100.0000</ROLLOVERCONTRIBPCT>
               <OTHERVESTPCT>100.0000</OTHERVESTPCT>
               </CONTRIBSECURITY>
               <CONTRIBSECURITY>
                  <SECID>
                     <UNIQUEID>74431M105</UNIQUEID>
                     <UNIQUEIDTYPE>CUSIP</UNIQUEIDTYPE>
                  </SECID>
                  <PRETAXCONTRIBPCT>25.0000</PRETAXCONTRIBPCT>
                  <PROFITSHARINGCONTRIBPCT>0.0000
</PROFITSHARINGCONTRIBPCT>
                  <ROLLOVERCONTRIBPCT>0.0000</ROLLOVERCONTRIBPCT>
                  <OTHERVESTPCT>0.0000</OTHERVESTPCT>
               </CONTRIBSECURITY>
               <CONTRIBSECURITY>
                  <SECID>
                     <UNIQUEID>743969107</UNIQUEID>
                     <UNIQUEIDTYPE>CUSIP</UNIQUEIDTYPE>
                  </SECID>
                  <PRETAXCONTRIBPCT>25.0000</PRETAXCONTRIBPCT>
                  <PROFITSHARINGCONTRIBPCT>0.0000
</PROFITSHARINGCONTRIBPCT>
                  <ROLLOVERCONTRIBPCT>0.0000</ROLLOVERCONTRIBPCT>
                  <OTHERVESTPCT>0.0000</OTHERVESTPCT>
               </CONTRIBSECURITY>
            </CONTRIBINFO>
            <INV401KSUMMARY>
               <YEARTODATE>
                  <DTSTART>20060101000000</DTSTART>
                  <DTEND>20060131000000</DTEND>
                  <CONTRIBUTIONS>
                     <PRETAX>843.2500</PRETAX>
                     <AFTERTAX>43.4200</AFTERTAX>
```

```
                    <MATCH>421.6200</MATCH>
                    <TOTAL>1308.2900</TOTAL>
                  </CONTRIBUTIONS>
              </INV401KSUMMARY>
            </INV401K>
          </INVSTMTRS>
      </INVSTMTTRNRS>
   </INVSTMTMSGSRSV1>
</OFX>
```

CHAPTER 14 BILL PRESENTMENT

14.1 Overview

Bill Presentment (PRES) is the electronic delivery of a bill from a biller to a customer.

Although some billers may provide Bill Presentment service themselves, many will choose to work with a bill publisher that provides Bill Presentment service on behalf of many billers. For this reason, Bill Presentment focuses on connecting customers to bill publishers.

14.1.1 Bill Presentment Model

This section summarizes the process of receiving bills electronically, starting with the steps required to find a bill publisher and set up Bill Presentment service.

To receive bills electronically, the client:

- Finds one or more billers by searching a biller directory server.
- Determines which bill publishers provide Bill Presentment service for the billers.
- Enrolls with a bill publisher for Bill Presentment service
- Signs on with the bill publisher and activates Bill Presentment service for one or more accounts with one or more billers.
- Requests electronic bills from the bill publisher.
- (Optionally) Pays bills using the OFX Bill Payment service.

14.1.2 Servers and Message Sets

During the billing process, the client typically communicates with two OFX servers:

- *Biller directory server:* An independent server that stores information about billers and bill publishers. Clients can query this server to find the bill publishers that serve the billers in which the customer is interested.
- *Bill publisher server:* The server that delivers bills to customers. A single bill publisher can provide Bill Presentment service for many billers. In some cases, a biller might act as its own bill publisher.

Although it is possible for a single server to perform both of the functions listed above, it is more likely that independent directory servers will provide clients with a single source for finding billers. To allow these functions to be routed separately by clients, Bill Presentment defines separate message sets for directory query and bill delivery.

- Biller Directory message set <PRESDIRMSGSETV1>
- Bill Delivery message set <PRESDLVMSGSETV1>

For additional information about the message sets defined for Bill Presentment, see section 14.7.

14.2 Biller Directory

To find billers, the client sends a <FINDBILLERRQ> request to the biller directory server. The biller directory server returns a <FINDBILLERRS> response.

<FINDBILLERRQ> and <FINDBILLERRS> are part of the Biller Directory message set <PRESDIRMSGSETV1>. The message set tags are <PRESDIRMSGSRQV1> and <PRESDIRMSGSRSV1>.

14.2.1 Client Signon to the Biller Directory Server

Because the client does not enroll with the biller directory server and the directory does not contain private data, the client may perform an anonymous signon (as described in section 2.5.1) when requesting this service. Unlike the FI Profile message set (see Chapter 7, especially section 7.1.4), this message set does not currently support customer-specific directories. The response should be identical whether or not the request arrived with an anonymous <SONRQ>. Compliant servers must support both request forms. For more information about signon requests, refer to Chapter 2, "Structure."

14.2.2 Search Arguments

If the client omits all elements in the <FINDBILLERRQ>, the client is requesting a complete directory of billers. Otherwise, the client wants to filter results based on the included elements. For each biller that matches the elements in the request, the biller directory server returns the complete name and address of the biller, plus the biller ID and bill publisher name. Servers can return information using case-insensitive matching, but this is not required.

14.2.3 Identification of Bill Publishers

Bill Publishers must be uniquely and consistently identified by name. Clients need some way to relate the bill publisher name given by a directory server to their own databases of known and approved bill publishers. Since the number of bill publishers is relatively small, and the number of directory servers that must be coordinated even smaller, the official corporate name of a bill publisher will serve as an ID for that publisher.

The name and address fields refer to the biller, except for <CONSUPOSTALCODE> which refers to the customer's address.

	Description
<FINDBILLERRQ>	
<DTUPDATE>	Date and time of last change to any biller entry as reported by the server on previous query, *datetime.* If present, <FINDBILLERRS> will include only Billers whose information has changed or been added since this time.
<BILLERID>	ID of this biller at this bill publisher, *A-32*
<NAME>	Biller's name, *A-32*
<ADDR1>	Biller's address line 1, *A-32*
<ADDR2>	Biller's address line 2, *A-32*
<ADDR3>	Biller's address line 3, *A-32*
<CITY>	Biller's city, *A-32*
<STATE>	Biller's state, *A-5*
<POSTALCODE>	Biller's postal code, *A-11*
<COUNTRY>	ISO/DIS-3166 3-letter country code standard, *A-3*
<SIC>	Standard Industry Code, *N-6*
<CONSUPOSTALCODE>	Postal code of customer, to allow server to filter out billers that do not do business in the customer's area, *A-11*
<INCIMAGES>	Y if the client wants images (logos) returned, *Boolean*
</FINDBILLERRQ>	

Note: Future versions of OFX will require <ADDR1> if <ADDR2> is specified and <ADDR2> if <ADDR3> is specified.

14.2.5 Find Biller Response <FINDBILLERRS>

<FINDBILLERRS> must appear within a <FINDBILLERTRNRS> transaction wrapper.

The response is a list of <BILLERINFO> aggregates.

	Description
<FINDBILLERRS>	
<DTUPDATE>	Date and time of last addition or modification to the entries in the directory, whether part of this response or not, *datetime*
<BILLERINFO>	Zero or more <BILLERINFO> aggregates
</BILLERINFO>	
</FINDBILLERRS>	

14.2.5.1 Biller Information <BILLERINFO>

<BILLERINFO> includes information about a single biller.

Besides basic name and address information, <BILLERINFO> includes the <BILLPUB> and <BILLERID> elements. These elements will be used with the customer's account number to identify the customer's account with the biller. For more information about the account-identification aggregates, refer to <PRESACCTFROM> and <PRESACCTTO> in section 14.3.2.2.

<BILLERINFO> can optionally include elements that specify the format of valid account numbers. <ACCTFORMAT> and <ACCTEDITMASK> provide information to the client. <HELPMESSAGE> provides a text message that the client can display to the customer.

To avoid the complications caused by invalid account numbers, <BILLERINFO> can also include a <VALIDATE>*URL* element that the client application can use to validate the customer's account number. See section 14.2.7 for more detail on this.

Tag	*Description*
<BILLERINFO>	
<BILLPUB>	Official standard name of the bill publisher, *A-32*
<BILLERID>	ID of this biller at this bill publisher, *A-32*
<NAME>	Name of the biller, *A-32*
<ADDR1>	Biller's address line 1, *A-32*
<ADDR2>	Biller's address line 2, *A-32*
<ADDR3>	Biller's address line 3, *A-32*

Tag	Description
<CITY>	Biller's city, *A-32*
<STATE>	Biller's state, *A-5*
<POSTALCODE>	Biller's postal code, *A-11*
<COUNTRY>	Biller's country; 3-letter country code from ISO/DIS-3166, *A-3*
<SIC>	Standard Industrial Classification Code, *N-6*
<PHONE>	Biller's phone number for customer information (if a special number exists for electronic billing information, use that number), *A-32*
<PAYMENTINSTRUMENTS>	Types of payment that the biller can accept electronically, see section 14.2.8.1
</PAYMENTINSTRUMENTS>	
<ACCTFORMAT>	Regular expression describing the account number format. For example, ^[0-9]{8,10}$ means the account number must be numbers only, and the length must be 8 to 10 numbers. *A-255*
<ACCTEDITMASK>	An alternative string describing the account number format. See below for details. The client can use the edit mask to assist the user in entering the account number. *A-255*
<HELPMESSAGE>	Human-readable message that the client can display to assist the customer in entering his or her account number. For example: "Enter in the last 10 digits of your account number without any spaces or dashes." This is defined by the biller during the implementation phase. *A-255*
<RESTRICT>	Human-readable description of any restrictions on who may sign up with this biller. For example: "Please be sure to enter each account number separately if you have more than one account with us. Your mail bill will be turned off only after another paper billing cycle has passed. Please note that this program is initially available for account numbers beginning with X Y and Z only." This is defined by the biller during the implementation phase. *A-255*
<LOGO>	URL of the biller's logo. If the client requested images, the logo should be included via multipart MIME in this response. *URL*
<VALIDATE>	URL for validation. The client application may use this to validate the customer's account number, see section 14.2.7. *URL*
<BILLERINFOURL>	URL of human-readable description of additional information the biller would like the customer to have with regard to signing up. *URL*
</BILLERINFO>	

Note: Future versions of OFX will require <ADDR1> if <ADDR2> is specified and <ADDR2> if <ADDR3> is specified.

While <ACCTFORMAT> uses Unix-style regular expressions to describe the account number format, <ACCTEDITMASK> provides a simpler, alternative method. It uses two special characters. The character @ matches one letter, upper or lowercase. The character # matches one number. All other characters match themselves (letters are case insensitive).

Usage examples:

The following represents a 16-digit account number:

################

A 16-digit account number, separated by hyphens into 4-digit chunks. First four characters must be 4128:

4128-####-####-####

4 letters, a hyphen, a number, a hyphen, 5 numbers:

@@@@-#-#####

10-digit account number that must begin with 153AG, and whose final 5 characters are numbers:

153AG#####

14.2.6 Status Codes <FINDBILLERRS>

	Meaning
0	Success (INFO)
2000	General error (ERROR)

14.2.7 Account Number Validation

Servers should implement a lightweight CGI (or equivalent) to validate account numbers. The URL provided in the <VALIDATE> can be accessed with an HTTP GET with three arguments: BILLERID, ACCOUNTNUMBER and CUSTOMERPOSTALCODE. The URL should respond with a text file that includes the following values:

1. Status: (Mandatory)

   Error: An error condition (wrong number of parameters, Database error, etc.). Clarifying text may accompany the error status.

   Passed: The account number is in an acceptable form for this biller (this is not a guarantee that the account will be accepted for the service).

   Failed: The account number does not correspond to an acceptable account number for this biller. Clarifying text may accompany the failed status.

2. Account: (Optional) The preferred format or version of the account number presented in the request.

   Heading: (Optional) Additional text to help explain problems to end-users.

Example:

<VALIDATE> = http://testit.com/validate.cgi

Client application uses HTTP GET with “http://testit.com/validate.cgi?billerid=5454&accountnumber=123-456-7890&customerpostalcode=12345”

The server would respond with one of these:

1. Error

   Content-type: text/plain

   <STATUS>error

   <HEADING>The server is unable to process your request at this time. Please resubmit.

2. Failure

   Content-type: text/plain

   <STATUS>failed

   <HEADING>123-456-7890 does not appear to be a valid account number

3. Passed

   Content-type: text/plain

   <STATUS>passed

   <ACCOUNT>1234567890

14.2.8.2 Payment Type and Brand <PAYMENTINSTRUMENT>

Each payment instrument is described by <PMTINSTRUMENTTYPE> and <BRAND>. If the server does not specify <BRAND>, the client assumes that a

14.3.1 Enrollment

To enroll with a bill publisher, the client uses the standard OFX enrollment aggregate <ENROLLRQ>. The bill publisher server returns an <ENROLLRS> that provides status about the enrollment and optionally returns a user ID and password to be used during subsequent signons.

14.3.2 Account Inquiry

To receive account information from a bill publisher, the client can use the standard OFX <ACCTINFORQ> aggregate, contained in the <ACCTINFOTRNRQ> wrapper.

The <ACCTINFORS> response returns a <PRESACCTINFO> aggregate for each of the customer's accounts with the billers at that bill publisher. Typically, the response will list only those accounts that have been activated for Bill Presentment service, not all available accounts.

Unlike a financial institution, bill publishers generally won't have information about all the accounts of its supported billers. Billers that also serve as their own bill publishers may be able return available accounts as well as activated accounts. The bill publisher can use the <AVAILACCTS> element in the profile for the Signup message set to indicate whether the server can return available account information.

If the server cannot return information about all available accounts, the client must ask customers for account information prior to requesting service activation for one or more accounts.

14.3.2.1 Bill Presentment Account Information <PRESACCTINFO>

The <PRESACCTINFO> aggregate appears within the <ACCTINFORS> aggregate.

	Description
<PRESACCTINFO>	Opening tag for bill presentment account information
<PRESACCTFROM>	Bill presentment account identification, see section 14.3.2.2
</PRESACCTFROM>	
<SVCSTATUS>	Status of the Bill Presentment service for this account– AVAIL, PEND, ACTIVE, or REJECTED
<REASON>	Relevant only if <SVCSTATUS>REJECTED is specified, *A-255*
</PRESACCTINFO>	Closing tag for bill presentment account information

14.3.2.2 Account Identification <PRESACCTFROM> <PRESACCTTO>

The <PRESACCTFROM> aggregate uniquely identifies a customer's account with a biller by the combination of bill publisher, biller ID, and account number. Biller IDs must be unique within a bill publisher.

Clients can optionally include a <USERID> in <PRESACCTFROM/TO> that is different from the one used in the <SONRQ>. This <USERID> supports account activation by a third party on behalf of a user. Based on access rights granted to the <SONRQ>, it is up to the server whether to honor such a request. More than one <SVCADD> can be present in a a single <OFX> request file on behalf of multiple <USERID>s.

The <USERID> element is disallowed for single-customer OFX request files. When sent by or returned to a client proxy (or, other group context), any <PRESACCTFROM/TO> aggregates must include the USERID element. A client proxy would include (and receive) this element when initiating an OFX session. But, customer-specific clients initiating a session with the same server would never use or see this information.

> **Note:** <USERID> is not intended to identify individual users of joint accounts. If a transaction might include two different USERIDs within otherwise identical <PRESACCTFROM> aggregates, servers should deliver two separate bills (download the same bill twice in <PRESLISTRS>), allow either to update the bill status (in <PRESNOTIFYRQ> or <BILLSTATUSMODRQ>), or deliver two separate <PRESACCTINFO> aggregates in <PRESGRPACCTINFOTRNRS>. It is up to the server to keep track of activity on joint accounts.

The Bill Publisher would not normally know the <PAYEEID> and <PAYEELSTID> (or their <SPNAME> context) information relevant to a particular client or user. But, after setting up (or changing) Biller as a payee with some Payment provider, a client may execute a <SVCCHG> <ACCTRQ> request that updates the corresponding <PRESACCTFROM> at the Bill Publisher. This request could pass payment identifiers

to the Bill Publisher. Since this information does not make the <PRESACCTFROM> more unique, clients may omit <SPNAME>, <PAYEEID> and <PAYEELSTID> when using <PRESACCTFROM> outside an <ACCTRQ> request. Servers should return this information in all contexts. After a client has supplied payee information, servers must not make server-initiated changes to the information unless the given (or implied) <SPNAME> is controlled by the same provider. Such servers may include (and update) payment identifiers from their Payment provider prior to the client including them in an <ACCTRQ> request.

	Description
<PRESACCTFROM>	
<BILLPUB>	Official standard name of bill publisher, *A-32*
<BILLERID>	ID of this biller at this bill publisher, *A-32*
<BILLERNAME>	Name of the biller; matches <NAME> element in <BILLERINFO>. See section 14.2.5.1, This element may be used only in cases where <PRESACCTFROM> is sent by the server (for example, in <PRESBILLINFO> or <PRESACCTINFO>), *A-32*
<ACCTID>	Account number, *A-22*
<PRESNAMEADDRESS>	Customer's name/address with the biller, see section 14.3.2.2.1
</PRESNAMEADDRESS>	
<USERID>	Customer's user ID, *A-32*
<SPNAME>	Service provider name. Used to scope the <PAYEEID> and /or <PAYEELSTID> (if provided) to a particular Payment service. Allowed only when <PAYEEID> or <PAYEELSTID> appear in <PRESACCTFROM>. *A-32* **Note:** <SPNAME> must be supplied with <PAYEELSTID> unless Bill Publisher also provides a payment service.
<PAYEEID>	Payee identifier. Identifies this Biller at the user's Payment provider. When sent in account activation, it is intended for storage on the Bill Presentment database, such that it can be returned in subsequent inquiries utilizing this aggregate. Used by clients to facilitate accurate Payee add or change requests when the Bill Presentment service (possibly, due to a prior client <ACCTRQ> request) knows the Payee ID at a Payment provider. The client may use this identifier to match the Biller as known by the Bill Presentment service to a Payee as known by the Payment provider.See section 14.5. *SRVRTID*
<PAYEELSTID>	Payee list identifier. Identifies this Biller on the user's payee list at their Payment provider. When sent in account activation, it is intended for storage on the Bill Presentment database, such that it can be returned in subsequent inquiries utilizing this aggregate. Used by clients to facilitate accurate Payee add or change requests when the Bill Presentment service (possibly, due to a prior client <ACCTRQ> request) knows the Payee List ID at a Payment provider. The client may use this identifier to match the Biller as known by the Bill Presentment service to a Payee as known by the Payment provider. See section 14.5. *SRVRTID*
</PRESACCTFROM>	

<PRESACCTTO> follows the same structure as <PRESACCTFROM>.

14.3.2.2.1 Customer Information with the biller <PRESNAMEADDRESS>

	Description
<PRESNAMEADDRESS>	Customer name and address information
<NAMEACCTHELD>	Customer's name as it appears on the account,

customer at the given biller. For example, some billers and service providers may require exact matches for remittance addresses. The user's enrollment data may specify the same location described in the biller's database, but not provide an exact match. In that case, <PRESNAMEADDRESS> may be required for successful service activation.

In cases where <PRESACCTTO> is forwarded to other servers but enrollment information was used rather than a client-provided <PRESNAMEADDRESS> aggregate, the <PRESNAMEADDRESS> information is neither stored at the server nor returned to the originating client with the <PRESACCTTO> aggregate. That is, clients which do not include the <PRESNAMEADDRESS> aggregate in their requests should never see it in a later response from the server.

14.3.3.2 Service Change <SVCCHG>

The server's profile indicates support for storing <PRESNAMEADDRESS> information by including <CANUPDATEPRESNAMEADDRESS>Y. In this case, <SVCCHG> requests may be used to update or delete the <PRESNAMEADDRESS> information stored by the server. <CHGUSERINFORQ> requests have no effect upon this information. Furthermore, no server-initiated transactions change the stored address data. Servers must always return <PRESNAMEADDRESS> data exactly as the client sent it. The presentment server may however forward a customer's change of address entered via <SVCCHG> to the proper biller.

For servers that support storing <PRESNAMEADDRESS>, the <PRESNAMEADDRESS> data can also be used to support users who receive bills at multiple locations.

If a server's profile includes <CANUPDATEPRESNAMEADDRESS>N, the <PRESNAMEADDRESS> should not be included in any <SVCCHG> requests. <PRESNAMEADDRESS> data is used only for the initial activation in this case.

14.3.4 Service Status Update for Groups of Customers

The service activation requested with <SVCADD> will often not happen immediately. In this case, a request for account information <ACCTINFORQ> will return an <ACCTINFORS> with a <SVCSTATUS>PEND. To find out whether the account has been activated, the client can either send <ACCTINFORQ> once per session until it returns <SVCSTATUS>ACTIVE, or it can include an <ACCTSYNCRQ> in each session to catch an unsolicited <ACCTRS> response to the <SVCADD> message.

This section describes a method of checking for status changes on behalf of a group of customers within the Bill Presentation service. This method is designed to be used by customer service representatives and client proxy systems. This method applies only where <SVC> is PRESSVC. To check status for a single customer, use the standard signup messages. For more information, see Chapter 8, "Activation & Account Information."

14.3.4.1 Account Information Request <ACCTINFORQ>

The client uses <ACCTINFORQ> to request information about accounts whose status has changed since the last time the request was made. This request is typically used to retrieve a list of accounts whose status has changed from <SVCSTATUS>PEND to <SVCSTATUS>ACTIVE.

For use in the Bill Presentation message set, the <ACCTINFORQ> request must appear within a <PRESGRPACCTINFOTRNRQ> transaction wrapper. This transaction wrapper includes the identifier for the requested group. The <ACCTINFORQ> request may also appear in the <ACCTINFOTRNRQ> wrapper described in Chapter 8, "Activation & Account Information."

	Description
<ACCTINFORQ>	Opening tag for billing account information request
<DTACCTUP>	Last <DTACCTUP> received in a response, *datetime*
<SVC>	Zero or more. Services to be included in <ACCTINFORS>. If absent, all supported services are being requested. BANKSVC = Banking service BPSVC = Payment service INVSVC = Investments PRESSVC = Bill Presentment service **Note:** If used in this message set, the value must be PRESSVC.
</ACCTINFORQ>	Closing tag for billing account information request

14.3.4.2 Account Information Response <ACCTINFORS>

The <ACCTINFORS> aggregate contains zero or more <ACCTINFO> aggregates, which provide the updated account information.

For use in this message set, the <ACCTINFORS> response must appear within a <PRESGRPACCTINFOTRNRS> transaction wrapper.

	Description
<ACCTINFORS>	Opening tag for billing account information response
<DTACCTUP>	Date and time of last update to account information on the server, *datetime*
<ACCTINFO>	Zero or more account information aggregates, see section 8.5.3. Each <ACCTINFO> aggregate contains at most one <PRESACCTINFO> aggregate, consistent with section 8.5.3. Left out of the response when no <SVC>PRESSVC accounts for the specified <USERID> or <GROUPID> or current user are found. **Note:** When <DTACCTUP> indicates the client is up-to-date, server should not return surrounding <ACCTINFORS>.
</ACCTINFO>	
</ACCTINFORS>	Closing tag for billing account information response

14.3.4.3 Group Account Information Transaction Request <PRESGRPACCTINFOTRNRQ>

As a special transaction wrapper for <ACCTINFORQ>, <PRESGRPACCTINFOTRNRQ> specifies whether the client is requesting account information for a single user or a group of users.

If the client specifies <GROUPID>, the client is requesting updated account information for a group of users. The server returns an <ACCTINFORS> response with a <PRESACCTINFO> aggregate for each account whose status has changed.

Standard signup messages are the preferred method for checking the status of a single customer, however <PRESGRPACCTINFOTRNRQ> may also be used for a single customer. (Standard signup messages are described in Chapter 8, "Activation & Account Information.")

The client should specify either <USERID> or <GROUPID>; if both are absent, the server uses the <USERID> from the signon request <SONRQ>.

	Description
<PRESGRPACCTINFOTRNRQ>	Opening tag for the transaction request
<TRNUID>	Client-assigned globally unique ID for this transaction, *trnuid*
<CLTCOOKIE>	Data to be echoed in the transaction response, *A-32*
<TAN>	Transaction authorization number,
Specify either <USERID> or <GROUPID>	

14.4 Bill Delivery

The Bill Delivery message set contains messages to obtain bills. The message set <PRESDLVMSGSETV1> contains the following aggregates: <PRESDLVMSGSRQV1> and <PRESDLVMSGSRSV1>.

14.4.1 Bill Delivery Process

Typically, the client periodically requests a list of bills from the bill publisher. The bill publisher responds with a list of bills, each of which contains summary data such as the due date and amount due. For each bill, the bill publisher might also return a URL to a Web site that contains an HTML-rendered version of the bill. Depending on the client's request, the server might also return structured bill detail for a given bill.

The aggregate for a bill list request is <PRESLISTRQ>. This request must be wrapped inside <PRESLISTTRNRQ>. There is no synchronization wrapper for bill list requests, since clients that require a list of bills can send another <PRESLISTRQ>.

The transaction wrapper <PRESLISTTRNRQ> contains optional elements that allow bills for one or more customers to be accessed by customer service representatives or client proxy systems. It is up to the server to decide who can access bills other than their own; it is recommended that all such access be logged in an audit trail.

14.4.2 Bill List Retrieval

<PRESLISTRQ> retrieves bills from the bill publisher. The bill publisher returns a <PRESLISTRS> response that contains a list of one or more bills.

14.4.2.1 Bill List Request <PRESLISTRQ>

The client requests bills from a bill publisher by date range. To specify the date range, clients use <DTSTART> and <DTEND>, as described in section 3.2.7. The date range includes all bills that were added or modified within the date range.

The bill publisher returns information sufficient to identify the biller and provide the amount due, due date, and remittance information so that a payment can be made to the biller. The bill publisher does not provide a viewable form of the bill, but returns a URL to an HTML rendering of the bill. Billing detail, such as individual purchases or transactions, can be included in the original response or obtained from a subsequent <PRESDETAILRQ>.

The <PRESLISTRQ> must be wrapped in the <PRESLISTTRNRQ> transaction wrapper.

Tag	*Description*
<PRESLISTRQ>	Opening tag for bill list request
<BILLPUB>	Official standard name of bill publisher, *A-32*
<DTSTART>	If present, indicates earliest date for which to include bills, *datetime*
<DTEND>	If present, indicates latest date for which to include bills, *datetime*
<DTDUEBY>	If present, indicates that the customer is requesting bills due on or before the date/time specified in <DTDUEBY>. *datetime*
<BILLERID>	Biller Identifier, If present, restricts the response to the given Biller, *A-32*
<BILLID>	If present, restrict response to given statement identifier, *A-32*
<BILLTYPE>	0 or more. If present, indicates which types of bills the customer is requesting. Possible values are: BILL = Invoice of an amount due to the biller that is payable. STATEMENT = History of activity on an account with the biller that is not payable. NOTICE = Generic letter from either the biller or the bill publisher that is not payable.
<BILLSTATUSCODE>	0 or more. If present indicates which bill statuses the customer is requesting. Possible values are: NEW = The server has not sent the bill to either the client or client proxy. This is the initial status code of a bill. DELIVERED = The server has sent the bill to either a client or client proxy. VIEWED = The customer has seen the bill. Implies previous status of DELIVERED. RETIRED = The customer no longer wishes to see this bill. Implies previous status of DELIVERED. WITHDRAWN = The biller or publisher no longer wishes this bill to be displayed. UNDELIVERABLE = Attempts to deliver this bill to the consumer in a timely fashion have failed. This status is not generally used when presenting a bill to a consumer. However, notifications using this status cover many useful cases.

Tag	Description
<BILLPMTSTATUSCODE>	0 or more. Bill payment status code. If present, indicates the customer is requesting bills matching the bill payment status code. Possible values are: NONE = There is neither a payment scheduled, nor has one been made against this bill. This may be the initial payment status of a bill. SCHEDULED = A payment has been scheduled, but not yet processed against this bill. PROCESSED = The payment has been processed against this bill, and can no longer be cancelled. POSTED = The biller has posted the payment against this bill. PAIDOUTOFBAND = A Payment has been initiated for this bill via a mechanism that does not report status via OFX. This status is intended to indicate the customer has paid the biller directly with cash or a check or has initiated an electronic payment through a mechanism that does not report payment status through OFX. AUTOPAY = The Biller or Service Provider will initiate the payment based on a pre-authorization by the customer, typically a “good until cancelled” instruction with no defined end date. In the US this is often implemented using a recurring pre-authorized ACH debit, though some Billers offer pre-authorized automatic payment through credit card. Examples include monthly deductions to cover a mortgage, regular payments from a checking account to a credit card, and the Automatic Payment Service (APS) offered by many utilities. Like NONE, this may be the initial payment status of the bill. CANCELLED = The customer cancelled the payment that was previously scheduled. UNPAYABLE = None of the Payment Instruments allowed for this bill are supported by the Payment Provider. This is intended to be used where the bill restricts payment to a subset of the Payment Instruments allowed in the Biller Directory entry. This could occur if the Payment Provider or the Biller changed their supported payment instrument types after enrollment and account activation.
<NOTIFYWILLING>	Flag indicating that client is prepared to send notifications of bill delivery, if desired (see section 14.4.5), *Boolean*
<INCLUDEDETAIL>	Flag indicating bill detail should be included too, *Boolean*
<INCLUDEBILLSTATUS>	Flag indicating bill status should be included too, *Boolean* Default is N.
<INCLUDEBILLPMTSTATUS>	Flag indicating bill payment status should be included, *Boolean* Default is N.
<INCLUDESTATUSHIST>	Flag indicating bill status history and/or bill payment status history should be included too. Only valid if <INCLUDEBILLSTATUS>Y and/or <INCLUDEBILLPMTSTATUS>Y are specified. *Boolean* Default is N.

Tag	Description
<INCLUDECOUNTS>	If Y, indicates that the response should include <PRESCOUNTS> and not <PRESBILLINFO>, *Boolean* May not be Y if <INCLUDESUMMARY>Y, <INCLUDEDETAIL>Y, <INCLUDEBILLSTATUS>Y, <INCLUDEBILLPMTSTATUS>Y, or <INCLUDESTATUSHIST>Y are specified. Default is N.
<INCLUDESUMMARY>	Include bill summaries (<PRESBILLINFO>), *Boolean.* May not be N if <INCLUDEDETAIL>Y, <INCLUDEBILLSTATUS>Y, <INCLUDEBILLPMTSTATUS>Y, <INCLUDESTATUSHIST>Y, or <INCLUDECOUNTS>N are specified. Unlike other boolean elements of this request, the default is Y.
</PRESLISTRQ>	Closing tag for bill list request

14.4.2.2 Bill List Response <PRESLISTRS>

The <PRESLISTRS> response must appear within a <PRESLISTTRNRS> transaction wrapper.

The <PRESLISTRS> response can contain zero or more bill summaries, with optional detail. Each bill summary corresponds to a (usually monthly) bill. When a server has no bills to return, it should return <STATUS><CODE>0 and leave out the <PRESLIST> within the <PRESLISTRS>.

When multiple selection criteria are used they are ANDed (as described in section 2.4.4.4). When counts are requested in the <PRESLISTRQ> (<INCLUDECOUNTS>), the server should include counts of each status requested where the bill's <BILLSTATUSCODE> matches one of those specified in the <PRESLISTRQ>, and the bill's <BILLPMTSTATUSCODE> matches one of those specified in the <PRESLISTRQ>. Thus, a request containing <BILLSTATUSCODE>NEW, <BILLSTATUSCODE>DELIVERED, <BILLSTATUSCODE>WITHDRAWN>, <BILLPMTSTATUSCODE>NONE, and <BILLPMTSTATUSCODE>CANCELLED>, will return three <BILLSTATUSCODE> counts and two <BILLPMTSTATUSCODE> counts. The sum of <BILLSTATUSCODE> counts will be equal to the sum of the <BILLPMTSTATUSCODE> counts. No inference can be drawn as to which bills have a combination of a specific <BILLSTATUSCODE> value and a specific <BILLPMTSTATUSCODE> value.

Tag	*Description*
<PRESLISTRS>	Opening tag for bill list response
<BILLPUB>	Official standard name of bill publisher, *A-32*
<USERID>	User whose bill data is being returned. Must match <USERID> provided in <PRESLISTTRNRQ> (if specified), "anonymous00000000000000000000000" (if <GROUPID> was specified in the <PRESLISTTRNRQ>), or the <USERID> for the authenticated user (otherwise), *A-32*
<DTSTART>	Start date of bills returned, *datetime*
<DTEND>	Date to present as start date for next request, *datetime*
<PRESLIST>	Bill summary list, see section 14.4.2.2.1
</PRESLIST>	
<PRESCOUNTS>	Bill Counts Aggregate
<BILLSTATUSCOUNTS>	Bill Status Counts, zero or more.
	The count(s) of all bills matching the given selection criteria, having a particular status(es). If <BILLSTATUSCODE> is not included in the request with <INCLUDECOUNTS>Y, counts are returned for every status with a non-zero count.
<BILLSTATUSCODE>	Bill Status Code, see section 14.4.2.2.3
<COUNT>	Count of Bills with the given Bill Status Code, *Integer*

Tag	Description
</BILLSTATUSCOUNTS>	
<BILLPMTSTATUSCOUNTS>	Bill Payment Status Counts, zero or more. The count(s) of all bills matching the given selection criteria, having a particular payment status(es). If <BILLPMTSTATUSCODE> is not included in the request with <INCLUDECOUNTS>Y, counts are returned for every payment status with a non-zero count.
<BILLPMTSTATUSCODE>	Bill Payment Status Code, see section 14.4.2.2.4
<COUNT>	Count of Bills with the given Bill Payment Status Code, *Integer*
</BILLPMTSTATUSSCOUNTS>	
</PRESCOUNTS>	
</PRESLISTRS>	Closing tag for bill list response

14.4.2.2.1 Bill List <PRESLIST>

The bill list aggregate <PRESLIST> contains a list of zero or more <PRESBILLINFO> aggregates.

	Description
<PRESLIST>	Opening tag for bill list
<PRESBILLINFO>	Bill information aggregate (zero or more that meet the selection criteria) While supported by the syntax of OFX, an empty <PRESLIST> aggregate should not be transmitted.
</PRESBILLINFO>	
</PRESLIST>	Closing tag for bill list

14.4.2.2.2 Bill Information <PRESBILLINFO>

The bill information aggregate <PRESBILLINFO> provides information about a single bill, including the amount due, date due, and pointers to more information.

If the client requested bill detail in the <PRESLISTRQ>, the bill publisher provides the detail in zero or more <BILLDETAILTABLE> aggregates. If the client did not request bill detail, the server should use the <DETAILAVAILABLE> flag to indicate whether the client can request bill detail at a later time using the <PRESDETAILRQ> aggregate.

The bill identifier <BILLID> must uniquely identify the bill with the bill publisher (not merely with the biller). The <BILLPUB> and <BILLID> combination must be globally unique, not the <FI> and <BILLID> combination.

The bill date <DTBILL> is usually a fixed number of days after the end of the bill period. It is not the date on which the bill publisher received the bill for publication.

Tag	Description
<PRESBILLINFO>	Opening tag for bill information
<BILLID>	Identifier for this bill within the bill publisher, *A-32*
<PRESACCTFROM>	Biller account information (see section 14.3.2.2)
</PRESACCTFROM>	
<PAYEEID>	Payee identifier. Specify only if the bill publisher is also provides Bill Payment service. See section 14.5.2. *SRVRTID*
<BILLREFINFO>	Biller-defined text to include with the payment, for the biller's Accounts Receivable reconciliation. Sections 14.5, 14.5.2. *A-80*
<AMTDUE>	Full payment amount due, *amount*
<MINAMTDUE>	Minimum payment amount due, *amount*
<DTPMTDUE>	Payment due date, *datetime*
<DTBILL>	Bill date, *datetime*
<DTOPEN>	Opening statement date, *datetime*
<DTCLOSE>	Closing statement date, *datetime*
<PREVBAL>	Balance of the account as of the previous period, *amount*
<ACTIVITY>	Net inflows and outflows for the account since the last period, *amount*
<ACCTBAL>	Balance of the account at the end of the current period, *amount*
<INVOICE>	Optional invoice data that the biller would like to receive with a payment (See 12.5.2.3). Client applications should allow the user to edit the amounts before returning this in a payment
</INVOICE>	
<NOTIFYDESIRED>	Indicator that a delivery notification (see section 14.4.5) is desired, *Boolean*
<BILLTYPE>	Bill Type. Possible values are: BILL = Invoice of an amount due to the biller that is payable. STATEMENT = History of activity on an account with the biller that is not payable. NOTICE = Generic letter from either biller or the bill publisher that is not payable.
<BILLSTATUS>	Zero or more bill status aggregates. See section 14.4.2.2.3.
</BILLSTATUS>	
<BILLPMTSTATUS>	Zero or more bill payment status aggregates. See section 14.4.2.2.4.

Tag	Description
</BILLPMTSTATUS> <STMNTIMAGE> </STMNTIMAGE> *Choose DETAILAVAILABLE or BILLDETAILTABLE, but not both*	Statement image aggregate, see section 14.4.2.2.5
<DETAILAVAILABLE> -or- **<BILLDETAILTABLE>** **</BILLDETAILTABLE>**	Indicator that structured detail is available, *Boolean* Bill details, when requested, see section 14.4.3.2.1
</PRESBILLINFO>	Closing tag for bill information

If <PREVBAL>, <ACTIVITY>, and <ACCTBAL> are all present, then <PREVBAL> and <ACTIVITY> must add up to <ACCTBAL>.

Note: This means payments from the consumer received by the biller are counted as negative activity.

14.4.2.2.3 Bill Status <BILLSTATUS>

	Description
<BILLSTATUS>	
<BILLSTATUSCODE>	Bill status code. Possible values are: NEW = The server has not sent the bill to either the client or client proxy. This is the initial status code of a bill. DELIVERED = The server has sent the bill to either a client or client proxy. VIEWED = The customer has seen the bill. Implies previous status of DELIVERED. RETIRED = The customer no longer wishes to see this bill. Implies previous status of DELIVERED. WITHDRAWN = The biller or publisher no longer wishes this bill to be displayed. UNDELIVERABLE = Attempts to deliver this bill to the consumer in a timely fashion have failed. This status is not generally used when presenting a bill to a consumer. However, notifications using this status cover many useful cases
<DTEFF>	Date/Time at which the status became effective (for example, the date and time a bill is created in the initial status description for a bill). *datetime*
<STATUSMODBY>	Status modified by. Servers are not required to store this information. Possible values are: CUSTOMER = customer. CUSTAGENT = An automated software agent acting on behalf of customer. BILLPUBLISHER = Bill Publisher. BILLPUBLISHERSR = Service representative acting on behalf of the payment provider. PMTPROVIDER = Payment Provider. PMTPROVIDERSR = Service representative acting on behalf of the payment provider. BILLER = biller. BILLERSR = Service representative acting on behalf of the biller.
</BILLSTATUS>	

14.4.2.2.4 Bill Payment Status <BILLPMTSTATUS>

Tag	*Description*
<BILLPMTSTATUS>	Zero or more bill payment status aggregates
<SRVRTID>	The server transaction ID of the payment against this bill (see section 3.2.2), *A-36*
<BILLPMTSTATUSCODE>	Bill payment status code. Possible values are: NONE = There is neither a payment scheduled, nor has one been made against this bill. SCHEDULED = A payment has been scheduled, but not yet processed against this bill. PROCESSED = The payment has been processed against this bill, and can no longer be cancelled. POSTED = The biller has posted the payment against this bill. PAIDOUTOFBAND = A Payment has been initiated for this bill via a mechanism that does not report status via OFX. This status is intended to indicate the customer has paid the biller directly with cash or a check or has initiated an electronic payment through a mechanism that does not report payment status through OFX. AUTOPAY = The Biller or Service Provider will initiate the payment based on a pre-authorization by the customer, typically a “good until cancelled” instruction with no defined end date. In the US this is often implemented using a recurring pre-authorized ACH debit, though some Billers offer pre-authorized automatic payment through credit card. Examples include monthly deductions to cover a mortgage, regular payments from a checking account to a credit card, and the Automatic Payment Service (APS) offered by many utilities. Like NONE, this may be the initial payment status of the bill. CANCELLED = The customer cancelled the payment that was previously scheduled. UNPAYABLE = None of the Payment Instruments allowed for this bill are supported by the Payment Provider. This is intended to be used where the bill restricts payment to a subset of the Payment Instruments allowed in the Biller Directory entry. This could occur if the Payment Provider or the Biller changed their supported payment instrument types after enrollment and account activation.
<DTEFF>	Date/Time at which the status became effective (for example, the date and time a bill is created in the initial status description for a bill). *datetime*

Tag	Description
<STATUSMODBY>	Status modified by. Servers are not required to store this information. Possible values are: CUSTOMER = customer. CUSTAGENT = An automated software agent acting on behalf of customer. BILLPUBLISHER = Bill Publisher. BILLPUBLISHERSR = Service representative acting on behalf of the payment provider. PMTPROVIDER = Payment Provider. PMTPROVIDERSR = Service representative acting on behalf of the payment provider. BILLER = biller. BILLERSR = Service representative acting on behalf of the biller.
</BILLPMTSTATUS>	

14.4.2.2.5 Statement Image <STMNTIMAGE>

The <STMNTIMAGE> aggregate provides one or more URLs that point to a fully rendered image of the bill, in HTML.

<IMAGEURL> accesses the complete bill image. This URL may contain navigation to other sites, or to other pages of bill images at the same site.

To support off-line viewing of the bill, the server may provide one or more additional URLs. Each <PREFETCHURL> points to a local Web page.

Each URL associated with <IMAGEURL> and <PREFETCHURL> must include an authentication token at the end (for example, ?authtoken=*randomString*). These embedded tokens guarantee that only the customer can access the Web page. Accessing the statement image requires SSL. The bill publisher might include an expiration date for the authentication token, and hence for the URLs. The expiration date could be quite short (for example, 1 hour) or quite long (for example, 1 month). After the expiration date, the client can obtain a new authentication token only by sending a new <PRESLISTRQ> request.

	Description
<STMNTIMAGE>	Opening tag for statement image
<IMAGEURL>	URL address for retrieving an image of the complete bill encoded as HTML. This can be cached by the client for later display, or it can be viewed live directly from the Web. *URL*
<PREFETCHURL>	Advice in support of off-line viewing. Zero or more. *URL*
<DTEXPIRE>	Date after which embedded authentication token expires, *datetime*
</STMNTIMAGE>	Closing tag for statement image

14.4.2.3 Bill List Transaction Request <PRESLISTTRNRQ>

As the transaction wrapper for <PRESLISTRQ>, this aggregate specifies whether the client is requesting bills for a single user or a group of users. The optional <USERID> and <GROUPID> elements support the following scenarios:

- **A customer requests his or her own bills from the bill publisher:** In this case, the client can optionally specify the customer's <USERID> in the <PRESLISTTRNRQ>. If the client does not specify <USERID>, the bill publisher uses the <USERID> in the signon request <SONRQ>.
- **A customer service representative requests a bill on behalf of a user:** The client sends the representative's <USERID> in the signon request <SONRQ>. To specify the user for which the representative is retrieving bills, the client sends the customer's <USERID> in the <PRESLISTTRNRQ>. The bill publisher must ultimately decide whether the customer service representative can access the requested bills. In its response, the bill publisher includes only those bills for which the requester has access privileges.
- **A client proxy system fetches bills on behalf of a group of users:** Instead of sending a <USERID>, the client sends the <GROUPID> that identifies the group of users. Within the <PRESLISTTRNRS>, the bill publisher returns a single <PRESLISTRS> containing zero or more <PRESBILLINFO> aggregates for each user in the named group. Individual customers are distinguished using the <USERID> element of each <PRESACCTFROM> in the <PRESBILLINFO> aggregates. Again, the bill publisher decides whether access should be granted. Bill publishers that support usage of <GROUPID> must maintain knowledge of which users are in which named group. The OFX specification does not provide a way to track membership in a named group. Any such management must happen out-of-band.

  Note: <USERID> is not intended to identify individual users of joint accounts. If a transaction might include two different USERIDs within otherwise identical <PRESACCTFROM> aggregates, servers should deliver two separate bills (download the same bill twice). It is up to the server to keep track of activity on joint accounts.

	Description
<PRESLISTTRNRQ>	Opening tag for bill list transaction request
<TRNUID>	Client-assigned globally unique ID for this transaction, *trnuid*
<CLTCOOKIE>	Data to be echoed in the transaction response, *A-32*
<TAN>	Transaction authorization number, *A-80*
Specify either <USERID> or <GROUPID>	
<USERID>	If present, the bill request is on behalf of this particular user, *A-32*
- or -	
<GROUPID>	If present, the bill request is on behalf of all users in the named group. If both <USERID> and <GROUPID> are absent, the <USERID> in the <SONRQ> is implied. *A-32*
<PRESLISTRQ>	Bill List Request Aggregate
</PRESLISTRQ>	
</PRESLISTTRNRQ>	Closing tag for bill list transaction request

14.4.2.4 Bill List Transaction Response <PRESLISTTRNRS>

Tag	Description
<PRESLISTTRNRS>	Opening tag for bill list transaction response
<TRNUID>	Client-assigned globally unique ID for this transaction *trnuid*
<STATUS>	Status aggregate
</STATUS>	
<CLTCOOKIE>	Client provided data. **REQUIRED** if provided in request *A-32*
<PRESLISTRS>	Bill list response aggregate, optional
</PRESLISTRS>	
</PRESLISTTRNRS>	Closing tag for bill list transaction response

14.4.2.5 Status Codes <PRESLISTRS>

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2003	Account not found (ERROR)
2004	Account closed (ERROR)
2005	Account not authorized (ERROR)
2020	Invalid date (ERROR)
2027	Invalid date range (ERROR)
15508	Transaction not authorized (ERROR)

14.4.3 Bill Detail Retrieval

If statement detail is available for a bill, the client can retrieve the detail using a bill detail request <PRESDETAILRQ>. One example of statement detail is the individual telephone calls from a telephone bill.

14.4.3.1 Bill Detail Request <PRESDETAILRQ>

The <PRESDETAILRQ> request must appear within a <PRESDETAILTRNRQ> transaction wrapper.

Tag	*Description*
<PRESDETAILRQ>	Opening tag for bill detail request
<BILLID>	Statement identifier from <PRESBILLINFO>, *A-32*
<BILLDETAILTABLETYPE>	If present, filters response to just tables of this type (See table 14.4.3.2.3), A-32.
</PRESDETAILRQ>	Closing tag for bill detail request

14.4.3.2 Bill Detail Response <PRESDETAILRS>

The <PRESDETAILRS> request must appear within a <PRESDETAILTRNRS> transaction wrapper.

The bill detail response contains zero or more <BILLDETAILTABLE> aggregates.

	Description
<PRESDETAILRS>	Opening tag for bill detail response
<PRESDETAIL>	Zero or more bill detail aggregates
<BILLID>	Statement identifier from <PRESBILLINFO>, *A-32*
<PRESACCTFROM>	Identifies biller account, see section 14.3.2.2. Must be included if in response to an <PRESDETAILRQ>, is redundant inside <PRESBILLINFO>
</PRESACCTFROM>	
<BILLDETAILTABLE>	Zero or more bill detail table aggregates. See section 14.4.3.2.1.
</BILLDETAILTABLE>	
</PRESDETAIL>	Closing tag for bill detail aggregate
</PRESDETAILRS>	Closing tag for bill detail response

14.4.3.2.1 Bill Detail Table <BILLDETAILTABLE>

The bill detail table allows billers to send tabular data to the customer in a flexible way. The table might contain phone calls from a telephone bill, or electrical meter readings for a utility bill.

A table consists of one or more rows, each having one or more columns. Within a table, all rows must have identical structures. The <BILLDETAILTABLETYPE> determines the "shape" or schema of the table. The <TABLENAME> gives a name to this table, and should be unique within an <PRESDETAILRS>

Note: The bill detail table may be redesigned in the future. Please consider this area "under construction.".

Tag	*Description*
<BILLDETAILTABLE>	Opening tag for bill detail table
<TABLENAME>	Name of bill detail table, *A-32*
<BILLDETAILTABLETYPE>	Type of bill detail table (See section 14.4.3.2.3), *A-32*.
<BILLDETAILROW>	Zero or more bill detail row aggregates, see section 14.4.3.2.2
</BILLDETAILROW>	
</BILLDETAILTABLE>	Closing tag for bill detail table

14.4.3.2.2 Bill Detail Row <BILLDETAILROW>

A <BILLDETAILTABLE> contains zero or more bill detail rows <BILLDETAILROW>.

A <BILLDETAILROW> contains zero or more columns <C>, whose meanings are specific to the type of table <BILLDETAILTABLETYPE> in which they occur. For the purpose of the DTD parser, all columns <C> are consider to be **Format: A-255**.

OFX requires all elements return data. If bill publishers do not use specific columns, they can return null columns, represented by the element <N>. All columns <N> are considered to be **Format: A-1.**

> **Note:** Bill publishers must include one character of data in a null column. Bill publishers can omit blank columns at the end of a <BILLDETAILROW>, tag and all. DTD should not enforce ordering, i.e. it should look like this:

