Open Group Technical Standard

Nostro Account Access

The Open Group

Open Group Technical Standard

Nostro Account Access

ISBN: 1-85912-217-5
Document Number: C012

Published in the U.K. by The Open Group, October 2001.

Any comments relating to the material contained in this document may be submitted to:

The Open Group
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

OGSpecs@opengroup.org

Contents

Preface

The Open Group

The Open Group is a vendor and technology-neutral consortium which ensures that multi-vendor information technology matches the demands and needs of customers. It develops and deploys frameworks, policies, best practices, standards, and conformance programs to pursue its vision: the concept of making all technology as open and accessible as using a telephone.

The mission of The Open Group is to deliver assurance of conformance to open systems standards through the testing and certification of suppliers' products.

The Open group is committed to delivering greater business efficiency and lowering the cost and risks associated with integrating new technology across the enterprise by bringing together buyers and suppliers of information systems.

Membership of The Open Group is distributed across the world, and it includes some of the world's largest IT buyers and vendors representing both government and commercial enterprises.

More information is available on The Open Group Web Site at *http://www.opengroup.org*.

Open Group Publications

The Open Group publishes a wide range of technical documentation, the main part of which is focused on development of Technical and Product Standards and Guides, but which also includes white papers, technical studies, branding and testing documentation, and business titles. Full details and a catalog are available on The Open Group Web Site at *http://www.opengroup.org/pubs*.

- Product Standards

  A Product Standard is the name used by The Open Group for the documentation that records the precise conformance requirements (and other information) that a supplier's product must satisfy. Product Standards, published separately, refer to one or more Technical Standards.

  The ''X'' Device is used by suppliers to demonstrate that their products conform to the relevant Product Standard. By use of the Open Brand they guarantee, through the Open Brand Trademark License Agreement (TMLA), to maintain their products in conformance with the Product Standard so that the product works, will continue to work, and that any problems will be fixed by the supplier. The Open Group runs similar conformance schemes involving different trademarks and license agreements for other bodies.

- Technical Standards (formerly CAE Specifications)

  Open Group Technical Standards, along with standards from the formal standards bodies and other consortia, form the basis for our Product Standards (see above). The Technical Standards are intended to be used widely within the industry for product development and procurement purposes.

  Technical Standards are published as soon as they are developed, so enabling suppliers to proceed with development of conformant products without delay.

  Anyone developing products that implement a Technical Standard can enjoy the benefits of a single, widely supported industry standard. Where appropriate, they can demonstrate product compliance through the Open Brand.

- CAE Specifications

  CAE Specifications and Developers' Specifications published prior to January 1998 have the same status as Technical Standards (see above).

- Preliminary Specifications

  Preliminary Specifications have usually addressed an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations. There is a strong preference to develop or adopt more stable specifications as Technical Standards.

- Consortium and Technology Specifications

  The Open Group has published specifications on behalf of industry consortia. For example, it published the NMF SPIRIT procurement specifications on behalf of the Network Management Forum (now TMF). It also published Technology Specifications relating to OSF/1, DCE, OSF/Motif, and CDE.

In addition, The Open Group publishes Product Documentation. This includes product documentation—programmer's guides, user manuals, and so on—relating to the DCE, Motif, and CDE. It also includes the Single UNIX Documentation, designed for use as common product documentation for the whole industry.

Versions and Issues of Specifications

As with all *live* documents, Technical Standards and Specifications require revision to align with new developments and associated international standards. To distinguish between revised specifications which are fully backwards compatible and those which are not:

- A new *Version* indicates there is no change to the definitive information contained in the previous publication of that title, but additions/extensions are included. As such, it *replaces* the previous publication.
- A new *Issue* indicates there is substantive change to the definitive information contained in the previous publication of that title, and there may also be additions/extensions. As such, both previous and new documents are maintained as current publications.

Corrigenda

Readers should note that Corrigenda may apply to any publication. Corrigenda information is published on The Open Group Web Site at *http://www.opengroup.org/corrigenda*.

Ordering Information

Full catalog and ordering information on all Open Group publications is available on The Open Group Web Site at *http://www.opengroup.org/pubs*.

This Document

This document describes the XML Formats, Conventions, and Semantics assumed in passing messages between a client (Bank holding a *Nostro* account) and a *Nostro* agent (Correspondent Bank managing the *Nostro* account on behalf of the client), that allow the client to receive information relating to the current transactions that have been entered into that *Nostro* account.

Trademarks

Motif®, OSF/1®, UNIX®, and the ‘‘X Device’’ are registered trademarks and IT DialTone™ and The Open Group™ are trademarks of The Open Group in the U.S. and other countries.

This list represents, as far as possible, those products that are trademarked. The Open Group acknowledges that there may be other products that might be covered by trademark protection and advises the reader to verify them independently.

Acknowledgements

The Open Group acknowledges the work of the members of the IBIX Group (Australia & New Zealand Banking Group (ANZ), Banco Espirito Santo, Barclays, Citigroup, JP Morgan Chase & Co.) and that of Gresham Computing plc in the creation of this Technical Standard.

Referenced Documents

The following documents are referenced in this Technical Standard:

1. Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000 (http://www.w3c.org)
2. Hypertext Transmission Protocol (HTTP): W3C Recommendation RFC 2616 HTTP 1.1 (http://www.w3c.org)
3. ISO 8601:2000, Data Elements and Interchange Formats — Information Interchange — Representation of Dates and Times (http://www.iso.ch)
4. ISO 4217:1995, Codes for the Representation of Currencies and Funds (http://www.iso.ch)
5. SWIFT Message Formats 2000 (http://www.swift.com)
6. European Directive on Cross Border Payments, EU Parliament, August 1999
7. Bank of International Settlement: Amendment to Capital Accord to Incorporate Market Risk (Report No. 24, January 1996) (http://www.bis.org)

Chapter 1

Introduction

1.1 Summary of Requirement

Many banks and large corporations wish to reduce the risk associated with the settlement of transactions in the areas of foreign exchange, cash management, and treasury, by being able, on demand, to have fast and continuous read access to transaction-level information on all of their Nostro accounts—held by various *Nostro* Agents in local back-office systems around the world.

Almost all financial institutions use *Nostro* accounts. As the requirement for faster transactions continues to rise, the increased intra-day exposure to counterparts has reached a point where banks desire to improve risk monitoring and controls. Access to virtual ‘‘near real-time’’ *Nostro* information plays a key role in this goal of reducing risk.

The recent widespread acceptance of Web technology, which enables businesses to publish information on the Internet using software from a range of suppliers, and which allows consumers of that information to access it using the browser of their choice, is made possible by the adoption of open standards for network transport (TCP/IP), protocol (HTTP), and data format (HTML), and the availability of a global public network—the Internet.

It is this that has encouraged financial institutions to consider that an open standards-based approach might enable them to meet their requirement for direct access to *Nostro* account information, using a similar vendor-independent model, and perhaps leveraging some of the existing low-cost ubiquitous Web technology.

There are, however, fundamental differences between the operating parameters of Web servers and browsers, and any proposed *Nostro* server with its associated client components. Some of these are listed below:

1. Most Web servers are public, with very little access control. Conversely all access to a *Nostro* server must be very strictly controlled—only fully authenticated clients can ever be allowed to retrieve account information.
2. Web browsers are designed for use by a human operator, and the HTML data format is primarily concerned with presentation. A *Nostro* client component will often be a background application, automatically accessing *Nostro* account servers around the world, and processing the data it retrieves without human intervention. In this case, the data format must be optimized for programmatic manipulation, with little if any presentation requirement.
3. For a Web server, throughput increases as a function of the number of browsers accessing it, as a single browser rarely exceeds one interaction every five seconds. For a *Nostro* server, a single client can generate a significant workload, as it programmatically fires a series of requests to the server.
4. A human Web browser user can adapt quickly and easily to changes in the data format, expressed in HTML and published by a Web server, especially as much of the content is concerned with presentation—the eye and brain are extremely good at locating the information of interest. A *Nostro* client application, however, must be programmed to handle specific data formats, and will be much more sensitive to format changes. Further, every response must be predetermined—programs are notoriously bad at dealing with the unexpected.

