# EXHIBIT OSS – 25

## TAG API Reference Guide

BELLSOUTH

BELLSOUTH DOCUMENT
BD-TAG-API-R7.5.0.10-28
ISSUE 28 FEBRUARY, 2001
RELEASE 7.5.0.10

# Telecommunications Access Gateway (TAG) API Reference Guide Part A

*For Release 7.5.0.10*

# Proprietary Statement

BellSouth Telecommunications reserves the right to revise this document for any reason, including but not limited to, conformity with standards promulgated by various government or regulatory agencies, utilization of advance in the state of the technical arts, or the reflection of changes in the design of any equipment, techniques, or procedures described or referred to herein.

LIABILITY TO ANYONE ARISING OUT OF USE OR RELIANCE UPON ANY INFORMATION SET FORTH HEREIN IS EXPRESSLY DISCLAIMED, AND NO REPRESENTATIONS OR WARRANTIES, EXPRESSED OR IMPLIED, ARE MADE WITH RESPECT TO THE ACCURACY OR UTILITY OF ANY INFORMATION SET FORTH HEREIN.

This document is not to be construed as a suggestion to any manufacturer to modify or change any of its products, nor does this document represent any commitment by BellSouth Telecommunications to purchase any product whether or not it provides the described characteristics.

Nothing contained herein shall be construed as conferring by implication, estoppel or otherwise, any license or right under any patent, whether or not the use of any information herein necessarily employs an invention of any existing or later issued patent.

## DOCUMENTATION PRODUCTION AND CONTROL
## NOTICE OF DOCUMENTATION UPDATE

Project Name Telecommunications Access Gateway

Type of Update

( ) Initial Release of New Document

(X) Maintenance Release to Existing Document

| | |
|---|---|
| Documentation Title | Telecommunications Access Gateway (TAG) API Reference Guide |
| Documentation Identification | BD-TAG-API-R7.5.0.10-28 |
| Issue/Revision, Date | Issue 28 FEBRUARY, 2001 |
| First/Current Compatible System Release/Version | Release 7.5.0.10 |
| Date Required at Destination | FEBRUARY, 2001 |

Supervisor
Initials/Date

Originator
Initials/Date

Project Approval

Date

**Reason For Update/Comments; Associated MRs:**

See Section 1.5 for History of changes.

Document Updated by:
Integrated Learning Solutions



Project funding year: 2001

---

# TAG API Reference Guide

## TABLE OF CONTENTS

# Figures

# Tables

# 1. About This Guide

## 1.1 Description

This document serves as a programmer's guide for the BellSouth Telecommunications Access Gateway (TAG) Client Application Program Interface (API).

## 1.2 Intended Audience

This guide is intended for developers who need to use the TAG Client API to develop applications that interface with the TAG Gateway.

## 1.3 Assumptions and Caveats

This document provides the reader with the information necessary for writing C++ applications using the TAG Client API.

It is expected that the reader have experience with the C++ language, as well as the knowledge needed to use their Integrated Development Environment (IDE). Specifically, the reader must know how to create and maintain makefiles or project files necessary for compiling and linking C++ programs.

The reader's knowledge of Pre-Order and Firm Order functionality is necessary to use this technology proficiently. This knowledge is also necessary to use the TAG Client APIs. No business rules are described in this guide. It is up to the reader to know and understand the parameters expected for each query and response.

It is helpful, but not essential, that the reader understand the client-server architecture model since the CLEC's application and the CLEC Notification Server will communicate with the TAG Server in a client-server environment.

Although the TAG Client API enforces encryption algorithms, security features, communication protocols, and uses Corba objects, the reader is not expected to have any knowledge of these technologies. The TAG Client API hides these facilities so the developer can focus their attention on Pre-Order and Firm Order tasks without worrying about the underlying architecture.

## 1.4 Organization

This guide is organized as described below. As of Release 7.1, this document is divided into two parts.

**PART A**

1. **About This Guide** provides an overview of this document, its audience, style, organization, and where to go for additional information. It also provides a summary or running history of changes between releases.

2. **Product Introduction** describes the TAG Client API and its relationship to the client application and TAG Gateway.
3. **TAG Client API** describes the C++ class library that implements the API.
4. **Configuration** explains how to configure the TAG Client API for runtime use.
5. **CLEC Notification Server** describes how to configure, start and stop the CLEC Notification Server.
6. **Appendix A - C++ Header Files** is a listing of the header files included by the client application in order to use the TAG Client API.
7. **Appendix B - Sample Code** contains all the source code for the Test Client. The code will illustrate how the API may be invoked by the client application.
8. **Appendix C - Errors** contains both message ids and error messages generated within the API.
9. **Appendix D - Pre-Installation Checklist** list items to be performed before installation.
10. **Appendix E - TAG Client API UNIX Installation** describes the script used to install the UNIX client software.
11. **Appendix F - TAG Client API Windows Installation** describes the client installation process on Windows.
12. **Appendix G - UNIX Kernel Parameters** lists UNIX kernel parameters.
13. **Appendix H - Test Client** explains how to run the Test Client application.

PART B

14. **Appendix I Test Cases for Pre-Order**
15. **Appendix J Firm Order Test Data**

## 1.5 History of TAG Changes

This Section describes the History of TAG releases and changes to the underlying Client API interface versions. It identifies potential CLEC application coding changes that will need to take place.

The following table represents changes to the underlying interfaces, which the Client API uses. All changes in the table, from one version of an interface to another, are source code compatible (i.e., CLEC source code will not have to change when compiling with the new version of the API) except where noted.

In general, if an interface version does not change from one release to another, a CLEC can continue to use their existing API to interface with a BellSouth TAG Server that supports the same version of the interface. In these cases, no changes or re-linking of their application is required.

When interface versions change, they are source code compatible with the prior version (this does not take into account how validations may require new fields to be populated), the CLEC may re-compile their Application with the new API, without code modification. This allows CLEC servers to communicate with the corresponding BellSouth TAG Server. They must then configure their TAG Client Application to interface with the new BellSouth TAG Server release.

In cases where the interface versions change and they require source code changes to be made, the CLEC must make the necessary code changes before re-compiling their Application with the new API, in order to communicate with the corresponding BellSouth TAG Server. They must then configure their TAG Client Application to interface with the new BellSouth TAG Server release.

## Table 1- TAG PreOrder Version History

| Release | xstAddress Validation | xstAppointment Scheduling | xstCustomer Record | XstService Availability | xstTelephone Number Assignment | xstLoop |
|---|---|---|---|---|---|---|
| 2.0.X | | | | | | |
| 2.1 – 2.1.0.7 | 1 | 1 | 1 | 1 | 1 | |
| 2.1.0.8 | 1 | 1 | 1 | 1 | 1 | |
| 2.2 | 2 | 1 | 1 | 1 | 1 | |
| 2.2.0.1 | 2 | 1 | 1 | 1 | 1 | |
| 2.2.0.2 | 2 | 1 | 1 | 1 | 1 | |
| 3.1 | 2 | 1 | 1 | 1 | 1 | |
| 3.1.0.1 | 2 | 1 | 1 | 1 | 2 | |
| 3.1.0.5 | 2 | 1 | 1 | 1 | 2 | |
| 3.1.1 | 2 | 1 | 1 | 1 | 2 | |
| 3.1.1.1 | 2 | 1 | 1 | 1 | 2 | |
| 7.1 | 2 | 1 | 1 | 1 | 2 | 1 |
| 7.1.0.1 | 2 | 1 | 1 | 1 | 2 | 1 |
| 7.1.1 | 2 | 1 | 1 | 1 | 2 | 1 |
| 7.1.2 | 3 | 1 | 1 | 1 | 2 | 1 |
| 7.1.2.1 | 3 | 1 | 1 | 1 | 2 | 1 |
| 7.5 | 3 | 1 | 1 | 1 | 2 | 1 |
| 7.5.0.10 | 3 | 1 | 1 | 1 | 2 | 1 |

**Table 2- TAG Firm Order Version History**

| Firm OrderVersion of Interface by Release | | | |
|---|---|---|---|
| **Release** | **xstOrder** | **xstOrder Notification** | **xstaDueDate** |
| **2.0.X** | | | |
| **2.1 – 2.1.0.7** | 1 | 1 | |
| **2.1.0.8** | 1a[1] | 1 | |
| **2.2** | 2[2] | 2 | |
| **2.2.0.1** | 2a[3] | 2 | |
| **2.2.0.2** | 2a | 2 | |
| **3.1** | 4[4] | 3[5] | |
| **3.1.0.1** | 4 | 3 | |
| **3.1.0.5** | 4 | 3 | |
| **3.1.1** | 4 | 4[6] | |
| **3.1.1.1** | 5 | 4 | |
| **7.1** | 6 | 5 | |
| **7.1.0.1** | 6 | 5 | |
| **7.1.1** | 6 | 5 | 2 |
| **7.1.2** | 8 | 6 | 2 |
| **7.1.2.1** | 8 | 6 | 3 |
| **7.5** | 9[7] | 6 | 3 |
| **7.5.0.10** | 9 | 6 | 3 |

Each cell in the above tables represents the interface numbering scheme for each interface a CLEC Application will use to perform PreOrder and Firm Order tasks. The column headings are the names of the interfaces, while the rows show which number each interface uses for a particular TAG release. For example, in TAG release 2.2.0.2, the xstOrder interface is xstOrder2a. For the TAG release 3.1 the xstOrder interface was modified so its new interface is xstOrder4. While compiling CLEC Applications it is important that the correct interface number be used for the respective TAG release.

[1] The ***orderLoopPort()*** method is not source code compatible with version, 1. If a CLEC uses this method, they must modifiy their Application code, before using this version of the TAG Client API.
[2] The ***orderLoopPort()*** method is not source code compatible with version, 1a. If a CLEC uses this method, they must modifiy their Application code, before using this version of the TAG Client API. It is, however, source code compatible with version 1.
[3] The ***orderLoopPort()*** method is not source code compatible with version, 2. If a CLEC uses this method, they must modifiy their Application code, before using this version of the TAG Client API. It is, however, source code compatible with version 1a.
[4] The xstOrder and xstOrderNotification interfaces are not source code compatible with earlier versions. If a CLEC uses this interface they must modify their application code before using this version of the TAG Client API.
[5] Same verbiage as footnote 4.
[6] Same verbiage as footnote 4.
[7] The order() and getDueDate() methods are not source code compatible with earlier versions due to the addition of the DID record. If a CLEC uses these methods they must modify their application code before using this version of the TAG Client API.