```
<!ELEMENT BILLDETAILROW          - - (C | N)*>
```

	Description
<BILLDETAILROW>	Opening tag for bill detail row
<C>	Zero or more column data elements, *A-255*
<N>	Zero or more column data elements, *A-1*
</BILLDETAILROW>	Closing tag for bill detail row

14.4.3.2.3 Table Types <BILLDETAILTABLETYPE>

OFX defines some common table types. Individual billers can define their own table types, and hence their own table structures, but must honor the custom tag naming convention outlined in section 2.7.

Value	*Description*
TransactionList	Table defined for “payment register”-style line items
CallLog	Table defined for record of telephone calls
ABC.Usage	Table defined by biller, not by OFX

14.4.3.2.3.1 TransactionList Table Type

<BILLDETAILTABLE> aggregates marked with <BILLDETAILTABLETYPE>TransactionLists have rows of 14 columns. The first column contains a unique identifier (like a BILLID), and must be present. Other columns may not always apply and can be left blank.

The TransactionList table type is a subset of the <STMTTRN> aggregate in section 11.4.3.

	Name	*Description*
1	BillId	Unique identifier token from server, *A-32*
2	TrnType	Transaction type (see section 11.4.3.1)
3	DtPosted	Date item was posted, *datetime*
4	DtUser	Date user initiated transaction, if known, *datetime*
5	TrnAmount	Amount of transaction, *amount*
6	CorrectBillId	If present, unique identifier of previously sent transaction that is corrected by this record, *A-32*
7	CorrectAction	Replace or delete. Specify only if column 6 is present.
8	CheckNum	Check or other reference number, *A-12*
9	RefNum	Other reference number, *A-12*
10	SIC	Standard Industrial Code, *N-6*
11	Name	Name of payee or description of transaction, *A-32*
12	Memo	Extra Information, *memo*
13	OrigCurSym	Original Currency Identifier (ISO 42173 3-letter), *A-3*
14	CurRate	Currency rate, ratio of currency to original currency, *rate*

14.4.3.2.3.2 CallLog Table Type

<BILLDETAILTABLE> aggregates marked with <BILLDETAILTABLETYPE>CallLog have rows of 10 columns.

	Name	Description
1	TNCalledFrom	Telephone number called from, *A-32*
2	CityStateFrom	City, state (or place, region) called from, *A-16*
3	TNCalled	Telephone number called, *A-32*
4	CityState	City, state (or place, region) called, *A-16*
5	Originated	Date/time call started, *datetime*
6	Type	Type of call, *A-8*
7	Rate	Rate (for example, Night, Day, Eve, Wknd), *A-5*
8	Duration	Duration of call in tenths of seconds, *N-6*
9	Cost	Cost of call, *amount*
10	TNChargedTo	Telephone number charged to, *A-32*

14.4.3.3 Status Codes <PRESDETAILRS>

Code	Meaning
0	Success (INFO)
2000	General error (ERROR)
2023	Unknown BILLID (ERROR)
10600	Table type not found (ERROR)

14.4.4 Table Structure Definition

Clients can obtain the definition of a table structure by sending a table structure request <BILLTBLSTRUCTRQ>.

Clients need only request the structure of tables it does not already know about. For instance, the client might request the structure of a biller-specific table that starts with the *x-* prefix. Knowing the structure of a table allows the client to display the data more clearly or store the data in a more compact form, such as a database table.

14.4.4.1 Table Structure Request <BILLTBLSTRUCTRQ>

To identify the table, the client includes the type of table <BILLDETAILTABLETYPE> and unique identifier <BILLID> for the table. Although <BILLDETAILTABLETYPE> uniquely identifies the table, OFX requires the <BILLID> as well to allow various server implementations.

The <BILLTBLSTRUCTRQ> request must appear within a <BILLTBLSTRUCTTRNRQ> transaction wrapper.

	Description
<BILLTBLSTRUCTRQ>	Opening tag for the table structure request
<BILLID>	Statement Identifier, *A-32*
<BILLDETAILTABLETYPE>	Table type for which the structure is requested (See table 14.4.3.2.3), A-32.
</BILLTBLSTRUCTRQ>	Closing tag for the table structure request

14.4.4.2 Table Structure Response <BILLTBLSTRUCTRS>

The <BILLTBLSTRUCTRS> response must appear within a <BILLTBLSTRUCTTRNRS> transaction wrapper.

The table structure response contains one or more column type definitions, which correspond positionally with the <C> aggregates in a <BILLDETAILROW> in a <BILLDETAILTABLE> of the corresponding <TABLETYPE>.

Tag	*Description*
<BILLTBLSTRUCTRS>	Opening tag for table structure response
<BILLID>	Table identifier, *A-32*
<BILLDETAILTABLETYPE>	Table type (See table 14.4.3.2.3), A-32.
<COLDEF>	Zero or more column definition aggregates (see section 14.4.4.2.1)
</COLDEF>	
</BILLTBLSTRUCTRS>	Closing tag for table structure response

14.4.4.2.1 Column Definition <COLDEF>

A column definition <COLDEF> associates a name and a data type with a column.

	Description
<COLDEF>	Opening tag for column definition
<COLNAME>	Column name, *A-32*
<COLTYPE>	Column type, valid values are (choose one): (A-255, D, N-6), *A-8.* Specifying D in this field means the column type is datetime.
</COLDEF>	Closing tag for column definition

14.4.4.3 Status Codes <BILLTBLSTRUCTRS>

Code	*Meaning*
0	Success (INFO)
2000	General error (ERROR)
2023	Unknown BILLID (ERROR)
10600	Table type not found (ERROR)

14.4.5 Delivery Notification

In OFX 1.6, a new Bill Status Modify transaction was added. This new transaction (see section 14.4.6) is a semantic superset of Delivery Notification. Bill Status Modify should be used instead of Delivery Notification.

The bill publisher can request delivery notification through the <NOTIFYDESIRED> flag in the <PRESBILLINFO> aggregate (see section 14.4.2.2.2). The bill publisher will expect to receive the delivery notification only if the <PRESLISTRQ> had the <NOTIFYWILLING> flag set.

The delivery notification request tells the bill publisher that the client has presented the specified bills to the customer. This is a stronger statement than acknowledging that the bills have been received by the client, specifically when the client software implements the pre-fetching or (push) model. Delivery notification should be sent only once for any given bill, and it should be sent the first time that the Bill Summary is displayed. Receipt of a delivery notification by the bill publisher has no legal significance. OFX does not define the maximum elapsed time between the presentation of the bill and the delivery notification.

14.4.5.2 Delivery Notification Response <PRESNOTIFYRS>

In OFX 1.6, a new Bill Status Modify request, <BILLSTATUSMODRS>, was added. This new request (see section 14.4.6.2) should be used instead of <PRESNOTIFYRS>.

The <PRESNOTIFYRS> response must appear within a <PRESNOTIFYTRNRS> transaction wrapper.

The delivery notification response lets the client know that the delivery notification request was received by the bill publisher.

	Description
<PRESNOTIFYRS>	Opening tag for Delivery Notification Response
<PRESDELIVERYID>	A bill delivery ID aggregate (See section 14.4.5.1.1)
</PRESDELIVERYID>	
</PRESNOTIFYRS>	Closing tag for Delivery Notification Response

14.4.5.3 Status Codes <PRESNOTIFYRS>

Code	*Meaning*
0	Success (INFO)
2000	General error (ERROR)
2023	BILLID not found (ERROR)
15508	Transaction not authorized (ERROR)

14.4.6 Bill Status Modification

14.4.6.1 Request <BILLSTATUSMODRQ>

The Bill Status Modify Request <BILLSTATUSMODRQ> must appear within a <BILLSTATUSMODTRNRQ> transaction wrapper.

14.4.6.3 Status Codes <BILLSTATUSMODRS>

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2023	BILLID not found (ERROR)
15508	Transaction not authorized (ERROR)

14.5 Bill Payment

To pay a bill received through a <PRESLISTRQ> request, the client can use the Bill Payment message set defined in Chapter 12, "Payments." To construct the payment information <PMTINFO> (see section 12.5.2), the client can use the bill information from <PRESBILLINFO>.

14.5.1 Remittance Information

The client should include the <BILLREFINFO> from the <PRESBILLINFO> aggregate as the <BILLREFINFO> in the <PMTINFO> aggregate. This token allows the biller to link the payment with the bill.

14.5.2 Payee Identification

Client software can produce <PAYEEID> or <PAYEE> in one of two ways.

If the same company provides Bill Presentment and Bill Payment services, the client can use the <PAYEEID> included in the <PRESBILLINFO> aggregate.

If the Bill Payment provider is a different company, the client must use information from the <PRESACCTINFO> to construct the <PAYEE> information.

14.6 Bill Presentment E-Mail

OFX currently defines a Bill Presentment e-mail message that clients can send to bill publishers. With this message, a customer can send a message to a bill publisher regarding one of his or her accounts.

The server acknowledges receipt of the message. The bill publisher then prepares a response that the client picks up when it synchronizes with the server. E-mail is subject to synchronization, using <PRESMAILSYNCRQ> (defined in section 14.6.4) and <PRESMAILSYNCRS> (defined in section 14.6.5.)

	Server Responds
Addressed message PRES account information	
	Acknowledgment
. . .	
Synchronization request	
	Response to customer

14.6.1 Bill Presentment Mail Request <PRESMAILRQ>

The client must identify the account to which account the customer query is related.

The <PRESMAILRQ> request must appear with a <PRESMAILTRNRQ> transaction wrapper.

	Description
<PRESMAILRQ>	PRES-e-mail-request aggregate
<PRESACCTFROM>	Account-from aggregate, see section 14.3.2.2
</PRESACCTFROM>	
<MAIL>	To, from, message information, see section 9.2.2
</MAIL>	
</PRESMAILRQ>	

14.6.2 Bill Presentment Mail Response <PRESMAILRS>

The <PRESMAILRS> request must appear with a <PRESMAILTRNRS> transaction wrapper.

	Description
<PRESMAILRS>	PRES-e-mail-response aggregate
<PRESACCTFROM>	Account-from aggregate, see section 14.3.2.2
</PRESACCTFROM>	
<MAIL>	To, from, message information, see section 9.2.2
</MAIL>	
</PRESMAILRS>	

14.6.3 Status Codes <PRESMAILRS>

	Meaning
0	Success (INFO)
2000	General error (ERROR)
2002	General account error (ERROR)
2003	Account not found (ERROR)
2004	Account closed (ERROR)
2005	Account not authorized (ERROR)
15508	Transaction not authorized (ERROR)
16500	HTML not allowed (ERROR)
16501	Unknown mail To: (ERROR)

14.6.4 Request <PRESMAILSYNCRQ>

	Description
<PRESMAILSYNCRQ> *Client synchronization option; <TOKEN>, <TOKENONLY>, or <REFRESH>*	Synchronization request aggregate
<TOKEN>	Previous value of <TOKEN> received for this type of synchronization request from server; 0 for first-time requests; *token*
<TOKENONLY>	Request for just the current <TOKEN> without the history, *Boolean*
<REFRESH>	Request for refresh of current state, *Boolean*
<REJECTIFMISSING>	If Y, do not process requests if client <TOKEN> is out of date, *Boolean*
<INCIMAGES>	Y if the client accepts mail with images in the message body. N if the client does not accept mail with images in the message body. *Boolean*
<USEHTML>	Y if client wants an HTML response, N if client wants plain text, *Boolean*
<PRESACCTFROM>	Account-from aggregate, see section 14.3.2.2
</PRESACCTFROM>	
<PRESMAILTRNRQ>	Bill presentment mail transactions (0 or more)
</PRESMAILTRNRQ>	
</PRESMAILSYNCRQ>	

14.6.5 Response <PRESMAILSYNCRS>.

	Description
<PRESMAILSYNCRS>	Synchronization response aggregate
<TOKEN>	New synchronization token, *token*
<LOSTSYNC>	Y if the token in the synchronization request is older than the earliest entry in the server's history table. In this case, some responses have been lost. N if the token in the synchronization request is newer than or matches a token in the server's history table. *Boolean*
<PRESACCTFROM>	Account-from aggregate, see section 14.3.2.2
</PRESACCTFROM>	
<PRESMAILTRNRS>	Bill presentment mail transactions (0 or more)
</PRESMAILTRNRS>	
</PRESMAILSYNCRS>	

14.7 Message Sets and Profile

OFX separates the messages that the client and server send into groups called message sets. In its profile response <PROFRS>, each bill publisher or other server provider defines the message sets that it supports and any options available for those message sets.

This section defines the message sets supported by Bill Presentment. It then describes the corresponding message set profile aggregates that can be provided in the profile response <PROFRS>. The message set profile aggregates for the <PROFRS> allow a bill publisher or other server provider to customize its use of OFX. For example, a server might support the Bill Delivery message set <PRESDLVMSGSET>, but not the Group Account Information message set <PRESDIRMSGSET>.

For general information about profiles, see Chapter 7, "FI Profile."

14.7.1 Message Sets and Messages

Bill Presentment defines the following message sets:

- Biller Directory message set <PRESDIRMSGSET>, which includes messages for finding billers and bill publishers
- Bill Delivery message set <PRESDLVMSGSET>, which includes messages for delivering bills and bill detail to customers, as well as messages for getting account information for a group of users

14.7.1.1 Biller Directory Message Set and Messages

The following tables show the OFX transactions and their corresponding transaction wrappers for a particular message set.

14.7.1.1.1 Biller Directory Request Messages

Message	
<PRESDIRMSGSET>	
<PRESDIRMSGSETV1>	
<PRESDIRMSGSRQV1>	FINDBILLERTRNRQ FINDBILLERRQ
</PRESDIRMSGSRQV1>	
</PRESDIRMSGSETV1>	
</PRESDIRMSGSET>	

14.7.1.1.2 Biller Directory Response Messages

Message	
<PRESDIRMSGSET>	
<PRESDIRMSGSETV1>	
<PRESDIRMSGSRSV1>	FINDBILLERTRNRS
	FINDBILLERRS
</PRESDIRMSGSRSV1>	
</PRESDIRMSGSETV1>	
</PRESDIRMSGSET>	

14.7.1.2 Bill Delivery Message Set and Messages

14.7.1.2.1 Bill Delivery Request Messages

	Message
<PRESDLVMSGSRQV1>	PRESLISTTRNRQ
	PRESLISTRQ
	PRESDETAILTRNRQ
	PRESDETAILRQ
	BILLTBLSTRUCTTRNRQ
	BILLTBLSTRUCTRQ
	BILLSTATUSMODTRNRQ
	BILLSTATUSMODRQ
	PRESNOTIFYTRNRQ
	PRESNOTIFYRQ
	PRESGRPACCTINFOTRNRQ
	ACCTINFORQ
	PRESMAILTRNRQ
	PRESMAILRQ
	PRESMAILSYNCRQ
</PRESDLVMSGSRQV1>	

14.7.1.2.2 Bill Delivery Response Messages

	Message
<PRESDLVMSGSRSV1>	PRESLISTTRNRS
	PRESLISTRS
	PRESDETAILTRNRS
	PRESDETAILRS
	BILLTBLSTRUCTTRNRS
	BILLTBLSTRUCTRS
	BILLSTATUSMODTRNRS
	BILLSTATUSMODRS
	PRESNOTIFYTRNRS
	PRESNOTIFYRS
	PRESGRPACCTINFOTRNRS
	ACCTINFORS
	PRESMAILTRNRS
	PRESMAILRS
	PRESMAILSYNCRS
</PRESDLVMSGSRSV1>	

14.7.2 Biller Directory Message Set Profile

This section defines the profile aggregate for the Biller Directory message set. This profile aggregate should be included in the <PROFRS> response for those servers that support the Biller Directory message set.

	Message
<PRESDIRMSGSET>	Opening tag for the Biller Directory message set profile
<PRESDIRMSGSETV1>	Version 1 of Biller Directory message set, one or more
<MSGSETCORE>	Common message-set core
</MSGSETCORE>	
<PRESDIRPROF>	Directory profile (if supported)
<PROCDAYSOFF>	Days of week that no processing occurs: MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, or SUNDAY. 0 or more <PROCDAYSOFF> can be sent.
<CANSUPPORTIMAGES>	Supports delivery of images as multipart MIME, *Boolean*
<PROCENDTM>	Time of day that day's processing ends, *time*
</PRESDIRPROF>	
</PRESDIRMSGSETV1>	
</PRESDIRMSGSET>	Closing tag for the Biller Directory message set profile

14.7.3 Bill Delivery Message Set Profile

This section defines the profile aggregate for the Bill Delivery message set. This profile aggregate should be included in the <PROFRS> response for those servers that support the Bill Delivery message set.

Tag	*Description*
<PRESDLVMSGSET>	Opening tag for the Bill Delivery message set profile
<PRESDLVMSGSETV1>	Version 1 of Bill Delivery message set, one or more
<MSGSETCORE>	Common message-set core
</MSGSETCORE>	
<PRESDLVPROF>	Bill Delivery profile (if supported)
<CANSUPPORTGROUPID>	Supports account information requests for a group of users, that is <PRESGRPACCTINFOTRNRQ> and <GROUPID> in

Tag	Description
<PROCDAYSOFF>	Days of week that no processing occurs: MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, or SUNDAY. 0 or more <PROCDAYSOFF> can be sent.
<CANSUPPORTIMAGES>	Supports delivery of images as multipart MIME, *Boolean*
<PROCENDTM>	Time of day that day's processing ends, *time*
<CANUPDATEPRESNAMEADDRESS>	Supports update of the PRESNAMEADDRESS associated with a particular bill, *Boolean* See section 14.3.3.1
<CANMODSTATUS>	Y if server supports the <BILLSTATUSMODRQ>. This must be explicitly supported by the server before it is used by the client. The default for this option is N. *Boolean* **Note:** Servers that support <BILLSTATUSMODRQ> are required to continue support for <PRESNOTIFYRQ>.
</PRESDLVPROF>	
<EMAILPROF>	E-mail profile
<CANEMAIL>	Supports generalized e-mail, *Boolean*
<CANNOTIFY>	Supports notification (of any kind), *Boolean*
</EMAILPROF>	
</PRESDLVMSGSETV1>	
</PRESDLVMSGSET>	Closing tag for the Bill Delivery message set profile

14.8 Bill Presentment Examples

14.8.1 Find Biller Examples

14.8.1.1 Get All Billers

The client sends a <FINDBILLERRQ> request, as shown in the following example, to retrieve all available billers.

> **Note:** These examples show the customer signing on anonymously. But, they may have enrolled in the past and choose to use their actual <USERID> and <USERPASS>. Anonymous signon is not required for use of the Biller Directory service (see section 14.2.1).