5. All Web browsers, regardless of supplier, do essentially the same thing—they render HTML to produce a visual representation of the document retrieved. Most *Nostro* client applications will be programmed specifically to the requirements of the *Nostro* account owner.
6. Most Web servers provide access to static documents, typically held one-for-one as files in the local filesystem of the server computer. A *Nostro* server will almost certainly need to retrieve account information from the *Nostro* agent's existing mission-critical back office systems, and without destabilizing them or degrading their performance.

The specification provides the solution proposed to the requirement of providing access to the *Nostro* accounts, while ensuring that the issues also presented are carefully considered and resolved.

The intention of this specification is to be compliant with the initiative of the Bank of International Settlement (BIS) to reduce settlement risk—the reduction of the period of settlement risk. With *Nostro* information provided in virtual real-time and a suitable control process in place the reduction of intra-day exposure is possible.

1.2 Inter-Bank Information eXchange (IBIX) Working Group

The purpose of the Inter-Bank Information eXchange Working Group is to act as an independent forum for financial institutions to produce open, interoperable specifications and protocols to enable the secure transfer of information from one financial institution to another.

Established as an independent working group under the legal framework of The Open Group, the members of the group are free to enter into projects related to information exchange between independent organizations. Projects may encompass specifications, implementations, conformance, testing, and promotion of the work of the group.

1.3 Business Requirements of Nostro Account Access

The current practice of reconciling *Nostro* accounts on *value day*+1 (with completion possibly as late as *value day*+2) is unacceptable. This was highlighted in the BIS Allsopp Report of 1996. Current systems generally are not capable of automated intra-day access to information at a *Nostro* agent, thus making it difficult, if not impossible, to do comprehensive and satisfactory intra-day monitoring of counter-party settlement risk and intra-day reconciliation.

In *Nostro* account access both the user and provider of *Nostro* services can benefit as discussed further in this business case.

The needs for the initial *Nostro* account access project have been defined as:

- The ability to directly access *Nostro* account information
- That the information is available in ''near real-time''
- That the information can be accessed flexibly and selectively, including all information
- The information includes a settlement-time time-stamp
- The protocol is based on the Open Standards of XML and TCP/IP

The benefits of this specification to an organization providing *Nostro* services to its customers have been identified as follows:

- **One Server for all Customers**

  Financial institutions have to provide services to their customers in a time of ever-increasing competition. They have to work faster and provide better service to maintain customer loyalty. The adoption of a *Nostro* server based on a standard will provide virtual real-time services that are in demand. It provides an added attraction to customers as they can adopt the same standard for all *Nostro* agents.

- **Capability to Add Additional Services**

  Once a suitable server engine is used to provide the access to *Nostro* account information, it will be possible to add further services that will add to maintaining customer loyalty, and add to the possibility of attracting new customers.

- **One Window on a Customer**

  By integrating a customer's position from all business areas, it is possible to present an integrated view of a customer's total position with the financial institution. This may include all positions held locally as well as positions held in other branches and subsidiaries. The benefit to the financial institution providing the information is that it can understand more fully and more quickly the exposure and risk that it has with a customer at any time.

- **Cross-Product Netting**

  If a customer's integrated position is known, it is possible to calculate a net settlement amount that will reduce the settlement exposure. A reduced amount to settle has obvious advantages, such as lower liquidity requirement, lower exposure should any form of default occur. From a legal point of view, an ISDA Master Agreement should cover netting in a default situation to protect the bank's net exposure.

- **Cross-Product Credit Line**

  If a customer's integrated position is known, it is possible to set one credit line across all business areas. The industry standard practice is to issue a credit line for each business area; often one business area is not aware of credit lines set in other business areas. This means that the potential total exposure to a customer, should all credit lines be fully used, may be greater than the financial institution is willing to accept. By aggregating all exposures in all business areas, it should be possible to set one credit line that would allow the same amount of trading but at a lower level than the sum of individual credit lines, thus lowering the potential maximum exposure.

- **Cross-Product Collateral**

  It will be possible to set one collateral requirement for a customer based on aggregated cross-product requirements. The effect will be to require less collateral (that is, less collateral than the sum of the individual collateral provided to each business area) from a customer while allowing the customer to trade at existing levels. Requiring less collateral (therefore less cost) from a customer is an attractive selling point when offering services.

- **Access to Payment Queue**

  At times there may be insufficient liquidity (in whatever form; for example, cash, collateral, credit line) to make payments on behalf of a customer. A service that may be attractive is the ability of the customer to access its payments queued at the *Nostro* agent with a view to setting priorities. On occasion, this may be important to the customer when a crucial payment is queued. By giving the customer the ability to change priorities directly, it will reduce the manual effort and responsibility of the financial institution of updating the queue. This manual updating is current practice. This facility will also be attractive to existing and prospective customers.

- **Open Architecture for Ease-of-Integration**

  Utilizing standard specifications and technologies based on those standards allows providers of information to deliver the same access and security to all customers requiring information. It also allows financial institutions to choose between different suppliers of technology, based on other criteria such as performance, reliability, and so on.

- **Open Network (Internet and Intranet) for Ease-of-Connectivity**

  The Web has delivered to all computer users the ability to deliver messages and information to all other users of computer technology. The technology, while not necessarily offering guaranteed delivery and service, is now pervasive and can be utilized by all. Establishing a network for communication is therefore not an issue for this specification. The new SWIFT Secure IP Network (SIPN) initiative is an obvious communication mechanism that will be readily adopted by almost all financial institutions.

The benefits of this specification to an organization that has access to and uses *Nostro* provided by a financial institution have been identified as follows:

- **Timely Information Access**

  The ability to access *Nostro* information in virtual real-time is necessary to monitor settlement in a timely manner as well as being an element in overall intra-day risk management. The payment services a bank performs on behalf of its customers means it is taking responsibility, and therefore risk.

  For example, typically a clearing bank will pay out on behalf of its customer up to a certain limit, usually based on anticipated funds being received by the end of the day. It is quite probable that the amount paid out, for some of the day, is over the limit permitted, until such time as covering funds are received. This may not be acceptable, but is allowed to happen, because it is not easy, or possible, to monitor the real-time situation without immediate access to *Nostro* information.

- **Reducing Risk**

  On a broader scale, a clearing bank may be exposed to the same customer through its branches and subsidiaries. There is a need for global risk management with access to information in branches and *Nostro* accounts relating to a customer, to make it possible to monitor customer risk across all currencies and all branches. Situations may arise where one branch is not receiving funds while another branch is paying out, thus creating an unacceptable exposure. With access to the *Nostro* agents it will be possible to delay payments in one currency until the funds in the other currency have been received.

  To minimize the effect of unacceptable intra-day exposures, it is essential to access *Nostro* information in virtual real-time, and ideally also access information from branches and subsidiaries. Various sections of an organization may benefit from directly accessing *Nostro* information.

- **Open Architecture for Ease-of-Integration**

  Again, just as the Financial Institution can benefit from an open architecture, the customer receiving *Nostro* information can benefit from a choice of suppliers and applications to process the information accordingly.

- **Open Network (Internet and Intranet) for Ease-of-Connectivity**

  Utilizing an Intranet, Extranet, or the Internet can help clients co-ordinate and disseminate *Nostro* information across the enterprise and partners. There is no need to make any special provision for proprietary communications mechanisms that receive *Nostro* account

information.

- **Informed Payment Decisions**

  With information available when it is needed, it is possible to make decisions on current factual information. Decisions to make or withhold a payment on behalf of a customer can only be made intelligently if up-to-date information is available. It is not uncommon at present for payments to be made regardless of the actual exposure, because the actual exposure is not known. Payments that are allowed to be made without up-to-date information may be putting the financial institution at risk unnecessarily. Up-to-date information will make it possible to make rational decisions.

- **Faster Resolution of Issues**

  Problems can be identified sooner so that action can be taken to resolve the issue. The current practice of reconciling on *value day*+1 means corrective action or intervention is done some time after the event. This delayed intervention may be of no effect or may make the issue more difficult to rectify due to the time elapsed. With up-to-the-minute information any discrepancy is identified very quickly, so corrective or intervention action will have proper effect.