### 1.5.1 TAG 7.5.0.10 Issue 28 from 7.5.0.10 Issue 26

Summary of Changes

| Section | Description |
|---|---|
| PART A | |
| Various | Changed Issue number |
| 2.4.3 | Added patch number "44" to ORBIXMT 3.0.1 |
| 2.5.2 | Added two macros |
| 2.5.3 | Corrected two library name typos |

### 1.5.2 TAG 7.5.0.10 Issue 26 from 7.5 Issue 25

Summary of Changes

| Section | Description |
|---|---|
| PART A | |
| Various | Changed 7.5 to 7.5.0.10 |
| Various | Changed Issue number from 25 to 26 |
| Various | Changed HP-UX 11.0 to HP-UX 11.00 |
| 1.5 | Added 7.5.0.10 version history to Tables 1 and 2 |
| 2.4.1 | Added patch number to Orbix version |
| 2.4.2 | Updated Orbix version and added patch number |
| 2.5.1 | Deleted the "-" in: xstClientApi –xscClientApi and xstCorbaClientStub - xswPv (TAG Client API) |
| 2.5.2 | Deleted the apostrophe in: RW_DON'T_USE_MEMPOOL |
| 3.5.2.7 | Updated struct Loop script to add two new fields; rz and cz |
| 6.10 | Updated xstaLoop.h to add LMU data items |
| 10.1.55 | Updated Orbix Home Directory to 3.0.1 |

### 1.5.3 TAG 7.5 Issue 25 from 7.5 Issue 24

Summary of Changes

| Section | Description |
|---|---|
| PART A | |
| Various | Updated the Issue numbers |
| Various | Updated the HP-UX version number to 11.00 |

### 1.5.4 TAG 7.5 Issue 24 from 7.5 Issue 23

Summary of Changes

| Section | Description |
|---|---|
| PART A | |

| | |
|---|---|
| Various | Changed Issue 23 to 24 |
| 2.5.2 | Added Note about Sun compiler option |

### 1.5.5 TAG 7.5 Issue 23 from 7.1.2.1

Summary of Changes

| Section | Description |
|---|---|
| PART A | |
| Various | Changed 7.1.2.1 to 7.5 |
| Various | Changed Issue 22 to 23 |
| 1.5 | Added a row for 7.5 to Table 1 |
| 1.5 | Added a row for 7.5 to Table 2 |
| 1.5 | Added a footnote for 9 in the new row for 7.5 in Table 2 |
| 2.4.1 | Updated software requirement version numbers |
| 2.4.1 | Replaced ANSI CFRONT C++ Complier to HP aCC C++ Compiler |
| 2.4.2 | Updated version numbers |
| 2.5.3 | Removed /D XST_EXPORTING flag |
| 2.5.3 | Added 6 other flags to the bottom of list |
| 2.5.3 | Added one row to Library table at the end |
| 2.5.3 | Changed the static link library file |
| 2.5.3 | Changed the dynamic link library file |
| 3.5.3.1 | Added a line in the xstaOrder::order section |
| 3.5.3.1 | Added a line in the xstaOrder::getDueDate section |
| 3.5.3.1 | Removed the bullet item, "Used to submit any one of the seven Order types". |
| 4.1 | Corrected grammer |
| 6.6 | Updated xstaAppointmentScheduling.h |
| 6.7 | Updated xstaCustomerRecord.h |
| 6.8 | Updated xstaServiceAvailability.h |
| 6.10 | Updated xstaLoop.h |
| 6.11 | Updated xstaDueDate.h |
| 6.12 | Updated xstaOrder.h |
| 8.3 | Added, changed and removed various rows from Table 17 |
| Appendix B | Updated Sample Code xstTestClient.C |

| Section | Description |
|---|---|
| PART B | |
| Various | Changed 7.1.2.1 to 7.5 |
| Various | Changed Issue 22 to 23 |
| Tables | Removed Table of Contents because all the tables are in Part A |

| | |
|---|---|
| Figures | Removed Table of Contents because all the figures are in Part A |
| 3.8.1 | Added the line "RequestType = loop_makeup_working;" |
| 3.8.1 | Removed the first two lines of the Output File |
| 3.8.2 | Added the line "RequestType = loop_makeup_spare;" |
| 3.8.2 | Removed the first two lines of the Output File |
| 3.8.3 | Removed the first two lines of the Output File |
| 3.8.4 | Replaced the first line in the Output File |

### 1.5.6 TAG 7.1.2.1 from 7.1.2

Summary of Changes

| Section | Description |
|---|---|
| | PART A |
| Various | Changed 7.1.2 to 7.1.2.1. |
| Various | Changed Issue 21 to 22 |
| Various | Updated page numbers where it says, "see configuration section on page..." |
| 1.5 | Added a row for 7.1.2.1 to Table 1 and Table 2 |
| 3.6.3 | Added line to Data Structures. |
| 4.1 | Modified the values for CLEC API configuration data.<br>Added an example. |
| 5.1 | Modified the syntax for CLEC Notification Server. |
| 6.11 | Updated sctDueDate number to "3".<br>Added line under XST_DECLSPEC |

| Section | Description |
|---|---|
| | PART B |
| Various | Changed 7.1.2 to 7.1.2.1 |
| 3.3-3.5 | Added SLTN line |
| 2.10 | Added npt = P; under xstTestClient Input for ESDQ |

### 1.5.7 TAG 7.1.2 from 7.1.1

Summary of Changes

| Section | Description |
|---|---|
| | PART A |
| Various | Changed 7.1.1 to 7.1.2 |
| 2.5.3 | Removed Orbix flag |

| | |
|---|---|
| Table 13 | Moved 'Request a formatted LSR' up one row – it belongs to xstaOrder |
| 3.5.3.2 | Updated struct and remarks in Order notification |
| 6.5 | Updated xstaAddressValidation.h |
| 6.9 | Updated xstaTelephoneNumberAssignment.h |
| 6.11 | Updated xstaOrder. h header file |
| 6.12 | Updated xstaOrderNotification.h header file |
| 8.3 | Two errors from a previous release were moved up to fit in alphabetically |

### 1.5.8 TAG 7.1.1 from 7.1.0.1

Summary of Changes

| Section | Description |
|---|---|
| PART A | |
| Various | fixed typos |
| 1.5 | Added xstDueDate to the Interface classes table |
| 2.6 | Added ESDQ to the Terminology table |
| 3.2 | Adjusted the Pre-Order Calculated Due Date class list to refer to xstaDueDate instead of xstOrder |
| 3.4 | Added xstaDueDate to the Class Hierarchy diagram |
| 3.5.2.2 | Deleted HouseNumberPrefix from the Address structure of the AddressValidation API |
| 3.5.2.8 | Added this section for the new Pre-Order class: xstaDueDate |
| 6.5 | Updated xstaAddressValidation.h with the new version |
| 6.11 | Added this section to define the new xstaDueDate.h header file |
| 6.12 | Updated xstaOrder.h with a new version |
| 7 | Updated the xstTestClient.C program with the new version |

| Section | Description |
|---|---|
| PART B | |
| Various | removed HouseNumberPrefix from all I/O messages |
| 2.10 | Added a section for the ESDQ input/output data |

### 1.5.9 TAG 7.1.0.1 from 7.1

Summary of Changes

| Section | Description |
|---|---|
| PART A | |

| | |
|---|---|
| Various | Updated 7.1 to 7.1.0.1 |
| Table 1 | Updated with 7.1.0.1 info |
| Table 2 | Updated with 7.1.0.1 info |
| 3.5.3.1 | Added View LSR<br>Added xsta::Status getFormattedLsr |
| Appendix B | New xstTestClient.C |
| 8.3 | Two new Firm Order validation errors added |
| 13.1.1 | Added description of the TAG – COG connection |
| 13.1.3 | Added View formatted LSR to Table 19 |

## TAG 7.1 from 3.1.1.1

Summary of Changes

Global: Non-technical edits to increase readability

**NOTE: As of Release 7.1,** the TAG API Guide has been divided into into two parts – Part A comprises of Section 1 – 13. Part B comprises of the Test cases for Pre Order and Firm Order.

| Section | Description |
|---|---|
| PART A | |
| 1.5 | Updated Interface Numbers and added xstaLoop as a new interface |
| 1.5.1 | Added this table |
| 2.2 | Added Look Makeup Pre Order query |
| 2.4 | Version upgrades:<br>OrbixMT – 2.3.4<br>Solaris – 2.5.1<br>Visual C++ - 5.0 |
| 2.5 | Modified the Linker macros |
| 2.5.1 | Added new libraries for HP, SUN and Windows NT platforms |
| 3.1 | new Pre Order LoopMakeup query |
| 3.2 | new class, xstaLoop |
| 3.4 | new class xstaLoop in hierarchy |
| 3.2.5.7 | Added the class description: xstaLoop |
| 3.5.3.1 | LSR structure change |
| 5.1 | ClecNotification configuration file change |
| 6.10 | xstaLoop.h added |
| 6.11 | xstaOrder.h – changed with a new interface number |
| 7 | xstTestClient.C – changed |
| 13.1.3 | Added four new Loop Queries |

| PART B | |
|---|---|
| 14 | add new loop query I/O samples |

## 1.5.10 TAG 3.1.1.1 from 3.1.1

Summary of Changes

| Section | Description |
|---|---|
| 1.4 | Fixed grammar |
| 1.5 | Added page break between tables 1 and 2 to fix footnote formatting. |
| 2.2 | Query order rearranged |
| 2.6 | Updated terminology to include all acronyms |
| 2.7 | Updated the Support Contact Informatiom |
| 3.1 | Query order rearranged |
| 3.2 | Query order rearranged |
| 3.3.2 | Fixed the line number reference |
| 3.4 | Changed class diagram to display classes in a consistent order with other queries listed in this guide. |
| 3.5.3.1 | Added dueDate parameter to the respective xstaOrder methods, and italicized parameter names |
| 6.10 | xstaOrder.h – updated to the new version |
| 8 | Added Error TAGR1392VAL, and TAGT0200VAL |
| 10.1.6 | Fixed the heading for the table of contents |
| 10.1.6.2 | Fixed the heading for the table of contents |
| 10.1.6.3 | Fixed the heading for the table of contents |