```
<OFX> <!-- Begin request data -->
  <SIGNONMSGSRQV1>
    <SONRQ>                    <!-- Begin anonymous signon -->
      <DTCLIENT>20050707202000</DTCLIENT><!-- Jul. 7, 2005, 8:20:00 PM
-->
      <USERID>anonymous00000000000000000000000</USERID>
      <USERPASS>anonymous00000000000000000000000</USERPASS>
      <LANGUAGE>ENG</LANGUAGE> <!-- Language used for text -->
      <FI>                     <!-- ID of receiving institution -->
        <ORG>NCH</ORG>         <!-- Name of ID owner -->
        <FID>1001</FID>        <!-- Actual ID -->
      </FI>
      <APPID>MyApp</APPID>
      <APPVER>0500</APPVER>
    </SONRQ>                   <!-- End of signon -->
  </SIGNONMSGSRQV1>
  <PRESDIRMSGSRQV1>
    <FINDBILLERTRNRQ>
      <TRNUID>1231239</TRNUID>
      <FINDBILLERRQ>           <!--Beginning of find biller request-->
        <INCIMAGES>N</INCIMAGES><!--LOGO Images are not requested-->
      </FINDBILLERRQ>          <!--End of request-->
    </FINDBILLERTRNRQ>
  </PRESDIRMSGSRQV1>
</OFX>
```

To keep the size of the example reasonable, we will assume that there are only four billers. Here is the server reply.

```
<OFX> <!-- Begin response data -->
  <SIGNONMSGSRSV1>
    <SONRS>                    <!-- Begin signon -->
      <STATUS>                 <!-- Begin status aggregate -->
        <CODE>0</CODE>         <!-- OK -->
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <DTSERVER>20050707202001</DTSERVER><!-- Jul. 7, 2005, 8:20:01 PM
-->
      <LANGUAGE>ENG</LANGUAGE> <!-- Language used in response -->
      <DTPROFUP>19990630000000</DTPROFUP> <!-- Last update to profile
-->
      <DTACCTUP>19990701233045</DTACCTUP> <!-- Last account update -->
    </SONRS>                   <!-- End of signon -->
  </SIGNONMSGSRSV1>
  <PRESDIRMSGSRSV1>
    <FINDBILLERTRNRS>
      <TRNUID>1231239</TRNUID>
      <STATUS>                 <!-- Begin status aggregate -->
        <CODE>0</CODE> <!-- OK -->
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <FINDBILLERRS>           <!--Beginning of response-->
        <DTUPDATE>20050415092000</DTUPDATE>
                               <!--Date last update 04/15/99 9:20am-->
        <BILLERINFO>
          <BILLPUB>Wepubbills</BILLPUB><!--Name of Bill Publisher-->
          <BILLERID>123456789</BILLERID><!--Biller ID at Wepubbills-->
          <NAME>RealBig Credit Co.</NAME>
                               <!--Name of biller-->
          <ADDR1>1324 Whatever St.</ADDR1>
                               <!--Street address of biller-->
          <CITY>MajorMetro</CITY><!--City of the Biller-->
          <STATE>OH</STATE>    <!--State of the biller-->
          <POSTALCODE>12345-1234</POSTALCODE>
                               <!--Postal code of biller-->
          <COUNTRY>USA</COUNTRY>
          <SIC>23</SIC>        <!--Standard Industry Code of biller-->
          <PHONE>614-235-2323</PHONE><!--Biller's phone number-->
          <PAYMENTINSTRUMENTS> <!--Type of payment accepted-->
            <PAYMENTINSTRUMENT>
```

```
        <PMTINSTRUMENTTYPE>CHECKINGACCOUNT</PMTINSTRUMENTTYPE>
      </PAYMENTINSTRUMENT>
      <PAYMENTINSTRUMENT>
        <PMTINSTRUMENTTYPE>CONCENTRATOR</PMTINSTRUMENTTYPE>
        <BRAND>CityBank</BRAND>
      </PAYMENTINSTRUMENT>
    </PAYMENTINSTRUMENTS>
    <ACCTFORMAT>([0-9]\{3\}-)\{3\}</ACCTFORMAT>
                          <!--Regular expression describing -->
                          <!--biller's account number-->
    <ACCTEDITMASK>###-###-####</ACCTEDITMASK>
                          <!--Edit mask for account number-->
    <LOGO>http://www.realbig.com/logo.gif</LOGO>
                          <!--URL to logo of biller-->
  </BILLERINFO>
  <BILLERINFO>
    <BILLPUB>Wepubbills</BILLPUB><!--Name of Bill Publisher-->
    <BILLERID>222334465</BILLERID><!--Biller ID at Wepubbills-->
    <NAME>Aphone Company</NAME><!--Name of biller-->
    <ADDR1>1324 Where Blvd<ADDR1>
                          <!--Street address of biller-->
    <CITY>Sometown</CITY><!--City of the biller-->
    <STATE>CA</STATE>    <!--State of the biller-->
    <POSTALCODE>10992-1234</POSTALCODE>
                          <!--Postal code of biller-->
    <COUNTRY>USA</COUNTRY>
    <SIC>39</SIC>        <!--Standard Industry Code of biller-->
    <PHONE>345-345-3489</PHONE><!--Biller's phone number-->
    <PAYMENTINSTRUMENTS> <!--Type of payment accepted-->
      <PAYMENTINSTRUMENT>
        <PMTINSTRUMENTTYPE>CHECKINGACCOUNT</PMTINSTRUMENTTYPE>
      </PAYMENTINSTRUMENT>
    </PAYMENTINSTRUMENTS>
    <ACCTFORMAT>([1-9]\{2\}-)\{2\}[0-9]\{3\}</ACCTFORMAT>
                          <!--Regular expression describing-->
                          <!--biller's account number-->
    <ACCTEDITMASK>##-##-###</ACCTEDITMASK>
                          <!--Edit mask for account number-->
    <LOGO>http://www.webup.com/aphone.gif</LOGO>
                          <!--URL to logo of biller-->
  </BILLERINFO>
  <BILLERINFO>
    <BILLPUB>Wepubbills</BILLPUB><!--Name of Bill Publisher-->
```

```
          <BILLERID>98765123454</BILLERID><!--Biller ID at Wepubbills-
->
          <NAME>Goodol Mortgage</NAME><!--Name of biller-->
          <ADDR1>8273 Magnolia St.</ADDR1>
                                <!--Street address of biller-->
          <CITY>Atlanta</CITY> <!--City of the Biller-->
          <STATE>GA</STATE>    <!--State of the biller-->
          <POSTALCODE>34342-6789</POSTALCODE>
                                <!--Postal code of biller-->
          <COUNTRY>USA</COUNTRY>
          <SIC>03</SIC>        <!--Standard Industry Code of biller-->
          <PHONE>864-234-6745</PHONE><!--Biller's phone number-->
          <PAYMENTINSTRUMENTS> <!--Type of payment accepted-->
            <PAYMENTINSTRUMENT>
              <PMTINSTRUMENTTYPE>CHECKINGACCOUNT</PMTINSTRUMENTTYPE>
            </PAYMENTINSTRUMENT>
          </PAYMENTINSTRUMENTS>
          <ACCTFORMAT>[0-1]\{12\}</ACCTFORMAT>
                                <!--Regular expression describing-->
                                <!--biller's account number-->
          <HELPMESSAGE>Enter the first 13 digits of your account
number</HELPMESSAGE>
                                <!--to help user key account number-->
          <RESTRICT> GA residents only.</RESTRICT>
                                <!--Indicate restricted availability-->
          <LOGO>http://www.wepub.com/mort.gif</LOGO>
                                <!--URL to logo of biller-->
        </BILLERINFO>
        <BILLERINFO>
          <BILLPUB>Wepubbills</BILLPUB><!--Name of Bill Publisher-->
          <BILLERID>32812816734</BILLERID><!--Biller ID at Wepubbills-
->
          <NAME>Sam's Widgets</NAME><!--Name of biller-->
          <ADDR1>Apt B3</ADDR1><!--Street address of biller-->
          <ADDR2>1267 Tank Rd</ADDR2>
          <CITY>Columbus</CITY><!--City of Biller-->
          <STATE>OH</STATE>    <!--State of the biller-->
          <POSTALCODE>77723-8989</POSTALCODE>
                                <!--Postal code of biller-->
          <COUNTRY>USA</COUNTRY>
          <SIC>12</SIC>        <!--Standard Industry Code of biller-->
          <PHONE>614-657-8934</PHONE><!--Biller's phone number-->
          <PAYMENTINSTRUMENTS> <!--Type of payment accepted-->
            <PAYMENTINSTRUMENT>
```

```
  <PRESDIRMSGSRQV1>
    <FINDBILLERTRNRQ>
      <TRNUID>1231245</TRNUID>
      <FINDBILLERRQ>
        <STATE>OH</STATE>
        <INCIMAGES>N</INCIMAGES>
      </FINDBILLERRQ>
    </FINDBILLERTRNRQ>
  </PRESDIRMSGSRQV1>
</OFX>
```

In the same circumstances as before, the response would be:

```
<OFX> <!-- Begin response data -->
  <SIGNONMSGSRSV1>
    <SONRS>                    <!-- Begin signon -->
      <STATUS>                 <!-- Begin status aggregate -->
        <CODE>0</CODE>         <!-- OK -->
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <DTSERVER>20050707202004</DTSERVER> <!-- Jul. 7, 2005, 8:20:04
PM
      <LANGUAGE>ENG</LANGUAGE> <!-- Language used in response -->
      <DTPROFUP>19990630000000</DTPROFUP> <!-- Last update to profile
-->
      <DTACCTUP>19990701233045</DTACCTUP> <!-- Last account update -->
    </SONRS>                   <!-- End of signon -->
  </SIGNONMSGSRSV1>
```

```
            <ADDR1>1324 Whatever St.</ADDR1>
                                <!--Street address of biller-->
            <CITY>MajorMetro</CITY><!--City of the Biller-->
            <STATE>OH</STATE>    <!--State of the biller-->
            <POSTALCODE>12345-1234</POSTALCODE>
                                <!--Postal code of biller-->
            <COUNTRY>USA</COUNTRY>
            <SIC>23</SIC>        <!--Standard Industry Code of biller-->
            <PHONE>614-235-2323</PHONE><!--Biller's phone number-->
            <PAYMENTINSTRUMENTS> <!--Type of payment accepted-->
               <PAYMENTINSTRUMENT>
                  <PMTINSTRUMENTTYPE>CHECKINGACCOUNT</PMTINSTRUMENTTYPE>
               </PAYMENTINSTRUMENT>
               <PAYMENTINSTRUMENT>
                  <PMTINSTRUMENTTYPE>CONCENTRATOR</PMTINSTRUMENTTYPE>
                  <BRAND>CityBank</BRAND>
               </PAYMENTINSTRUMENT>
            </PAYMENTINSTRUMENTS>
            <ACCTFORMAT>([0-1]\{3\}-)\{3\}</ACCTFORMAT>
         <!--Regular expression describing-->
                                <!--biller's account number-->
            <ACCTEDITMASK>###-###-###-</ACCTEDITMASK>
                                <!--Edit mask for account number-->
            <LOGO>http://www.relbig.com/logo.gif</LOGO>
                                <!--URL to logo of biller-->
          </BILLERINFO>
        </FINDBILLERRS>
      </FINDBILLERTRNRS>
    </PRESDIRMSGSRSV1>
</OFX>
```

14.8.2 Enrollment Examples

In this example, the client wants to enroll with a bill publisher.

14.8.2.1 Enrollment Request

```
<OFX>
  <SIGNONMSGSRQV1>              <!--Signon Request-->
    <SONRQ>
      <DTCLIENT>20050307022243</DTCLIENT><!--Timestamp, 3/07/99,
2:22:43am-->
      <USERID>anonymous00000000000000000000000</USERID>
      <USERPASS>anonymous00000000000000000000000</USERPASS>
      <LANGUAGE>ENG</LANGUAGE>
      <APPID>OFXAPP</APPID>
      <APPVER>0201</APPVER>
    </SONRQ>
  </SIGNONMSGSRQV1>
  <SIGNUPMSGSRQV1>              <!--Enrollment Request-->
    <ENROLLTRNRQ>
      <TRNUID>10001</TRNUID>
      <ENROLLRQ>
        <FIRSTNAME>Cindy</FIRSTNAME>
        <MIDDLENAME>P</MIDDLENAME>
        <LASTNAME>Williams</LASTNAME>
        <ADDR1>123 Oak St</ADDR1>
        <CITY>San Jose</CITY>
        <STATE>CA</STATE>
        <POSTALCODE>94111<POSTALCODE>
        <COUNTRY>USA</COUNTRY>
        <DAYPHONE>415-555-0123</DAYPHONE>
        <EVEPHONE>408-555-2323</EVEPHONE>
        <EMAIL>cindy@aol.com</EMAIL>
        <USERID>cindyid</USERID>
        <TAXID>111-33-5555</TAXID>
        <SECURITYNAME>wynona</SECURITYNAME>
        <DATEBIRTH>19650402</DATEBIRTH>
      </ENROLLRQ>
    </ENROLLTRNRQ>
  </SIGNUPMSGSRQV1>
</OFX>
```

14.8.2.2 Enrollment Response

For this example, the server responds with immediate acceptance. In practice, many servers would send an enrollment status code of 13000 and send the user ID and password in a welcome letter.

```
<OFX>
  <SIGNONMSGSRSV1>              <!--Signon response->
    <SONRS>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <DTSERVER>20050307081437</DTSERVER><!--Timestamp, 3/07/05,
8:14:37am-->
      <LANGUAGE>ENG</LANGUAGE>
      <DTPROFUP>19990301070000</DTPROFUP><!--Timestamp, 3/01/99,
7:00:00am-->
      <DTACCTUP>19990301070000</DTACCTUP><!--Timestamp, 3/01/99,
7:00:00am-->
      </SONRS>
  </SIGNONMSGSRSV1>
  <SIGNUPMSGSRSV1>              <!--Enrollment response-->
    <ENROLLTRNRS>
      <TRNUID>10001</TRNUID>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <ENROLLRS>
        <TEMPPASS>y12345</TEMPPASS>
        <USERID>cindyid</USERID>
        <DTEXPIRE>20050407</DTEXPIRE><!--When Temp Password Expires-->
      </ENROLLRS>
    </ENROLLTRNRS>
  </SIGNUPMSGSRSV1>
</OFX>
```

14.8.3 Activation Example

After enrollment, Cindy wants to sign up with a biller, presumably found with the directory services, with biller ID 415-552-9923 of bill publisher Publisher, Inc.

14.8.3.1 Activation Request

```
<OFX>
  <SIGNONMSGSRQV1>              <!--Signon Request-->
    <SONRQ>                     <!-- ...Sign on request. For a
                                complete example, see section
                                11.14.1-->
    </SONRQ>
  </SIGNONMSGSRQV1>
  <SIGNUPMSGSRQV1>              <!--Activation Request-->
    <ACCTTRNRQ>
      <TRNUID>10002</TRNUID>
      <ACCTRQ>                  <!—-Activate biller acct -->
        <SVCADD>
          <PRESACCTTO>
            <BILLPUB>Publisher, Inc.</BILLPUB>
            <BILLERID>415-552-9923</BILLERID>
            <ACCTID>4128 9343 2324 2314</ACCTID>
           <PRESNAMEADDRESS>
            <NAMEACCTHELD>Cindy P Williams</NAMEACCTHELD><!--Name as
on biller’s statement-->
            <ADDR1>123 Oak St</ADDR1><!--Address as on statement-->
            <CITY>San Jose</CITY>
            <STATE>CA</STATE>
            <POSTALCODE>94111</POSTALCODE>
            <COUNTRY>USA</COUNTRY>
            <DAYPHONE>408-555-2323</DAYPHONE>
           </PRESNAMEADDRESS>
            <USERID>cindyid</USERID>
          </PRESACCTTO>
        </SVCADD>
        <SVC>PRESSVC</SVC>
      </ACCTRQ>
    </ACCTTRNRQ>
  </SIGNUPMSGSRQV1>
</OFX>
```

14.8.3.2 Activation Response

```
<OFX>
  <SIGNONMSGSRSV1>              <!--Signon Response->
    <SONRS>                     <!-- ...Sign on response. For a
                                complete example, see section
                                11.14.1-->
    </SONRS>
  </SIGNONMSGSRSV1>
  <SIGNUPMSGSRSV1>              <!--Enrollment Response-->
    <ACCTTRNRS>                 <!--Service Activation Response-->
      <TRNUID>10002</TRNUID>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <ACCTRS>
        <SVCADD>
          <PRESACCTTO>
            <BILLPUB>Publisher, Inc.</BILLPUB>
            <BILLERID>415-552-9923</BILLERID>
            <ACCTID>4128 9343 2324 2314</ACCTID>
           <PRESNAMEADDRESS>
            <NAMEACCTHELD>Cindy P Williams</NAMEACCTHELD>
            <ADDR1>123 Oak St</ADDR1>
            <CITY>San Jose</CITY>
            <STATE>CA</STATE>
            <POSTALCODE>94111</POSTALCODE>
            <COUNTRY>USA</COUNTRY>
            <DAYPHONE>408-555-2323</DAYPHONE>
           </PRESNAMEADDRESS>
            <USERID>cindyid</USERID>
          </PRESACCTTO>
        </SVCADD>
        <SVC>PRESSVC</SVC>
      </ACCTRS>
    </ACCTTRNRS>
  </SIGNUPMSGSRSV1>
</OFX>
```

14.8.4 Bill Delivery Examples

14.8.4.1 Customer Bill Delivery

The customer, Dan North, wants to see his bills since 3/1/99, which is the last time he asked to see his bills.

14.8.4.1.1 Customer Bill Delivery Request

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>                    <!-- ...Sign on request. For a
                               complete example, see section
                               11.14.1-->
    </SONRQ>
  </SIGNONMSGSRQV1>
  <PRESDLVMSGSRQV1>
    <PRESLISTTRNRQ>
      <TRNUID>12345</TRNUID>
      <PRESLISTRQ>
        <BILLPUB> ABillPublisher</BILLPUB>
        <DTSTART>20050301000000</DTSTART><!--Get Dan's bills since 3/
1/05-->
        <NOTIFYWILLING>Y</NOTIFYWILLING>
        <INCLUDEDETAIL>Y</INCLUDEDETAIL>
      </PRESLISTRQ>
    </PRESLISTTRNRQ>
  </PRESDLVMSGSRQV1>
</OFX>
```

14.8.4.1.2 Customer Bill Delivery Response

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>                    <!-- ...Sign on response. For a
                               complete example, see section
                               11.14.1-->
    </SONRS>
  </SIGNONMSGSRSV1>
  <PRESDLVMSGSRSV1>
    <PRESLISTTRNRS>
      <TRNUID>12345</TRNUID>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY
      </STATUS>
```

```
      <PRESLISTRS>
        <BILLPUB>ABillPublisher</BILLPUB>
        <USERID>MyUserID</USERID>
        <DTSTART>20050301000000</DTSTART><!--Same as in request: no
data loss-->
        <DTEND>20050409090000</DTEND>
                              <!--Value for DTSTART next time-->
        <PRESLIST>
          <PRESBILLINFO>
            <BILLID>65432</BILLID>
            <PRESACCTFROM>
              <BILLPUB>ABillPublisher</BILLPUB>
              <BILLERID>1001</BILLERID><!--Biller id for Power Inc-->
              <ACCTID>1245678GL7</ACCTID><!--Dan North's acct w/
Power Inc-->
            </PRESACCTFROM>
            <BILLREFINFO>1234678GL7970501</BILLREFINFO>
            <AMTDUE>124.24</AMTDUE><!--Dan North to pay $124.24-->
            <DTPMTDUE>20050501</DTPMTDUE><!--by 5/1/05 -->
            <DTBILL>20050401</DTBILL>
            <NOTIFYDESIRED>N</NOTIFYDESIRED>
            <STMNTIMAGE>
              <IMAGEURL>
https://www.Power.com/bills/apr/dannorth.htm?authtoken=65j3ltfm7
              <DTEXPIRE>200504101200</DTEXPIRE>
                              <!--Must visit url by 4/10/05 12am-->
            </STMNTIMAGE>
            <BILLDETAILTABLE>
            <TABLENAME>usage</TABLENAME>
            <BILLDETAILTABLETYPE>x_Power_usage</BILLDETAILTABLETYPE>
                              <!--Power Inc format for usage-->
              <BILLDETAILROW>
                      <C>elec</C><!--Consumable-->
                      <C>20050228</C>
                          <!--Date meter reading start of period-->
                      <C>65543</C>
                            <!--Meter reading at start of period-->
                      <C>20050328</C>
                            <!--Date meter reading end of period-->
                      <C>65643</C><!--Meter reading at end of
period-->
                      <C>100</C><!--Difference in meter readings-->
                      <C>KWH</C><!--Units-->
```