- **Improved Cash Management**

  The *Nostro* account information will enable cash management to be done on real-time factual information rather than anticipated cash movement. Current practice of cash management is based on cash movements anticipated during a day. With up-to-the-minute information, it is possible for a bank to make cash management decisions at various times during a day, thereby maximizing the use of funds.

- **Improved Risk Management**

  As well as reducing risk, *Nostro* account information availability will enable risk management to be done on real-time information. As with cash management, risk management can be done up-to-the-minute rather than in retrospect. There is a changing paradigm, where daily or periodic risk management, that has been standard practice, is no longer acceptable. The amounts of exposure, and the speed at which credit worthiness can change, are such that monitoring and managing risk intra-day is becoming essential, not just nice to have. The types of risk an organization may wish to improve its coverage of include liquidity, operational, and credit.

- **Monitoring Payment Performance**

  This initiative will enable payment performance to be monitored. Using a settlement-time time-stamp of when irrevocable credit has been granted, it is possible to monitor the payment performance of a customer. By comparing the time of credit and the time when payments have been made on behalf of the customer, the amount and time of exposure can be calculated. This information can assist with setting true intra-day limits, collateral requirements, and even whether pre-funding is required in the worst cases.

Chapter 2

Overview of Solution

This chapter defines the elements considered to be part of the proposed solution for providing *Nostro* account information to the client.

2.1 Elements of the Specification

The elements of this specification are defined as follows:

- Any physical private, public, or virtual private network which supports TCP/IP transport
- TCP/IP transport to run over that network
- Security services that are provided as part of the network, or that are based on The Open Group CDSA (Common Data Security Architecture)
- HTTP request-response connectionless application-to-application protocol
- A core set of named *Nostro* account access information services, each expressed in terms of named XML inputs and outputs
- For each information service, request and response data structures expressed in XML (Extensible Markup Language) fully described as XML DTDs (or in XML Schema Language (XSL))

For illustration purposes only, the minimum set of elements that could be used in an implementation is:

- The public Internet
- A TCP/IP transport layer employing point-to-point security by means of Secure Sockets Layer and Digital Certificates
- A *Nostro* server component, offering the core set of information services to authenticated clients using HTTP protocol over TCP/IP transport, with data expressed in XML, and with enterprise application integration facilities to enable it to retrieve the appropriate account information from an existing back-office application
- A *Nostro* client component, developed specifically to meet the requirements of the bank requiring access to their *Nostro* account information, capable of locating and communicating with a number of *Nostro* servers on the network, meeting the authentication requirements of those servers, submitting requests in XML to such servers using the HTTP protocol over TCP/IP protocol, and understanding the data structures returned, as expressed in XML

2.2 Status of the Elements of the Specification

Some of the specification components listed above are generic and already available. Others are specific to this requirement, and will need to be developed and implemented by *Nostro* agents or clients wishing to provide or access *Nostro* account information.

Once all the specification components are publicly available, they may be used by any party (bank, software vendor, system integrator) to build conforming implementation components of a working system, which will interoperate with other conforming implementation components in a consistent and deterministic way.

Banks wishing to offer *Nostro* server facilities need only implement a server component which conforms to the protocol and data structure specifications, publishes the core set of information services, and is accessible to clients on a conforming network.

Banks may choose to implement such a *Nostro* server themselves, or purchase some or all of the conforming implementation components from a software vendor.

It is anticipated that an architecture involving a third-party—other than the *Nostro* agent or client bank—may implement a central clearing system to allow a client to interrogate all of his *Nostro* account information through this one central server. This distributed architecture will also provide additional security to the *Nostro* agent, as all details of the transaction are expected to be ‘‘pushed’’ from the *Nostro* agent rather than ‘‘requested’’ by the central server.

Banks or enterprises wishing to act as clients, and make use of the *Nostro* server facilities made accessible by banks acting as *Nostro* agents (described above) need only implement client applications which conform to the protocol and data structure specifications, and which are able to access the *Nostro* servers they have an interest in, by being on the same or a connected network, and by being in possession of the necessary location and authentication information.

Banks or enterprises may choose to implement such a *Nostro* client application themselves, or purchase some or all of the conforming implementation components from a software vendor.

Chapter 3

Specification Components

This chapter presents a detailed description of the specification components of the proposed solution.

3.1 A Physical Network Supporting TCP/IP

- Public Internet
- Private network (for example, the new SWIFT Secure IP Network, SIPN)
- Virtual private network

3.2 TCP/IP Transport

- Socket-based TCP
- Unique server identification: IP address and port number

3.3 Security Services Based on CDSA

CDSA security services may need to be employed to ensure some or all of the following:

Authentication Both client and server can be assured of the true identity of the party they are communicating with.

Privacy Sensitive data passing over the network cannot be seen by unrelated third parties.

Integrity The application data cannot be modified in any way as it passes over the network.

Various levels of security may be applied, depending on the requirements of the parties involved.

As a minimum, an authentication mechanism which enables a *Nostro* server to verify that a client making a request for information about a specific *Nostro* account is in fact authorized to receive that information. Authentication is usually achieved by the client obtaining from the server an authentication token, which identifies the client as an authenticated user of services provided by the server. Each request is accompanied by the authentication token, which the server will use in this case to check that the client is allowed to access the account information required.

Authentication tokens can be transient objects, obtained during some kind of ‘‘logging in’’ process and having a limited lifespan, or they can be semi-permanent objects, such as Digital Certificates, obtained by the client from a Certification Authority.

This specification does not impose any specific authentication mechanism on the user or provider of *Nostro* account services. It is to be agreed between the user and the provider, prior to the implementation of the service.

3.4 HTTP

Hypertext Transmission Protocol is the underlying mechanism that will allow the *Nostro* account access services to be passed between the client and *Nostro* server. The XML defined in this specification represents an extension of the message formats used in HTML 4.0 or later. These extensions are used for:

- Input-identifier
- Output-identifier
- Authentication-token
- Error handling
- Request submission
- Protocol errors
- Accounts unique within the *Nostro* server
- Location of the server
- Identification of the information services

3.5 Integrated Services

Intergrated services, such as SwiftNET Link, a required part of the SWIFT solution, can be used to implement the *Nostro* account access solution. SwiftNET Link provides complete infrastructure support for the transport, security, and HTTP/XML handling of the messages.

3.6 Core Information Services

The following core information service requests must be interpreted and responded to by a conforming server-side implementation component:

getNostroAccountBalance
Returns basic account details, with the current balance.

getNostroAccountTransactions
Returns all or selected transactions for a specific account.

getNostroTransactionStatusUpdate
Returns an update of the status of the selected transactions.

A client-side implementation component will rely on the standard definition of these information services to determine the level of service it can expect from any server-side implementation component.

Therefore, it is very important that there is a common, core definition of these information services that all server-side implementation components will implement, in a fully deterministic way. This standard definition is provided below, in terms of the input and output interfaces to the information services, and a detailed and comprehensive description of the behavior of the information services under certain conditions.

Chapter 4

Definitions

The following terms and identifiers used in this specification are defined as follows:

accountIdentifier
An identifier for the *Nostro* account being interrogated by the client. The format of this identifier is decided by the *Nostro* agent, and provided to the client as part of the process to establish a Service Level Agreement (SLA). It may be encrypted. The client should have no need to interpret this identifier. The **accountIdentifier** is always a mandatory parameter, and is used by the server to uniquely identify the account for which a balance or some transaction information is required.

AmountFormat
Amount items will contain only numeric characters, must contain at least one significant digit before the decimal comma, must contain a decimal comma, and the number of digits after the decimal comma must not exceed the maximum number allowed for that specific currency, as defined in ISO 4217.

balanceDate
An ISO 8601 (UTC) date time field which indicates to the *Nostro* agent the exact date and time for which the requested Book Amount Balance and Good Value Balance is being requested.

If the **balanceDate** entry is before the current day at the *Nostro* agent, then the time element of the field shall be ignored, and the values returned for **balanceBookAmount** and **balanceGoodValueAmount** will represent the values at Close of Business in the timezone of the *Nostro* agent on the day requested.