## 1.5.11 TAG 3.1.1 from 3.1.0.5

Summary of Changes

| Section | Description |
|---|---|
| 2.6 | Terminology updated |
| 3.2 | Fixed Class Description |
| 3.5.2.6 | XstaTelephoneNumberAssignment – updated remarks on class method |
| 4 | Fixed grammar |
| 5.2 | Fixed spelling of Daemon |
| 6.2 | xstaTAP.h updated for new version |
| 6.11 | XstaOrderNotification.h – updated to reflect the new version |
| 7 | xstTestClient.C – updated to the latest version |
| 8.1 | Added description of API error messages |
| 8.2 | Added description of Pre-Order error messages |
| 8.3 | Added description of Firm Order error messages |

| | |
|---|---|
| 8.4 | Added description of TAG Gateway error messages |
| 8.3 | Added new LNP Error messages (9000 series) |
| 14.2 | Changed NPA-TTX to NPA-NXX |
| 14.3 | Fixed description of Customer Service test case |
| 14.3.5 | Fixed description of test case |
| 14.4 | Fixed description of Service Availability test case |
| 14.5.2 | Expanded "TN" to Telephone Number to match others |
| 14.5.3 | Expanded "TN" to Telephone Number to match others |
| 14.8 and 14.8.2 | Corrected spelling mistake |

### 1.5.12 TAG 3.1.0.5 from 3.1.0.1

Summary of Changes

| Section | Description |
|---|---|
| 8.3 | Changes Error Message TAGS3105VAL |

### 1.5.13 TAG 3.1.0.1 from 3.1

Summary of All Changes by Interface Type

| Function | Summary of Changes |
|---|---|
| **PRE-ORDER** | |
| Address Validation | None |
| Appointment Availability | None |
| Customer Record | None |
| Service Availability | None |
| Telephone Number – Direct-in-Dial | None |
| Telephone Number – General Pool | None |
| Telephone Number - Multi-line Hunt | None |
| Telephone Number – Miscellaneous<br><br>(see Appendix A for the data structures defined in the API header files.) | Added GetAvailableMiscTelephoneNumbers() method to the XstTelephoneNumberAssignment object. |
| **ORDER** | |
| Order | None |
| Directory Listing | None |
| Number Portability | None |
| Loop | None |
| Loop – Number Portability | None |
| Loop – Port | None |
| Port | None |
| Resale | None |
| Notifications | None |
| PreOrder Due Date | None |

### 1.5.13.1 TAG 3.1.0.1 from 3.1

Summary of Changes

| Section | Description |
|---|---|
| various | some formatting changes made |
| 3.5.1.1 | routingInfo( ) method added to xstaServerConnection class<br>clecId( ) method changed in xstaServerConnection class |
| 3.5.2.6 | GetAvailableMiscTelephoneNumbers( ) method added to xstaTelephoneNumberAssignment class |
| 6.1 | xstaCommon.h – changed |
| 6.3 | XstaServerConnection.h – changed |
| 6.9 | XstTelephoneNumberAssignment.h – changed |
| 7 | xstTestClient.C – changed |
| 13.1.3 | Added TNAQ_MISC to Table 19 |
| 14.8 | Added TelephoneNumberAssignment (Miscellaneous) examples |

### 1.5.14 TAG 3.1

Summary of All Changes by Interface Type

| Function | Summary of Changes |
|---|---|
| **PRE-ORDER** | |
| Address Validation | None |
| Appointment Availability | None |
| Customer Record | None |
| Service Availability | None |
| Telephone Number – Direct-in-Dial | None |
| Telephone Number – General Pool | None |
| Calculated Due Date | DELETED |
| Telephone Number - Multi-line Hunt | None |
| **ORDER** | |
| Order<br><br>(see Appendix A for the data structures defined in the API header files) | The new function, order(), replaces all of the orderLoop(), orderLoopNp(), orderNp(), orderResale(), orderPort(), orderDirectoryListingAndAssistance(), and orderLoopPort(). |
| Directory Listing<br><br>(see Appendix A for the data structures defined in the API header files) | This function is now obsolete, replaced with the new function order(). It is still available in this release but takes an additional argument. |
| Number Portability<br><br>(see Appendix A for the data structures defined in the API header files) | This function is now obsolete, replaced with the new function order(). It is still available in this release but takes an additional argument. |
| Loop | This function is now obsolete, replaced with the new function order(). It is still available in this release but |

| | |
|---|---|
| (see Appendix A for the data structures defined in the API header files) | takes an additional argument. |
| Loop - Number Portability<br><br>(see Appendix A for the data structures defined in the API header files) | This function is now obsolete, replaced with the new function order(). It is still available in this release but takes an additional argument. |
| Loop – Port<br><br>(see Appendix A for the data structures defined in the API header files) | This function is now obsolete, replaced with the new function order(). It is still available in this release but takes an additional argument. |
| Port<br><br>(see Appendix A for the data structures defined in the API header files) | This function is now obsolete, replaced with the new function order(). It is still available in this release but takes an additional argument. |
| Resale<br><br>(see Appendix A for the data structures defined in the API header files) | This function is now obsolete, replaced with the new function order(). It is still available in this release but takes an additional argument. |
| Notifications<br><br>(see Appendix A for the data structures defined in the API header files) | The data structures and member functions have changed drastically to support OSS'99. |
| PreOrder Due Date<br><br>(see Appendix A for the data structures defined in the API header files) | The new function, getDueDate(), provides Pre-Order due date. |

### 1.5.14.1 TAG 3.1 from 2.2.0.2

Summary of Changes

| Section | Description |
|---|---|
| various | Added new verbage to better describe each section |
| various | Changed formatting to be consistant throughout |
| various | Removed reference to "Interim" Number Portability |
| 1.5 | Interfaces Changed:<br>xstaOrder (2a to 4)<br>xstaOrderNotification (2 to 3) |
| 1.5.1.1 | This "Summary of Changes" section is new as of 3.1 |
| 2.2 | Added Calculated Due Date to the Pre-Order query list |
| 3.3.2 | Explained the while loop sample code |
| 3.4 | Removed xstaDueDate from the diagram |
| 3.5.2.1 – 3.5.2.4 | Removed references to xstaDueDate |
| 3.5.2.5 | Removed the xstaDueDate class description |

| | |
|---|---|
| 3.5.2.5 (new section) | Removed references to the xstaDueDate |
| 3.5.2.6 | Removed references to the xstaDueDate |
| 3.5.3.1 | xstaOrder class changes are as follows: |
| | Removed Retail( ) |
| | Added order( ) |
| | Added getDueDate( ) |
| | orderLoop( )<br>removed LoopList argument<br>added DSDL argument<br>added LOOP argument |
| | orderLoopWithInp( ) renamed to orderLoopWithNp( )<br>added DSDL argument<br>added LSNP argument<br>removed DirectoryList argument<br>removed LoopInpList argument |
| | Deleted orderRetailBundled( ) |
| | orderResale( )<br>removed DirectoryList argument<br>removed ResaleList argument<br>added DSDL argument<br>added Resale argument |
| | orderPort( )<br>removed DiretoryList argument<br>removed PortList argument<br>added PORT argument<br>added DSDL argument |
| | OrderDirectoryListingAndAssistance( )<br>removed DirectoryList argument<br>removed ResaleList argument<br>added DSDL argument |
| | orderLoopPort( )<br>removed DirectoryList argument<br>removed LoopList argument<br>removed PortList argument<br>added DSDL argument<br>added PORT argument<br>added Resale argument |
| 3.5.3.2 | XstaOrderNotification class are as follows: |
| | readNotification( )<br>removed blocking integer<br>changed Notification argument to NotificationList argument |

| | |
|---|---|
| | Added readNotificationList( ) |
| | Added acknowledgeNotificationList( ) |
| 5.1 | Added "GroupList" parameter to the xst_ClecNotification configuration file. |
| 5.2 – 5.3 | Separated instructions by UNIX and MS-Windows |
| 5.6.4 | Changed the Transaction Log Records T8 – T10. |
| 6.1 | Changed xstaCommon.h<br>added PreOrderDueDate=181 (among other changes) |
| 6.3 | Changed xstaServerConnection.h |
| 6.8 | Deleted xstaDueDate.h |
| 6.9 | Changed xstaOrder.h |
| 6.11 | Changed xstaOrderNotification.h |
| 7 | Changed the xstTestClient utility |
| 8 | Errors:<br>Numbers are now 11 characters, new ones added and some text changes were made<br>Sorted tables by Error Number |
| 9 | Removed BSAFE from the Pre Installation Check List |
| 13.1 | xstTestClient:<br>query_type command line argument removed (the query_type is now part of the input file) |
| 13.1.3 | REQTYPES:<br>LOOPINP changed to LOOPNP<br>INP changed to NP<br>RETAIL deleted<br>ORDER added |
| 13.1.4 | xstTestNotification added –g option |
| 14.8 | Added Pre-Order Calculated Due Date example |
| 15.8 | Added Firm Order Calculated Due Date example |

## 1.6 Print Type Styles

This guide uses the following print type styles.

**Table 3- Print Type Styles**

| Convention | Indicates | Example |
|---|---|---|
| **Bold** | In prototypes, methods and data types | **xstaServerConnection(**<br>**const char ****applid*,<br>**const char ****applpass*,<br>**const char ****userid* **);** |
| *Italic* | File name or directory | The *bin* directory contains |
| *Italic* | Arguments | *applid* |
| `Monospaced` | Text that is displayed exactly as is. For example, source code, input/output data and error messages | `for(i=0; i<10; i++)...` |

## 1.7 Environmental Factors

Environmental factors can result in the program output and screens appearing different from those illustrated in this guide.