```
                        <C>.8934</C><!--Rate (price per unit)-->
                        <C>89.34</C><!--Charge -->
                    </BILLDETAILROW>
                    <BILLDETAILROW>
                        <C>gas</C><!--Consumable -->
                        <C>20050226</C>
                              <!-Date meter reading start of
                              period-->
                        <C>509843</C>
                              <!--Meter reading at start of period-->
                        <C>20050327</C>
                              <!--Date meter reading end of period-->
                        <C>510843</C><!--Meter reading at end of
period->
                        <C>1000</C><!--Difference in meter readings -
->
                        <C>Therms</C><!--Units -->
                        <C>.02543</C><!--Rate (price per unit) -->
                        <C>25.43</C><!--Charge -->
                    </BILLDETAILROW>
                </BILLDETAILTABLE>
            </PRESBILLINFO>
            <PRESBILLINFO>
                <BILLID>65436</BILLID>
                <PRESACCTFROM>
                      <BILLPUB>ABillPublisher</BILLPUB>
                      <BILLERID>2021</BILLERID>
                                  <!--Biller id of FluteRental, Inc. -->
                      <ACCTID>8765XY95</ACCTID>
                                  <!--Dan North's account number -->
                </PRESACCTFROM>
                <BILLREFINFO>8765XY95970428</BILLREFINFO>
                <AMTDUE>16.21</AMTDUE><!--Total to be paid -->
                <DTPMTDUE>20050428</DTPMTDUE><!--by 4/28/05 -->
                <DTBILL>20050408</DTBILL>
                <NOTIFYDESIRED>N</NOTIFYDESIRED>
                <STMNTIMAGE>
                      <IMAGEURL>
                      https://www.FluteRental.com/95rs3vlx/bill.asp</
IMAGEURL>
                      <DTEXPIRE>20050601</DTEXPIRE><!--Must visit url by
6/1/05-->
                </STMNTIMAGE>
```

```
              <DETAILAVAILABLE>N</DETAILAVAILABLE><!--No structured
detail exists-->
           </PRESBILLINFO>
         </PRESLIST>
       </PRESLISTRS>
     </PRESLISTTRNRS>
  </PRESDLVMSGSRSV1>
</OFX>
```

14.8.4.2 Bill Delivery for Customer Service Representative

This example assumes that Dan North calls Power Inc with a question about his power bill. Power's customer service representative, Maria Smith, uses a similar application and a similar OFX request to see the same bill that Dan sees.

14.8.4.2.1 Bill Delivery Request Example for a Customer Service Representative

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>                    <!-- ...Sign on request. For a
                               complete example, see section
                               11.14.1-->
    </SONRQ>
  </SIGNONMSGSRQV1>
  <PRESDLVMSGSRQV1>
    <PRESLISTTRNRQ>
      <TRNUID>23456/<TRNUID>
      <USERID>MyUserID</USERID><!--Asks for Dan North's bills -->
      <PRESLISTRQ>
        <BILLPUB> ABillPublisher</BILLPUB>
        <DTSTART>20050330</DTSTART><!--Approximate date -->
        <DTEND>2005040410</DTSTART><!--Approximate date -->
        <NOTIFYWILLING>N</NOTIFYWILLING>
        <INCLUDEDETAIL>Y</INDLUDEDETAIL>
      </PRESLISTRQ>
    </PRESLISTTRNRQ>
  </PRESDLVMSGSRQV1>
</OFX>
```

14.8.4.2.2 Bill Delivery Response Example for a Customer Service Representative

The response from the server includes the Power Incorporated bill, but not the FluteRental bill. This is because the server decides that Maria Smith's credentials are good enough to see Dan North's Power Inc bill, but not good enough to see anything else.

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>                    <!-- ...Sign on response. For a
                               complete example, see section
                               11.14.1-->
    </SONRS>
  </SIGNONMSGSRSV1>
  <PRESDLVMSGSRSV1>
    <PRESLISTTRNRS>
      <TRNUID>23456</TRNUID>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <PRESLISTRS>
        <BILLPUB>ABillPublisher</BILLPUB>
        <USERID>MyUserID</USERID><!--Dan North's userid, so Dan's
bill-->
        <DTSTART>20050328</DTSTART><!--Same as in request: no data
loss-->
        <DTEND>20050409</DTEND><!--Same as in request-->
        <PRESLIST>
          <PRESBILLINFO>
            <BILLID>65432</BILLID>
            <PRESACCTFROM>
              <BILLPUB>ABillPublisher</BILLPUB>
              <BILLERID>1001</BILLERID><!--Biller id for Power Inc-->
              <ACCTID>1245678GL7</ACCTID>
                              <!--Dan's account with Power Inc-->
              <USERID>MyUserID</USERID><!--Dan North's userid-->
            </PRESACCTFROM>
            <BILLREFINFO>1234678GL7970501</BILLREFINFO>
            <AMTDUE>124.24</AMTDUE><!--Dan to pay $124.24-->
            <DTPMTDUE>20050501</DTPMTDUE><!--by 5/1/05 -->
            <DTBILL>20050401</DTBILL>
            <NOTIFYDESIRED>N</NOTIFYDESIRED>
            <STMNTIMAGE>
              <IMAGEURL>https://www.Power.com/bills/apr/
dannorth.htm?authtoken=987ab6gr8y</IMAGEURL>
              <DTEXPIRE>200504111200</DTEXPIRE>
                              <!--Must visit url by 4/11/05 12am-->
            </STMNTIMAGE>
            <BILLDETAILTABLE>
```

```
                <TABLENAME>usage</TABLENAME>
                <BILLDETAILTABLETYPE>x_Power_usage</
BILLDETAILTABLETYPE>      <!--Power Inc format for usage-->
                <BILLDETAILROW>
                          <C>elec</C><!--Consumable-->
                          <C>20050228</C>
                         <!--Date meter reading start of period-->
                           <C>65543</C>
                         <!--Meter reading at start of period-->
                           <C>20050328</C>
                         <!--Date meter reading end of period-->
                           <C>65643</C>
                         <!--Meter reading at end of period-->
                         <C>100</C><!--Difference in meter readings-->
                          <C>KWH</C><!--Units-->
                          <C>.8934</C><!--Rate (price per unit)-->
                          <C>89.34</C><!--Charge-->
                </BILLDETAILROW>
                <BILLDETAILROW>
                          <C>gas</C><!--Consumable-->
                          <C>20050226</C>
                              <!--Date meter reading start of period-->
                          <C>509843</C>
                              <!--Meter reading at start of period-->
                          <C>20050327</C>
                              <!--Date meter reading end of period-->
                          <C>510843</C>
                              <!--Meter reading at end of period-->
                         <C>1000</C><!--Difference in meter readings-->
                         <C>Therms</C><!--Units-->
                         <C>.02543</C><!--Rate (price per unit)-->
                         <C>25.43</C><!--Charge-->
                </BILLDETAILROW>
              </BILLDETAILTABLE>
            </PRESBILLINFO>
          </PRESLIST>
        </PRESLISTRS>
      </PRESLISTTRNRS>
    </PRESDLVMSGSRSV1>
</OFX>
```

14.8.4.3 Bill Delivery for a Group of Users

In this example, Realtors Company downloads the phone bills for the employees' office phones by asking the bill publisher to see the bills for the group RealtorsEmployees. The composition of the group RealtorsEmployees has been agreed upon between Realtors Company and the bill publisher; moreover, the bill publisher has agreed to grant Realtors Company access rights to the RealtorsEmployees group. All this took place outside of OFX.

14.8.4.3.1 Bill Delivery Request Example for a Group of Users

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>                    <!-- ...Sign on request. For a
                               complete example, see section
                               11.14.1-->
    </SONRQ>
  </SIGNONMSGSRQV1>
  <PRESDLVMSGSRQV1>
    <PRESLISTTRNRQ>
      <TRNUID>34567<TRNUID>
      <GROUPID>RealtorsEmployees</GROUPID><!--Asks for Employee's
phone bills-->
      <PRESLISTRQ>
        <BILLPUB> ABillPublisher</BILLPUB>
        <DTSTART>20050430</DTSTART><!--since 4/30/2005-->
        <NOTIFYWILLING>N</NOTIFYWILLING>
        <INCLUDEDETAIL>Y</INCLUDEDETAIL>
      </PRESLISTRQ>
    </PRESLISTTRNRQ>
  </PRESDLVMSGSRQV1>
</OFX>
```

14.8.4.3.2 Bill Delivery Response Example for a Group of Users

The response, not shown here, will include several bills each marked with its own <USERID>. The only bills returned will be the employees' phone bills for their office phones, since those bills are the only ones to which Realtor Company has access rights.

14.8.4.4 Group Account Information

This is an example of a client proxy system that is tracking changes to the accounts of a group of users.

14.8.4.4.1 Group Account Information Request

```
<OFX>
  <SIGNONMSGSRQV1>              <!--Signon Request->
    <SONRQ>                     <!-- ...Sign on request. For a
                                complete example, see section
                                11.14.1-->
    </SONRQ>
  </SIGNONMSGSRQV1>
  <PRESDLVMSGSRQV1>             <!--Group Account Info Request-->
    <PRESGRPACCTINFOTRNRQ>
      <TRNUID>10001</TRNUID>
      <GROUPID>AClientProxysCustomers</GROUPID>
                                <!--Predefined group of customers-->
      <ACCTINFORQ>
        <DTACCTUP>19990104</DTACCTUP><!--Last DTACCTUP received for
group-->
      </ACCTINFORQ>
    </PRESGRPACCTINFOTRNRQ>
  </PRESDLVMSGSRQV1>
</OFX>
```

14.8.4.4.2 Group Account Information Response

```
<OFX>
  <SIGNONMSGSRSV1>              <!--Signon Response->
    <SONRS>                     <!-- ...Sign on response. For a
                                complete example, see section
                                11.14.1-->
    </SONRS>
  </SIGNONMSGSRSV1>
  <PRESDLVMSGSRSV1>             <!--Group Account Info Response-->
    <PRESGRPACCTINFOTRNRS>
      <TRNUID>10001</TRNUID>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <ACCTINFORS>
        <DTACCTUP>19990122092431</DTACCTUP>
        <ACCTINFO>
```

```
            <PRESACCTINFO>
               <PRESACCTFROM>
                  <BILLPUB>PUBLISHER, INC.</BILLPUB>
                  <BILLERID>415-552-9923</BILLERID>
                  <ACCTID>408-555-4342-M132</ACCTID>
                  <USERID>bygeorge</USERID>
                        <!--User from group with new status-->
                  </PRESACCTFROM>
                  <SVCSTATUS>ACTIVE</SVCSTATUS>
            </PRESACCTINFO>
          </ACCTINFO>
        </ACCTINFORS>
        <ACCTINFORS>
          <DTACCTUP>19990123082423</DTACCTUP>
          <ACCTINFO>
            <PRESACCTINFO>
               <PRESACCTFROM>
                     <BILLPUB>PUBLISHER, INC.</BILLPUB>
                     <BILLERID>415-552-9923</BILLERID>
                       <ACCTID>408-555-2341-U421</ACCTID>
               <USERID>1324-42</USERID>
                     <!--User from group with new status-->
               </PRESACCTFROM>
               <SVCSTATUS>REJECTED</SVCSTATUS>
               <REASON>ACCOUNT NOT FOUND</REASON>
                                 <!--User supplied account with biller->
                                 <!--didn't match biller's records-->
            </PRESACCTINFO>
          </ACCTINFO>
        </ACCTINFORS>
      </PRESGRPACCTINFOTRNRS>
    </PRESDLVMSGSRSV1>
</OFX>
```

CHAPTER 15 IMAGES

15.1 Overview

As more financial services companies offer users access to information over the internet, paper statements are being replaced with electronic statement images. Additionally, with the passing of federal law "Check 21", financial institutions now have an ability to truncate checks and convert them to images at the originating institutions. Consequently, users may no longer receive cancelled checks, deposit tickets, and monthly statements via regular mail.

OFX addresses a growing need for communicating images information in an electronic format by allowing clients to request transaction and statement images as part of the statement and closing statement requests, respectively. The server has several options available to support image download requests (see sections 15.2.1 through 15.2.3).

15.2 Image Download Options

This section describes the process of downloading images associated with statements and transactions, beginning with requesting and receiving image identifier data, followed by retrieving the actual images.

The client requests image identifier data by sending an OFX statement download or closing request with the tags <INCTRANIMG>Y or <INCSTMTIMG>Y, respectively. (The statement download and/or closing requests are among those found in the Banking, Credit Card, Investments and/or Loan message sets.) If an image is available, the server sends back one or more <IMAGEDATA> aggregates in the response. (More than one <IMAGEDATA> aggregate might be returned for check images.) The <IMAGEDATA> aggregate contains data necessary for retrieving the actual image. Each image is represented by one <IMAGEDATA> aggregate. (See section 3.1.6.1 for a description of the <IMAGEDATA> aggregate.)

The retrieval method used to get the actual image is dependent on the value of <IMAGEREFTYPE> in the <IMAGEDATA> aggregate. These values are OPAQUE, URL or FORMURL and the corresponding methods are described in sections 15.2.1, 15.2.2 and 15.2.3, below.

15.2.1 Image Retrieval Method # 1: Reference Type OPAQUE

The <IMAGEDATA> aggregate returned by the server contains the information needed for retrieving an image. One retrieval method is specified in the < IMAGEDATA> aggregate by the following OFX Reference Type tag containing the value OPAQUE:

<IMAGEREFTYPE>OPAQUE

The OPAQUE moniker requires the client to send an OFX request to access an image. This request will be in the form of a normal OFX request (complete with Signon) and includes an <IMAGERQ> request as described in section 15.2.1.1. However, whereas the request file contains typical OFX syntax, the successful response returned is in the form of raw bytes. If a failure condition occurs, the response should be as described in Section 15.2.1.3.

15.2.1.1 OFX Image Request <IMAGERQ>

The client sends a single <IMAGERQ> request corresponding to each <IMAGEDATA> aggregate associated with the OPAQUE moniker. Each <IMAGERQ> request must be sent in its own OFX request file. Only one image is returned for each <IMAGERQ>, that is, for each OFX request file.

The <IMAGERQ> request must appear within the <IMAGETRNRQ> transaction wrapper.

	Description
<IMAGERQ>	Image request aggregate
<IMAGEREF>	Image identifier from <IMAGEDATA> aggregate, see section 3.1.6.1, *A-1024*
</IMAGERQ>	End of image request aggregate

15.2.1.2 OFX Image Response - Successful Response

The successful response to an OFX <IMAGERQ> request is a complete HTTP response followed by a series of raw bytes. The HTTP response contains content-type set to a supported image type, e.g. image/jpeg. The allowed types in the response are shown in the table below.

Type	*Description*
image/jpeg	JPEG images
image/tiff	TIFF images
Application/pdf	PDF documents
image/png	PNG images

15.2.1.3 OFX Image Response - Unsuccessful Response

An unsuccessful response to an OFX <IMAGERQ> request is a complete HTTP response followed by an error string, “OFX-STATUS=XXXX”. The possible error status code values for OFX-STATUS are shown in the table below.

	Meaning
15000	Must change USERPASS (ERROR)
15500	Signon Invalid (ERROR)
15502	USERPASS Lockout (ERROR)
16510	Image not available yet; try again later (ERROR)
16511	Image expired (ERROR)
16512	Image Reference Identifier not found (ERROR)
16513	Image server is not available; try again later (ERROR)

```
Here is a typical unsuccessful response:

HTTP 1.0 200 OK                 HTTP headers
Content-Type: application/x-ofx
Content-Length: 20

OFX-STATUS=15500                OFX status
```

15.2.2 Image Retrieval Method # 2: Reference Type URL

The <IMAGEDATA> aggregate returned by the server contains the information needed for retrieving an image. One retrieval method is specified in the <IMAGEDATA> aggregate by the following OFX Reference Type tag containing the value URL:

<IMAGEREFTYPE>URL

The URL moniker does not require the client to send an OFX request to access an image. Instead, the client issues an HTTP request to the URL specified as the value of the <IMAGEREF> tag in the <IMAGEDATA> aggregate. The client will then authenticate. Once this authentication takes place, the image can be displayed.

15.2.3 Image Retrieval Method # 3: Reference Type FORMURL

The <IMAGEDATA> aggregate returned by the server contains the information needed for retrieving an image. One retrieval method is specified in the < IMAGEDATA> aggregate by the following OFX Reference Type tag containing the value FORMURL:

<IMAGEREFTYPE>FORMURL

The FORMURL moniker does not require the client to send an OFX request to access an image. Instead, the client issues an HTTP request, with encoded data specified in the URL retrieved from the <IMAGEREF> tag in the <IMAGEDATA> aggregate. The image can then be displayed. See Section 15.2.3.1, below, for more information about the HTTP POST request.

15.2.3.1 HTTP POST Request

Clients use the HTTP POST command to send a request to the previously acquired URL for the desired financial institution. The URL identifies a service on an FI server that can accept an image request and produce a response.

The POST identifies the data as being of content-type *application/x-www-form-urlencoded*. The OFX client will send a request using HTTP POST (over SSL) formatted as an HTML FORM. The form may include financial institution identification, client application identification, user ID, password, and image reference fields as shown in the table below.

	Description
<input name="APPID">	ID of client application, *A-5*
<input name="APPVER">	Version of client application (6.00 encoded as 0600), *N-4*
<input name="ORG">	Organization defining this FI namespace, *A-32*
<input name="FID">	ID of the financial institution, *A-32*
<input name="USERID">	User website identification string, *String–255*
<input name="USERPASS">	User password on website, *String-255*
<input name="CRED2">	Secondary user website identification string, if required, *String-255*
<input name="CRED3">	Tertiary user website identification string, if required, *String-255*
<input name="IMAGEREF">	Unique identifier for the image, *A-1024*

Some web sites require more than the standard <USERID> <USERPASS> provided by OFX. This specification includes optional secondary credentials <CRED2> and <CRED3> for financial institution websites that require them. Special characters and spaces within the user ID or password will be encoded using standard URL encoding.

15.3 Message Sets and Profile

The following tables show the OFX image transactions and their corresponding transaction wrappers for a particular message set.

15.3.1 Image Message Set Request Messages

Message	
<IMAGEMSGSRQV1>	
	IMAGETRNRQ
	IMAGERQ
</IMAGEMSGSRQV1>	

15.3.2 Image Message Set Response Messages

There are no standard OFX Image Message Set responses.

15.3.3 Image Message Set Profile

The following table shows the Image Message Set profile. Note that there is one URL associated with all images, i.e. the <IMAGERQ> request is always sent to the same URL regardless of the message set the image is associated with.

Tag	*Description*
<IMAGEMSGSET>	Message set for images
<IMAGEMSGSETV1>	Version 1 of message set.
<MSGSETCORE>	Common message set core
</MSGSETCORE>	
<DFLTIMAGEDELAY>	Default number of calendar days from <DTSERVER> (for statement images) or <DTPOSTED> (for transaction images) when image will become available. If not present, the value is presumed to be 0, *N-5*
	Can be overridden, for each image, by either <IMAGEDELAY> or <DTIMAGEAVAIL> in the <IMAGEDATA> aggregate.
<DFLTIMAGETTL>	Number of calendar days the image will remain available on the host once the image becomes available. If not present, the value is presumed to be indefinitely, *N-5*
	Can be overridden, for each image, by <IMAGETTL> in the <IMAGEDATA> aggregate.

Tag	*Description*
</IMAGEMSGSETV1> **</IMAGEMSGSET>**	

15.4 Examples

15.4.1 Transaction Image Example

This example represents a customer who requests a statement download for a checking account.

The request includes a request for transaction images. The response contains an updated balance for the account, two transactions and image information for one of the transactions.

The first request file:

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>
        <DTCLIENT>20050929101000.000[-4:EDT]</DTCLIENT>
        <USERID>JSmith123</USERID>
        <USERPASS>MyPassword</USERPASS>
        <LANGUAGE>ENG</LANGUAGE>
        <FI>
            <ORG>NCH</ORG>
            <FID>1001</FID>
        </FI>
        <APPID>MyApp</APPID>
        <APPVER>0500</APPVER>
    </SONRQ>
  </SIGNONMSGSRQV1>
  <BANKMSGSRQV1>
    <STMTTRNRQ>
        <TRNUID>1001</TRNUID>
        <STMTRQ>
          <BANKACCTFROM>
            <BANKID>121099999</BANKID>
            <ACCTID>999988</ACCTID>
            <ACCTTYPE>CHECKING</ACCTTYPE>
          </BANKACCTFROM>
          <INCTRAN>
            <DTSTART>20050927090000.000[-4:EDT]</DTSTART>
            <INCLUDE>Y</INCLUDE>
          </INCTRAN>
          <INCTRANIMG>Y</INCTRANIMG><!-- request transaction images -->
        </STMTRQ>
    </STMTTRNRQ>
  </BANKMSGSRQV1>
```

```
</OFX>
```

The first response file:

```
The response contains retrieval information for a check image.
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <DTSERVER>20050929101100.000[-4:EDT]</DTSERVER>
      <LANGUAGE>ENG</LANGUAGE>
      <FI>
        <ORG>NCH</ORG>
        <FID>1001</FID>
      </FI>
    </SONRS>
  </SIGNONMSGSRSV1>
  <BANKMSGSRSV1>
    <STMTTRNRS>
      <TRNUID>1001</TRNUID>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
      </STATUS>
      <STMTRS>
        <CURDEF>USD</CURDEF>
        <BANKACCTFROM>
          <BANKID>121099999</BANKID>
          <ACCTID>999988</ACCTID>
          <ACCTTYPE>CHECKING</ACCTTYPE>
        </BANKACCTFROM>
        <BANKTRANLIST>
          <DTSTART>20050927090000.000[-4:EDT]</DTSTART>
          <DTEND>20050929090000.000[-4:EDT]</DTEND>
          <STMTTRN>
            <TRNTYPE>CHECK</TRNTYPE>
            <DTPOSTED>20050928090000.000[-4:EDT]</DTPOSTED>
            <TRNAMT>-1000.00</TRNAMT>
            <FITID>28</FITID>
            <CHECKNUM>2010</CHECKNUM>
```

```
            <IMAGEDATA><!-- front of check image -->
                <IMAGETYPE>TRANSACTION</IMAGETYPE>
                <IMAGEREF>11111</IMAGEREF>
                <IMAGEREFTYPE>OPAQUE</IMAGEREFTYPE>
                <CHECKSUP>FRONTONLY</CHECKSUP>
            </IMAGEDATA>
            <IMAGEDATA><!-- back of check image -->
                <IMAGETYPE>TRANSACTION</IMAGETYPE>
                <IMAGEREF>2222222222</IMAGEREF>
                <IMAGEREFTYPE>OPAQUE</IMAGEREFTYPE>
                <CHECKSUP>BACKONLY</CHECKSUP>
            </IMAGEDATA>
          </STMTTRN>
          <STMTTRN>
            <TRNTYPE>FEE</TRNTYPE>
            <DTPOSTED>20050928090000.000[-5:EST]</DTPOSTED>
            <TRNAMT>-7.55</TRNAMT>
            <FITID>35</FITID>
          </STMTTRN>
        </BANKTRANLIST>
        <LEDGERBAL>
          <BALAMT>58336.07</BALAMT>
          <DTASOF>20050929101100.000[-4:EDT]</DTASOF>
        </LEDGERBAL>
        <AVAILBAL>
          <BALAMT>58336.07</BALAMT>
          <DTASOF>20050929101100.000[-4:EDT]</DTASOF>
        </AVAILBAL>
      </STMTRS>
    </STMTTRNRS>
  </BANKMSGSRSV1>
</OFX>
```

The second request file:

Now the client requests the image using the information in the previous response file. Since the server supports returning both front and back of the check separately, two requests are sent by the client. The following request asks for the front of the check.

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>
        ...
```

```
    </SONRQ>
  </SIGNONMSGSRQV1>
  <IMAGEMSGSRQV1>
    <IMAGETRNRQ>
        <TRNUID>145</TRNUID>
        <IMAGERQ>
            <IMAGEREF>11111</IMAGEREF>
        </IMAGERQ>
    </IMAGETRNRQ>
  </IMAGEMSGSRQV1>
</OFX>
```

The second response file:

The response is a complete HTTP response with a content-type set to a supported image type. The actual content, in the form of raw bytes, is not displayed here.

The third request file:

The next request asks for the back of the check.

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>
        ...
    </SONRQ>
  </SIGNONMSGSRQV1>
  <IMAGEMSGSRQV1>
    <IMAGETRNRQ>
        <TRNUID>146</TRNUID>
        <IMAGERQ>
            <IMAGEREF>2222222222</IMAGEREF>
        </IMAGERQ>
    </IMAGETRNRQ>
  </IMAGEMSGSRQV1>
</OFX>
```

The third response file:

The response is a complete HTTP response with a content-type set to a supported image type. The actual content, in the form of raw bytes, is not displayed here.

15.4.2 Statement Closing Image Example

This example represents a customer who requests a closing statement for a checking account. The request includes a request for a statement image. The response contains closing information for the account plus image information for that statement image. The image will be available for 90 days.

The request file:

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>
        ...
    </SONRQ>
  </SIGNONMSGSRQV1>
  <BANKMSGSRQV1>
    <STMTENDTRNRQ>
        <TRNUID>1002</TRNUID>
        <STMTENDRQ>
          <BANKACCTFROM>
            <BANKID>121099999</BANKID>
            <ACCTID>999987</ACCTID>
            <ACCTTYPE>CHECKING</ACCTTYPE>
          </BANKACCTFROM>
          <INCSTMTIMG>Y</INCSTMTIMG>
        </STMTENDRQ>
    </STMTENDTRNRQ>
  </BANKMSGSRQV1>
</OFX>
```

The response file:

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>
        ...
    </SONRS>
  </SIGNONMSGSRSV1>
  <BANKMSGSRSV1>
    <STMTENDTRNRS>
        <TRNUID>1002</TRNUID>
        <STATUS>
            <CODE>0</CODE>
            <SEVERITY>INFO</SEVERITY>
        </STATUS>
```

```
        <STMTENDRS>
          <CURDEF>USD</CURDEF>
          <BANKACCTFROM>
            <BANKID>121099999</BANKID>
            <ACCTID>999987</ACCTID>
            <ACCTTYPE>CHECKING</ACCTTYPE>
          </BANKACCTFROM>
          <CLOSING>
              <FITID>2005-09-20 15:47:40.76</FITID>
              <DTCLOSE>20050920154740.760[-4:EDT]</DTCLOSE>
              <BALCLOSE>336.07</BALCLOSE>
              <DTPOSTSTART>20050101000000.000[-5:EST]</DTPOSTSTART>
              <DTPOSTEND>20050920000000.000[-4:EDT]</DTPOSTEND>
              <IMAGEDATA>
                  <IMAGETYPE>STATEMENT</IMAGETYPE>
                  <IMAGEREF>4448444</IMAGEREF>
                  <IMAGEREFTYPE>OPAQUE</IMAGEREFTYPE>
                  <IMAGETTL>90</IMAGETTL>
              </IMAGEDATA>
          </CLOSING>
        </STMTENDRS>
    </STMTENDTRNRS>
  </BANKMSGSRSV1>
</OFX>
```

The request file:

Now the client requests the image using the information in the response file.

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>
        ...
    </SONRQ>
  </SIGNONMSGSRQV1>
  <IMAGEMSGSRQV1>
    <IMAGETRNRQ>
        <TRNUID>99887</TRNUID>
        <IMAGERQ>
            <IMAGEREF>4448444</IMAGEREF>
        </IMAGERQ>
    </IMAGETRNRQ>
```

```
  </IMAGEMSGSRQV1>
</OFX>
```

The response file:

The response is a complete HTTP response with a content-type set to a supported image type. The actual content, in the form of raw bytes, is not displayed here.

15.4.3 Transaction FORMURL Example

The following shows the FORMURL method of image retrieval, assuming the initial OFX response contains <IMAGEREFTYPE>FORMURL</IMAGEREFTYPE>.