If the **balanceDate** entry is the current day at the *Nostro* agent, then:

- If the time field is not present or set to 00:00:00, the **balanceBookAmount** and **balanceGoodValueBalance** fields will be set to the value within the account at the time the request is received by the *Nostro* agent.
- If the time field is set to a specific time during the current day that is earlier than the current time at the *Nostro* agent, the **balanceBookAmount** and **balanceGoodValueBalance** fields will be set to the value within the account at the specific time requested.
- If the time field is set to a specific time during the current day that is later than the current time at the *Nostro* agent, then an error will be returned.

If the **balanceDate** is recognized to be at a time in the future, an error will be returned.

ISO 8601 (UTC)
All date references within this specification shall be represented by the ISO 8601:2000 (E) extended date format of:

```
YYYY-MM-DD
```

where:

YYYY Represents the year in question.

MM Represents the month.

DD Represents the day within the month.

All characters in the date format shall be numeric in the range 0-9 or the '-' character.

All time references within this specification shall be represented by the ISO 8601:2000 (E) Coordinated Universal Time (UTC) extended format or local time of:

```
hh:mm:ssZ or
hh:mm:ss+xx:yy
hh:mm:ss
```

where:

hh Represents the hours.

mm Represents the minutes.

ss Represents the seconds.

xx Represents the number of hours different from GMT.

yy Represents the number of minutes different from GMT.

For example, 9am on 25th December in New York may be represented as:

```
2001-12-25T14:00:00Z (UTC) or
2001-12-25T09:00:00-05:00
2001-12-25T09:00:00
```

No abbreviations or truncated formats are permitted by this specification. All characters in the time format shall be numeric in the range 0-9 or the ':' character. The 'T' character is present when the date and time formats are present in the same character string. The local time zone—where no designation is present—shall be agreed in the SLA.

number format

Unless indicated otherwise, the format of all numbers detailed in this specification shall be defined as ''one or more digits optionally followed by a decimal point character and one or more digits''.

searchString

The search string used in several parameters to the **getNostroAccountTransactionInput** request. The format of the string may contain the following special characters:

* When preceded and succeeded by a string of characters, means that the string will match with zero or more characters in the transaction record being searched that are between the preceding and/or succeeding characters surrounding the '*'.

? When preceded and succeeded by a string of characters, means that the string will match with exactly one character in the transaction record being searched that is between the preceding and/or succeeding characters surrounding the '?'.

Chapter 5

Input and Output Interfaces

The interface to each core information service is fully defined by a single request XML data structure and a set of response XML data structures, exactly one of which will be returned in a specific request-response pair. The application protocol allows a client application to determine which response data structure has been returned, and react accordingly.

Each data structure has a unique name, and is fully defined by a metadata description, expressed as an XML DTD (Document Type Description), or in XML Schema Language.

The following data structures have been identified for the core information services:

getNostroAccountBalance

- *getNostroAccountBalanceInput*
- *getNostroAccountBalanceOutput*
- *getNostroAccountBalanceError*

getNostroAccountTransactions

- *getNostroAccountTransactionsInput*
- *getNostroAccountTransactionsOutput*
- *getNostroAccountTransactionsError*
- *getNostroAccountTransactionsStatusUpdate*

The formal DTDs which describe these data structures are described below.

5.1 getNostroAccountBalanceInput

```
<!DOCTYPE getNostroAccountBalanceInput
[ <!ELEMENT getNostroAccountBalanceInput
    (
        accountIdentifier+,
        balanceDate?,
        returnAmountBookBalance?
    )>
    <!ATTLIST getNostroAccountBalanceInput
        elementID CDATA #FIXED "IBIX0100"
        version CDATA #FIXED "1.0">
    <!ELEMENT accountIdentifier (#PCDATA)>
        <!ATTLIST accountIdentifier
            elementID CDATA #FIXED "IBIX0101"
            type CDATA #FIXED "string"
            version CDATA #FIXED "1.0">
    <!ELEMENT balanceDate (#PCDATA)>
        <!ATTLIST balanceDate
            elementID CDATA #FIXED "IBIX0102"
            type CDATA #FIXED "date"
            version CDATA #FIXED "1.0">
    <!ELEMENT returnAmountBookBalance (#PCDATA)>
```

```
        <!ATTLIST returnAmountBookBalance
            elementID CDATA #FIXED "IBIX0103"
            type CDATA #FIXED "string"
            version CDATA #FIXED "1.0">
]>
```

5.2 getNostroAccountBalanceOutput

```
<!DOCTYPE getNostroAccountBalanceOutput
[ <!ELEMENT getNostroAccountBalanceOutput
    (
        accounts*
    )>
    <!ATTLIST getNostroAccountBalanceOutput
        elementID CDATA #FIXED "IBIX0200"
        version CDATA #FIXED "1.0">

    <!ELEMENT accounts
        (
            accountIdentifier,
            balance
        )>
            <!ATTLIST accounts
                elementID CDATA #FIXED "IBIX0210"
                version CDATA #FIXED "1.0">
        <!ELEMENT accountIdentifier (#PCDATA)>
            <!ATTLIST accountIdentifier
                elementID CDATA #FIXED "IBIX0211"
                type CDATA #FIXED "string"
                version CDATA #FIXED "1.0">
        <!ELEMENT balance
            (
                balanceValueDate,
                balanceDateTime,
                balanceAmountBookBalance?,
                balanceGoodValueBalance,
                lastTransactionIdentifier
            )>
            <!ATTLIST balance
                elementID CDATA #FIXED "IBIX0220"
                version CDATA #FIXED "1.0">
            <!ELEMENT balanceValueDate (#PCDATA)>
            <!ATTLIST balanceValueDate
                elementID CDATA #FIXED "IBIX0221"
                type CDATA #FIXED "date"
                version CDATA #FIXED "1.0">
            <!ELEMENT balanceDateTime (#PCDATA)>
                <!ATTLIST balanceDateTime
                    elementID CDATA #FIXED "IBIX0222"
                    type CDATA #FIXED "date"
                    version CDATA #FIXED "1.0">
            <!ELEMENT balanceAmountBookBalance (#PCDATA)>
```

```
                <!ATTLIST balanceAmountBookBalance
                    elementID CDATA #FIXED "IBIX0223"
                    type CDATA #FIXED "decimal"
                    version CDATA #FIXED "1.0">
            <!ELEMENT balanceGoodValueBalance (#PCDATA)>
                <!ATTLIST balanceGoodValueBalance
                    elementID CDATA #FIXED "IBIX0224"
                    type CDATA #FIXED "decimal"
                    version CDATA #FIXED "1.0">
            <!ELEMENT lastTransactionIdentifier (#PCDATA)>
                <!ATTLIST lastTransactionIdentifier
                    elementID CDATA #FIXED "IBIX0225"
                    type CDATA #FIXED "string"
                    version CDATA #FIXED "1.0">
]>
```

5.3 getNostroAccountBalanceError

```
<!DOCTYPE getNostroAccountBalanceError
[ <!ELEMENT getNostroAccountBalanceError
    (
        errors*
    )>
    <!ATTLIST getNostroAccountBalanceError
        elementID CDATA #FIXED "IBIX0300"
        version CDATA #FIXED "1.0">
    <!ELEMENT errors
        (
            accountIdentifier,
            errorCode,
            errorDetail?
        )>
        <!ATTLIST errors
            elementID CDATA #FIXED "IBIX0310"
            version CDATA #FIXED "1.0">
        <!ELEMENT accountIdentifier (#PCDATA)>
            <!ATTLIST accountIdentifier
                elementID CDATA #FIXED "IBIX0211"
                type CDATA #FIXED "string"
                version CDATA #FIXED "1.0">
        <!ELEMENT errorCode (#PCDATA)>
            <!ATTLIST errorCode
                elementID CDATA #FIXED "IBIX0311"
                type CDATA #FIXED "integer"
                version CDATA #FIXED "1.0">
        <!ELEMENT errorDetail (#PCDATA)>
            <!ATTLIST errorDetail
                elementID CDATA #FIXED "IBIX0312"
                type CDATA #FIXED "string"
                version CDATA #FIXED "1.0">
]>
```