For UNIX® clients, the directory represented by the environment variable $XST_HOME is the root of the TAG directory tree.

# 2. Product Introduction

## 2.1 Background

As part of the Telecommunications Act of 1996, the Federal Communication Commission (FCC) requires Incumbent Local Exchange Carriers (ILECs) to make Operational Support Systems (OSSs) available to Competitive Local Exchange Carriers (CLEC). This type of gateway access enables the CLECs to provide customer service that can equal the service offered by the ILEC.

The FCC order mandates that the gateway be an electronic, machine-to-machine interface that does not rely on human intervention in the ultimate transfer of information from the ILEC OSSs to the CLEC.

In addition, the gateway must provide standard interfaces so all carriers can develop their OSS infrastructure while continuing to exchange information necessary to service customers in a competitive local service market.

BellSouth offers the TAG Gateway as its transaction-based interface between BellSouth OSSs and clients at external organizations. Typically these clients wish to do business with the BellSouth Wholesale organization, and the gateway enables access for the purpose of providing Pre-Order and Order functionality. The gateway provides protocol translation and security services, but does not directly provide any of the business functionality.

The Pre-Order and Order functionality enable local service providers and network providers to exchange information about current and future services, unbundled network elements, and combinations of network elements. The Pre-Order functionality permits a service negotiator to assemble the data required to provide service to a customer.

## 2.2 Description

The TAG Client API is a C++ class library that provides the ability to implement client applications that interface with the TAG Gateway. The TAG Client API transparently provides basic request validation, security, server connection management, and encryption/decryption of transmitted data.

The TAG Client API allows the application to run as a Common Object Request Broker Architecture (CORBA) client interacting with the TAG Gateway that is running as a CORBA server using the IONA Technologies PLC ORBIX® implementation of the CORBA standard.

Client applications using the TAG Client API may be run on one of following computing platforms: SUN with Solaris 7, HP9000 with HP-UX™ 11.00, or a Pentium®-based PC with Windows NT™ 4.0 or Windows® 95.

The client application built with the TAG Client API communicates with the TAG Gateway. The TAG Gateway provides the following gateway

services:

- Throttle control
- Authentication/authorization
- Data mapping
- Transaction/error logging
- Encryption/decryption
- Message Routing

**Pre-Order** queries supported by TAG Client API:

- Address Validation
- Appointment Scheduling
- Customer Record
- Service Availability
- Telephone Number Assignment.
- Due Date Calculation
- Loop Makeup

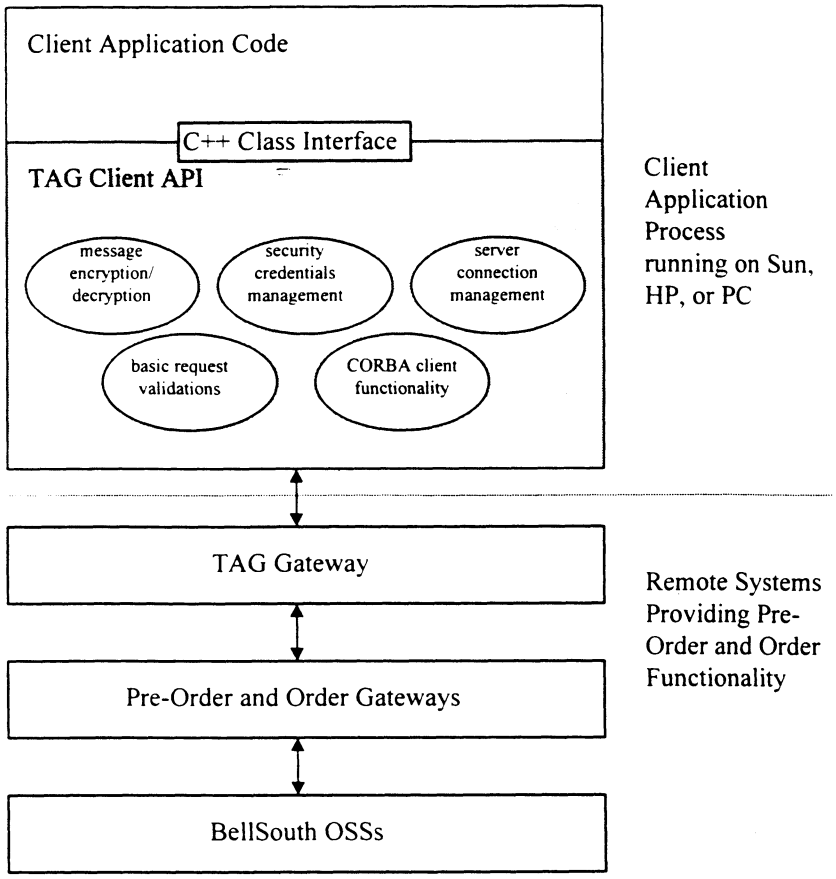
**Firm Order** functionality supported by TAG Client API:

- Submit a Local Service Request (LSR).
- Request a list of Purchase Order Numbers.
- Request Service Order Status
- Receive notification of Rejects, Firm Order Confirmation, and Completion Notices.

## 2.3 Components

Figure 1 represents a high-level overview of the TAG architecture. In particular, this figure illustrates the relationship of the client application and the TAG Client API.

**Figure 1 TAG Components**



## 2.4 Required Software

The tables in this section list the software systems necessary to use the TAG Client API in a development and a run-time environment on the supported platforms. It is the customer's responsibility to obtain this software and all necessary licenses.

A run-time environment is one in which no C++ development will occur. This information is included so the CLEC can develop their applications on a separate machine from their production machine.

### 2.4.1 HP Client Environment

**Table 4- HP Client Development Environment**

| Component | Release | License |
|---|---|---|
| HP UX | 11.00 | Yes |
| HP Softbench | 6.5X | Yes |
| HP aCC C++ Compiler | A.03.13 | Yes |
| ORBIXMT | 3.0.1 OTM Patch 44 | Yes |
| HP Gold and Gold+ Tape | 4.1 | Yes |

**Table 5- HP Client RunTime Environment**

| Component | Release | License |
|---|---|---|
| HP UX | 11.00 | Yes |
| ORBIXMT | 3.0.1 OTM Patch 44 | Yes |
| HP Gold and Gold+ Tape | 4.1 | Yes |

### 2.4.2 Sun Client Environment

**Table 6- Sun Client Development Environment**

| Component | Release | License |
|---|---|---|
| Sun Solaris | 7 | Yes |
| Visual Workshop for C++ | 5.0 | Yes |
| SunPro | 5.0 | Yes |
| ORBIXMT | 3.0.2 OTM Patch 45 | Yes |

**Table 7- Sun Client RunTime Environment**

| Component | Release | License |
|---|---|---|
| Sun Solaris | 7 | Yes |
| ORBIXMT | 3.0.1 OTM Patch 45 | Yes |

2.4.3 MS-Windows 95/NT Client Environment

**Table 8- Windows 95/NT Client Development Environment**

| Component | Release | License |
|---|---|---|
| MS Windows | NT 4.0 | Yes |
| MS Visual C++ | 6.0 | Yes |
| ORBIXMT | 3.0.1 OTM Patch 44 | Yes |

**Table 9- Window 95/NT Client RunTime Environment**

| Component | Release | License |
|---|---|---|
| MS Windows | NT 4.0 | Yes |
| ORBIXMT | 3.0.1 OTM Patch 44 | Yes |

## 2.5 Compiler and Linker Requirements

The following compiler and linking information in this section must be used on each of the client development platforms.

2.5.1 HP

All C++ code linked with the TAG Client API library must be compiled with CC +eh.

The following macros must be defined during compilation of client code (with the –D command line option):

- _REENTRANT
- _HPUX_SOURCE
- _THREAD_SAFE
- RW_MULTI_THREAD
- RW_POSIX_THREADS
- HAVE_BOOL
- PTHREAD_SAFE

- RWSTD_MULTI_THREAD
- RW_DONT_USE_MEMPOOL
- RW_NO_STL
- VAR_FIX
- _KERNEL_THREADS
- _POSIX_C_SOURCE=199506L
- _TMPROTOTYPES
- _HPACC_USING_MULTIPLES_IN_FUNCTIONAL

The following libraries must be linked with the client code:

(with the –l command line option):

- orbixmt (ORBIX)
- pthread stream Csup (HP)
- xstClientApi xscClientApi xstCorbaClientStub xswPv (TAG Client API)
- bsafe (BSAFE libraries)
- rwtool (Rogue Wave)
- xscRxp
- CorbaxscCogBridgeClientStub
- xstCorbaONClientStub
- xstCorbaLookupClientStub
- xstCorbaLookupCommonClientStub and
- bsafe (RSA)
- ldap lber.

### 2.5.2 SUN

Note: There is a temporary bug in the SUN compiler. In order to compile and link with TAG 7.5 for the SUN machine, a special compiler option must be used. The option is: xO0 (small x, capital O, zero).