The request file:

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>
        ...
    </SONRQ>
  </SIGNONMSGSRQV1>
  <BANKMSGSRQV1>
    <STMTENDTRNRQ>
        <TRNUID>1002</TRNUID>
        <STMTENDRQ>
          <BANKACCTFROM>
            <BANKID>121099999</BANKID>
            <ACCTID>999987</ACCTID>
            <ACCTTYPE>CHECKING</ACCTTYPE>
          </BANKACCTFROM>
          <INCSTMTIMG>Y</INCSTMTIMG>
        </STMTENDRQ>
    </STMTENDTRNRQ>
  </BANKMSGSRQV1>
</OFX>
```

The response file:

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>
        ...
    </SONRS>
  </SIGNONMSGSRSV1>
  <BANKMSGSRSV1>
    <STMTENDTRNRS>
        <TRNUID>1002</TRNUID>
        <STATUS>
            <CODE>0</CODE>
            <SEVERITY>INFO</SEVERITY>
        </STATUS>
        <STMTENDRS>
          <CURDEF>USD</CURDEF>
```

```
          <BANKACCTFROM>
            <BANKID>121099999</BANKID>
            <ACCTID>999987</ACCTID>
            <ACCTTYPE>CHECKING</ACCTTYPE>
          </BANKACCTFROM>
          <CLOSING>
              <FITID>2005-09-20 15:47:40.76</FITID>
              <DTCLOSE>20050920154740.760[-4:EDT]</DTCLOSE>
              <BALCLOSE>336.07</BALCLOSE>
              <DTPOSTSTART>20050101000000.000[-5:EST]</DTPOSTSTART>
              <DTPOSTEND>20050920000000.000[-4:EDT]</DTPOSTEND>
              <IMAGEDATA>
                  <IMAGETYPE>STATEMENT</IMAGETYPE>
                  <IMAGEREF>11111</IMAGEREF>
                  <IMAGEREFTYPE>FORMURL</IMAGEREFTYPE>
                  <IMAGETTL>90</IMAGETTL>
              </IMAGEDATA>
          </CLOSING>
        </STMTENDRS>
    </STMTENDTRNRS>
  </BANKMSGSRSV1>
</OFX>
```

The request file:

Now the client requests the image with HTTP content-type set to *application/x-www-form-urlencoded*.

```
<form name="RequestOFXImages" method="post">
APPID=MyOFXApp&APPVER=0600&ORG=NCH&FID=1100&USERID=MyID&USERPASS=My$ec
retPwd&IMAGEREF=11111
</form>
```

Successful response:

The response is a complete HTTP response.

Unsuccessful response to same request that fails because the image is not available yet:

OFX-STATUS=16510

15.4.4 Profile Request and Response Showing Image Support

The following shows a profile request and response that shows support for banking, credit cards, loans, and images for these message sets.

The following are OFX fragments; the missing OFX is covered in other request/response examples. (For readability, the end tags for element tags are not shown.)

Request:

```
<PROFMSGSRQV1>
<PROFTRNRQ>
  <TRNUID>974495445
  <PROFRQ>
    <CLIENTROUTING>MSGSET
    <DTPROFUP>20021107103100.000[-5:EST]
  </PROFRQ>
</PROFTRNRQ>
</PROFMSGSRQV1>
```

Response:

```
<PROFMSGSRSV1>
  <PROFTRNRS>
  <TRNUID>974495445
  <STATUS>
    <CODE>0
    <SEVERITY>INFO
  </STATUS>
  <PROFRS>
  <MSGSETLIST>
    <SIGNONMSGSET><SIGNONMSGSETV1>
        <MSGSETCORE>...</MSGSETCORE>
    </SIGNONMSGSETV1></SIGNONMSGSET>
    <BANKMSGSET><BANKMSGSETV1>
      <MSGSETCORE>...</MSGSETCORE>
      <CLOSINGAVAIL>Y
      <XFERPROF>
        ...
      </XFERPROF>
      <EMAILPROF>
        ...
      </EMAILPROF>
```

```
      <IMAGEPROF>
        <CLOSINGIMGAVAIL>Y
        <TRANIMGAVAIL>Y
      </IMAGEPROF>
    </BANKMSGSETV1></BANKMSGSET>
    <CREDITCARDMSGSET><CREDITCARDMSGSETV1>
      <MSGSETCORE>...</MSGSETCORE>
      <CLOSINGAVAIL>Y
      <IMAGEPROF>
        <CLOSINGIMGAVAIL>Y
        <TRANIMGAVAIL>N  <!-- Not supported for CCs -->
      </IMAGEPROF>
    </CREDITCARDMSGSETV1></CREDITCARDMSGSET>
    <LOANMSGSET><LOANMSGSETV1>
      <MSGSETCORE>
        ...
      </MSGSETCORE>
      <TRANDNLD>Y
      <CLOSINGAVAIL>Y
      <AMRTAVAIL>Y
      <CANEMAIL>N
      <IMAGEPROF>
        <CLOSINGIMGAVAIL>Y
        <TRANIMGAVAIL>Y
      </IMAGEPROF>
    </LOANMSGSET></LOANMSGSETV1>
    <IMAGEMSGSET><IMAGEMSGSETV1>
      <MSGSETCORE>...</MSGSETCORE>
      <DFLTIMAGETTL>90  <!-- Image will be available for 90 days -->
    </IMAGEMSGSETV1></IMAGEMSGSET>
    <PROFMSGSET>
      ...
    </PROFMSGSET>
  </MSGSETLIST>
  <SIGNONINFOLIST>
    ...
  </SIGNONINFOLIST>
  </PROFTRNRS>
</PROFMSGSRSV1>
</OFX>
```

CHAPTER 16 AUTOMATIC 1-WAY OFX

In support of extending the work already developed on financial institution web sites to download OFX formatted statement data, OFX provides a standard for OFX clients to request this data from a web site on behalf of customers.

16.1 RequestOFX Form

The following table outlines the components of the HTTP form request. Tags in bold are required.

<form name ="RequestOFX" method="post">	***Description***
<input name="APPID">	ID of client application, *A-5*
<input name="APPVER">	Version of client application (6.00 encoded as 0600), *N-4*
<input name="ORG">	Organization defining this FI namespace, *A-32*
<input name="FID">	ID of the financial institution, *A-32*
<input name="USERID">	User website identification string, *String–255*
<input name="USERPASS">	User password on website, *String-255*
<input name="CRED2">	Secondary user website identification string, if required, *String-255*
<input name="CRED3">	Tertiary user website identification string, if required, *String-255*
<CLIENTUID>	Unique ID identifying OFX client, *A-36*
<MFA_XXX>	0 or more MFA Challenge Question answers. The tag name must correspond to the MFAPHRASEID in the last received MFACHALLENGERS.
<input name="DTSTART">	Start date of statement requested using ISO 8601 standard YYYY-MM-DD, e.g. 2005-07-16
<input name="DTEND">	End date of statement requested using ISO 8601 standard YYYY-MM-DD, e.g. 2005-07-16
<input name="INCTRAN">	Include-transaction aggregate If value = Y, website should include the following in the OFX formatted file when available: <BANKTRANLIST> in <STMTRS> <BANKTRANLIST> in <CCSTMTRS> <INVTRANLIST> in <INVSTMTRS> <LOANTRANLIST> in <LOANSTMTRS> <AMRTTRANLIST> in <AMRSTMTRS>

The tag names for the challenge question answers are constructed from the <MFAPHRASEID>s of the associated challenge questions. For example, if one of the challenge questions has an <MFAPHRASEID>

The client will send zero or one <ACCTID> tag.

- If the client includes the <ACCTID> tag in the Form request, the website should interpret this as a request for statement data for an account with only this account ID.
- If the client does not include the <ACCTID> tag, the client is requesting statement data for all accounts associated with this <USERID> available via <OFX> download from the website.
- If the website is unable to compile an OFX response for more than one account, the website should respond to the request for multiple data with an “OFX-STATUS = 15509” in the HTTP response. Refer to section 16.2.2.

16.1.5.1 Unrecognized Account IDs

Upon receipt of a request for a statement for an invalid account, the website should respond to the request with an "OFX-STATUS=2003" in the HTTP response.

16.1.6 Account Type <ACCTTYPE>

An account type value may be sent by the client to disambiguate two accounts that have the same account ID, or simply to clarify the request. Clients sending <ACCTTYPE> must include <ACCTID> in the request.

The website should ignore this value if it does not coincide with the account type on record, if possible. Otherwise, it may respond to the request with an "OFX-STATUS = 2003" in the HTTP response. Refer to section 16.2.2.

16.2 HTTP Form Response

16.2.1 Response to RequestOFX Form

Data returned by way of HTTP follows the standard HTTP 1.0 specification. Place the standard HTTP result code on the first line. HTTP defines a number of status codes. Servers can return any standard HTTP result, including the following suggested values.

	Meaning	*Action*
200	OK	The request was processed and a valid Open Financial Exchange result is returned.
400s	Bad request	The request was invalid and was not processed. Clients will report an internal error to the user.
500s	Server error	The server is unavailable. Clients should advise the user to retry shortly.

Open Financial Exchange requires the following HTTP standard headers in the response returned by the website:

Code	*Value*	*Explanation*
Content-type	application/x-ofx	The MIME type for Open Financial Exchange
Content-length	length	Length (given in bytes) of the data after removing HTTP headers

Having authenticated the request, the website responds with an OFX download, just as if the customer had manually entered the request on the site.

Here is a typical successful response:

```
HTTP 1.0 200 OK                    HTTP headers
Content-Type: application/x-ofx
Content-Length: 8732

<?OFX OFXHEADER="200" VERSION="211" SECURITY="NONE" OLDFILEUID="NONE"
NEWFILEUID="NONE"?>

<OFX>
... Open Financial Exchange response following the OFX specification...
</OFX>
```

16.2.2 Status Responses

If the financial institution website recognizes the request, but is unable to return an OFX response for some reason, the website should respond to the HTTP request with a complete HTTP response including an error, "**OFX-STATUS=XXXX**", indicating the appropriate status. Status codes are discussed further in appropriate sections of Chapter 2.

Enumerated values for OFX-STATUS:

	Meaning
0	Success (INFO) - Return this when there is no statement available for the specified timeframe.
2000	General error (ERROR)
2003	Account not found (ERROR)
3000	User credentials are correct, but further authentication is needed (ERROR)
15500	Signon invalid (ERROR)
15501	Customer account already in use (ERROR)
15502	USERPASS lockout (ERROR)
15509	Multiple accounts not supported (ERROR)
15511	User should contact financial institution (ERROR)

Here is a typical unsuccessful response:

```
HTTP 1.0 200 OK                    HTTP headers
Content-Type: application/x-ofx
Content-Length: 17

OFX-STATUS=15500                   OFX status
```

16.2.3 Challenge Response

If the user credentials supplied on the original request are correct but not sufficient to authenticate the user, servers may respond with a status code of 3000. When the status code of 3000 is sent, the server must also include an <MFACHALLENGERS> (see Section 2.5.4.2) containing questions for the user to answer.

16.3 Automatic 1-Way URL

In order to make it simpler for OFX clients to take advantage of the new automatic 1-way capabilities of financial institution websites, financial institution may choose to use the suggested common URL format for automatic 1-way requests: https://websiteurl.com/ofx.XXX, where “websiteurl” is the homepage of the financial institution website and “XXX” would be the type of page required by the financial institution web site. Examples page types could include, but are not limited to, the following: .cgi, .asp, .aspx, .jsp, and html. Environmental variables may be included at the end of the specified URL, but should be static.

16.4 Redirection and Cookies

Some FIs may be set up to implement the authentication on one server and then redirect the client to another URL for the actual web application. In such solutions, the redirection carries a proof of authentication token (either on the URL or in a cookie) so that the web application can validate that the user has been successfully authenticated.

To support such solutions, Automatic 1-way OFX clients need to handle redirection and cookies just like a generic web browser would.

It is up to the FI, however, to include the details of the 1-way OFX request together with the proof of authentication token to the application site where the request is processed and the response is generated.

The client only knows about the initial URL where the authentication happens. If the processing needs to be done on a different URL, the client does support redirection and cookies but it does not post the request details again to the redirect target URL.

Clients use the HTTP POST command to send a request to the previously acquired URL for the desired financial institution. The URL presumably identifies a Common Gateway Interface (CGI) or other process on an FI server that can accept an Automatic 1-Way OFX request and produce a response.

The POST identifies the data as being of type *application/x-www-form-urlencoded*. The OFX client will send a request using HTTP POST (over SSL) formatted as an HTML FORM. The form includes client application identification, user ID, password, date range, and account ID fields. When sending the request, the client should omit fields with NULL values.

The following are examples of request and response pairs using this new format. For simplicity, the HTTP Headers are left out of these examples. Responses are indented for readability.

16.5 Examples

For readability purposes, end tags for element tags are not shown in the following examples.

16.5.1 Example - Failed Multiple Statements Request

This is an example where the request does not include an <ACCTID>. The website responds with an "**OFX-STATUS=15509**" indicating that it is unable to return multiple accounts. The client should thereafter send requests for one account at a time.

16.5.1.1 Request

```
APPID=MyOFXApp&APPVER=0600&USERID=MyID&USERPASS=My$ecretPwd&INCTRAN=Y
```

16.5.1.2 Response

OFX-STATUS=15509

16.5.2 Example - MFA Challenge Response

This is an example where the server requires the client to answer <MFACHALLENGERS> questions.

16.5.2.1 Request

APPID=MyOFXApp&APPVER=0600&USERID=MyID&USERPASS=My$ecretPwd&INCTRAN=Y

16.5.2.2 Response

```
<?OFX OFXHEADER="200" VERSION="211" SECURITY="NONE" OLDFILEUID="NONE"
NEWFILEUID="NONE"?>
<OFX>
<SIGNONMSGSRSV1>
<SONRS>
  <STATUS>
    <CODE>3000</CODE>
    <SEVERITY>ERROR</SEVERITY>
  </STATUS>
  <DTSERVER>20050831165153.000[-8:PST]</DTSERVER>
  <LANGUAGE>ENG</LANGUAGE>
</SONRS>
<MFACHALLENGETRNRS>
  <MFACHALLENGERS>
    <MFACHALLENGE>
      <MFAPHRASEID>MFA_4</MFAPHRASEID>
```

```
        <MFAPHRASELABEL>Father's middle name</MFAPHRASELABEL>
      </MFACHALLENGE>
      <MFACHALLENGE>
        <MFAPHRASEID>MFA_12</MFAPHRASEID>
        <MFAPHRASELABEL>Last four digits of your home phone number</
MFAPHRASELABEL>
      </MFACHALLENGE>
      <MFACHALLENGE>
        <MFAPHRASEID>MFA_18</MFAPHRASEID>
        <MFAPHRASELABEL>Name of the company where you had your first
job</MFAPHRASELABEL>
      </MFACHALLENGE>
    </MFACHALLENGE>
</MFACHALLENGETRNRS>
</SIGNONMSGSRSV1>
</OFX>
```

16.5.2.3 Request 2

APPID=MyOFXApp&APPVER=0600&USERID=MyID&USERPASS=My$ecretPwd&DTSTART=2004-08-01&DTEND=2004-08-31&INCTRAN=Y&MFA_4=William&MFA_12=1234&MFA_18=Acme

16.5.2.4 Response 2

OFX-Status=0

16.5.3 Banking and Credit Card Statements

This is an example request asking for statement information from August 1 to August 31, 2005 for all accounts. The customer has a checking, savings, and credit card account, but the savings account has no data within the given timeframe. The financial institution downloads statements in OFX 2.1.1 format and does not return any statements for accounts that have no data within the specified period.

In this example, end tags are omitted for readability purposes.

16.5.3.1 Request

```
APPID=MyOFXApp&APPVER=0600&USERID=MyID&USERPASS=My$ecretPwd&DTSTART=20
05-08-01&DTEND=2005-08-31&INCTRAN=Y
```

16.5.3.2 Response

```
<?xml version="1.0" encoding="USASCII"?>
<?OFX OFXHEADER="200" VERSION="211" SECURITY="NONE" OLDFILEUID="NONE"
NEWFILEUID="NONE"?>
```

```
<OFX>
    <SIGNONMSGSRSV1>
        <SONRS>
            <STATUS>
                <CODE>0</CODE>
                <SEVERITY>INFO</SEVERITY>
            </STATUS>
            <DTSERVER>20050831165153.000[-8:PST]</DTSERVER>
            <LANGUAGE>ENG</LANGUAGE>
        </SONRS>
    </SIGNONMSGSRSV1>
    <BANKMSGSRSV1>
        <STMTTRNRS>
            <TRNUID>0</TRNUID>
            <STATUS>
                <CODE>0</CODE>
                <SEVERITY>INFO</SEVERITY>
            </STATUS>
            <STMTRS>
                <CURDEF>USD</CURDEF>
                <BANKACCTFROM>
                    <BANKID>000000123</BANKID>
                    <ACCTID>123456</ACCTID>
                    <ACCTTYPE>CHECKING</ACCTTYPE>
                </BANKACCTFROM>
                <BANKTRANLIST>
                    <DTSTART>20050801</DTSTART>
                    <DTEND>20050831165153.000[-8:PST]</DTEND>
                    <STMTTRN>
                        <TRNTYPE>POS</TRNTYPE>
                        <DTPOSTED>20050824080000</DTPOSTED>
                        <TRNAMT>-80</TRNAMT>
                        <FITID>219378</FITID>
                        <NAME>FrogKick Scuba Gear</NAME>
                    </STMTTRN>
                </BANKTRANLIST>
                <LEDGERBAL>
                    <BALAMT>2156.56</BALAMT>
                    <DTASOF>20050831165153</DTASOF>
                </LEDGERBAL>
            </STMTRS>
        </STMTTRNRS>
```

```
    </BANKMSGSRSV1>
    <CREDITCARDMSGSRSV1>
        <CCSTMTTRNRS>
            <TRNUID>0</TRNUID>
            <STATUS>
                <CODE>0</CODE>
                <SEVERITY>INFO</SEVERITY>
            </STATUS>
            <CCSTMTRS>
                <CURDEF>USD</CURDEF>
                <CCACCTFROM>
                    <ACCTID>123412341234</ACCTID>
                </CCACCTFROM>
                <BANKTRANLIST>
                    <DTSTART>20050801</DTSTART>
                    <DTEND>20050831165153.000[-8:PST]</DTEND>
                    <STMTTRN>
                        <TRNTYPE>INT</TRNTYPE>
                        <DTPOSTED>20050811080000</DTPOSTED>
                        <TRNAMT>-23.00</TRNAMT>
                        <FITID>219867</FITID>
                        <NAME>Interest Charge</NAME>
                    </STMTTRN>
                    <STMTTRN>
                        <TRNTYPE>CREDIT</TRNTYPE>
                        <DTPOSTED>20050811080000</DTPOSTED>
                        <TRNAMT>350.00</TRNAMT>
                        <FITID>219868</FITID>
                        <NAME>Payment - Thank You</NAME>
                    </STMTTRN>
                </BANKTRANLIST>
                <LEDGERBAL>
                    <BALAMT>-562.00</BALAMT>
                    <DTASOF>20050831165153</DTASOF>
                </LEDGERBAL>
            </CCSTMTRS>
        </CCSTMTTRNRS>
    </CREDITCARDMSGSRSV1>
</OFX>
```

16.5.3.3 Request 2

```
APPID=MyOFXApp&APPVER=0600&USERID=MyID&USERPASS=My$ecretPwd&DTSTART=20
5-08-01&DTEND=20050831&INCTRAN=Y&MFA_4=William&MFA_12=1234&MFA_18=Acme
```

16.5.3.4 Response 2

```
<?xml version="1.0" encoding="US-ASCII"?>
<?OFX OFXHEADER="200" VERSION="211" SECURITY="NONE" OLDFILEUID="NONE"
NEWFILEUID="NONE"?>
<OFX>
    <SIGNONMSGSRSV1>
        <SONRS>
            <STATUS>
                <CODE>0</CODE>
                <SEVERITY>INFO</SEVERITY>
            </STATUS>
            <DTSERVER>20050831165153.000[-8:PST]</DTSERVER>
            <LANGUAGE>ENG</LANGUAGE>
        </SONRS>
    </SIGNONMSGSRSV1>
    <BANKMSGSRSV1>
        <STMTTRNRS>
            <TRNUID>0</TRNUID>
            <STATUS>
                <CODE>0</CODE>
                <SEVERITY>INFO</SEVERITY>
            </STATUS>
            <STMTRS>
                <CURDEF>USD</CURDEF>
                <BANKACCTFROM>
                    <BANKID>000000123</BANKID>
                    <ACCTID>123456</ACCTID>
                    <ACCTTYPE>CHECKING</ACCTTYPE>
                </BANKACCTFROM>
                <BANKTRANLIST>
                    <DTSTART>20040801</DTSTART>
                    <DTEND>20050831165153.000[-8:PST]</DTEND>
                    <STMTTRN>
                        <TRNTYPE>POS</TRNTYPE>
                        <DTPOSTED>20050824080000</DTPOSTED>
                        <TRNAMT>-80</TRNAMT>
                        <FITID>219378</FITID>
                        <NAME>FrogKick Scuba Gear</NAME>
```

```
                </STMTTRN>
            </BANKTRANLIST>
            <LEDGERBAL>
                <BALAMT>2156.56</BALAMT>
                <DTASOF>20050831165153</DTASOF>
            </LEDGERBAL>
        </STMTRS>
    </STMTTRNRS>
</BANKMSGSRSV1>
<CREDITCARDMSGSRSV1>
    <CCSTMTTRNRS>
        <TRNUID>0</TRNUID>
        <STATUS>
            <CODE>0</CODE>
            <SEVERITY>INFO</SEVERITY>
        </STATUS>
        <CCSTMTRS>
            <CURDEF>USD</CURDEF>
            <CCACCTFROM>
                <ACCTID>123412341234</ACCTID>
            </CCACCTFROM>
            <BANKTRANLIST>
                <DTSTART>20050801</DTSTART>
                <DTEND>20050831165153.000[-8:PST]</DTEND>
                <STMTTRN>
                    <TRNTYPE>INT</TRNTYPE>
                    <DTPOSTED>20050811080000</DTPOSTED>
                    <TRNAMT>-23.00</TRNAMT>
                    <FITID>219867</FITID>
                    <NAME>Interest Charge</NAME>
                </STMTTRN>
                <STMTTRN>
                    <TRNTYPE>CREDIT</TRNTYPE>
                    <DTPOSTED>20050811080000</DTPOSTED>
                    <TRNAMT>350.00</TRNAMT>
                    <FITID>219868</FITID>
                    <NAME>Payment - Thank You</NAME>
                </STMTTRN>
            </BANKTRANLIST>
            <LEDGERBAL>
                <BALAMT>-562.00</BALAMT>
                <DTASOF>20050831165153</DTASOF>
```

```
    </SONRS>
</SIGNONMSGSRSV1>
<BANKMSGSRSV1>
    <STMTTRNRS>
        <TRNUID>0
        <STATUS>
            <CODE>0
            <SEVERITY>INFO
        </STATUS>
        <STMTRS>
            <CURDEF>USD
            <BANKACCTFROM>
                <BANKID>000000123
                <ACCTID>123456
                <ACCTTYPE>CHECKING
            </BANKACCTFROM>
            <BANKTRANLIST>
                <DTSTART>20050801
                <DTEND>20050831165056.000[-8:PST]
                <STMTTRN>
                    <TRNTYPE>PAYMENT
                    <DTPOSTED>20050824080000
                    <TRNAMT>-80.32
                    <FITID>219378
                    <CHECKNUM>1044
                    <NAME>FrogKick Scuba Gear
                </STMTTRN>
            </BANKTRANLIST>
            <LEDGERBAL>
                <BALAMT>2156.56
                <DTASOF>20050831165056
            </LEDGERBAL>
        </STMTRS>
    </STMTTRNRS>
    <STMTTRNRS>
        <TRNUID>0
        <STATUS>
            <CODE>0
            <SEVERITY>INFO
        </STATUS>
        <STMTRS>
            <CURDEF>USD
```

```
                <BANKACCTFROM>
                    <BANKID>000000123
                    <ACCTID>654321
                    <ACCTTYPE>SAVINGS
                </BANKACCTFROM>
                <BANKTRANLIST>
                    <DTSTART>20050801
                    <DTEND>20050831165056.000[-8:PST]
                </BANKTRANLIST>
                <LEDGERBAL>
                    <BALAMT>3452.00
                    <DTASOF>20050831165056
                </LEDGERBAL>
            </STMTRS>
        </STMTTRNRS>
    </BANKMSGSRSV1>
</OFX>
```

16.5.5 Example - Banking, Balance only

This is a request for current balance information only for a user who only has one line of credit account with the financial institution. This financial institution website provides LEDGERBAL, AVAILBAL and BALLIST aggregates. The response is returned in OFX 2.1.1 format.

16.5.5.1 Request

```
APPID=MyOFXApp&APPVER=0600&USERID=MyID&USERPASS=My$ecretPwd&ACCTID=123
456-90
```

16.5.5.2 Response

```
<?xml version="1.0" encoding="USASCII"?>
<?OFX OFXHEADER="200" VERSION="211" SECURITY="NONE" OLDFILEUID="NONE"
NEWFILEUID="NONE"?>

<OFX>
    <SIGNONMSGSRSV1>
        <SONRS>
            <STATUS>
                <CODE>0</CODE>
                <SEVERITY>INFO</SEVERITY>
            </STATUS>
            <DTSERVER>20050831165153.000[-8:PST]</DTSERVER>
            <LANGUAGE>ENG</LANGUAGE>
```

```
      </SONRS>
    </SIGNONMSGSRSV1>
    <BANKMSGSRSV1>
      <STMTTRNRS>
        <TRNUID>0</TRNUID>
        <STATUS>
          <CODE>0</CODE>
          <SEVERITY>INFO</SEVERITY>
        </STATUS>
        <STMTRS>
          <CURDEF>USD</CURDEF>
          <BANKACCTFROM>
            <BANKID>000000123</BANKID>
            <ACCTID>123456-90</ACCTID>
            <ACCTTYPE>CREDITLINE</ACCTTYPE>
          </BANKACCTFROM>
          <LEDGERBAL>
            <BALAMT>-4367.00</BALAMT>
            <DTASOF>20050831165153</DTASOF>
          </LEDGERBAL>
          <AVAILBAL>
            <BALAMT>5633.00</BALAMT>
            <DTASOF>20050831165153</DTASOF>
          </AVAILBAL>
          <BALLIST>
            <BAL>
              <NAME>Interest</NAME>
              <DESC> Interest Year To Date</DESC>
              <BALTYPE>DOLLAR</BALTYPE>
              <VALUE>117.89</VALUE>
              <DTASOF>20050831165153</DTASOF>
            </BAL>
          </BALLIST>
        </STMTRS>
      </STMTTRNRS>
    </BANKMSGSRSV1>
</OFX>
```

16.5.6 Investment Statement, Balances and Positions

This is a request for statement, positions and balances for one investment account. The financial institution site requires that their customers log in with an extra User ID; their primary account ID. The response is returned in OFX 1.0.2 format.

16.5.6.1 Request

APPID=MyOFXApp&APPVER=0600&USERID=MyID&USERPASS=My$ecretPwd&CRED2=1256734&DTSTART=20050801&DTEND=20050831&INCTRAN=Y&INCPOS=Y&INCBAL=Y&ACCTID=1256734

16.5.6.2 Response

```
OFXHEADER:100
DATA:OFXSGML
VERSION:102
SECURITY:NONE
ENCODING:USASCII
CHARSET:1252
COMPRESSION:NONE
OLDFILEUID:NONE
NEWFILEUID:NONE

<OFX>
    <SIGNONMSGSRSV1>
        <SONRS>
            <STATUS>
                <CODE>0
                <SEVERITY>INFO
            </STATUS>
            <DTSERVER>20050831165324.000[-8:PST]
            <LANGUAGE>ENG
        </SONRS>
    </SIGNONMSGSRSV1>
    <INVSTMTMSGSRSV1>
        <INVSTMTTRNRS>
            <TRNUID>0
            <STATUS>
                <CODE>0
                <SEVERITY>INFO
            </STATUS>
            <INVSTMTRS>
```

```
                        <DTPRICEASOF>20050831
                        <MEMO>MONEY MARKET FUND
                    </INVPOS>
                </POSMF>
            </INVPOSLIST>
            <INVBAL>
                <AVAILCASH>370.87
                <MARGINBALANCE>0.00
                <SHORTBALANCE>0.00
            </INVBAL>
        </INVSTMTRS>
    </INVSTMTTRNRS>
</INVSTMTMSGSRSV1>
<SECLISTMSGSRSV1>
    <SECLIST>
        <MFINFO>
            <SECINFO>
                <SECID>
                    <UNIQUEID>808515100
                    <UNIQUEIDTYPE>CUSIP
                </SECID>
                <SECNAME>MONEY MARKET FUND
                <TICKER>SWMXX
            </SECINFO>
            <MFTYPE>OPENEND
        </MFINFO>
    </SECLIST>
</SECLISTMSGSRSV1>
</OFX>
```

16.5.6.3 Example - Investment Positions and Balances Only

This is a request for positions and balances only for an investment account. For clarify, the client specifies that this is a retirement account. The response is returned in OFX 2.1.1 format.