5.4 getNostroAccountTransactionsInput

```
<!DOCTYPE getNostroAccountTransactionsInput
[ <!ELEMENT getNostroAccountTransactionsInput
    (
        accountIdentifier+,
        startTransactionIdentifier?,
        lastTransactionIdentifier?,
        maximumTransactions?,
        selectionByDateTime?,
        selectionByReference?,
        selectionByOriginator?,
        selectionByBeneficiary?,
        selectionByAmountRange*,
        selectionByTypeIdentificationCode?,
        selectionByStatus?
        selectionByContent?
    )>
    <!ATTLIST getNostroAccountTransactionsInput
        elementID CDATA #FIXED "IBIX0400"
        version CDATA #FIXED "1.0">
    <!ELEMENT accountIdentifier (#PCDATA)>
        <!ATTLIST accountIdentifier
            elementID CDATA #FIXED "IBIX0211"
            type CDATA #FIXED "string"
            version CDATA #FIXED "1.0">
    <!ELEMENT startTransactionIdentifier (#PCDATA)>
        <!ATTLIST startTransactionIdentifier
            elementID CDATA #FIXED "IBIX0401"
            type CDATA #FIXED "string"
            version CDATA #FIXED "1.0">
    <!ELEMENT lastTransactionIdentifier (#PCDATA)>
        <!ATTLIST lastTransactionIdentifier
            elementID CDATA #FIXED "IBIX0402"
            type CDATA #FIXED "string"
            version CDATA #FIXED "1.0">
    <!ELEMENT maximumTransactions (#PCDATA)>
        <!ATTLIST maximumTransactions
            elementID CDATA #FIXED "IBIX0403"
            type CDATA #FIXED "integer"
            version CDATA #FIXED "1.0">
    <!ELEMENT selectionByDateTime
        (
            startDateTime,
            endDateTime
        )>
        <!ATTLIST selectionByDateTime
            elementID CDATA #FIXED "IBIX0410"
            version CDATA #FIXED "1.0">
        <!ELEMENT startDateTime (#PCDATA)>
            <!ATTLIST startDateTime
                elementID CDATA #FIXED "IBIX0411"
                type CDATA #FIXED "date"
```

```
            version CDATA #FIXED "1.0">
    <!ELEMENT endDateTime (#PCDATA)>
        <!ATTLIST endDateTime
            elementID CDATA #FIXED "IBIX0412"
            type CDATA #FIXED "date"
            version CDATA #FIXED "1.0">
<!ELEMENT selectionByReference (#PCDATA)>
    <!ATTLIST selectionByReference
        elementID CDATA #FIXED "IBIX0404"
        type CDATA #FIXED "string"
        version CDATA #FIXED "1.0">
<!ELEMENT selectionByOriginator (#PCDATA)>
    <!ATTLIST selectionByOriginator
        elementID CDATA #FIXED "IBIX0405"
        type CDATA #FIXED "string"
        version CDATA #FIXED "1.0">
<!ELEMENT selectionByBeneficiary (#PCDATA)>
    <!ATTLIST selectionByBeneficiary
        elementID CDATA #FIXED "IBIX0406"
        type CDATA #FIXED "string"
        version CDATA #FIXED "1.0">
<!ELEMENT selectionByAmountRange
    (
        creditIndicator,
        lowAmount,
        highAmount
    )>
    <!ATTLIST selectionByAmountRange
        elementID CDATA #FIXED "IBIX0420"
        version CDATA #FIXED "1.0">
    <!ELEMENT creditIndicator (#PCDATA)>
        <!ATTLIST creditIndicator
            elementID CDATA #FIXED "IBIX0421"
            type CDATA #FIXED "string"
            version CDATA #FIXED "1.0">
    <!ELEMENT lowAmount (#PCDATA)>
        <!ATTLIST lowAmount
            elementID CDATA #FIXED "IBIX0422"
            type CDATA #FIXED "decimal"
            version CDATA #FIXED "1.0">
    <!ELEMENT highAmount (#PCDATA)>
        <!ATTLIST highAmount
            elementID CDATA #FIXED "IBIX0423"
            type CDATA #FIXED "decimal"
            version CDATA #FIXED "1.0">
<!ELEMENT selectionByTypeIdentificationCode (#PCDATA)>
    <!ATTLIST selectionByTypeIdentificationCode
        elementID CDATA #FIXED "IBIX0407"
        type CDATA #FIXED "string"
        version CDATA #FIXED "1.0">
<!ELEMENT selectionByStatus (#PCDATA)>
    <!ATTLIST selectionByStatus
```

```
            elementID CDATA #FIXED "IBIX0408"
            type CDATA #FIXED "string"
            version CDATA #FIXED "1.0">
        <!ELEMENT selectionByContent
            (
            contentLocation,
            contentValue?
            )>
            <!ATTLIST selectionByContent
            elementID CDATA #FIXED "IBIX0430"
            version CDATA #FIXED "1.0">
            <!ELEMENT contentLocation (#PCDATA)>
            <!ATTLIST contentLocation
                elementID CDATA #FIXED "IBIX0431"
                type CDATA #FIXED "string"
                version CDATA #FIXED "1.0">
            <!ELEMENT contentValue (#PCDATA)>
            <!ATTLIST contentValue
                elementID CDATA #FIXED "IBIX0432"
                type CDATA #FIXED "string"
                version CDATA #FIXED "1.0">
]>
```