The following macros must be defined during compilation of client code (with the –D command line option):

- __EXTENSIONS__
- THREAD_SAFE
- RW_MULTI_THREAD
- RWSTD_MULTI_THREAD
- RW_NO_STL
- RW_DONT_USE_MEMPOOL
- RW_POSIX_THREADS
- HAVE_BOOL
- _TMPROTOTYPES
- RW_NO_EXCEPTIONS

The following compiler flags must be used during compilation of client code:

- -mt
- -library=rwtools7,iostream

The following libraries must be linked with the client code:

(with the –l command line option):

- orbixmt (ORBIX)
- xstClientApi xscClientApi
  xstCorbaClientStub xswPv (TAG Client API)
- bsafe (BSAFE libraries)
-
- pthread
- socket
- nsl
- thread
- CorbaxscCogBridgeClientStub
- xscRxp CorbaxscCogRelayClientStub
- ldapssl40

### 2.5.3 Windows NT

- The following macros must be defined during compilation of client code: /D WIN32
- /MD
- /D WINDOWS

- 
- /D RW_NO_STL
- /D XSTA_TRACE
- /D XST_DLL
- /D ORBIX_DLL
- /D RSA_PLATFORM=RSA_I386_486
- /D RW_DONT_USE_MEMPOOL
- /D RW_MULTI_THREAD

The following libraries must be linked with the client code:

- CorbaxscCogBridgeClientStub.lib
- xscClientApi.lib
- xscCnsApi.lib
- xscRxp.lib
- nsldapssl32v30.lib
- wsock32.lib
- kernel32.lib
- user32.lib
- gdi32.lib
- winspool.lib
- comdlg32.lib
- advapi32.lib
- shell32.lib
- ole32.lib
- oleaut32.lib
- uuid.lib
- odbc32.lib
- odbccp32.lib

The following libraries must be linked with the client application to use the TAG API software system:

| Library | Description |
|---|---|
| Itmi.lib | Orbix |
| xstClientApi.lib<br>or<br>xstClientApidll.lib | TAG APIs<br>(see the section below on "Static verses Dynamic Link Libraries to determine which of these libraries to use). |
| xstConfig.lib<br>or<br>xstConfigdll.lib | (see the section below on "Static verses Dynamic Link Libraries to determine which of these libraries to use). |
| | |
| xstCorbaClientStub.lib | TAG Corba |
| xswPV.lib | TAG Program Validations |
| Bsafe43.lib | BSAFE |
| Tls12d.lib | Rogue Wave |
| wsock32.lib | Standard Visual C++ |
| user32.lib | Standard Visual C++ |
| advapi32.lib | Standard Visual C++ |

#### 2.5.3.1 Static verses Dynamic Link Libraries

There are two types of API libraries supported in the MS-Windows environment;

- Static Libraries: If a program uses an API stored in a static library, that API is copied into the program at compile/link time. This makes the resulting program much larger (on the disk and in memory). The advantage, there is only one file (the executable program) to be shipped to the production machine.

- Dynamic Link Libraries (DLL). Dynamic Link Libraries include APIs that are not copied into an executable at compile/link time, but rather brought into memory only when called by the program during execution. The advantage is less disk space and memory is wasted for each API, especially when several programs using the same API are running simultaneously. One disadvantage is the Dynamic Link Library must be shipped to the production machine along with the executable programs that use the DLL. A second disadvantage of DLLs is there are a few extra steps to linking and running a program that uses a DLL. It is the reader's responsibility however, to know these steps.

For customer convenience, the TAG Client API is supplied as both, a static library and a dynamic link library.

To use the static library, link with the file,

- xstClientApi.lib
- xstConfig.lib

To use the dynamic link library, link with the file;

- xstClientApidll.lib
- xstConfigdll.lib

## 2.6 Terminology

This guide uses the following terms and acronyms.

**Table 10- Terminology**

| Term | Description |
|---|---|
| AAQ | Appointment Availability Query requests appointment availability information based on the NPA/NXX transmitted by the CLEC. |
| API | Application Program Interface. |
| AVQ | Address Validation Query verifies a customer address. |
| CDD | Due Date calculation query provides a due date for completion of a service request based on the desired due date. |
| CLEC | Competitive Local Exchange Carrier (or Company) is a carrier or company that is certified by the appropriate public service commission to provide local exchange service within a given state. |
| CLLI | Common Language Location Identifier is the standard identifier of a specific location for a piece of equipment. The CLLI is sometimes referred to as the Local Service Termination (LST.) |
| CORBA | Common Object Request Broker Architecture. |
| CSRQ | Customer Record Query is used to obtain customer information including customer billing. |
| DLL | Dynamic Link Library |
| ESDQ | Estimated Service Date Query |
| FCC | Federal Communications Commission |
| IDE | Integrated Development Environment |
| ILEC | Incumbent Local Exchange Carrier (or Company) is the primary company that provides local exchange service. |
| LSR | Local Service Request |
| NP | Number Portability |
| NPA | Numbering Plan Area is the area code portion of a telephone number. |
| NXX | Number Exchange is the exchange portion of a telephone number. |
| OSS | Operational Support Systems. |
| PON | Purchase Order Number |
| PV | Programmable Validation |
| SAQ | Service Availability Query is used to obtain product, feature and PIC information for a specific BellSouth switch. |

| Term | Description |
|---|---|
| SOS | Service Order Status |
| TAG | Telecommunications Access Gateway. |
| TNAQ | Telephone Number Assignment Query requests and reserves telephone numbers. |
| TNCAN | Telephone Number Cancellation requests the cancellation of a reservation. |
| TNSQ | Telephone Number Selection Query requests a change to a previous reservation. |

## 2.7 Support

Should you need support for a specific component, refer to the table below for your contact.

**Table 11- Support**

| Component | Contact | Phone Number |
|---|---|---|
| TAG Production | BellSouth SPOC – EC Support | 1-888-462-8030 |
| TAG Testing | SAIC | 404-927-2825 (primary)<br>404-927-3331 (alternate) |
| TAG Client API | BellSouth/Onsite SAIC | |

# 3. TAG Client API

## 3.1 Introduction

The TAG Client API is a C++ class library that allows a client application to interface with the TAG Gateway while hiding the interface details from the client application programmer. The TAG Client API hides the following:

Underlying communication details

Management of security credentials

Much of the TAG Gateway server connection details

Encryption/decryption schemes and the associated DES security key management

For **Pre-Order** request the TAG Client application can generate the following:

Address Validation Queries (AVQ)

Appointment Availability Queries (AAQ)

Customer Record Queries (CSRQ)

Service Availability Queries (SAQ)

Telephone Number Assignment Queries (TNAQ)

Due Date Calculation (CDD)

LoopMakeup

For **Firm Order** requests the TAG Client application can generate the following:

LOOP Services Order Request

LOOP Services with Number Portability (NP) Order Request

Number Portability (NP) Order Request

Resale Order Request

PORT Services Order Request

Directory Listing and Assistance Order Request

LOOP/PORT Combination Services Order Request

Purchase Order Number (PON) List Query

Service Order Status Query

In general, to use the API the client application should follow these steps:

- Construct the query object
- Supply the query arguments
- Invoke the query method
- Catch any exceptions thrown by the method

In processing a request, the TAG Client API manages the following steps:

Performs basic validations for each request type, checking for missing and invalid data.

Provides security through authentication and authorization checks.

Connects to the appropriate TAG Gateway server.

Performs encryption/decryption of data transmitted between the client application and the TAG Gateway.

## 3.2 Class Overview

The base class for common functionality is the ***xstaServerConnection*** object. It records information used for server connection, e.g TAG Security Server name and host name. These items and the CLEC ID are read from a configuration file upon the first attempted connection to ***readConfig()***.

The API's are broken down into 7 classes that support both Pre-Order and Firm Order functions:

**Table 12- Pre-Order Classes**

| Pre-Order Function | Class Name |
|---|---|
| Address Validation | xstaAddressValidation |
| Appointment Availability | xstaAppointmentScheduling |
| Customer Record | xstaCustomerRecord |
| Service Availability | xstaServiceAvailability |
| Telephone Number Assignment | xstaTelephoneNumberAssignment |
| Due Date Calculation | xstaDueDate |
| Loop Makeup | xstaLoop |

Firm Order classes handled by the API.

**Table 13- Firm Order Classes**

| Order Function | Class Name |
|---|---|
| Submit a Local Service Request (LSR). | xstaOrder |
| Request a list of Purchase Order Numbers. | |
| Request Service Order Status | |
| Request a formatted LSR | |
| Receive notification of Rejects, Firm Order Confirmation, and Completion Notices. | xstaOrderNotification |
| Request a formatted LSR | |

The client application code provides each query class with essential information, such as Ids and passwords, when the object is instantiated.

Multiple query objects (of the same or different class) can exist simultaneously in the client application.

Multiple threads can use query objects, but the object can only be used by one thread and can't be shared among threads.

## 3.3 Common Data Structures

This section describes the TAG Client API data structures that require special handling.

### 3.3.1 xstaString Class

XstaString is a straightforward string class that dynamically allocates sufficient memory and frees it when done. xstaString supports the following operations:

xstaString( );
Default constructor initializes to the empty string.

xstaString( const char *str );
Initializes to the null terminated string at str. If str==0, initializes to the empty string.

void operator = ( const char *str );
Assigns to the null terminated string at str. If str==0, assigns to the empty string.

void operator += ( const char *str );
Appends to the current value the null terminated string at str. Does nothing if str==0.

operator const char * ( ) const { return data_; }
Returns a pointer to the null terminated string data.

int isEmpty ( ) const { return data_ && *data_; }
Returns 0 if the string is not empty or 1 if it is empty.

### 3.3.2 Lists

Both in and out arguments and exceptions make use of lists. Lists are declared with the xstaLIST_DECLARATION macro. The **length()** member must be used to find out how many elements are present in an "out" argument whereas **length(n)** must be used to set the number of elements in an "in" argument. Thus the length( ) method that takes no arguments is the **"get"** method, while the length( ) method that expects one argument is used as the **"set"** method.

The subscript operator may be used to either read or write the element. Subscript beyond the current length() of the list results in undefined behavior.

Example using "/* out */ xstaCustomerRecord::CustomerTextList &customerTextList":

```
unsigned long i;
for (i=0; i < customerTextList.length(); i++) {
cout << (const char *) customerTextList[i];
}
```

Example using "/* in */ const xstaTelephoneNumberAssignment::TelephoneNumberList &telephoneNumbers":

```
1. unsigned long i = 0;
2. while( ... ) {
3.      telephoneNumbers.length( i + 1 );
4.      telephoneNumbers[i] = newNumber;
5. }
```

The above code snippet loops through the Telephone Number list. The first statement inside the loop (line 3) sets the length of the telephoneNumber List to (i+1). The second statement (line 4) sets the ith element of telephoneNumber to the value of "newNumber."

### 3.3.3 Scope

The Client API uses many types (both typedefs and structs) defined in the scope xstaCommon (see xstaCommon.h in "Appendix 1 - C++ Header Files)." Client code must say xstaCommon::Status to declare a structure.

```
xstaCommon::Status queryStatus;
```

### 3.3.4 Data-Type Checking

C++ is not C. The C++ language enforces a much more stringent set of rules on data-types. The C++ compiler will complain if one data-type is used in an expression where a different data-type is expected.

Please pay particular attention to the data-types required by the TAG APIs.