16.5.6.4 Request

```
APPID=MyOFXApp&APPVER=0600&USERID=MyID&USERPASS=My$ecretPwd&INCPOS=Y&
INCBAL=Y&ACCTID=12679356&ACCTTYPE=RETIREMENT
```

16.5.6.5 Response

```
<?xml version="1.0" encoding="USASCII"?>
<?OFX OFXHEADER="200" VERSION="211" SECURITY="NONE" OLDFILEUID="NONE"
NEWFILEUID="NONE"?>
```

```
<OFX>
    <SIGNONMSGSRSV1>
        <SONRS>
            <STATUS>
                <CODE>0
                <SEVERITY>INFO
            </STATUS>
            <DTSERVER>20050831165324.000[-8:PST]
            <LANGUAGE>ENG
        </SONRS>
    </SIGNONMSGSRSV1>
    <INVSTMTMSGSRSV1>
        <INVSTMTTRNRS>
            <TRNUID>0
            <STATUS>
                <CODE>0
                <SEVERITY>INFO
            </STATUS>
            <INVSTMTRS>
                <DTASOF>20050831165324.000[-8:PST]
                <CURDEF>USD
                <INVACCTFROM>
                    <BROKERID>Broker Site
                    <ACCTID>1256734
                </INVACCTFROM>
                <INVTRANLIST>
                    <DTSTART>20050101
                    <DTEND>20050831165324.000[-8:PST]
                    <BUYMF>
                        <INVBUY>
                            <INVTRAN>
                                <FITID>40025098199608029240
                                <DTTRADE>20030802
                                <MEMO>MONEY MARKET FUND
                            </INVTRAN>
                            <SECID>
                                <UNIQUEID>808515100
                                <UNIQUEIDTYPE>CUSIP
                            </SECID>
                            <UNITS>120.00
                            <UNITPRICE>1.0000
```

```
                    <TICKER>SWMXX
                </SECINFO>
                <MFTYPE>OPENEND
            </MFINFO>
        </SECLIST>
    </SECLISTMSGSRSV1>
</OFX>
```

APPENDIX A STATUS CODES

The following table provides a complete list of the status codes that can be returned by a server. For each status code, the table includes the following information:

- Number of the status code
- Meaning of the status code
- Conditions under which a server must return the status code

Note: If a server must send an unsupported message to an earlier client, it should include a <MESSAGE> element describing the general error.

Code	*Meaning*	*Condition*
Code	Meaning	Condition
0	Success (INFO)	The server successfully processed the request.
1	Client is up-to-date (INFO)	Based on the client timestamp, the client has the latest information. The response does not supply any additional information.
2000	General error (ERROR)	Error other than those specified by the remaining error codes. **Note:** Servers should provide a more specific error whenever possible. Error code 2000 should be reserved for cases in which a more specific code is not available.
2001	Invalid account (ERROR)	
2002	General account error (ERROR)	Account error not specified by the remaining error codes.
2003	Account not found (ERROR)	The specified account number does not correspond to one of the user's accounts.
2004	Account closed (ERROR)	The specified account number corresponds to an account that has been closed.
2005	Account not authorized (ERROR)	The user is not authorized to perform this action on the account, or the server does not allow this type of action to be performed on the account.
2006	Source account not found (ERROR)	The specified account number does not correspond to one of the user's accounts.
2007	Source account closed (ERROR)	The specified account number corresponds to an account that has been closed.

Code	Meaning	Condition
2008	Source account not authorized (ERROR)	The user is not authorized to perform this action on the account, or the server does not allow this type of action to be performed on the account.
2009	Destination account not found (ERROR)	The specified account number does not correspond to one of the user's accounts.
2010	Destination account closed (ERROR)	The specified account number corresponds to an account that has been closed.
2011	Destination account not authorized (ERROR)	The user is not authorized to perform this action on the account, or the server does not allow this type of action to be performed on the account.
2012	Invalid amount (ERROR)	The specified amount is not valid for this action; for example, the user specified a negative payment amount.
2014	Date too soon (ERROR)	The server cannot process the requested action by the date specified by the user.
2015	Date too far in future (ERROR)	The server cannot accept requests for an action that far in the future.
2016	Transaction already committed (ERROR)	Transaction has entered the processing loop and cannot be modified/cancelled using OFX. The transaction may still be cancelled or modified using other means (for example, a phone call to Customer Service).
2017	Already canceled (ERROR)	The transaction cannot be canceled or modified because it has already been canceled.
2018	Unknown server ID (ERROR)	The specified server ID does not exist or no longer exists.
2019	Duplicate request (ERROR)	A request with this <TRNUID> has already been received and processed.
2020	Invalid date (ERROR)	The specified datetime stamp cannot be parsed; for instance, the datetime stamp specifies 25:00 hours.
2021	Unsupported version (ERROR)	The server does not support the requested version. The version of the message set specified by the client is not supported by this server.
2022	Invalid TAN (ERROR)	The server was unable to validate the TAN sent in the request.

Code	Meaning	Condition
2023	Unknown FITID (ERROR) [BILLID not found (ERROR) in the billing message sets]	The specified FITID/BILLID does not exist or no longer exists.
2025	Branch ID missing (ERROR)	A <BRANCHID> value must be provided in the <BANKACCTFROM> aggregate for this country system, but this field is missing.
2026	Bank name doesn't match bank ID (ERROR)	The value of <BANKNAME> in the <EXTBANKACCTTO> aggregate is inconsistent with the value of <BANKID> in the <BANKACCTTO> aggregate.
2027	Invalid date range (ERROR)	Response for non-overlapping dates, date ranges in the future, et cetera.
2028	Requested element unknown (WARNING)	One or more elements of the request were not recognized by the server or the server (as noted in the FI Profile) does not support the elements. The server executed the element transactions it understood and supported. For example, the request file included private tags in a <PMTRQ> but the server was able to execute the rest of the request.
3000	MFA Challenge authentication required (ERROR)	User credentials are correct, but further authentication required. Client should send <MFACHALLENGERQ>.
3001	MFA Challenge authentication invalid (ERROR)	<MFACHALLENGEA> contains invalid information.
6500	<REJECTIFMISSING>Y invalid without <TOKEN> (ERROR)	This error code may appear in the <SYNCERROR> element of an <xxxSYNCRS> wrapper (in <PRESDLVMSGSRSV1> and V2 message set responses) or the <CODE> contained in any embedded transaction wrappers within a sync response. The corresponding sync request wrapper included <REJECTIFMISSING>Y with <REFRESH>Y or <TOKENONLY>Y, which is illegal.
6501	Embedded transactions in request failed to process: Out of date (WARNING)	<REJECTIFMISSING>Y and embedded transactions appeared in the request sync wrapper and the provided <TOKEN> was out of date. This code should be used in the <SYNCERROR> of the response sync wrapper.

6502	Unable to process embedded transaction due to out-of-date <TOKEN> (ERROR)	Used in response transaction wrapper for embedded transactions when <SYNCERROR>6501 appears in the surrounding sync wrapper.
10000	Stop check in process (INFO)	Stop check is already in process.
10500	Too many checks to process (ERROR)	The stop-payment request <STPCHKRQ> specifies too many checks.
10501	Invalid payee (ERROR)	Payee error not specified by the remaining error codes.
10502	Invalid payee address (ERROR)	Some portpo frhoyr correactfor 6(ou)-(nknob)191(wn.)

1053	va1 i payn	eaccouni numl (ERROR)	Theaccouni numl<RCCT> of(ti)-4.8he requessed-payiRvali

Code	Meaning	Condition
10516	Wire beneficiary invalid (ERROR)	The specified wire beneficiary does not exist or no longer exists.
10517	Invalid payee name (ERROR)	The server does not recognize the specified payee name.
10518	Unknown model ID (ERROR)	The specified model ID does not exist or no longer exists.
10519	Invalid payee list ID (ERROR)	The specified payee list ID does not exist or no longer exists.
10600	Table type not found (ERROR)	The specified table type is not recognized or does not exist.
12250	Investment transaction download not supported (WARN)	The server does not support investment transaction download.
12251	Investment position download not supported (WARN)	The server does not support investment position download.
12252	Investment positions for specified date not available (WARN)	The server does not support investment positions for the specified date.
12253	Investment open order download not supported (WARN)	The server does not support open order download.
12254	Investment balances download not supported (WARN)	The server does not support investment balances download.
12255	401(k) not available for this account (ERROR)	401(k) information requested from a non-401(k) account.
12500	One or more securities not found (ERROR)	The server could not find the requested securities.
13000	User ID & password will be sent out-of-band (INFO)	The server will send the user ID and password via postal mail, e-mail, or another means. The accompanying message will provide details.
13500	Unable to enroll user (ERROR)	The server could not enroll the user.
13501	User already enrolled (ERROR)	The server has already enrolled the user.
13502	Invalid service (ERROR)	The server does not support the service <SVC> specified in the service-activation request.
13503	Cannot change user information (ERROR)	The server does not support the <CHGUSERINFORQ> request.

Code	Meaning	Condition
13504	<FI> Missing or Invalid in <SONRQ> (ERROR)	The FI requires the client to provide the <FI> aggregate in the <SONRQ> request, but either none was provided, or the one provided was invalid.
14500	1099 forms not available (ERROR)	1099 forms are not yet available for the tax year requested.
14501	1099 forms not available for user ID (ERROR)	This user does not have any 1099 forms available.
14600	W2 forms not available (ERROR)	W2 forms are not yet available for the tax year requested.
14601	W2 forms not available for user ID (ERROR)	The user does not have any W2 forms available.
14700	1098 forms not available (ERROR)	1098 forms are not yet available for the tax year requested.
14701	1098 forms not available for user ID (ERROR)	The user does not have any 1098 forms available.
15000	Must change USERPASS (INFO)	The user must change his or her <USERPASS> number as part of the next OFX request.
15500	Signon invalid (ERROR)	The user cannot signon because he or she entered an invalid user ID or password.
15501	Customer account already in use (ERROR)	The server allows only one connection at a time, and another user is already signed on. Please try again later.
15502	USERPASS lockout (ERROR)	The server has received too many failed signon attempts for this user. Please call the FI's technical support number.
15503	Could not change USERPASS (ERROR)	The server does not support the <PINCHRQ> request.
15504	Could not provide random data (ERROR)	The server could not generate random data as requested by the <CHALLENGERQ>.
15505	Country system not supported (ERROR)	The server does not support the country specified in the <COUNTRY> field of the <SONRQ> aggregate.
15506	Empty signon not supported (ERROR)	The server does not support signons not accompanied by some other transaction.
15507	Signon invalid without supporting pin change request (ERROR)	The OFX block associated with the signon does not contain a pin change request and should.

15508	Transaction not authorized. (ERROR)	Current user is not authorized to perform this action on behalf of the <USERID>.
15509	Multiple accounts not supported (ERROR)	1-Way server does not support returning multiple accounts in response.
15510	CLIENT UID invalid (ERROR)	CLIENTUID error. User must register the Client UID.
15511	User contact required (ERROR)	User should contact financial institution
15512	<AUTHTOKEN> required (ERROR)	User needs to contact financial institution to obtain AUTHTOKEN. Client should send it in the next request.
15513	<AUTHTOKEN> invalid (ERROR)	<AUTHTOKEN> sent in signon request was invalid
16500	HTML not allowed (ERROR)	The server does not accept HTML formatting in the request.
16501	Unknown mail To: (ERROR)	The server was unable to send mail to the specified Internet address.
16502	Invalid URL (ERROR)	The server could not parse the URL.
16503	Unable to get URL (ERROR)	The server was unable to retrieve the information at this URL (e.g., an HTTP 400 or 500 series error).
16510	Image not available yet (ERROR)	The image is not available yet; try again later.
16511	Image Expired (ERROR)	The image was available, but is no longer available (has expired).
16512	IMAGEREF not found (ERROR)	Image Reference Id

APPENDIX B SUGGESTED NAME VALUES FOR THE BANKING <BALLIST>

Suggested values for the <NAME> and <DESC> fields of the <BAL> aggregates used in bank and credit card statement download are shown below. Of course any values may be used in these fields; however, to promote consistency, it is suggested that these strings be used when applicable. This list will be expanded to cover as many unique different balance types as necessary

B.1 Usage Field Values

The Usage field contains abbreviations for the type of account each balance should be applied to. The values can be as follows:

	Abbreviation
Credit Card	CC
Checking	CH
Savings	SA
Loan / Mortgage / Credit Line	LMC
TD / CD	TD

B.2 Name and Description Table

Note: In the table below the <NAME> values are currently the same as the <DESC> values. Ideally the <NAME> value should be as descriptive as possible while staying within the 32 character limit. If a longer string needs to be conveyed to the user to describe the type of balance, the longer <DESC> field should be used.

<NAME> Values	*<DESC> Values*	*Usage*
Amount Deposited This Month	Amount Deposited This Month	CH, SA, LMC, CD
Amount Overpaid	Amount Overpaid	CC
Amount Withdrawn This Month	Amount Withdrawn This Month	CH, SA, LMC
Available Credit Amount	Available Credit Amount	CC, CH, LMC
Available Credit Financing	Available Credit Financing	CC
Available Credit Installment	Available Credit Installment	CC

<NAME> Values	<DESC> Values	Usage
Available Instant Cash	Available Instant Cash	CH
Cash Advance Amount	Cash Advance Amount	CC
Cash Outstanding	Cash Outstanding	CC
Commission Fee	Commission Fee	LMC
Credit Line Amount	Credit Line Amount	CC, CH, LMC
Credit Line Financing	Credit Line Financing	CC
Credit Line Installment	Credit Line Installment	CC
Credit Line Used	Credit Line Used	CC, LMC
Current Statement Balance	Current Statement Balance	CC
Deposit Year To Date	Deposit Year To Date	CH, SA, TD
Deposits Last Year	Deposits Last Year	CH, SA, TD
Equivalent Principal Amount	Equivalent Principal Amount	SA, TD
Federal Tax	Federal Tax	CH, SA, TD
Interest Due Amount	Interest Due Amount	LMC, TD
Interest Earned This Period	Interest Earned This Period	CH, SA
Interest Paid Last Year	Interest Paid Last Year	CH, SA
Interest Paid Year To Date	Interest Paid Year To Date	CH, SA, TD
Interest Penalty Amount	Interest Penalty Amount	LMC
Interest Since Last Statement	Interest Since Last Statement	LMC
Interest Value at Maturity	Interest Value at Maturity	TD
Last Bill Balance	Last Bill Balance	CC
Last Deposit Amount	Last Deposit Amount	CH, SA, TD
Late Charges	Late Charges	LMC
Latest Payment Amount	Latest Payment Amount	CC, LMC
Maturity Value	Maturity Value	SA, TD
Minimum Amount Due	Minimum Amount Due	CC, LMC
Monthly Payment	Monthly Payment	SA, LMC
Net To Close Amount	Net To Close Amount	LMC
Next Payment Amount	Next Payment Amount	CC, LMC
Next Payment Credit Life	Next Payment Credit Life	LMC

<NAME> Values	<DESC> Values	Usage
Next Payment Escrow	Next Payment Escrow	LMC
Next Payment Principal	Next Payment Principal	LMC
New Charges	New Charges	CC
New Payments Or Credits	New Payments Or Credits	CC
Notice Amount	Notice Amount	SA
Oldest Installment Due Amount	Oldest Installment Due Amount	LMC
Original Principal Amount	Original Principal Amount	SA, LMC, TD
Overdraft Coverage Used	Overdraft Coverage Used	CH, SA, LMC
Overdraft Limit	Overdraft Limit	CH, SA
Paid Principal	Paid Principal	LMC
Past Due Amount	Past Due Amount	CC, LMC
Pending Credits	Pending Credits	CH, SA
Pending Debits	Pending Debits	CH, SA
Previous Statement Balance	Previous Statement Balance	CC
Total Amount Due Now	Total Amount Due Now	CC, LMC
Total Amount Owed	Total Amount Owed	CC, LMC
Total Available Balance	Total Available Balance	CH
Total Security Value	Total Security Value	LMC
Total To Date Interest	Total To Date Interest	CH, SA, LMC, TD
Unbilled Charges	Unbilled Charges	CC
Unbilled Loan Amount	Unbilled Loan Amount	CC
Uncollected Amount	Uncollected Amount	LMC
Unpaid Balance	Unpaid Balance	CC, CH, LMC
Unpaid Principal	Unpaid Principal	LMC

APPENDIX C CHANGE HISTORY

C.1 OFX 1.6 to 2.0

This appendix describes the revisions made to Open Financial Exchange that occurred between version 1.6 and version 2.0. Major changes include:

1) The Data Formatting standard has changed from that of SGML to XML.

2) V2 message sets have been removed.

3) 401(k) support has been added to Investments.

4) A new Tax OFX Addendum which adds support for 1099 and W2 download has been added and is available separately bound from this document.

Note that some detailed changes are not shown below if they are part of one of the changes above and thus are identified that way. 401(k) changes have been itemized in full, however.

C.1.1 Specification Changes by Chapter

Location	*Subject*	*Change Type*	*Change*
Global	SGML->XML	Change	The Data Formatting standard has changed from that of SGML to XML.
Global	End tags required	Change	Ends tags are now required. All examples were changed as well as textual references to end tags.
Global	Header changed	Change	The OFX header has changed from that of an SGML-type header to a standard XML declaration syntax. Examples were changed as appropriate.
Global	V2 eliminated	Change	V2 tags and message sets have been eliminated. <SRVRTID2>, <TOKEN2>, <MESSAGE2> and <URL2> are among those tags that have been deleted.
Global	"1.6 add" tags eliminated	Change	All "1.6 add" tags were eliminated except those in the bill presentment message set and <REFRESHSUPT> in <MSGSETCORE>
2.3	Special character handling	Change	Special characters are predefined in XML.
3.1.2.1	Handling of user values	Clarification	This section was rewritten to clarify rules for handling user values by a client or server.

Location	Subject	Change Type	Change
6.4	Token and Synchronization Summary	Addition	New section added to clarify rules for token and synchronization handling.
11.7	Immediate Transfers	Clarification	Clarified handling of immediate transfers which are batched and done that night or next day.
11.12.2	INTRASYNCRQ/S	Addition	CCACCTFROM aggregate added
11.12.3	INTERSYNCRQ/S	Addition	CCACCTFROM aggregate added
11.12.5	RECINTRASYNCRQ/S	Addition	CCACCTFROM aggregate added
11.12.6	RECINTERSYNCRQ/S	Addition	CCACCTFROM aggregate added
12.2.2	scope of PAYEELSTID	Clarification	Clarified handling of PAYEELSTID scoping
12.9.2.1	<PAYEEMODRQ>	Change	Optional tag <MODPENDING> was removed.
12.9.2.2	<PAYEEMODRS>	Change	Optional tag <MODPENDING> was removed.
13.3.3	change to signage rules	Change	PENALTY, WITHHOLDING and STATEWITHHOLDING were added to list of elements affecting signage of units and totals.
13.6.3.1	New 401(k) section	Addition	Entitled "401(k) Accounts"
13.6.3.2	New section on download detail	Addition	Entitled, "Note on Downloading Positions and Transaction Detail for Investment Accounts"
13.7.1.1	New profile tag	Addition	New tag <INV401KDNLD>
13.8.4	SECLIST return info	Clarification	better description of when to return <SECLIST>
13.9.1.2	new <INVSTMTRQ> tags	Addition	<INC401K> and <INC401KBAL>
13.9.2.2	New <INVSTMTRS> aggregates	Addition	<INV401K> and <INV401KBAL>
13.9.2.4	New elements added to verbiage on transactions	Addition	PENALTY, WITHHOLDING and STATEWITHOLDING included in text
13.9.2.4	New elements added to computation of total	Addition	PENALTY, WITHHOLDING and STATEWITHOLDING included in text
13.9.2.4.2	New elements added	Addition	DTPAYROLL, INV401KSOURCE, LOANINTEREST, LOANID, LOANPRINCIPAL, PENALTY, PRIORYEARCONTRIB, STATEWITHHOLDING
13.9.2.4.2	Total description changed	Change	calculation changed
13.9.2.4.2	WITHHOLDING description changed	Change	Indicates that this element is for federal taxes only.
13.9.2.4.3	401(k) verbiage added	Addition	Describes funding loans or other withdrawals

Location	Subject	Change Type	Change
13.9.2.4.3	Some elements added	Change	New elements added to INVBUY and INVSELL
13.9.2.4.4	New element added	Change	<INV401KSOURCE> added to INCOME, INVEXPENSE, REINVEST, RETOFCAP, SPLIT and TRANSFER aggregates
13.9.2.6.1	New element added	Change	<INV401KSOURCE> added to INVPOS aggregate
13.9.2.7	</BAL> no longer bolded	Correction	Corrected mismatch between beginning/ending tags
13.9.2.8	New section	Addition	Entitled, "401(k) Balances <INV401KBAL>"
13.9.2.9	Status code added to table	Addition	Error status code 12255 added
13.9.3	New section	Addition	Entitled "401(k) Account Information"
13.11	New section (example)	Addition	Entitled "Complete 401(k) Example"
Appendix A	Status code 12254	Addition	Investment Balances Download not supported (WARN)
Appendix A	Status code 14500	Addition	1099 forms not available (ERROR)
Appendix A	Status code 14501	Addition	1099 forms not available for user ID (ERROR)
Appendix A	Status code 14600	Addition	W2 forms not available (ERROR)
Appendix A	Status code 14601	Addition	W2 forms not available for user ID (ERROR)

C.2 OFX 2.0 to 2.0.1

This appendix describes the revisions made to Open Financial Exchange that occurred between version 2.0 and version 2.0.1. Major changes include:

1) Reversals to investments transactions.

2) Extended balance information in banking and credit card statements.

3) Bill publisher information in payments.

C.2.1 Specification Changes by Chapter

Section	Subject	Change Type	Change
2.2	file format	Correction	Changed wording to not require separate lines for headers and also to make some xml declaration items optional
2.5.2	pin change and error recovery	Clarification	Paragraph 3 (regarding status code 15000) clarified
3.2.9.2	signage in amounts	correction/ clarification	Added explanatory text for when amounts are signed and how
3.2.9.2	investment reversals	Deletion	Removed sentences referring to how to do investment reversals since REVERSALFITID is now a reversal option in investments.
6.6.1	<LOSTSYNC>	Correction	Changed <LOSTSYNC>N to <LOSTSYNC>Y in 3 places.
6.10.2	Lite Synch	Clarification	Reworded because confusing
6.10.2.1	Lite Synch and Older Clients/ Servers	Clarification	Reworded because confusing
8.8	Signup profile information	Clarification	Slight change to clarify what is required in the Signup profile response.
10.3.1	models and sync behavior	Addition	New section clarifying proper behavior for models
11.4.1 11.4.2 13.9.1.2	statement download request (banking, credit cards and investments)	Clarification	Clarification to second paragraph: what to do if transactions are requested and none are available.
11.4.1.2 11.4.2.2	statement response aggregate	Addition, DTD Change	Added <BALLIST> to <STMTRS> and <CCSTMTRS>.
11.4.3	<SRVRTID> in <STMTTRN>	Clarification	The use of <SRVRTID> in <STMTTRN> was clarified
11.7	immediate transfers	Clarification	Immediate transfers should appear in the sync if error recovery is not supported.
11.7.2.2	<INTRAMODRS> introduction	Correction	Servers may, but are not required to, notify clients of changes in transfer status.
11.13.1	Message Sets and Messages	Clarification	Fixed format of tables to be clearer.

Section	Subject	Change Type	Change
11.14.1	Example	Correction	The <FI> aggregate was added to the signon response.
11.14.4	Example	Correction	The <SRVRTID> value in a <RECINTRARS> was changed to be the same as the <RECSRVRTID>.
12.2.2	shared accounts	Clarification	More verbiage about shared accounts and <PAYEELSTID>s
12.3	shared accounts	Clarification	More verbiage about shared accounts and <PAYEELSTID>s
12.5.2	removed sentence from paragraph 1	Correction	It implied that <PAYEE> or <PAYEEID> is optional when one of the other is required.
12.5.2	new tag in <PMTINFO> aggregate	Addition, DTD change	Added <BILLPUBINFO> aggregate.
12.9.1.2	<PAYEERS>	Clarification/ Correction	Paragraph 2: corrected (<PAYEELSTID> changed to <PAYEEID>) + small clarification
12.11.1	Message Sets and Messages	Clarification	Fixed format of table to be clearer.
12.11.2	bill pay profile	Addition, Correction (no DTD change)	Added <DIFFLASTPMT> tag to profile
12.11.2	bill pay profile	Addition, DTD change	Added <BILLPUBCONTEXT> to profile
12.11.2	bill pay profile	Clarification:	The use of <PROCENDTM> in the profile was clarified.
12.12	bill pay examples	Corrections	fixed some bugs in bill pay examples
13.7.1.2	Message Sets and Messages	Clarification	Fixed format of table to be clearer.
13.7.2.2	Message Sets and Messages	Clarification	Fixed format of table to be clearer.
13.3.3	Signs	Clarification, Deletion	Deleted section where it talks about how to do reversals since REVERSALFITID is included as an option now.
13.9.2.4.1	another reversal option added	Addition, DTD change	Added <REVERSALFITID> to <INVTRAN>
13.9.2.4.1.1	investment reversals	Addition	New section: Transaction Reversals
13.9.2.4.4	correction to tables	Correction	"<INV401KSOURCE aggregate" was changed to "<INV401KSOURCE>" since it is not an aggregate.

Section	Subject	Change Type	Change
13.9.2.6.1	<DTPRICEASOF> clarification	Correction	Removed text that said this datetime value could be zero if unknown.
14.7.1.2	Message Sets and Messages	Clarification	Fixed format of table to be clearer.
Appendix A	Added error codes	Addition	14700 and 14701 (for 1098 tax)
Appendix B	New Appendix	Addition	Suggested banking <BALLIST> values
Appendix C, "Diff. Between OFX 1.6 and 2.0"	Added <CCACCTFROM> change	Correction	In OFX 2.0, CCACCTFROM was added to all transfer sync RQ/RS's but this was mistakenly left out of the 1.6 -> 2.0 change appendix.

C.3 OFX 2.0.1 to 2.0.2

C.3.1 Specification Changes by Chapter

Section	Subject	Change Type	Change
2.4.6.1	Tag ordering fixed	Correction	<STATUS> and <CLTCOOKIE> were transposed in the "typical response"
2.4.6.1	Text unbolded	Correction	The text "Response aggregate" was unbolded in the "typical response" since a response aggregate is not required
3.2.2	Joint Accounts	Clarification	Clarified that <SRVRTID>s should be unique across users of joint accounts
3.2.7	<DTSTART>/<DTEND> usage	Clarification	Clarified the meaning of <DTSTART> and <DTEND>
3.2.8.1	Timezone name not required	Clarification	Clarification that timezone name is not required in a timezone
3.2.8.2	Example added	Addition	Example added to clarify difference between date and datetime in <DTSTART>, <DTEND>
6.4.3	Addition to table of synchronizable types	Addition	Added LOANSYNCRQ/S to table
7.2	Typo	Correction	Word in <SIGNONINFO> description changed from "profile" to "provide"
9.4.2	New tables	Addition	Requests/Responses and transaction wrapper tables added
12.7.3.3	Addition to Status Code Table	Addition	2017 ("Already Cancelled") added to <RECPMTCANCRS> table
12.11.2	<PMTBYPAYEEID> verbiage fixed	Correction	Changed words "user supplied" to "server supplied"
12.11.2	<STSVIAMODS> type missing	Correction	Added *Boolean*
13.3.3	Signage in investment transactions	Addition	Sentence changed to clarify signage of <UNITS> and <TOTAL>
13.6.3.2	Note on downloading positions and transaction detail	Correction	The Note in this section was changed to refer to all accounts, not just 401K accounts

Section	Subject	Change Type	Change
13.9.2.4.1.1	Removed sentence	Deletion	Removed a sentence because MARGININTEREST should be reversed using INVBANKTRAN, not INVEXPENSE (since there is no security involved)
13.9.2.5.1	<OO><UNITS> type change	Correction	Changed to *Quantity*
13.9.2.6.1	Value of <DTPRICEASOF>	Clarification	If this date is unknown, use 19900101 as the placeholder; do not use 0
13.11	Signage of <TOTAL> in examples	Correction	Changed the signage of <TOTAL> from positive to negative for <BUYSTOCK> and <BUYMF> aggregates
13.11	Closing tag fixed in example	Correction	A closing tag for <DTSTART> was changed to </DTSTART>
14.7.3	<CANUPDATEPRESNAMEADDRESS> type missing	Correction	Added *Boolean*

C.3.2 DTD Changes

These are the corrections to the DTD files to ensure that the DTD files match the specification. They are corrections only and so do not constitute changes to the standard itself. The changes are:

1) The spelling of ANUALLY was changed to ANNUALLY.

2) ANUALLY was changed to ANNUAL and SEMIANNUALLY was changed to SEMIANNUAL in the COUPONFREQ enumeration type.

3) The ordering of tags INV401KDNLDand CANEMAIL was reversed; CANEMAIL should come first.

C.4 OFX 2.0.2 to 2.1

This appendix describes the revisions made to Open Financial Exchange that occurred between version 2.0.2 and version 2.1. Major changes include:

1) The addition of Loan statement, closing, and amortization schedule download plus loan email

2) The addition of image download in banking, credit card, investment and loan message sets (new chapter).

3) The addition of Automatic 1-Way OFX functionality (new chapter).

4) Addition of investment closing request/response to facilitate image download of closing statement in Investment Message Set.