5.5 getNostroAccountTransactionsOutput

```
<!DOCTYPE getNostroAccountTransactionsOutput
[ <!ELEMENT getNostroAccountTransactionsOutput
    (
        accounts*
    )>
    <!ATTLIST getNostroAccountTransactionsOutput
        elementID CDATA #FIXED "IBIX0500"
        version CDATA #FIXED "1.0">
    <!ELEMENT accounts
        (
            accountIdentifier,
            accountName?,
            accountCurrency?,
            startTransactionIdentifier?,
            lastTransactionIdentifier?,
            transaction*
        )>
        <!ATTLIST accounts
            elementID CDATA #FIXED "IBIX0510"
            version CDATA #FIXED "1.0">
        <!ELEMENT accountIdentifier (#PCDATA)>
            <!ATTLIST accountIdentifier
                elementID CDATA #FIXED "IBIX0211"
                type CDATA #FIXED "string"
                version CDATA #FIXED "1.0">
        <!ELEMENT accountName (#PCDATA)>
```

```
    <!ATTLIST accountName
        elementID CDATA #FIXED "IBIX0511"
        type CDATA #FIXED "string"
        version CDATA #FIXED "1.0">
<!ELEMENT accountCurrency (#PCDATA)>
    <!ATTLIST accountCurrency
        elementID CDATA #FIXED "IBIX0512"
        type CDATA #FIXED "string"
        version CDATA #FIXED "1.0">
<!ELEMENT startTransactionIdentifier (#PCDATA)>
    <!ATTLIST startTransactionIdentifier
        elementID CDATA #FIXED "IBIX0513"
        type CDATA #FIXED "string"
        version CDATA #FIXED "1.0">
<!ELEMENT lastTransactionIdentifier (#PCDATA)>
    <!ATTLIST lastTransactionIdentifier
        elementID CDATA #FIXED "IBIX0514"
        type CDATA #FIXED "string"
        version CDATA #FIXED "1.0">
<!ELEMENT transaction
    (
        transactionIdentifier,
        valueDate,
        valueDateTimeClearingSystem?,
        valueDateTimeNostro?,
        accountOwnerReference,
        accountServicingInstitutionReference,
        supplementaryDetailsReference,
        transactionTypeIdentificationCode,
        bookEntryDate,
        amount,
        debitCreditIndicator,
        status,
        charges,
        narrative?,
        beneficiary,
        originator
    )>
    <!ATTLIST transactions
        elementID CDATA #FIXED "IBIX0520"
        version CDATA #FIXED "1.0">
    <!ELEMENT transactionIdentifier (#PCDATA)>
        <!ATTLIST transactionIdentifier
            elementID CDATA #FIXED "IBIX0521"
            type CDATA #FIXED "string"
            version CDATA #FIXED "1.0">
    <!ELEMENT valueDate (#PCDATA)>
        <!ATTLIST valueDate
            elementID CDATA #FIXED "IBIX0522"
            type CDATA #FIXED "date"
            version CDATA #FIXED "1.0">
    <!ELEMENT valueDateTimeClearingSystem (#PCDATA)>
```

```
    <!ATTLIST valueDateTimeClearingSystem
        elementID CDATA #FIXED "IBIX0523"
        type CDATA #FIXED "date"
        version CDATA #FIXED "1.0">
<!ELEMENT valueDateTimeNostro (#PCDATA)>
    <!ATTLIST valueDateTimeNostro
        elementID CDATA #FIXED "IBIX0524"
        type CDATA #FIXED "date"
        version CDATA #FIXED "1.0">
<!ELEMENT accountOwnerReference (#PCDATA)>
    <!ATTLIST accountOwnerReference
        elementID CDATA #FIXED "IBIX0525"
        type CDATA #FIXED "string"
        version CDATA #FIXED "1.0">
<!ELEMENT accountServicingInstitutionReference (#PCDATA)>
    <!ATTLIST accountServicingInstitutionReference
        elementID CDATA #FIXED "IBIX0526"
        type CDATA #FIXED "string"
        version CDATA #FIXED "1.0">
<!ELEMENT supplementaryDetailsReference (#PCDATA)>
    <!ATTLIST supplementaryDetailsReference
        elementID CDATA #FIXED "IBIX0527"
        type CDATA #FIXED "string"
        version CDATA #FIXED "1.0">
<!ELEMENT transactionIdentificationCode (#PCDATA)>
    <!ATTLIST transactionIdentificationCode
        elementID CDATA #FIXED "IBIX0528"
        type CDATA #FIXED "string"
        version CDATA #FIXED "1.0">
<!ELEMENT bookEntryDate (#PCDATA)>
    <!ATTLIST bookEntryDate
        elementID CDATA #FIXED "IBIX0529"
        type CDATA #FIXED "date"
        version CDATA #FIXED "1.0">
<!ELEMENT amount (#PCDATA)>
    <!ATTLIST amount
        elementID CDATA #FIXED "IBIX052A"
        type CDATA #FIXED "decimal"
        version CDATA #FIXED "1.0">
<!ELEMENT debitCreditIndicator (#PCDATA)>
    <!ATTLIST debitCreditIndicator
        elementID CDATA #FIXED "IBIX052B"
        type CDATA #FIXED "string"
        version CDATA #FIXED "1.0">
<!ELEMENT status (#PCDATA)>
    <!ATTLIST status
        elementID CDATA #FIXED "IBIX052C"
        type CDATA #FIXED "string"
        version CDATA #FIXED "1.0">
<!ELEMENT charges (#PCDATA)>
    <!ATTLIST charges
        elementID CDATA #FIXED "IBIX052D"
```

```
                    type CDATA #FIXED "string"
                    version CDATA #FIXED "1.0">
                <!ELEMENT narrative (#PCDATA)>
                    <!ATTLIST narrative
                        elementID CDATA #FIXED "IBIX052E"
                        type CDATA #FIXED "string"
                        version CDATA #FIXED "1.0">
                <!ELEMENT beneficiary (#PCDATA)>
                    <!ATTLIST beneficiary
                        elementID CDATA #FIXED "IBIX052F"
                        type CDATA #FIXED "string"
                        version CDATA #FIXED "1.0">
                <!ELEMENT originator (#PCDATA)>
                    <!ATTLIST originator
                        elementID CDATA #FIXED "IBIX052G"
                        type CDATA #FIXED "string"
                        version CDATA #FIXED "1.0">
]>
```

5.6 getNostroAccountTransactionsError

```
<!DOCTYPE getNostroAccountTransactionsError
[ <!ELEMENT getNostroAccountTransactionsError
    (
        errors*
    )>
        <!ATTLIST getNostroAccountTransactionsError
            elementID CDATA #FIXED "IBIX0600"
            version CDATA #FIXED "1.0">
    <!ELEMENT errors
        (
            accountIdentifier,
            errorCode,
            errorDetail?
        )>
        <!ATTLIST errors
            elementID CDATA #FIXED "IBIX0310"
            version CDATA #FIXED "1.0">
        <!ELEMENT accountIdentifier (#PCDATA)>
            <!ATTLIST accountIdentifier
                elementID CDATA #FIXED "IBIX0211"
                type CDATA #FIXED "string"
                version CDATA #FIXED "1.0">
        <!ELEMENT errorCode (#PCDATA)>
            <!ATTLIST errorCode
                elementID CDATA #FIXED "IBIX0311"
                type CDATA #FIXED "integer"
                version CDATA #FIXED "1.0">
        <!ELEMENT errorDetail (#PCDATA)>
            <!ATTLIST errorDetail
                elementID CDATA #FIXED "IBIX0312"
```

```
                type CDATA #FIXED "string"
                version CDATA #FIXED "1.0">
]>
```

5.7 getNostroAccountTransactionStatusUpdate

```
<!DOCTYPE getNostroAccountTransactionsStatusUpdate
[ <!ELEMENT getNostroAccountTransactionsStatusUpdate
    (
        transactionIdentifier+
    )>
    <!ATTLIST getNostroAccountTransactionsStatusUpdate
        elementID CDATA #FIXED "IBIX0700"
        version CDATA #FIXED "1.0">
    <!ELEMENT transactionIdentifier (#PCDATA)>
        <!ATTLIST transactionIdentifier
            elementID CDATA #FIXED "IBIX0521"
            type CDATA #FIXED "string"
            version CDATA #FIXED "1.0">
]>
```

Chapter 6

Semantics of Interfaces

This chapter describes the required behavior of the elements in the XML messages implemented by a server-side *Nostro* agent implementation, or when invoked by a client-side implementation.

6.1 getNostroAccountBalance

The **getNostroAccountBalance** service enables a client to request balance information for an account to be returned from the server.

This information service requires the XML document *getNostroAccountBalanceInput* to provide the input data needed to control its behavior.

If the server is able to satisfy the request for a balance, it will return the XML document *getNostroAccountBalanceOutput*; otherwise, it will return the error XML document *getNostroAccountBalanceError*.

The server must respond to **getNostroAccountBalance** requests at least as often as the Minimum Response Times detailed in the Service Level Agreement (SLA).

The input document *getNostroAccountBalanceInput* contains the following input fields:

accountIdentifier
A list of one or more identifiers used to detail to the server the accounts for which a balance is being requested; see Chapter 4 (on page 11).

balanceDate
(Mandatory) An ISO 8601 (UTC) date-time as defined in Chapter 4 (on page 11). If the content of the **balanceDate** field is set to `"0000-00-00T00:00:00"`, then the request is for the current balance available at the *Nostro* agent at the time when the request is received.

returnBookAmountBalance (Optional) If supplied, a string containing the word `"yes"` to indicate that the **balanceAmountBookBalance** value should be returned to the client. Any other value in the string will be ignored.

The output document *getNostroAccountBalanceOutput* contains the following:

accountIdentifier
(Mandatory) Usually the account identifier as input to the request, but may be as normalized by the server.

A returned balance is grouped in its own balance tag, each with the following fields:

balanceValueDate
(Mandatory) Specifies the actual (ISO 8601 (UTC)) date of the balance.

balanceDateTime
(Mandatory) Specifies the actual (ISO 8601 (UTC)) date-time of the balance. This date-time combination may be altered by the *Nostro* agent to the Close-of-Business balance on the day in question, if the server is not able to return the intra-day balance. The SLA should define the behavior of balance requests for any day preceding the current day.

balanceAmountBookBalance
(Optional) The monetary amount of the balance. This string will contain only numeric characters, and an optional decimal point. This balance may include forward value items.

balanceGoodValueBalance

(Mandatory) The monetary amount of the balance of cleared and irrevocable funds. This string, in **numberFormat**, will contain only numeric characters, and an optional decimal point. The good value balance contained in this field may change in a subsequent request as a result of:

— A ''Revocable'' transaction becoming ''Good Value''

— A later posting to this account ''back valued'' to a date and time that is earlier than the input **balanceDateTime** field value.

The value returned will reflect the good value on the account at the **balanceDate** (date and time) requested by the input message. This balance explicitly excludes any forward value items.