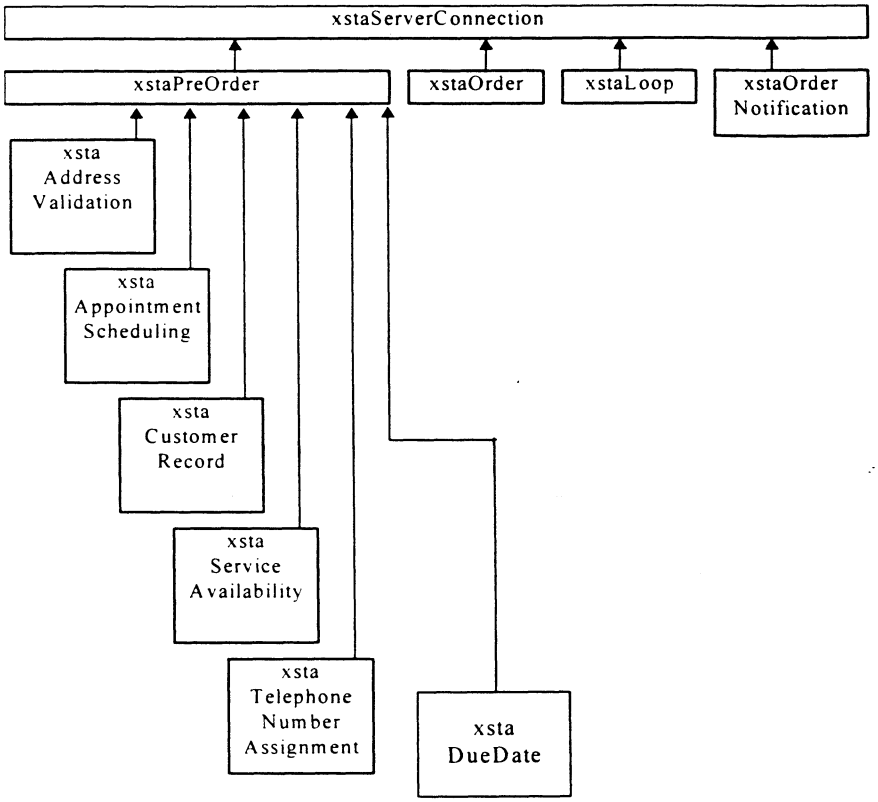
## 3.4 Class Hierarchy

All classes are derived from the base class xstaServerConnection.

The xstaServerConnection class handles a connection to a particular server. The derived classes make use of this class to connect and reconnect to their respective servers.

The xstaPreOrder class performs validations common to all Pre-Order query classes.

**Figure 2 Class Hierarchy**



## 3.5 Class Descriptions

This section describes the C++ classes that constitute the TAG Client API. For each class, the following is provided (where applicable):

Inheritance path

- Description of the class
- Included files needed to use the class
- "See Also" list of related classes
- Methods and data members publicly available from the class

### 3.5.1 Base Classes

The TAG Client API provides one base class, xstaServerConnection. All other TAG Client API classes can be traced back to this class.

#### 3.5.1.1 xstaServerConnection

Inheritance

> xstaServerConnection

Description

The xstaServerConnection base class provides common functionality and handles a connection to a particular server. The derived query classes make use of this class to connect and reconnect to their respective servers.

Include Files

```
#include "xstaServerConnection.h"
```

Methods

| XstaServerConnection :: applicationId |
|---|

```
void applicationId(/* in */ const char *applid ):
```

Remarks

- Sets the value of *applid.*
- *applid* must match value stored in the BellSouth security system.
- Re-acquires security credentials after a SecurityException occurs with an exception type of AuthenticationFailed or ThrottleControlFailed.
- Updates local value when *applid* changes in the BellSouth security system.

```
const char *applicationId( );
```

Remarks

- Gets the value of *applid.*

XstaServerConnection :: applicationPassword

xstaServerConnection::applicationPassword

void applicationPassword(
/* in */ const char **applpass* );

Remarks

- Sets the value of *applpass*.
- *applpass* must match value stored in The BellSouth security system.
- Re-acquires security credentials after a SecurityException occurs with an exception type of AuthenticationFailed or ThrottleControlFailed.
- Updates local value when *applpass* changes in the BellSouth security system.

```
xstaServerConnection::applicationPassword
      const char *applicationPassword();
```

Remarks

- Gets the value of applpass.

xstaServerConnection::userId

void userID(

/* in */ const char **userid* );

Remarks

- Sets the value of *userid.*
- *userid* is an arbitrary value logged by the TAG Gateway.

xstaServerConnection::userId

const char *userId( );

Remarks

- Gets the value of *userid.*

xstaServerConnection::clecId

void clecId(
/* in */ const char **ClecId* );

Remarks

- Sets the per-object value of *ClecId.*
- Updates local value when API configuration file changes (see

Configuration section on page 84). Changes to the configuration file are not loaded into the API automatically.

- Overrides the value of *ClecId* loaded from the API configuration file.

### xstaServerConnection::apiVersion

```
static const char *apiVersion( );
```

Remarks

- Returns the current version of the API.

### xstaServerConnection::changePassword

```
void changePassword (
/* in */ const char *applid,
/* in */ const char *oldPassword,
/* in */ const char *newPassword
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

Remarks

- Used to change the password for *applid* from *oldPassword* to *newPassword*

### xstaServerConnection::routingInfo

```
void routingInfo (
const xstaCommon::AttributeList &routing_list
);
```

Data Structures

routing_list

routing_list is a list of xstaCommon::Attribute

Return Value

none

Remarks

- Sets the routing information list used when acquiring credentials from the TAG Security Server.
- Overrides any routing information provided in the configuration file.
- Re-acquires security credentials at next request.

xstaServerConnection::finalize

```
static void finalize ( );
```

Remarks

- Performs ORBIX cleanup before exiting.
- Required only on the Windows NT platform; has no effect on other platforms.

xstaServerConnection::surrenderCredentials

```
void surrenderCredentials ()
throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::GatewayTimeout
);
```

Remarks

- If an extended period of no activity (i.e., overnight) is known, avoids the server idle time-out. Idle time-out raises a SecurityException.

xstaServerConnection::readConfig

```
static void readConfig( )
throw (
        xstaCommon::InvalidData
);
```

Remarks

- Updates local configuration values when API configuration file changes (see "Configuration" section on page 84.). Changes to the configuration file are not loaded into the API automatically.
- Called automatically at the first instantiation of a derived class.

### 3.5.2 Pre-Order Functionality Classes

The following 6 classes make up the Pre-Order business rules. The ***xstaPreOrder*** class inherits from ***xstaServerConnection*** and serves as the base class for the other 6 classes in this section.

#### 3.5.2.1 xstaPreOrder

**Inheritance**

> xstaServerConnection

> xstaPreOrder

Description

The xstaPreOrder base class performs validations common to all Pre-Order query classes.

### Include Files

```
#include "xstaPreOrder.h"
```

### Methods

No public methods exist for this class.

## 3.5.2.2 xstaAddressValidation

### Inheritance

> xstaServerConnection

> xstaPreOrder

> xstaAddressValidation

### Description

The xstaAddressValidation class is for Address Validation Queries (AVQ).

### Include Files

```
#include "xstaAddressValidation.h"
```

### See Also

xstaAppointmentScheduling
xstaCustomerRecord

xstaServiceAvailablity
xstaTelephoneNumberAssignment

### Methods

#### xstaAddressValidation::xstaAddressValidation

```
xstaAddressValidation (
/* in */ const char *applid,
/* in */ const char *applpass,
/* in */ const char *userid
) throw ( );
```

### Remarks

- Creates an instance of xstaAddressValidation.
- Sets initial values from provided strings.
- If any string is empty or zero, the corresponding set method must be called before issuing the first query. For example: xstaServerConnection::userid.
- *applid* and *applpass* must match values stored in the BellSouth security system.
- *userid* is an arbitrary value logged by the TAG Gateway.

## xstaAddressValidation::~xstaAddressValidation

```
~xstaAddressValidation( )
throw (
        xstaCommon::SecurityException,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

### Remarks

- Destroys an instance of xstaAddressValidation.
- Internally calls xstaServerConnection::surrenderCredentials.

## xstaAddressValidation::verifyWithAddress

```
xstaCommon::Status verifyWithAddress (
/* in */ const xstaCommon::Address &queryAddress,
/* in */ const char *inquiryNumber,
/* out */ xstaCommon::MsgHeader &messageHeader,
/* out */ xstaAddressValidation::AlternativeAddressList &alternativeAddresses,
/* out */ xstaAddressValidation::AddressInfoHeader &addressInfoHeader
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

### Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*queryAddress:*

```
struct Address {
 HouseNumber      houseNumber;
HouseNumberSuffix houseNumberSuffix;
StreetDirectional streetDirectional;
StreetThoroughfare streetThoroughfare;
StreetName       streetName;
StreetSuffix     streetSuffix;
Room             room;
Building         building;
Floor            floor;
DescriptiveLocation descriptiveLocation;
City             city;
State            state;
ZipCode          zipCode;
UnnumberedHouse  unnumberedHouseIndicator;
CrossBoundary    crossBoundary;
PostalBox        postalBox;
Route            route;
RateZone                 rateZone;
DriveInstructions driveInstructions;
CompanyIndicator  companyIndicator;
AddressPatternList addressPattern;
Address ();
};
```

*messageHeader:*

```
struct MsgHeader {
xstaString inquiryNumber;
DateTime  dateSent;
};
```

*alternativeAddresses:*

AlternativeAddressList is a list of AlternativeAddressInfo:

```
struct AlternativeAddressInfo {
AlternativeAddress  alternativeaddress;
NpaNxx npanxx;
LocalServiceTermination          localServiceTermination;
ServiceAddressTelephoneInfoList                    telephoneInfo;
AreaTransferInfo      areaTransferInto;
};
```

*addressInfoHeader:*

```
struct  AddressInfoHeader {
ServiceRestrictionInfoList        serviceRestrictionInfoList;
xstaString serviceInstructionText;
};
```

Return Value

```
struct Status {
xstaString msgId;
xstaString msgTxt;
};
```

Remarks

- Performs Address Validation given a street address.
- Providing a value for *inquiryNumber* indicates a resend. A null or blank *inquiryNumber* indicates an initial send. The *inquiryNumber* is returned in *messageHeader*.
- To find out what type is in each element of the AlternativeAddressList, use alternativeAddresses[i]._d() as discriminant.