C.4.1 Specification Changes by Chapter

Section	Subject	Change Type	Change
Global	OFX header	Change	OFX header: value of VERSION changed from "202" to "210" throughout
Global	Using Social Security Number as UserID	Change	Removed references to SSN from examples and text to discourage use of SSNs for identification purposes.
Global	Dates in examples	Change	An attempt was made to update examples with more recent dates (specifically, year 2005).
1.1.1	Added references to loans, images and 1-Way OFX	Addition	Added 3 bullets under *Broad Range of Financial Activities*
1.1.1	Changed wording	Change	Removed OS-specific references in *Platform Independent* bullet
1.2.3	Automatic 1-Way OFX	Addition	New section to discuss 1-Way OFX
1.4	OFX Versions	Addition	Added reference to OFX 2.1
2.2.2	VERSION	Change	Updated this section for 2.1
2.4.5.2, 2.4.5.3, 2.8.4	New Message sets	Addition	Added references to Loan and Image message sets
2.5.1, 2.5.1.1	Using SSNs for UserID/ Password	Addition	Added verbiage to discourage Financial Institutions from using Social Security Number for Customer ID or PIN/Password.
3.1.6	New Images section	Addition	Added new common aggregates <IMAGEDATA> and <IMAGEPROF>
6.4.3	Synchronizable objects table	Addition	Added <LOANMAILSYNCRQ/S>
7.1.1.1	New Section	Addition	Profile Support for Downloading Images
8.4.1	Using SSNs for UserID/ password	Addition	Verbiage added to discourage Financial Institutions from using Social Security Number for Customer ID or PIN/Password.
8.4.2	Certain tags unbolded	Change	Some <ENROLLRQ> tags changed to optional
8.5	<DTACCTUP> in requests	Clarification	Clarified rules around value of <DTACCTUP> in <ACCTINFORQ>
9.2.5.1	Correction (typo)	Correction	Changed <MSGBODY>> to <MSGBODY>
11.1.1	Loan Data	Addition	Loan data paragraph added

Section	Subject	Change Type	Change
11.3.5	Changes to <XFERINFO> aggregate	Addition	Added <LOANACCTFROM>, <LOANACCTTO> and <LOANTRNAMT> to <XFERINFO>. New verbiage after table.
11.3.7 - 11.3.9	New Sections for Loan definitions	Addition	<LOANACCTFROM>, <LOANACCTINFO>, etc. (all loan related)
11.4.1	Added verbiage about image download of transactions	Addition	Paragraph added at end of section
11.4.1.1	New tag in <STMTRQ>	Addition	Added <INCTRANIMG> to <STMTRQ>
11.4.3	New tag and aggregate in <STMTTRN>	Addition	Added <EXTDNAME> and <IMAGEDATA> to <STMTTRN>
11.4.4 - 11.4.6	New sections for Loan Stmt Download	Addition	<LOANSTMTRQ>, <LOANSTMTTRN>, etc. added
11.5	Added verbiage about image download of statements	Addition	Paragraph added at end of section
11.5.1.1	New tag in <STMTENDRQ>	Addition	Added <INCSTMTIMG>
11.5.2	New aggregate in <CLOSING>	Addition	Added <IMAGEDATA>
11.5.3	New tag in <CCSTMTENDRQ>	Addition	Added <INCSTMTIMG>
11.5.4.2	New aggregate in <CCCLOSING>	Addition	Added <IMAGEDATA>
11.5.5 - 11.5.8	New sections for Loan Stmt Closing	Addition	<LOANSTMTENDRQ> and related aggregates added
11.11.4	New section for Loan e-mail	Addition	<LOANMAILRQ> and related aggregates
11.12.2, 11.12.3	Added LOANACCTFROM	Addition	Added LOANACCTFROM to all transfer sync requests/responses
11.12.2 11.12.3.2 11.12.5.2	What transfers to synchronize	Clarification	Clariified note after table about synchronizing only on <xxxACCTFROM> account
11.12.8	LOANMAILSYNC	Addition	New LOANMAILSYNCRQ/S aggregates
11.13	Loan support verbiage	Addition	Added bullet describing new loan support
11.13.1.5	Loan RQ/RS tables added	Addition	New tables showing loan requests/responses
11.13.2.1	New aggregate in profile for banking	Addition	<IMAGEPROF> added

Section	Subject	Change Type	Change
11.13.2.2	New tags in <XFERPROF>	Addition	<CANLOAN>, <CANSCHEDLOAN> and <CANRECURLOAN>
11.13.2.4	Changed description	Clarification	Clarified meaning of <CANNOTIFY>
11.13.3	New aggregate in profile for credit cards	Addition	<IMAGEPROF> added
11.13.6	New loan profile	Addition	Added <LOANMSGSET>
11.14.5	New examples	Addition	Examples showing various Loan requests/ responses
11.14.6	New examples	Addition	Example showing Image support
13.1	New bullet	Addition	Added bullet for image download
13.7.1.1	New aggregate in profile for images	Addition	<IMAGEPROF> added
13.7.1.1	New tag in profile for closing	Addition	<CLOSINGAVAIL> added
13.7.1.2	Added new closing request/ response to table	Addition	INVSTMTENDTRNRQ/S added
13.9.1	Added verbiage about image download of statements	Addition	Paragraph added at end of section
13.9.1.2	Added verbiage for images	Addition	Paragraph added just before table
13.9.1.2	New tag in <INVSTMTRQ>	Addition	Added <INCTRANIMG>
13.9.2.4.4	REINVEST description	Clarification	Changed *INVTRAN* to *INVTRANLIST transaction* in definition of REINVEST
13.10	New Closing rq/rsp	Addition	New section for investment closing
15	New Images Chapter	Addition	New chapter describing Image Message Set
16	New 1-Way Chapter	Addition	New chapter describing 1-Way OFX
Appendix A	Status code 15509 for 1-Way	Addition	Multiple accounts not supported (ERROR)
Appendix A	Status code 16510 for Images	Addition	Image not available yet; try again later (ERROR)
Appendix A	Status code 16511 for Images	Addition	Image has expired; not longer available (ERROR)
Appendix A	Status code 16512 for Images	Addition	Image Reference Identifier not found (ERROR)
Appendix A	Status code 16513 for Images	Addition	Image server is not available; try again later (ERROR)

C.5 OFX 2.1 to 2.1.1

This appendix describes the changes made to Open Financial Exchange between version 2.1 and version 2.1.1 The major change is the introduction of Multi-Factor Authentication.

C.5.1 Specification Changes by Chapter

Section	Subject	Change Type	Change
Global	OFX header	Change	OFX header: value of VERSION changed from "210" to "211" throughout
2.5	MFA	Addition	Added verbiage stating that MFA is added to Signon message set functionality
2.5.1	Preventing identity fraud	Change	Added verbiage discouraging use of SSN for USERID or USERPASS.
2.5.1.1	New Section	Addition	New section entitled *Multi-Factor Authentication*
2.5.1.2	New tags in <SONRQ> for MFA	Addition	New tags added to Signon Request <SONRQ>: *CLIENTUID, USERCRED1, USERCRED2, AUTHTOKEN, ACCESSKEY, MFACHALLENGEA*
2.5.1.3	New tag in <SONRS> for MFA	Addition	New tag added to Signon Response <SONRS>: *ACCESSKEY*
2.5.1.4	New error codes	Addition	New MFA error codes added to <SONRS> Status Code table
2.5.4	New OFX request/response	Addition	New section added for MFA request/response <MFACHALLENGERQ>, <MFACHALLENGERS>
2.5.6	New examples	Addition	Added examples for MFA
3.1.5	New entries in status code table	Addition	Added some error codes for MFA to <SONRS> status code table
7.2.2	Signon realms	Addition	New tags added to <SIGNONINFO> aggregate: *USERCRED1LABEL, USERCRED2LABEL, CLIENTUIDREQ, AUTHTOKENFIRST, AUTHTOKENLABEL, AUTHTOKENINFOURL, MFACHALLENGESUPT, MFACHALLENGEFIRST*

Section	Subject	Change Type	Change
7.2.2	<CHGPINFIRST> description	Change	Text was added to state that if the profile tag MFACHALLENGEFIRST is 'Y' the pin change request should be sent immediately after the session containing MFACHALLNEGE authentication.
10.2.1	Adjusting due date of a payment	Clarification	Models are free to adjust the due date of a payment if this date falls on a non-processing day.
16.1	Added tags to Auto 1-Way Request for MFA	Addition	Added CLIENTUID and MFA_XXX tags
16.1.1	<CLIENTUID> text	Addition	New text describing use of CLIENTUID
16.1.2	New section for MFA	Addition	New section "Challenge Question Answers"
16.2.2	Status code table	Addition	New status codes for MFA
16.2.3	MFA Response description	Addition	New section "Challenge Response"
16.5.2	Example for MFA	Addition	Added example showing use of MFA challenge questions/answers
Appendix A	New Status code 3000	Addition	User credentials are correct, but further authentication required. Client should send MFACHALLENGERQ (ERROR)
Appendix A	New Status code 3001	Addition	MFACHALLENGEA contains invalid information (ERROR)
Appendix A	New Status code 15510	Addition	CLIENTUID error. User must register the Client UID (ERROR)
Appendix A	New Status code 15511	Addition	User should contact financial institution (ERROR)
Appendix A	New Status code 15512	Addition	User needs to contact financial institution to obtain AUTHTOKEN. Client should send it in the next request (ERROR)
Appendix A	New Status code 15513	Addition	AUTHTOKEN invalid (ERROR)

TAG INDEX

Conventions

This index uses the conventions shown in the following table to identify different entry types.

	Text Style
Tag definition	**Bold text**
Tag shown in example	*Italic text*
Tag used within body of text	Plain text

Symbols

61

A

ACCESSKEY **49**, 50, 51
ACCRDINT **457**, 462, 465
ACCTBAL **528**, 529
ACCTEDITMASK 507, **508**, 509, *560*, *562*, *564*
ACCTFORMAT 507, **508**, 509, *560*, *561*, *564*
ACCTID *22*, *126*, *127*, *148*, *154*, *177*, *178*, *182*, **186**, 188, 190, 206, 213, 214, 233, *316*, *317*, *319*, *320*, *321*, *323*, *324*, *328*, *329*, *330*, *332*, *333*, *408*, *409*, *410*, *411*, *412*, *413*, *414*, *415*, *418*, *419*, *420*, *421*, *425*, *426*, 431, *490*, *491*, *496*, *497*, 515, *567*, *568*, *570*, *571*, *573*, *577*, 600
ACCTINFO 145, 146, **147**, *148*, 432, 519, *576*, *577*
ACCTINFORQ 146, **146**, *148*, 513, 518, **518**, 518, 519, 520, *576*
ACCTINFORS 145, 146, **146**, 146, *148*, 431, 513, 514, 518, **519**, 519, 520, 521, *576*, *577*, **580**
ACCTINFOTRNRQ 146, *148*, 513, 518
ACCTINFOTRNRS 146, *148*
ACCTKEY **187**, 188, 190, 206, 213, 214, 216
ACCTREQUIRED 158
ACCTRQ 149, **149**, 153, *154*, 158, 514, 515, 516, *567*
ACCTRS 151, **151**, 153, *154*, 518, *568*
ACCTSYNCRQ 114, 153, **153**, 518
ACCTSYNCRS 114, 153, **153**
ACCTTRNRQ 149, *154*, *567*
ACCTTRNRS 151, *154*, *568*
ACCTTYPE *22*, *126*, *127*, *148*, *154*, *177*, *178*, *182*, 187, 188, **189**, *316*, *317*, *319*, *320*, *321*, *323*, *324*, *325*, *328*, *329*, *330*, *332*, *333*, *408*, *409*, *410*, *411*, *412*, *413*, *414*, *415*, *418*, *419*, *420*, *421*, *425*, *426*, 600
ACTIVITY **528**, 529
ADDR1 133, 142, *144*, 155, 156, 259, 354, *408*, *409*, *423*, *424*, 506, 507, 508, 516, *559*, *560*, *561*, *564*, *565*, *567*, *568*
ADDR2 133, 142, 155, 156, 259, 354, *423*, *424*, 506, 507, 508, 516, *561*
ADDR3 133, 142, 155, 156, 259, 354, 506, 507, 508, 516
ADJAMT **357**, 357
ADJDATE **357**, 357
ADJDESC **357**, 357
ADJNO **357**, 357
ADJUSTMENT 356, **357**, 357
AFTERTAX **476**, 483, *500*
AFTERTAXCONTRIBAMT **479**
AFTERTAXCONTRIBPCT **479**
AMRTAVAIL 315
AMRTSMTRQ *337*
AMRTSMTRS *337*
AMRTSMTTRN *337*
AMRTSTMTRQ **219**
AMRTSTMTRS **219**
AMRTSTMTTRN 219, **221**
AMRTTRANLIST 219
AMRTTYPE **221**
AMTDUE **528**, *570*, *571*, *573*
APPID *22*, **50**, *316*, *558*, *562*, *565*, 583, 599
APPVER *22*, 30, **50**, *316*, *558*, *562*, *565*, 583, 599
ASSETCLASS **446**, **447**, 448, 449, *494*, *495*
AUCTION **470**
AUTHTOKEN **48**, 50

BILLTBLSTRUCTTRNRQ 541
BILLTBLSTRUCTTRNRS 541
BILLTYPE **523**, 528
BODY *167*, *168*, *169*
BPACCTINFO **351**, 351
BRANCHID **186**, 188, 625
BRAND 512, **512**, *560*, *562*, *564*
BROKERCONTACTINFO **477**
BROKERID **431**, *490*, *491*, *496*, *497*
BUSNAMEACCTHELD **516**
BUYDEBT **462**
BUYMF **462**
BUYOPT **462**
BUYOTHER 431, **462**
BUYPOWER **474**
BUYSTOCK **462**, *491*
BUYTYPE **457**, 462, 470, *492*, *494*

C

C 538, **538**, 541, *570*, *571*, *574*
CALLPRICE **446**
CALLTYPE **446**
CANADDPAYEE **407**
CANBILLPAY **313**
CANCELWND **313**
CANEMAIL 312, 315, **435**, 557
CANLOAN 311
CANMODMDLS 311, **407**
CANMODPMTS **407**
CANMODSTATUS **557**
CANMODXFERS **584**
CANNOTIFY **312**, 557
CANPENDING *180*, **270**, 270, 277, *328*, *329*, 379, 380, *422*, *423*
CANRECUR **311**, **584**
CANRECURLOAN 311
CANSCHED **311**, 311, 314, 315
CANSCHEDLOAN 311
CANSUPPORTGROUPID **556**
CANSUPPORTIMAGES **556**, 557
CANUPDATEPRESNAMEADDRESS 517, **557**
CANUSEDESC **311**
CANUSERANGE **311**
CASESEN **136**
CASHBAL **476**
CCACCTFROM 80, 186, **190**, 190, 192, 193, 201, 206, 207, 213, 214, 216, 225, 226, 229, 233, 243, 252, 280, 281, 289, 290, 291, 292, 294, 295, 296, 297, 298, 299
CCACCTINFO **192**
CCACCTTO **190**, 190, 193, 211, 217
CCCLOSING 226, **226**, 226, 227
CCSTMTENDRQ 201, **225**, 225, 229
CCSTMTENDRS **226**, 226
CCSTMTENDTRNRQ 225
CCSTMTENDTRNRS 226
CCSTMTRQ 205, **206**, 206, 216, 227
CCSTMTRS **207**, 207
CCSTMTTRNRQ 206
CCSTMTTRNRS 207
CHALLENGERQ **56**, 56, 100, 102, 628
CHALLENGERS **56**, 56, 100, 102
CHALLENGETRNRQ 56
CHALLENGETRNRS 56
CHARTYPE **136**
CHECKING **432**
CHECKNUM 210, 216, 236, 237, 283, *318*, *322*, **361**, 369, *417*
CHECKSUP **80**, *588*
CHGPINFIRST **137**
CHGUSERINFO **158**
CHGUSERINFORQ 155, **155**, 516, 517, 627
CHGUSERINFORS **156**
CHGUSERINFOSYNCRQ 114, 155, **157**
CHGUSERINFOSYNCRS 114, 155, **157**
CHGUSERINFOTRNRQ 155, 157
CHGUSERINFOTRNRS 155, 157
CHKANDDEB **224**
CHKDESC 235, **236**, 236
CHKERROR **237**
CHKMAILRS **283**, 283
CHKNUMEND **235**, *321*
CHKNUMSTART **235**, *321*
CHKRANGE 235, **235**, 235, *321*
CHKSTATUS **237**, *322*
CITY 133, 142, *144*, 155, 156, 259, 354, *408*, *409*, *423*, *424*, 506, 508, 516, *559*, *560*, *561*, *564*, *565*, *567*, *568*
CLIENTACTREQ **158**
CLIENTENROLL 158, **158**
CLIENTROUTING **132**
CLIENTUID **48**, 50, 599
CLIENTUIDREQ **137**

CLOSING 223, **224**, 224, 484
CLOSINGAVAIL **310**, 312, 315
CLOSINGIMGAVAIL **81**, *596*
CLOSUREOPT **462**
CLTCOOKIE 43, 45, 442, 443, 451, 453, 520, 521, 535
CODE *37*, 45, 52, 57, *62*, **78**, 79, **81**, *144*, *148*, *154*, *168*, *169*, *171*, *182*, 240, 250, *317*, *320*, *321*, *324*, *329*, *330*, *332*, 360, *409*, *411*, *413*, *414*, *416*, *417*, *418*, *420*, *422*, *424*, *426*, *491*, *497*, 526, *559*, *563*, *566*, *568*, *569*, *573*, *576*, 625
COLDEF 541, **542**, 542
COLNAME **542**
COLTYPE **542**
COMMISSION 455, **457**, 460, 461, 464, *492*
CONFMSG **261**
CONSUPOSTALCODE 506, **506**
CONTRIBINFO **478**, *500*
CONTRIBSECURITY **478**, *500*
CONTRIBUTIONS 480, 481, **482**, *500*
CORRECTACTION **210**, 210, 216
CORRECTFITID *84*, **210**, 216
COUNT **526**, 527
COUNTRY 133, 142, *144*, 155, 156, 259, 354, 506, 508, 516, *559*, *560*, *561*, *564*, *565*, *567*, *568*, 628
COUPONFREQ **446**
COUPONRT **446**
CRED2 583, 599
CRED3 583, 599
CREDITCARDMSGSET **312**
CREDITCARDMSGSETV1 42, **312**
CREDITCARDMSGSRQV1 305
CREDITCARDMSGSRSV1 305
CREDITLIMIT **227**
CSPHONE **134**
CURDEF **105**, 105, 106, 204, 207, 214, 219, 223, 226, 229, 236, 241, 250, 261, *317*, *320*, *321*, *324*, *332*, **361**, 373, *409*, *411*, *419*, *426*, 453, 484, 485, *491*, *497*
CURRATE 105, **106**
CURRENCY 77, **105**, 105, **106**, 106, 211, 217, 225, 227, 232, 237, 445, 460, 461, 463, 464, 465, 466, 469, 472
CURRENTLOANBAL **480**
CURRENTVESTPCT **479**
CURSYM 106, **106**

D

DATEBIRTH **142**, *144*, *565*
DAYPHONE **142**, *144*, 155, 156, 516, *565*, *567*, *568*
DAYSTOPAY **358**, 363, 406, *410*, *411*, *424*, *426*
DAYSWITH 311, **406**, 406
DEBADJ **227**
DEBTCLASS **446**
DEBTINFO 444, **446**, 446
DEBTTYPE **446**
DEFERPCTAFTERTAX **478**
DEFERPCTPRETAX **478**
DEND 213
DENOMINATOR 466
DEPANDCREDIT **224**
DEPMAILRS **284**, 284
DESC **77**, **80**, 147, *148*, *493*
DETAILAVAILABLE 527, **529**, *572*
DFLTDAYSTOPAY 311, **406**, 406, 585
DFLTIMAGEDELAY **584**
DFLTIMAGETTL **584**
DIFFFIRSTPMT **407**
DIFFLASTPMT **407**
DISCOUNT **356**, 356
DOMXFERFEE **313**, 314, 315
DSCAMT **356**, 356
DSCDATE **356**, 356
DSCDESC **356**, 356
DSCRATE **356**, 356
DTACCTUP 51, 145, **146**, 146, *148*, *317*, 518, 519, 520, *559*, *563*, *566*, *576*, *577*, 580
DTASOF 77, 199, 200, 204, 207, 214, 233, *318*, **445**, 451, 452, 453, *491*, *493*, *497*
DTAUCTION **470**
DTAVAIL 210
DTBILL **528**, 528, *570*, *571*, *573*
DTCALL **446**
DTCHANGED **55**
DTCLIENT *22*, **49**, *316*, *558*, *562*, *565*
DTCLOSE **224**, 227, 231, 528
DTCOUPON **446**
DTCREATED **162**, *167*, *168*, *169*
DTDUE *177*, *178*, **193**, 193, 258, 261, *324*, *325*, 353, 375, 406, *408*, *409*, *410*, *411*, *412*, *413*, *414*, *415*, *418*, *419*, *420*, *421*, *426*
DTDUEBY **523**, 523

DTEFF **530**, 531
DTEND **86**, 86, 201, 203, 204, 206, 207, 214, 219, 223, 224, 225, 227, 229, 316, *318*, 451, 452, 453, 484, *491*, 522, 523, 526, *570*, *572*, *573*, 599
DTEXPIRE 143, *144*, **448**, *495*, **533**, *566*, *570*, *571*, *573*
DTIMAGEAVAIL **80**
DTINFOCHG **156**
DTLOANMATURITY **197**
DTLOANSTART **197**
DTMAT **446**
DTNEXT **224**, 227, 231
DTOPEN **224**, 227, 231, 528
DTPAYROLL 460, *498*
DTPLACED **469**, *494*
DTPMTDUE 90, **200**, **227**, 528, *570*, *571*, *573*
DTPMTPRC **358**, 364, *410*, *412*, *413*, *415*, *417*, *426*
DTPOSTED **210**, 216, 241, 250, 261, *318*, *492*
DTPOSTEND 224, **225**, 227, 232
DTPOSTSTART 224, **225**, 227, 232
DTPRICEASOF **472**, *493*
DTPROFUP 51, **132**, 133, *317*, *559*, *563*, *566*
DTPURCHASE **457**, 466
DTSEEN **543**, 543
DTSERVER **51**, *317*, *559*, *563*, *566*
DTSETTLE **456**, *491*
DTSTART **86**, 86, 201, 203, 204, 206, 207, 213, 214, 219, 223, 224, 225, 227, 229, 316, *317*, 451, 452, 453, 484, *490*, *491*, *496*, 522, 523, 526, *569*, *570*, *572*, *573*, *575*, 599
DTTRADE **456**, *491*
DTUPDATE 505, **506**, 507, *559*, *563*
DTUSER **210**, 216, 236, 237, 283, 284, *318*, *492*
DTXFERPRC **195**, 195, **196**, **199**, **200**, *331*, *332*
DTXFERPRJ 241, **241**, 241, 250, 261, *320*, *333*
DTYIELDASOF **447**, **449**
DURATION **469**, *494*

E

EARNINGS 481, **483**
EMAIL **134**, **142**, *144*, 155, 156, *565*
EMAILMSGSET 173, **174**
EMAILMSGSETV1 42, **174**
EMAILPROF 310, **312**, 312, 557
EMPLOYERCONTACTINFO **477**
EMPLOYERNAME **477**, *499*
ENROLLRQ 47, **141**, **142**, *144*, 150, 155, 513, 516, *565*
ENROLLRS **142**, **143**, *144*, 513, *566*
ENROLLTRNRQ 142, *144*, *565*
ENROLLTRNRS 143, *144*, *566*
ESCRWAMT 200, 201, **201**
ESCRWBAL **233**
ESCRWFEES **201**
ESCRWFEESBAL **233**
ESCRWINS **201**
ESCRWINSBAL **233**
ESCRWOTHER **201**
ESCRWOTHERBAL **233**
ESCRWPMI **201**
ESCRWPMIBAL **233**
ESCRWTAX **201**
ESCRWTAXBAL **233**
ESCRWTOTAL **201**
ESTPAYOFF 231, **233**
ESTPAYOFFBAL **233**
EVEPHONE **142**, *144*, 155, 156, 516, *565*
EXTBANKACCTTO 625
EXTBANKDESC 258, **259**, 259, 261
EXTDNAME 210, 216
EXTDPAYEE 350, **358**, 358, 361, 363, 373, 388, 391, 406, *409*, *411*, *424*, *426*
EXTDPMT 343, 352, **354**, 354
EXTDPMTCHK **354**
EXTDPMTDSC 354, **354**
EXTDPMTFOR **354**
EXTDPMTINFO 352
EXTDPMTINV 354, 355, **356**, 356, 357

F

FAXPHONE **134**
FEE **236**, 260, **261**, **283**, 284, *322*
FEEMSG **236**, *322*
FEES 455, **457**, 460, 461, 464
FI *22*, 50, 51, **53**, 53, *316*, *317*, 527, *558*, *562*
FIASSETCLASS **446**, **447**, 448, 449
FICERTID 56, **56**, 56
FID *22*, **53**, *316*, *317*, *558*, *562*, 583, 599

FIID 442, **442**, **445**, *494*, *495*
FIMFASSETCLASS **447**
FINALAMT 372, **373**, 376, 377
FINAME **133**
FINCHG **227**
FINDBILLERRQ 47, 504, **505**, 505, 506, 558, *558*, *563*
FINDBILLERRS 504, **507**, 507, *559*, *563*
FINDBILLERTRNRQ 505, *558*, *563*
FINDBILLERTRNRS 507, 509, *559*, *563*
FIPORTION **447**
FIRSTNAME **142**, *144*, 155, 156, *565*
FITID 83, **83**, 83, *84*, 112, 209, 210, 216, 224, 226, 227, 231, *318*, 450, 456, 469, 485, *491*, *492*, *494*
FRACCASH **457**, 466
FREQ 176, **176**, *177*, *178*, *323*, *324*, 375, *418*, *419*, *420*, *421*, 626
FROM **162**, *167*, *168*, *169*

G

GAIN **457**, 461, 462
GENUSERKEY 47, **50**
GETMIMERQ 70, 170, **170**, 170, *171*
GETMIMERS 70, *70*, 170, **170**, 170, *172*
GETMIMESUP **174**
GETMIMETRNRQ *171*
GETMIMETRNRS *171*
GROUPID 519, 520, **520**, 526, 534, 535, **535**, 535, *575*, *576*

H

HASEXTDPMT **407**
HELDINACCT **472**, *492*, *493*
HELPMESSAGE 507, **508**, *561*
HTML *167*, *168*, *169*, *172*

I

IDSCOPE 358, **358**, *410*, *411*, *424*, *426*
IMAGEDATA **79**, 211, 217, 225, 227, 232, 485, 580, 582, 583, *588*
IMAGEDELAY **80**
IMAGEMSGSET 584, *596*
IMAGEPROF **81**, 310, 312, 315, 435, *596*
IMAGEREF **80**, 580, 583, *588*, *589*
IMAGEREFTYPE 80, 580, 582, 583, *588*
IMAGERQ **580**, *589*
IMAGETTL **80**
IMAGETYPE **79**, *588*
IMAGEURL 533, **533**, *570*, *571*, *573*
INC401K **452**, *496*
INC401KBAL **452**, *496*
INCBAL **452**, *490*, 600
INCEPTODATE **481**
INCIMAGES 162, **163**, 163, 166, *167*, *168*, *169*, 298, 300, 384, 488, 506, 550, *558*, *563*
INCLUDE *22*, 203, **203**, 206, 213, *316*, 452, *490*, *496*
INCLUDEBILLPMTSTATUS **524**, 524, 525
INCLUDEBILLSTATUS **524**, 524, 525
INCLUDECOUNTS *525*, 526
INCLUDEDETAIL **524**, 525, *569*, *572*, *575*
INCLUDESTATUSHIST **524**, 525
INCLUDESUMMARY **525**, *525*
INCOME **463**
INCOMETYPE **457**, 463, 464
INCOO **452**, *490*
INCPOS 451, **452**, *490*, *496*, 600
INCSTMTIMG 223, 225, 229, 484, *590*
INCTRAN *22*, 203, **203**, 206, 213, 214, 216, *316*, **452**, *490*, *496*
INCTRANIMG 203, 213, 452, *586*
INITIALAMT 372, **373**, 376, 377
INITIALLOANBAL **480**
INSURANCE 201
INTAMT **201**
INTERCANRQ **255**, 255
INTERCANRS **255**, 255
INTERMODRQ **252**, 252
INTERMODRS **253**, 253
INTERRQ **249**, 249, 272, 274
INTERRS **250**, 250, 272, 275
INTERSYNCRQ 114, **291**, 291, 296
INTERSYNCRS 114, **292**, 292
INTERTRNRQ 249, 252, 255, **291**
INTERTRNRS 250, 253, 255, **292**
INTERXFERMSGSET **313**
INTERXFERMSGSETV1 42, **313**
INTERXFERMSGSRQV1 306
INTERXFERMSGSRSV1 307