The output document *getNostroAccountBalanceError* contains the following:

accountIdentifier

(Mandatory) Usually the account identifier as input to the request, but may be as normalized by the server.

errorCode

(Mandatory) A unique code which specifies the type of error that occurred. The errors that may be reported are:

Error Code	Type of Error
01	*Nostro* account information not available on-line.
02	*Nostro* account server closed for maintenance.
03	Too many **getNostroAccountBalance** requests.
05	Request outside scope of the SLA.
06	*Nostro* server busy—please try later.
07	Invalid XML parameter.
11	Invalid **accountIdentifier** parameter.
12	**accountIdentifier** not supplied.
21	Invalid **balanceDate** parameter.
22	**balanceDate** not supplied.
23	Future **balanceDate** not allowed.
9x	*Nostro* agent error—specific to *Nostro* agent.

errorDetail

(Optional) For certain errors, an **errorDetail** field may be returned with additional detailed information.

6.2 getNostroAccountTransactions

The **getNostroAccountTransactions** service enables a client to request detailed information on transactions for an account to be returned from the server.

This information service requires the XML document *getNostroAccountTransactionsInput* to provide the input data needed to control its behavior. If the server is able to satisfy the request for transactions (that is, at least one transaction is returned), it will return the XML document *getNostroAccountTransactionsOutput*; otherwise, it will return the error XML document *getNostroAccountTransactionsError*.

The input document *getNostroAccountTransactionsInput* contains the following input fields:

accountIdentifier
(Mandatory) Used by the server to uniquely identify the account for which transactions are required. Any sub-structure within this string is specific to the server implementation, and should not be manipulated by the client. The *Nostro* access service provider will make valid **accountIdentifier** values available to its authorized clients, and they should be used exactly as supplied.

startTransactionIdentifier
(Optional) Used by the server to identify the start position for any search of transaction records within the *Nostro* account. The value supplied here by the client will be a value returned from a server in a previous call to this service. The SLA will detail the meaning of the value supplied. The actual start position is the next logical transaction in date and time order of entry into the *Nostro* account after the transaction identified by the provision of this parameter. Where the client does not provide this parameter, the default start position for searching is assumed to be the first transaction entered into the account *Nostro* agent during the current day, or of the specific day entered in the **startDateTime** parameter.

lastTransactionIdentifier
(Optional) Used by the server to identify the last transaction in the sequence identified by the combination with **startTransactionIdentifier**, that will be searched for transactions to be returned. The value supplied here by the client will be a value returned from a server in a previous call to this service. Where the client does not provide this parameter, the default is assumed to be the most recent transaction entered into the account by the *Nostro* agent, or the last transaction on the specific day entered in the **endDateTime** parameter.

maximumTransactions
(Optional) Specifies the maximum number of transactions that the client wants to receive in the response to this request. Often this field will be used on conjunction with the **startTransactionIdentifier** and **lastTransactionIdentifier** fields, to retrieve a defined number of transactions. The default maximum number of transactions should be defined by the SLA, but if this is not, and if this field is not supplied, the default is to return a maximum of 100 transactions.

selectionByDateTime
(Optional) A group which contains 2 optional fields, one of which must be present:

— **startDateTime**

— **endDateTime**

either or both of which can contain an ISO 8601 (UTC) date-time. If **selectionByDateTime** is present, then one of the **startDateTime** or **endDateTime** fields must be present. This selection criteria is used by the client to specify a date-time range for transactions. If a **startDateTime** is not specified, then the value of the date field in **startDateTime** defaults to that of the **endDateTime**, and the value of the time field defaults to the start of that day at

the *Nostro* server. If an **endDateTime** is not specified, then the value of the date field in the **endDateTime** field defaults to that of the **startDateTime** field, and the time field in **endDateTime** will default, for a day other than the current day, to end of day at the *Nostro* server, or the current time at the *Nostro* server.

If **selectionByDateTime** is not supplied, the value of this selection criteria defaults to the start of the current day to the current time at the *Nostro* server.

selectionByReference
(Optional) A field, of format **searchString**, which contains a value which uniquely identifies a single transaction within the account. Equivalent to the information in SWIFT Field 20 or 72.

selectionByOriginator
(Optional) A field, of format **searchString**, which contains a value which identifies one or more transactions by an originator. Used by the server to select only those transactions that originate from the party identified by the information in SWIFT Field 52 Option A or Option D. The A option (BIC code) should always be used when such information is available.

selectionByBeneficiary
(Optional) A field, of format **searchString**, which contains a value which identifies a beneficiary of one or more transactions. Used by the server to select only those transactions for the party identified by the information that is contained in SWIFT Fields 58 or 59.

selectionByStatus
(Optional) A field which, if present, must contain only one of the terms ‘‘Good Value’ or ‘‘Revocable’’. No abbreviation is allowed.

selectionByAmountRange
(Optional) A list of zero or more value range groups, each of which, if specified, must contain one mandatory field:

debitCreditIndicator
(Mandatory) Must be provided and contain the value D (Debit), C (Credit), RC (Reverse Credit), or RD (Reverse Debit), and 2 optional fields:

— **lowAmount**(optional)

— **highAmount**(optional)

which specify an inclusive value range. If only **lowAmount** is supplied, then this shall be interpreted as all transactions of value greater than or equal to **lowAmount**. If only **highAmount** is defined, then this shall be interpreted as all transactions of value greater than zero and less than or equal to **highAmount**.

A **debitCreditIndicator** of C (Credit) will match with all RD (Reverse Debit) transactions. Similarly, a value of D (Debit) will include all RC (Reverse Credit) transactions.

selectionByTypeIdentificationCode
(Optional) A field which contains a value which identifies a transaction type, used by the server to select only those transactions of that type. The acceptable transaction types are those defined by SWIFT Field 61 subfield 6.

selectionByContent
(Optional) A group which contains 2 mandatory fields:

— **contentLocation**(mandatory)

— **contentValue**(mandatory)

which specify a server-defined content location within the transaction record, and a content value used by the server to select only those transactions containing the specified content in the specified location. The extent of locations and values that may be used by the client shall be defined in the SLA.

Note: If multiple selection criteria are specified (for example, **selectionByOriginator** and **selectionByValue**), only transactions which meet all sets of criteria are returned. Transactions returned by this message will always be returned in date and time order, with the older messages appearing first.

The output document *getNostroAccountTransactionsOutput* contains the following for each account being reported on:

accountIdentifier
(Mandatory) Usually the account identifier as input to the request, but may be as normalized by the server.

accountName
(Optional) A text name of the account, as known by the server.

accountCurrency
(Optional) An ISO 4217 (alphabetic) code representing the currency for which the account is held.

lastTransactionIdentifier
(Mandatory) Contains a server-defined identifier to the last transaction returned in this response. This is typically used in a subsequent request, in the **startTransactionIdentifier** or **lastTransactionIdentifier** field of the input document, to request transactions following this transaction. If there are any transactions to be returned to the client, then **lastTransactionIdentifier** must be completed by the server and returned in the output message. If no transactions are returned this field must be set to the value ‘"No Matching Transactions’’ by the server.