```
xstaAddressValidation::AlternativeAddressList
   altAddrs;
unsigned long i;
for (i=0; i < altAddrs.length(); i++) {
  switch (altAddrs[i].alternativeaddress._d()) {
  case xstaCommon::RangedChoice:
     // access altAddrs[i].
     // alternativeaddress.addresswithrange();
     break;
  case xstaCommon::FieldedChoice:
     // access altAddrs[i].
     // alternativeaddress.fieldedaddress();
     break;
  }
}
```

If the length() function returns zero, no addresses were found.

- queryAddress.houseNumberPrefix is not used.

Example

See "Appendix B - Sample Code".

## xstaAddressValidation::verifyWithTelephone

```
xstaCommon::Status verifyWithTelephone (
/* in */ const char *queryTelephone,
/* out */ xstaCommon::MsgHeader &messageHeader,
/* out */ xstaAddressValidation::AlternativeAddressList &alternativeAddresses
/* out */ xstaAddressValidation::AddressInfoHeader &addressInfoHeader
) throw (
        xstaCommon::SecurityException,
```

xstaCommon::InvalidData,
xstaCommon::BackendResourceLimitation,
xstaCommon::GatewayTimeout
);

Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*messageHeader:*

See description in xstaAddressValidation::verifyWithAddress.

*alternativeAddresses:*

See description in xstaAddressValidation::verifyWithAddress.

*addressInfoHeader:*

See description in xstaAddressValidation::verifyWithAddress.

Return Value

See description in xstaAddressValidation::verifyWithAddress.

Remarks

- Performs Address Validation given a telephone number.
- See description in xstaAddressValidation::verifyWithAddress to handle the AlternativeAddressList.
- Only returns FieldedChoice or empty list only. Does not return Ranged Choice.

Example

See "Appendix B - Sample Code".

### 3.5.2.3 xstaAppointmentScheduling

**Inheritance**

> xstaServerConnection

> xstaPreOrder

> xstaAppointmentScheduling

Description

The xstaAppointmentScheduling class is for Appointment Scheduling Queries (AAQ).

Include Files

#include "xstaAppointmentScheduling.h"

See Also

xstaAddressValidation
xstaCustomerRecord

xstaServiceAvailablity
xstaTelephoneNumberAssignment

## Methods

### xstaAppointmentScheduling::xstaAppointmentScheduling

```
xstaAppointmentScheduling (
/* in */ const char *applid,
/* in */ const char *applpass,
/* in */ const char *userid)
throw ( );
```

## Remarks

- Creates an instance of xstaAppointmentScheduling.
- Sets initial values from provided strings.
- If any string is empty or zero, the corresponding set method must be called before issuing the first query (e.g., xstaServerConnection::userid()).
- *applid* and *applpass* must match values stored in the BellSouth security system.
- *userid* is an arbitrary value logged by the TAG Gateway.

### xstaAppointmentScheduling::~xstaAppointmentScheduling

```
~xstaAppointmentScheduling( )
throw (
        xstaCommon::SecurityException,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

## Remarks

- Destroys an instance of xstaAppointmentScheduling.
- Internally calls xstaServerConnection::surrenderCredentials.

### xstaAppointmentScheduling::verifyAvailAppts

```
xstaCommon::Status verifyAvailAppts (
/* inout */ xstaString &npanxx,
/* out */ xstaCommon::MsgHeader &messageHeader,
/* out */ xstaAppointmentScheduling::AppointmentData &availableAppointments
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

## Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*messageHeader:*

```
struct MsgHeader {
```

```
xstaString inquiryNumber;
DateTime  dateSent;
};
```

*availableAppointments:*

```
struct AppointmentData {
AvailableDaysList coAvailability;
AvailableDaysList pvAvailability;
IntervalList interval;
HolidayList holidays;
CloseDateList closeDates;
};
```

Return Value

```
struct Status {
xstaString msgId;
xstaString msgTxt;
};
```

Remarks

- Retrieves a list of available appointments for the given npa/nxx list.

Example

See "Appendix B - Sample Code".

### 3.5.2.4 xstaCustomerRecord

**Inheritance**

> xstaServerConnection

> xstaPreOrder

> xstaCustomerRecord

Description

The xstaCustomerRecord class is for Customer Record Queries (CSRQ).

Include Files

```
#include "xstaCustomerRecord.h"
```

See Also

```
xstaAddressValidation
xstaAppointmentScheduling
xstaServiceAvailablity
xstaTelephoneNumberAssignment
```

Methods

xstaCustomerRecord::xstaCustomerRecord

```
xstaCustomerRecord (
/* in */ const char *applid,
/* in */ const char *applpass,
/* in */ const char *userid
) throw ( );
```

Remarks

- Creates an instance of xstaCustomerRecord.

- Sets initial values from provided strings.
- If any string is empty or zero, the corresponding set method must be called before issuing the first query. For example: xstaServerConnection::userid.
- *applid* and *applpass* must match values stored in the BellSouth security system.
- *userid* is an arbitrary value logged by the TAG Gateway.

### xstaCustomerRecord::~xstaCustomerRecord

```
~xstaCustomerRecord( )
throw (
        xstaCommon::SecurityException,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

Remarks

- Destroys an instance of xstaCustomerRecord.
- Internally calls xstaServerConnection::surrenderCredentials.

### xstaCustomerRecord::retrieveCustomerRecord

```
xstaCommon::Status retrieveCustomerRecord (
/* in */ const xstaCustomerRecord::CustomerRequest &customerRequest,
/* out */ xstaCommon::MsgHeader &messageHeader,
/* out */ xstaCustomerRecord::CustomerTextList &customerTextList
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*customerRequest:*

```
struct CustomerRequest {
xstaString  stateCode;
xstaString customerCode;
xstaString  exchangeCompanyCircuitId;
xstaCommon::TelephoneNumber          accountTelephoneNumber;
xstaString agencyAuthorizationStatus;
xstaString authorizationName;
xstaCommon::DateTime authorizationDate;
unsigned char        lsilndicator;
CustomerRequest();
};
```

*messageHeader:*

```
struct MsgHeader {
xstaString inquiryNumber;
DateTime  dateSent;
};
```

*customerTextList:*

CustomerTextList is a list of CustomerText:

```
typedef xstaString CustomerText;
```

Return Value

```
struct Status {
xstaString msgId;
xstaString msgTxt;
};
```

Remarks

- Retrieves a list of text strings for a Customer Record.
- When invoked using *stateCode* and *exchangeCompanyCircuitId*, a telephone number is returned rather than a customer record. The client application must reinvoke the method using the returned telephone number in order to retrieve the customer record.
- The CustomerRequest constructor sets *lsiIndicator* to 0 (false); the client Application should set it to non-zero to make an LSI request.

Example

See "Appendix B - Sample Code".

### 3.5.2.5 xstaServiceAvailability

Inheritance

> xstaServerConnection

> xstaPreOrder

> xstaServiceAvailability

Description

The xstaServiceAvailability class is for Service Availability Queries (SAQ).

Include Files

```
#include "xstaServiceAvailability.h"
```

See Also

```
xstaAddressValidation
xstaAppointmentScheduling
xstaCustomerRecord
xstaTelephoneNumberAssignment
```

Methods

**xstaServiceAvailability::xstaServiceAvailability**

```
xstaServiceAvailability (
/* in */ const char *applid,
/* in */ const char *applpass,
/* in */ const char *userid
) throw ( );
```

Remarks

- Creates an instance of xstaServiceAvailability.
- Sets initial values from provided strings.
- If any string is empty or zero, the corresponding set method must be called before issuing the first query. For example: xstaServerConnection::userid.
- *applid* and *applpass* must match values stored in the BellSouth security system.
- *userid* is an arbitrary value logged by the TAG Gateway.

### xstaServiceAvailability::~xstaServiceAvailability

```
~xstaServiceAvailability( )
throw (
        xstaCommon::SecurityException,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

Remarks

- Destroys an instance of xstaServiceAvailability.
- Internally calls xstaServerConnection::surrenderCredentials.

### xstaServiceAvailability::verifyService

```
xstaCommon::Status verifyService (
/* inout */ xstaString &clli,
/* in */ const char *npanxx,
/* in */ const char *picServiceOffering,
/* in */ const xstaServiceAvailability::ServiceAbbreviationList
&ServiceAbbreviationList,
/* out */ xstaCommon::MsgHeader &messageHeader,
/* out */ xstaServiceAvailability::SwitchInfo &switchInfo,
/* out */ xstaServiceAvailability::ProductFeatureInfoList &productFeatureInfoList,
/* out */ xstaServiceAvailability::CarrierInfoList &carrierInfoList,
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*messageHeader:*

```
struct MsgHeader {
xstaString inquiryNumber;
DateTime        dateSent;
};
```

*switchInfo:*

```
struct SwitchInfo {
xstaString      switchType;
```

```
xstaString          isdnIndicator;
SwitchNpaNxxList    SwitchNpaNxxList;
xstaString          eightHundredServingOffice;
xstaString          watsServingOffice;
SwitchCliiList      switchCliiList;
};
```

*productFeatureInfoList:*

ProductFeatureInfoList is a list of ProductFeatureInfo:

```
struct ProductFeatureInfo {
xstaString productId;
xstaString productName;
xstaString featureTitle;
xstaString centralOfficeFeatureAvail;
xstaCommon DateTime featureEffectiveDate;
xstaString accessNumber;
xstaString tariffStatus;
xstaCommon DateTime tariffEffectiveDate;
xstaString..taiffNotes
ProductFeatureUsocList productFeatureUsocList;
};
```

*carrierInfoList:*

CarrierInfoList is a list of CarrierInfo:

```
struct CarrierInfo {
xstaString cic;
xstaString serviceOfferingType;
xstaString accessCarrierName;
xstaString accessCarrierNameAbbreviation;
xstaString accessCarrierTelephoneNumber;
};
```

Return Value