INTLXFERFEE **313**, 314, 315
INTRACANRQ **246**, 246
INTRACANRS **246**, 246
INTRAMODRQ **243**, 243
INTRAMODRS **244**, 244, *330*, *332*
INTRARQ **240**, 240, 265, 267, *319*, *323*
INTRARS **240**, 241, 265, 268, *320*, *324*, *332*
INTRASYNCRQ 114, **289**, 289, 294
INTRASYNCRS 114, **290**, 290, 292
INTRATRNRQ *41*, 240, 243, 246, **289**, *319*
INTRATRNRS 241, 244, 246, 290, *319*, *330*, *331*
INV401K 454, **477**, *499*
INV401KBAL 454, **476**, *499*
INV401KDNLD 435
INV401KSOURCE **457**, 460, 461, 463, 464, 465, 466, 469, 472, *498*
INV401KSUMMARY **480**, *500*
INVACCTFROM 142, 150, 431, **431**, 431, 432, 451, 453, 466, 484, 486, 487, 488, 489, *490*, *491*, *496*, *497*
INVACCTINFO 431, **432**, 432
INVACCTTO 150, **431**
INVACCTTYPE **432**
INVALIDACCTTYPE **310**
INVBAL 454, **474**, 474, *493*
INVBANKTRAN 454, **455**, 455, *492*
INVBUY **459**, 462, *491*
INVCLOSING 484, **485**
INVDATE **356**, 356
INVDESC **356**, 356
INVEXPENSE **463**
INVMAILRQ **486**, 486, 550
INVMAILRS 487, **487**, **551**
INVMAILSYNCRQ 114, 486, **488**, 488
INVMAILSYNCRS 114, 486, **489**, 489
INVMAILTRNRQ **488**
INVMAILTRNRS **489**
INVNO **356**, 356
INVOICE 355, **356**, 356, 528
INVOOLIST **454**, *494*
INVPAIDAMT **356**, 356
INVPOS **472**, 472, 473, *492*, *493*
INVPOSLIST 431, **454**, *492*
INVSELL **459**, 465
INVSTMTENDRQ **484**
INVSTMTENDRS **484**
INVSTMTMSGSET **435**, 435
INVSTMTMSGSETV1 42, 434
INVSTMTMSGSRQV1 436, *490*, *496*
INVSTMTMSGSRSV1 437, *491*, *497*
INVSTMTRQ 451, **451**, 451, *490*, *496*
INVSTMTRS 453, **453**, 453, *491*, *497*
INVSTMTTRNRQ **451**, 451, *490*, *496*
INVSTMTTRNRS **453**, 453, *491*, *497*
INVTOTALAMT **356**, 356
INVTRAN **456**, 456, 460, 461, 462, 463, 464, 465, 466, *491*
INVTRANLIST **453**, *491*, *497*

J

JRNLFUND **463**
JRNLSEC **464**

L

LANGUAGE *22*, **50**, 51, 135, *316*, *317*, *558*, *559*, *562*, *563*, *565*, *566*
LASTNAME **142**, *144*, 155, 156, *565*
LEDGERBAL 105, **204**, 207, *318*
LIMITPRICE **469**, *494*
LINEITEM 356, **357**, 357
LITMAMT **357**, 357
LITMDESC **357**, 357
LOAD 455, **457**, 460, 461, 464
Loan Closing **231**
LOANACCTFROM 193, **196**, 197, 213, 214, 219, 229, 285, 289, 290, 291, 292, 300, 301
LOANACCTID **196**, 196
LOANACCTINFO **197**, *333*
LOANACCTTO 193, **196**, 217
LOANACCTTYPE 196, **196**
LOANCLOSING 229, **231**, *341*
LOANDESC **480**
LOANFREQ 197, **198**
LOANID **457**, 460, 461, **480**, *498*
LOANINFO **480**
LOANINITBAL 197, 221
LOANINITNUMPMTS **197**
LOANINT 197, **199**, 231
LOANINTEREST **457**, 460, *498*
LOANINTERESTTODATE **480**

LOANINTLTD **199**
LOANINTPRJ **199**
LOANINTRATE **199**
LOANINTYTD **199**
LOANIRATE 197, **199**, 221, 231
LOANMAILRQ **285**
LOANMAILRS **285**
LOANMAILSYNCRQ **300**
LOANMAILSYNCRS **301**
LOANMAILTRNRQ 300
LOANMAILTRNRS 301
LOANMATURITYDATE **480**
LOANMSGSET **315**
LOANMSGSETV1 **315**
LOANNEXTPMTDATE **480**
LOANPMT 197, **200**, 231
LOANPMTAMT **480**
LOANPMTFREQ **480**
LOANPMTSINITIAL **480**
LOANPMTSREMAINING **480**
LOANPMTTYPE **200**
LOANPRINCIPAL **457**, 460, *498*
LOANRATE **480**
LOANRMNPMTS **197**, 231
LOANSMTRQ *334*
LOANSMTRS *335*
LOANSMTTRN *335*
LOANSTARTDATE **480**
LOANSTMTENDRQ **229**, *341*
LOANSTMTENDRS 229, **229**, *341*
LOANSTMTRQ **213**
LOANSTMTRS **213**
LOANSTMTTRN 214, 216, **216**
LOANTOTALPROJINTEREST **480**
LOANTRANLIST **214**
LOANTRNAMT 193, **201**, 216, 221
LOANTRNTYPE 216, **218**
LOANTYPE **197**
LOGO **508**, *560*, *561*, *562*, *564*
LOSTSYNC 112, 118, *126*, *127*, **153**, 157, 167, 288, 290, 292, 293, 295, 297, 299, 301, 385, 396, 399, 401, 489, 551

M

MAIL 162, **162**, 164, *167*, *168*, *169*, 280, 281, 283, 284, 285, 381, 382, 486, 487, 548
MAILRQ **164**, 164, *167*, 174
MAILRS **164**, 164, 166, *168*, *169*
MAILSUP **174**
MAILSYNCRQ 114, 164, **166**, 166, *168*, 174
MAILSYNCRS 114, 164, **166**, 167, *168*
MAILTRNRQ 164, **166**, *167*
MAILTRNRS 164, **167**, *168*
MARGINBALANCE **474**, *493*
MARGININTEREST **464**
MARKDOWN **457**, 461
MARKUP **458**, 460
MATCH **476**, 482, 483, *501*
MATCHCONTRIBAMT **479**
MATCHCONTRIBPCT **479**
MATCHINFO **478**, *499*
MATCHPCT **478**, *499*
MAX 47, **136**
MAXMATCHAMT **478**
MAXMATCHPCT **478**
MEMO **85**, 85, *177*, *178*, 211, 217, 259, 353, *408*, *409*, *410*, *411*, *412*, *413*, *414*, *415*, *418*, *419*, *420*, *421*, *426*, 445, 456, 469, 472, *492*, *493*
MEMO2 **85**
MESSAGE 37, *38*, **78**, 79, 143, **158**, 164
MFA_XXX 599
MFACHALLENGE **58**
MFACHALLENGEA 50, **61**
MFACHALLENGEFIRST **137**
MFACHALLENGERQ **58**, 58
MFACHALLENGERS **58**
MFACHALLENGESUPT **137**
MFAPHRASEA **61**
MFAPHRASEID 58, **59**, 61
MFAPHRASELABEL **58**
MFASSETCLASS **447**
MFINFO 444, **447**, 447
MFTYPE **447**
MIDDLENAME **142**, *144*, 155, 156, *565*
MIN 30, **136**
MINAMTDUE **528**
MINPMTDUE **227**
MINUNITS **469**
MKTGINFO **205**, 208, 214, 219, 225, 227, 232, 454
MKTVAL **472**, *493*
MODELWND 311, 370, **406**

MODPENDING **267**, **268**, 274, 275, 375, 376, 377, *420*, *421*
MSGBODY **162**, *167*, *168*, *169*
MSGSETCORE 43, 61, 97, 99, 130, 134, 135, **135**, 135, 136, 138, 158, 174, 310, 312, 313, 314, 315, 406, 435, 438, 556
MSGSETLIST 61, **133**, 138, 158, 173

N

N 538, **538**
NAME 77, 79, 210, 211, 216, 236, 237, 259, 344, 354, 358, 364, 375, *408*, *409*, *410*, *411*, *423*, *424*, *426*, *492*, *493*, 506, 507, 515, *559*, *560*, *561*, *563*
NAMEACCTHELD **516**, *567*, *568*
NEWUNITS **458**, 466

PAYEERQ 347, 349, 363, **387**, 387, *423*
PAYEERS 347, 364, 375, **388**, 388, *424*
PAYEESYNCRQ 114, **153**, 153, **157**, 157, 350, **395**, 395
PAYEESYNCRS 114, **153**, 153, **157**, 157, 346, 363, **396**, 396
PAYEETRNRQ 39, 387, 389, 393, **395**, *423*
PAYEETRNRS 350, 388, 391, 394, **396**, *424*
PAYINSTRUCT **258**, **261**
PAYMENTINSTRUMENT 511, **512**, 512, *559*, *560*, *561*, *562*, *564*
PAYMENTINSTRUMENTS 508, 511, **511**, 511, 512, *559*, *560*, *561*, *564*
PENALTY 461
PERCENT **447**, 447
PERIODTODATE **481**
PHONE 147, *148*, 259, 354, *408*, *409*, *423*, *424*, 508, *559*, *560*, *561*, *564*
PINCH **137**
PINCHRQ **54**, 54, *62*, 102, 104, 122, 137, 628
PINCHRS **54**, **55**, 55, 58, 61, *62*
PINCHTRNRQ *62*
PINCHTRNRS *62*
PLANID **477**, *499*
PLANJOINDATE **477**, *499*
PMTAMT **200**
PMTBYADDR **406**
PMTBYPAYEEID **407**
PMTBYXFER **407**
PMTCANCRQ 39, 350, 359, **367**, 367, *415*
PMTCANCRS *182*, 350, 367, **368**, 368, *416*
PMTINFO *177*, *178*, **352**, 352, 360, 361, 363, 364, 365, 372, 373, 375, 376, 377, 381, 382, *408*, *409*, *410*, *411*, *412*, *413*, *414*, *415*, *418*, *419*, *420*, *421*, *426*, 546
PMTINQRQ 39, 349, 350, **369**, 369, *416*
PMTINQRS **369**, 369, *417*
PMTINQTRNRQ 39, 369, *416*
PMTINQTRNRS 369, *417*
PMTINSTRUMENTTYPE 512, **512**, 512, *560*, *561*, *562*, *564*
PMTMAILRQ **381**, 381
PMTMAILRS **382**, 382
PMTMAILSYNCRQ 114, 381, **384**, 384, **548**
PMTMAILSYNCRS 114, **385**, 385
PMTMAILTRNRQ 381, **384**
PMTMAILTRNRS 382, **385**
PMTMODRQ 39, 343, 346, 347, 349, 359, **364**, 364, 365, *412*, *414*
PMTMODRS 349, 350, 364, **365**, 365, 367, 375, 397, *413*, *415*
PMTNUMBER **221**
PMTPRCCODE 358, 360, **364**, *410*, *412*, *413*, *415*, *417*, *426*
PMTPRCSTS **358**, 358, 360, **361**, 365, 367, 369, *410*, *411*, *413*, *415*, *417*, *426*
PMTRQ 39, *127*, 346, 347, 349, 350, **360**, 360, 363, 367, 387, 388, 391, *408*, *410*
PMTRS *126*, *127*, 349, 350, 352, 360, **360**, 360, **361**, 363, 364, 397, *409*, *411*, *425*, *426*
PMTSYNCRQ 113, 114, *126*, *127*, *182*, 350, **398**, 398, 402, *425*
PMTSYNCRS 114, *126*, *127*, *182*, 349, 350, 367, **399**, 399, 402, *425*
PMTTRNRQ 39, *127*, 360, 365, 367, **398**, *408*, *410*, *412*, *414*, *415*
PMTTRNRS *126*, *127*, *182*, 360, 365, 368, **399**, *409*, *411*, *413*, *414*, *416*, *426*
PORTION **447**
POSDEBT **473**
POSDNLD **435**
POSMF **473**
POSOPT **473**, *493*
POSOTHER 431, **473**
POSSTOCK **473**, *492*
POSTALCODE 133, 142, *144*, 155, 156, 259, 354, *408*, *409*, *423*, *424*, 506, 508, 516, *559*, *560*, *561*, *564*, *565*, *567*, *568*
POSTPROCWND **406**
POSTYPE **458**, 466, **472**, *492*, *493*
PREFETCHURL 533, **533**
PRESACCTFROM 507, 514, **514**, 514, 515, 516, 528, 534, 537, 543, 548, 550, 551, *570*, *571*, *573*, *577*
PRESACCTINFO 513, **514**, 514, 515, 519, 520, 546, *577*
PRESACCTTO 507, **514**, 514, 516, 517, *567*, *568*
PRESBILLINFO 515, 525, 527, **527**, 527, 528, 534, 536, 537, 542, 546, *570*, *571*, *573*
PRESCOUNTS 525, **526**
PRESDELIVERYID 543, **543**, 543, 544
PRESDETAIL **537**
PRESDETAILRQ 522, 527, 536, **536**, 536, 537
PRESDETAILRS **537**, 537

PRESDETAILTRNRQ 536
PRESDETAILTRNRS 537, 540
PRESDIRMSGSET 552, 553, **556**
PRESDIRMSGSETV1 42, 504, 552, 553, **556**
PRESDIRMSGSRQV1 504, 552, *558*, *563*
PRESDIRMSGSRSV1 504, 553, *559*, *563*
PRESDIRPROF **556**
PRESDLVMSGSET 552, **556**
PRESDLVMSGSETV1 42, 504, 522, **556**
PRESDLVMSGSRQV1 522, 554, *569*, *572*, *575*, *576*
PRESDLVMSGSRSV1 522, 555, *569*, *573*, *576*, 625
PRESDLVPROF **556**
PRESGRPACCTINFOTRNRQ 518, **519**, 519, 520, *576*
PRESGRPACCTINFOTRNRS 519, **520**, 520, **521**, *576*
PRESLIST 526, **527**, 527, *570*, *573*
PRESLISTRQ 522, **522**, 523, 526, 527, 533, 534, 535, 542, 546, *569*, *572*, *575*
PRESLISTRS 522, **526**, 526, 534, 535, *570*, *573*
PRESLISTTRNRQ 522, 523, 526, **534**, 534, 535, *569*, *572*, *575*
PRESLISTTRNRS 526, 534, **535**, 535, 536, *569*, *573*
PRESMAILRQ **548**, 548
PRESMAILRS **548**, 548
PRESMAILSYNCRQ 114, 547, **550**, 550
PRESMAILSYNCRS 114, 547, **551**, 551

114. 56 (40)Ti /F3 1373 10.

RECPMTRQ *177*, 346, 347, **372**, 372, *418*
RECPMTRS *83*, *178*, 352, **373**, 373, 375, *418*
RECPMTSYNCRQ 114, **400**, 400, 402
RECPMTSYNCRS 114, **401**, 401, 402
RECPMTTRNRQ 372, 376, 379, **400**, *418*, *419*, *422*
RECPMTTRNRS 373, 377, 380, **401**, *418*, *420*, *422*
RECSRVRTID *83*, 178, *178*, 180, *180*, 241, 250, 265, 267, 268, 270, 272, 274, 275, *277*, *324*, *328*, *329*, *333*, 347, **361**, 373, 376, 377, 379, 380, *418*, *420*, *421*, *422*
RECURRINST **176**, 176, *177*, *178*, 265, 267, 268, 272, 274, 275, *323*, *324*, 372, 373, 375, 376, 377, *418*, *419*, *420*, *421*
REFNUM **210**, 216, 250
REFRESH 116, 117, **117**, 117, 124, 135, 153, 157, 166, 287, 289, 291, 293, 294, 296, 298, 300, 384, 395, 397, 398, 400, 488, 550, 625
REFRESHSUPT **135**, 135
REINVCG **473**
REINVDIV **473**, 473
REINVEST **464**
REJECTIFMISSING 113, 116, **117**, *126*, *127*, 153, 157, 166, *168*, *182*, 287, 289, 291, 293, 294, 296, 298, 300, *328*, 384, 395, 398, 400, *425*, 488, 550, 625
RELFITID *84*, **458**, 462, 465
RELTYPE **458**, 465
RESPFILEER **135**
RESTRICT **508**, *561*
RESTRICTION **469**, *494*
RETOFCAP **465**
REVERSALFITID *84*, **456**, 456
ROLLOVER **476**, 482, 483, *499*
ROLLOVERCONTRIBAMT **479**
ROLLOVERCONTRIBPCT **479**, *500*

S

SECID **441**, 441, 442, **442**, 444, 445, 448, 460, 461, 462, 463, 464, 465, 466, 469, 470, 472, *492*, *493*, *494*, *495*
SECINFO **445**, 445, 446, 447, 448, 449, *494*, *495*
SECLIST 440, 443, **444**, 444, *494*
SECLISTMSGSET **438**, 438
SECLISTMSGSETV1 42, **438**
SECLISTMSGSRQV1 40, 439, 440, 444
SECLISTMSGSRSV1 40, 440, 444, *494*
SECLISTRQ 442, **442**, 442, 443
SECLISTRQDNLD **438**
SECLISTRS 443, **443**, 443
SECLISTTRNRQ **442**, 442, 444
SECLISTTRNRS **443**, 443
SECNAME **445**, *494*, *495*
SECRQ **442**
SECURED **458**, 465, 473
SECURITYNAME **142**, *144*, *565*
SELLALL **470**
SELLDEBT **465**
SELLMF **465**
SELLOPT **465**
SELLOTHER 431, **465**
SELLREASON **458**, 465
SELLSTOCK **465**
SELLTYPE **458**, 465, 470
SESSCOOKIE 47, **50**, 51
SEVERITY 37, *38*, 43, *62*, 78, **78**, **81**, *144*, *148*, *154*, *168*, *169*, *172*, *182*, *317*, *320*, *321*, *324*, *329*, *330*, *332*, *409*, *411*, *413*, *414*, *416*, *417*, *418*, *420*, *422*, *424*, *426*, *491*, *497*, *559*, *563*, *566*, *568*, *569*, *573*, *576*
SHORTBALANCE **474**, *493*
SHPERCTRCT **448**, **458**, 462, 465, *495*
SIC 210, **506**, 508, *559*, *560*, *561*, *564*
SIGNONINFO 47, 133, **136**, 137
SIGNONINFOLIST **133**
SIGNONMSGSET 61, **61**, 131
SIGNONMSGSETV1 42, **61**, 130
SIGNONMSGSRQV1 *22*, *40*, *316*, *318*, *320*, *323*, *325*, *326*, *328*, *329*, *408*, *410*, *412*, *414*, *415*, *416*, *417*, *419*, *422*, *423*, *425*, *490*, *496*, *558*, *562*, *565*, *567*, *569*, *572*, *575*, *576*
SIGNONMSGSRSV1 *317*, *319*, *321*, *324*, *328*, *330*, *409*, *411*, *413*, *414*, *416*, *417*, *418*, *420*, *422*, *424*, *425*, *491*, *497*, *559*, *563*, *566*, *568*, *569*, *573*, *576*
SIGNONREALM 135, **136**
SIGNUPMSGSET **158**
SIGNUPMSGSETV1 42, **158**
SIGNUPMSGSRQV1 40, 139, *565*, *567*
SIGNUPMSGSRSV1 40, *566*, *568*

SONRQ *22*, *40*, **46**, 46, 47, **49**, 49, 51, 54, 97, 102, 104, 123, *126*, 131, 141, 167, *316*, *319*, *320*, *323*, *325*, *326*, *328*, *329*, *408*, *410*, *412*, *414*, *415*, *416*, *417*, *419*, *422*, *423*, *424*, *425*, *490*, *491*, *496*, *497*, 504, 514, 520, 534, 535, *558*, *562*, *565*, *567*, *569*, *572*, *575*, *576*, 628
SONRS **46**, 46, 47, 50, **51**, 51, 124, *317*, *319*, *321*, *324*, *328*, *330*, *409*, *411*, *413*, *414*, *416*, *417*, *418*, *420*, *424*, *425*, *491*, *497*, *559*, *563*, *566*, *567*, *568*, *569*, *572*, *573*, *575*, *576*
SPACES **136**
SPECIAL **136**
SPLIT **466**
SPNAME 51, **135**, 348, 514, 515
SRVRTID **82**, 82, *83*, 112, 116, 120, *182*, 210, 216, 241, 243, 244, 246, 250, 252, 253, 255, 261, 262, 263, 265, 272, *320*, *324*, *330*, *332*, 347, 349, 350, **361**, 365, 367, 368, 369, 381, 382, 402, 403, *409*, *411*, *412*, *413*, *414*, *415*, *416*, *417*, *426*, 456, 469
SRVRTID2 **82**, *83*, 531
STARTOFYEAR **478**
STATE 133, 142, *144*, 155, 156, 259, 354, *408*, *409*, *423*, *424*, 506, 508, 516, *559*, *560*, *561*, *563*, *564*, *565*, *567*, *568*
STATEWITHHOLDING **458**, 461
STATUS *37*, 43, 45, 51, 52, 57, *62*, **78**, 78, 81, *126*, 133, 143, *144*, *148*, *154*, *168*, *169*, *171*, *182*, 240, 250, *317*, *320*, *321*, *324*, *329*, *330*, *332*, 360, *409*, *411*, *413*, *414*, *416*, *417*, *418*, *420*, *422*, *424*, *426*, 443, 453, *491*, *497*, 510, 521, 526, 535, *559*, *563*, *566*, *568*, *569*, *573*, *576*
STATUSMODBY **530**, 532
STMNTIMAGE 529, **533**, 533, **533**, 533, *570*, *571*, *573*
STMTENDRQ 39, 222, **222**, 222, 225, 484
STMTENDRS 222, **223**, 223, 226, 484, 485
STMTENDTRNRQ 39, 222
STMTENDTRNRS 223
STMTRQ *22*, 27, 38, 203, **203**, 203, 205, 224, 225, *316*, 430
STMTRS 27, 38, 105, 203, **204**, 204, 207, *317*, 430
STMTTRN 204, 207, **209**, 209, 210, *318*, 455, *492*, 539
STMTTRNRQ *22*, *41*, 203, *316*
STMTTRNRS 204, *317*
STOCKINFO 444, **449**, 449, *494*
STOCKTYPE **449**
STOPPRICE **469**
STPCHKFEE **311**
STPCHKNUM 236, **237**, 237, *322*
STPCHKPROF 310, **311**, 311
STPCHKRQ **235**, 235, *321*, 626
STPCHKRS **236**, 236, *321*
STPCHKSYNCRQ 114, **287**, 287
STPCHKSYNCRS 114, **288**, 288
STPCHKTRNRQ 235, **287**, *321*
STPCHKTRNRS 236, **288**, *321*
STRIKEPRICE **448**, *495*
STSVIAMODS **406**
SUBACCT **469**, *494*
SUBACCTFROM **458**, 463, 464
SUBACCTFUND 455, **458**, 460, 461, 463, 464, 465, 466, *492*
SUBACCTSEC **458**, 460, 461, 462, 463, 464, 465, 466, *492*
SUBACCTTO **458**, 463, 464
SUBJECT **162**, *167*, *168*, *169*
SUPTXDL *148*, **191**, 192, 197
SVC **149**, 151, *154*, 516, 518, 519, *567*, *568*, 627
SVCADD 149, **150**, 150, **150**, 151, *154*, 514, 516, **516**, 516, 518, *567*, *568*
SVCCADD **149**
SVCCHG 149, **150**, 151, 514, 517
SVCDEL 149, **150**, 151
SVCSTATUS 145, **147**, *148*, 151, *154*, 191, 192, 198, 351, 432, *577*
SVCSTATUS2 514, 518
SWITCHALL **470**
SWITCHMF **470**
SYNCERROR 625
SYNCMODE **135**, 135

T

TABLENAME 537, **537**, *570*, *574*
TABLETYPE **537**, 541
TAN 43, 45, **442**, 442, 451, 520, 535

TAXES 455, **458**, 460, 461, 464
TAXEXEMPT **458**, 461, 463, 464
TAXID **142**, *144*, *565*
TEMPPASS **143**, 143, *144*, *566*
TFERACTION **459**, 466
TICKER 442, **442**, **445**, *494*, *495*
TO **162**, *167*, *168*, *169*
TOKEN 79, **85**, 85, 110, 111, 112, 113, 116, 117, 118, 124, *126*, *127*, 152, 153, 156, 157, 165, 166, 167, *168*, 168, *168*, *182*, 238, 242, 245, 247, 251, 254, 256, 262, 263, 266, 269, 271, 273, 276, 278, 281, 282, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, *328*, 362, 366, 368, 370, 374, 378, 380, 383, 384, 385, 389, 392, 394, 395, 396, 398, 399, 400, 401, 403, *425*, 487, 488, 489, 625, 626
TOKEN2 **85**, 153, 550, 551
TOKENONLY 117, **117**, 153, 157, 166, 287, 289, 291, 293, 294, 296, 298, 300, 384, 395, 398, 400, 488, 550, 625
TOTAL **458**, 460, 461, 462, 463, 464, 465, *492*
TOTALFEES **224**
TOTALINT **225**
TRANDNLD 315, **435**
TRANIMGAVAIL **81**, *596*
TRANSFER **466**
TRANSPSEC **97**, 97, **135**, 135
TRNAMT **85**, 85, *177*, *178*, 193, 210, 216, 236, 237, 258, 261, 283, 284, *318*, *319*, *320*, *323*, *325*, *331*, *332*, *333*, 352, 375, *408*, *409*, *410*, *411*, *412*, *413*, *414*, *415*, *418*, *419*, *420*, *421*, *426*, *492*
TRNTYPE 210, **212**, *318*, *492*
TRNUID *22*, 43, 45, *62*, **82**, 82, 112, 116, 118, 120, 122, *126*, *127*, *144*, *148*, *154*, 161, 164, *167*, *168*, *169*, *171*, *316*, *317*, *319*, *321*, *323*, *324*, *328*, *329*, *330*, *331*, 347, 402, 403, *408*, *409*, *410*, *411*, *412*, *413*, *414*, *415*, *416*, *417*, *418*, *420*, *422*, *423*, *424*, *425*, *426*, 442, 443, 451, 453, *490*, *491*, *496*, *497*, 520, 521, 535, *558*, *559*, *563*, *565*, *566*, *567*, *568*, *569*, *572*, *573*, *575*, *576*, 624
TSKEYEXPIRE **51**
TSPHONE **134**
TYPEDESC **448**

U

UNIQUEID **441**, *492*, *493*, *494*, *495*
UNIQUEIDTYPE **441**, *492*, *493*, *494*, *495*
UNITPRICE 431, **445**, **459**, 460, 461, 464, 466, **472**, *492*, *493*
UNITS 431, **459**, 460, 461, 462, 464, 466, **469**, **472**, *492*, *493*, *494*
UNITSSTREET **473**, 473
UNITSUSER **473**, 473
UNITTYPE **459**, 470
URL *70*, 130, **134**, **135**, 136, **158**, 170, *171*, *172*
USEHTML 162, **163**, 163, 164, 166, *167*, *168*, *169*, 298, 300, 384, 488, 550
USERCRED1 **48**, 50
USERCRED1LABEL **137**
USERCRED2 **48**, 50
USERCRED2LABEL **137**
USERID *22*, 47, 49, **49**, 54, 55, 56, 58, 59, 61, *62*, 131, 140, 142, 143, *144*, 162, *167*, *168*, *169*, *316*, 348, 514, 515, 519, 520, 526, 534, 535, 558, *558*, 562, *562*, *565*, *566*, *567*, *568*, *570*, *572*, *573*, *575*, *577*, 583, 599, 629
USERKEY 47, 49, **50**, 51
USERPASS *22*, 47, 49, **50**, 102, 104, 131, 140, *316*, 558, *558*, 562, *562*, *565*, 583, 599, 628
USPRODUCTTYPE 107, 432, **433**, 433

V

VALIDATE 507, **508**, 510, *562*
VALUE **77**, *493*
VER **135**, 135, 136
VESTDATE **479**
VESTINFO **479**
VESTPCT **479**

W

WEBENROLL 158, **158**
WIREBENEFICIARY 258, **259**, 259, 261
WIRECANRQ **262**, 262
WIRECANRS **263**, 263
WIREDESTBANK 258, **258**, 261

WIRERQ **258**, 258
WIRERS **260**, 261
WIRESYNCRQ 114, **293**, 293
WIRESYNCRS 114, **293**, 293
WIRETRNRQ 258, 262, **293**
WIRETRNRS 261, 263, **293**
WIREXFERMSGSET **314**
WIREXFERMSGSETV1 42, 302, **314**, **315**
WIREXFERMSGSRQV1 308, 309, 584
WIREXFERMSGSRSV1 308, 309, 584
WITHDRAWALS 481, **483**
WITHHOLDING **459**, 461, 463

X

XFERDAYSWITH **406**
XFERDEST *148*, **191**, 192, 197
XFERDFLTDAYSTOPAY **406**
XFERINFO **192**, 192, 193, 240, 241, 243, 244, 248, 249, 250, 252, 253, *319*, *320*, *323*, *324*, *330*, *332*
XFERPRCCODE 195, **195**, 196, **196**, 199, 200, 241, 250, *331*, *332*
XFERPRCSTS **195**, 195, 196, 199, 200, 241, 244, 250, 253, *331*, *332*
XFERPROF 310, **311**, 311, 313, 584
XFERSRC *148*, **191**, 192, 197

Y

YEARTODATE **480**, *500*
YIELD **447**, **449**, *494*, *495*
YIELDTOCALL **446**
YIELDTOMAT **446**