A server may return several transactions, and must return all transactions that match the selection criteria, up to the maximum number defined by the parameters. If no transactions match the selection criteria, then no transactions need be returned. A returned transaction is a group of fields in its own transaction tag, with the following definitions. Where the server is capable of returning any of the following optional fields, the SLA will define whether it is mandatory that the field be returned to the client:

identifier
(Mandatory) Contains a server-defined unique identifier to this transaction.

valueDate
(Mandatory) Contains the ISO 8601 (UTC) value date for this transaction. Equivalent to SWIFT Field 61 Sub-field 1.

valueDateTimeClearingSystem
(Optional) Contains the ISO 8601 (UTC) value date-time for this transaction where the clearing bank communicates the time of the debit or credit of the funds to the account.

valueDateTimeNostro
(Optional) Contains the ISO 8601 (UTC) value date-time for this transaction of the time the *Nostro* agent is deemed to have credited or debited the transaction to the account.

accountOwnerReference
(Optional) Contains a server-defined reference to the owner of this account for this transaction. Equivalent to SWIFT Field 61 Sub-field 7.

accountServiceInstitutionReference
(Optional) Contains a server-defined reference to the servicing institution of this account for this transaction. Equivalent to SWIFT Field 61 Sub-field 8.

supplementaryDetailsReference
(Optional) Contains a server-defined reference to supplementary details for this transaction. Equivalent to SWIFT Field 61 Sub-field 9.

transactionIdentificationCode
(Optional) Contains a server-defined reference to the type of this transaction. Equivalent to SWIFT Field 61 Sub-field 6.

bookEntryDate
(Mandatory) Contains the ISO 8601 (UTC) book entry date for this transaction.

amount
(Mandatory) Contains the value amount of this transaction, in the currency of the account. This string will be of format **amountFormat**. Equivalent to SWIFT Field 61 Sub-field 5.

debitCreditIndicator
(Mandatory) Contains D, C, RC, or RD. Equivalent to SWIFT Field 61 Sub-field 3.

status
(Mandatory) Contains the status of this transaction. This field may only contain the values ''Good Value'' or ''Revocable''. Where the *Nostro* agent does not hold or is unable to determine the status of a transaction, ''Revocable'' should be returned.

charges
(Optional) Contains the value of any charges applied for this transaction, by the *Nostro* agent only, in the currency of the account. This string will contain only numeric characters, and an optional decimal point.

narrative
(Optional) Contains a text narrative for this transaction. Equivalent to SWIFT Field 86 or Field 72 on an MT900/910 Message.

beneficiary
(Optional) Contains a server-defined reference to the beneficiary of this transaction. Equivalent to SWIFT Field 52 Option A or Option D on an MT900/910 Message.

originator
(Optional) Contains a server-defined reference to the originator of this transaction. Equivalent to SWIFT Field 52 Option A or Option D on an MT900/910 Message.

Note: The number of transactions returned is the number of transaction groups in the output document.

The output document *getNostroAccountTransactionsError* contains the following:

accountIdentifier
(Mandatory) Usually the account identifier as input to the request, but may be as normalized by the server.

errorCode
(Mandatory) A unique code which specifies the type of error that occurred:

Error Code	Type of Error
01	*Nostro* account information not available on-line.
02	*Nostro* account server closed for maintenance.
04	Too many **getNostroAccountTransaction** requests.
05	Request outside the scope of the SLA.
06	*Nostro* server busy—please try later.
07	Invalid XML parameter.
11	Invalid **accountIdentifier** parameter.
12	**accountIdentifier** not supplied.
31	Invalid **startTransactionIdentifier** parameter.
32	Invalid **lastTransactionIdentifier** parameter.
33	Invalid **maximumTransactions** parameter.
34	Invalid **selectionByDateTime** parameter.
35	Invalid **selectionByReference** parameter.
36	Invalid **selectionByOriginator** parameter.
37	Invalid **selectionByBeneficiary** parameter.
38	Invalid **selectionByStatus** parameter.
39	Invalid **selectionByAmountRange** parameter.
40	Invalid **selectionByTypeIdentificationCode** parameter.
41	Invalid **selectionByContent** parameter.
42	Invalid **startDateTime** parameter.
43	Invalid **endDateTime** parameter.
44	Invalid **debitCreditIndicator** parameter.
45	Invalid **lowAmount** parameter.
46	Invalid **highAmount** parameter.
47	Invalid **contentLocation** parameter.
48	Invalid **contentValue** parameter.
49	Invalid **transactionIdentifier** parameter.
51	**startDateTime** or **endDateTime** required parameter missing.
52	**debitCreditIndicator** required parameter missing.
53	**contentLocation** or **contentValue** required parameter missing.
9x	*Nostro* agent error—specific to *Nostro* agent.

errorDetail

(Optional) For certain errors, an **errorDetail** field may be returned with additional detailed information.

6.3 getNostroAccountTransactionsStatusUpdate

The **getNostroAccountTransactionsStatusUpdate** service enables a client to request detailed information on a list of known transactions for an account to be returned from the server. It is particularly used to check on whether a transaction has changed from ‘‘Revocable’’ to that of ‘‘Good Value’’.

This information service requires the XML document *getNostroAccountTransactionStatusUpdate* to provide the unique reference to the transactions for which an update of the status is required. If the server is able to satisfy the request for at least one of the transactions specified, it will return the XML document *getNostroAccountTransactionsOutput*; otherwise, it will return the error XML document *getNostroAccountTransactionsError*.

The input document *getNostroAccountTransactionsStatusUpdate* contains the following input field:

transactionIdentifier
(Mandatory) Used by the server to uniquely identify one or more transactions for which an updated status of each transaction is required.

The transactions returned by *getNostroAccountTransactionsOutput* will contain all transactions that match **transactionIdentifier**. The SLA should agree which field or fields in the transaction record the **transactionIdentifier** of a transaction being reversed is place, such that this permits an efficient search of the *Nostro* account to identify a revocable transaction which is then subsequently revoked.

It is the responsibility of the client to ensure that a ‘‘Revocable’’ transaction is matched to any corresponding reverse transaction that removes the transaction from the account.

Chapter 7

Service Level Agreement

This section of the specification highlights the requirements that should be considered in a Service Level Agreement (SLA) that will need to be entered into between the client and the *Nostro* agent.

1. **Authentication**

   The SLA is required to agree the authentication mechanism that will be used between the client and the *Nostro* agent. This specification imposes no restriction on the mechanisms and utilities that may be used to ensure that the *Nostro* agent is talking to the appropriate client.

   The *Nostro* agent, for example, may encode the IP Address of the client in the **accountIdentifier**.

2. **accountIdentifier**

   This identifier is assigned by the *Nostro* agent. The format is to be agreed between the client and the *Nostro* agent, and may include any such information that the *Nostro* agent may require to ensure that the *Nostro* agent is communicating with the correct client.

   The format of this identifier may include information such as an encoded IP address to receive the response from the request for account information.

3. **Balance**

   The SLA should agree on the type of balance requests that can be made by the client. Many *Nostro* agents cannot provide an intra-day balance for any previous day, and in these circumstances, the end-of-day closing balance should be returned.

4. **transactionIdentifier**

   This identifier is assigned by the *Nostro* agent. The format is to be agreed between the client and the *Nostro* agent, and may include any such information that the transaction can be uniquely identified by both the client and the *Nostro* agent.

   The client and *Nostro* agent must agree that once assigned, a **transactionIdentifier** cannot be removed or deleted from an account. The effect of the transaction can only be removed by a matching reverse transaction also posted onto the account.

5. **Frequency of Requests**

   The SLA should define an agreed level of service between the client and the *Nostro* agent. As a minimum, one **getBalanceAccountInput** should be allowed each minute.

6. **Minimum Response Times**

   The SLA should define the minimum response time to the client for any request, even if this is to return a *Nostro* server busy message. The number of times, within a given period, it is acceptable to receive a *Nostro* server busy should also be defined.

7. **Default Operation**

   By default, the **getNostroAccountTransactions** call will return a maximum of the last 100 transactions on the current day; however, the SLA should permit the client and *Nostro* agent to agree different modes of operation and default values.

Where more than the maximum number of transactions are to be returned by any single message, the SLA should define whether the transactions are returned from the first group of transactions in date and time order or the last group of transactions in date and time order.

When more than one account is being searched by the **getNostroAccountTransactions** message, the maximum number of transactions value applies individually to each account returning transactions.

8. **Returned Transaction Information**

   The SLA should detail for the client which of the following optional fields that may be returned for a transaction are changed to ‘‘Mandatory, if available’’, and must be returned to the client by the server creating the response message:

   - **valueDateTimeCentral**
   - **valueDateTimeClearing**
   - **accountOwnerReference**
   - **accountServiceInstitutionReference**
   - **supplementaryDetailsReference**
   - **transactionTypeIdentificationReference**
   - **charges**
   - **narrative**
   - **beneficiary**
   - **originator**

9. **Selection By Content**

   The SLA should define those areas of the transaction information that are available to the client to be selected as part of the request for transaction information.

10. **Timezone for Date and Time**

    Where the client and server are situated in different timezones, the SLA should agree the default local timezone in which all requests for information and the responses to those requests are expressed.

Index