```
struct Status {
xstaString msgId;
xstaString msgTxt;
};
```

Remarks

- Provides Service Availability for a given CLLI and NPA-NXX.
- ServiceAbbreviationList allows Client Application to query by specific service abbreviation.

Example

See "Appendix B - Sample Code".

### 3.5.2.6 xstaTelephoneNumberAssignment

Inheritance

> xstaServerConnection

> xstaPreOrder

> xstaTelephoneNumberAssignment

Description

The xstaTelephoneNumberAssignment class is for Telephone Number Assignment Queries (TNAQ).

Include Files

```
#include "xstaTelephoneNumberAssignment.h"
```

## See Also

xstaAddressValidation
xstaAppointmentScheduling
xstaCustomerRecord
xstaServiceAvailability

## Methods

### xstaTelephoneNumberAssignment:: xstaTelephoneNumberAssignment

```
xstaTelephoneNumberAssignment (
/* in */ const char *applid,
/* in */ const char *applpass,
/* in */ const char *userid
) throw ( );
```

## Remarks

- Creates an instance of xstaTelephoneNumberAssignment.
- Sets initial values from provided strings.
- If any string is empty or zero, the corresponding set method must be called before issuing the first query. For example: xstaServerConnection::userid.
- *applid* and *applpass* must match values stored in the BellSouth security system.
- *userid* is an arbitrary value logged by the TAG Gateway.

### xstaTelephoneNumberAssignment:: ~xstaTelephoneNumberAssignment

```
~xstaTelephoneNumberAssignment( )
throw (
        xstaCommon::SecurityException,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

## Remarks

- Destroys an instance of xstaTelephoneNumberAssignment.
- Internally calls xstaServerConnection::surrenderCredentials.

### xstaTelephoneNumberAssignment::getAvailableTelephoneNumbers

```
xstaCommon::Status getAvailableTelephoneNumbers (
/* in */ const char *npanxx,
/* in */ unsigned short quantityRequested,
/* in */ const char *exceptionChar,
/* in */ xstaTelephoneNumberAssignment::TNOptions &options,
/* in */ const char *requestedNumberLow
/* in */ const char *typeOfService
/* inout */ xstaString &clli,
/* out */ xstaCommon::MsgHeader &messageHeader,
/* out */ xstaString &confirmationNumber,
/* out */ xstaTelephoneNumberAssignment::TelephoneNumberList &telephoneNumbers
) throw (
```

```
                xstaCommon::SecurityException,
                xstaCommon::InvalidData,
                xstaCommon::BackendResourceLimitation,
                xstaCommon::GatewayTimeout
    );
```

Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*options:*

```
enum TNOptions{None, Easy, Coin, SeqLine, AscendingLineDigits,
DescendingLineDigits, IdenticalLineDigits};
```

*messageHeader:*

```
struct MsgHeader {
xstaString inquiryNumber;
DateTime  dateSent;
};
```

*telephoneNumbers:*

TelephoneNumberList is a list of TelephoneNumber:

```
typedef xstaString TelephoneNumber;
```

Return Value

```
struct Status {
xstaString msgId;
xstaString msgTxt;
};
```

Remarks

- Retrieves a list of telephone numbers that satisfy the query.

Example

See "Appendix B - Sample Code".

## xstaTelephoneNumberAssignment:: getAvailableMLHTelephoneNumbers

```
xstaCommon::Status getAvailableMLHTelephoneNumbers (
/* in */ const char *npanxx,
/* in */ const char *leadTelephoneNumber,
/* in */ unsigned short quantityInTerminals,
/* in */ unsigned short quantityOutTerminals,
/* inout */ xstaTelephoneNumberAssignment::TerminalCodeList
&existingMLHNumberList,
/* in */ const char *lastInTerminal,
/* in */ const char *lastOutTerminal,
/* inout */ xstaString &clli,
/* out */ xstaCommon::MsgHeader &messageHeader,
/* out */ xstaTelephoneNumberAssignment::HuntGroupNumberList
&huntGroupNumberList,
) throw (
            xstaCommon::SecurityException,
            xstaCommon::InvalidData,
            xstaCommon::BackendResourceLimitation,
            xstaCommon::GatewayTimeout
);
```

Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*existingMLHNumberList*

TerminalCodeList is a list of TerminalCode:

```
typedef xstaString TerminalCode;
```

*messageHeader:*

See description in xstaTelephoneNumberAssignment::getAvailableTelephoneNumbers

*huntGroupNumberList:*

HuntGroupNumberList is a list of HuntGroupNumber:

```
struct HuntGroupNumber {
MLHNumber huntGroupNumber;
xstaCommon::TelephoneNumber leadTelephoneNumber;
TerminalRange inTerminalRange;
TerminalRange  outTerminalRange;
};
```

Return Value

See description in xstaTelephoneNumberAssignment::getAvailableTelephoneNumbers.

Remarks

- Retrieves a list of terminal ranges that satisfy the query.
- There may be 0 or 1 elements in existingMLHNumberList on input and 0, 1, or 2 elements on output.

Example

See "Appendix B - Sample Code".

## xstaTelephoneNumberAssignment:: getAvailableDIDTelephoneNumbers

```
xstaCommon::Status
getAvailableDIDTelephoneNumbers (
/* in */ const char *npanxx,
/* in */ xstaTelephoneNumberAssignment::TelephoneNumber telephoneNumber,
/* in */ const char *exceptionChar,
/* in */ unsigned short quantityRequested,
/* in */ long routeIndex,
/* inout */ xstaString &clli,
/* out */ xstaCommon::MsgHeader &messageHeader,
/* out */ xstaString &confirmationNumber,
/* out */ unsigned short &quantityProvided,
/* out */ xstaTelephoneNumberAssignment::TelephoneNumberRangeList
&telephoneNumberRangeList,
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*messageHeader:*

See description in xstaTelephoneNumberAssignment::getAvailableTelephoneNumbers

*telephoneNumberRangeList:*

TelephoneNumberRangeList is a list of TelephoneNumberRange:

```
struct TelephoneNumberRange {
xstaCommon::TelephoneNumber lowValue;
xstaCommon::TelephoneNumber highValue;
};
```

### Return Value

See description in xstaTelephoneNumberAssignment::getAvailableTelephoneNumbers.

### Remarks

- Retrieves a list of DID telephone number ranges that satisfy the query.

### Example

See "Appendix B - Sample Code".

## xstaTelephoneNumberAssignment:: getAvailableMiscTelephoneNumbers

```
xstaCommon::Status getAvailableMiscTelephoneNumbers (
        /* in */ const char *npanxx,
        /* in */ const char *city,
        /* in */ const char *state,
        /* in */ unsigned short quantityRequested,
        /* out */ xstaCommon::MsgHeader &messageHeader,
        /* out */ unsigned short &quantityProvided,
        /* out */ xstaCommon::NpaNxxList &npanxxList,
        /* out */ xstaTelephoneNumberAssignment::TelephoneNumberList
&telephoneNumberList
        ) throw (
                xstaCommon::SecurityException,
                xstaCommon::InvalidData,
                xstaCommon::BackendResourceLimitation,
                xstaCommon::GatewayTimeout
);
```

### Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

```
messageHeader:
struct MsgHeader {
        xstaString inquiryNumber;
        DateTime  dateSent;
        };
```

```
npanxxList:
npanxxList is a list of xstaCommon::NpaNxx:
typedef xstaString NpaNxx;

telephoneNumberList:
TelephoneNumberList is a list of TelephoneNumber:
typedef xstaString TelephoneNumber;
```

### Return Value

```
struct Status {
        xstaString msgId;
        xstaString msgTxt;
        };
```

### Remarks

- If city and state are provided, returns a list of miscellaneous NPANXXs.
- If npanxx is provided, returns a list of up to 25 miscellaneous account numbers.

### Example

See "Appendix B - Sample Code".

## xstaTelephoneNumberAssignment:: selectAvailableTelephoneNumbers

```
xstaCommon::Status selectAvailableTelephoneNumbers (
/* in */ const char *npanxx,
/* in */ const xstaTelephoneNumberAssignment::TelephoneNumberList
&telephoneNumbers,
/* in  */ xstaTelephoneNumberAssignment::TelephoneOption option,
/* in */ const char *confirmationNumber,
/* in */ const char *reserveUntilDate,
/* inout */ xstaString &clli,
/* out */ xstaCommon::MsgHeader &messageHeader
) throw (
        xstaCommon::SecurityException,
        xstCommon::InvalidData,
        xstCommon::BackendResourceLimitation,
        xstCommon::GatewayTimeout
);
```

### Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*telephoneNumbers:*

TelephoneNumberList is a list of TelephoneNumber:

```
typedef xstaString TelephoneNumber;
```

*option:*

```
enum TelephoneOption{ TN, DID };
```

*messageHeader:*

See description in xstaTelephoneNumberAssignment::getAvailableTelephoneNumbers

Return Value

See description in xstaTelephoneNumberAssignment::getAvailableTelephoneNumbers.

Remarks

- Extends the reservation up to 1 year for a single TN, a list of TNs, or a confirmation number.

Example

See "Appendix B - Sample Code".

## xstaTelephoneNumberAssignment::cancelTelephoneNumbers

```
xstaCommon::Status cancelTelephoneNumbers (
/* in */ const char *npanxx,
/* in */ const char *confirmationNumber,
/* in  */ xstaTelephoneNumberAssignment::TelephoneNumberList
&telephoneNumberList,
/* inout */ xstaString &clli,
/* out */ xstaCommon::MsgHeader &messageHeader
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*telephoneNumberList:*

TelephoneNumberList is a list of TelephoneNumber:

```
typedef xstaString TelephoneNumber;
```

*messageHeader:*

See description in xstaTelephoneNumberAssignment::getAvailableTelephoneNumbers.

Return Value

See description in xstaTelephoneNumberAssignment::getAvailableTelephoneNumbers.

Remarks

- Cancels a reservation by confirmation number.

Example

See "Appendix B - Sample Code".

## xstaTelephoneNumberAssignment::cancelDIDTelephoneNumbers

```
xstaCommon::Status cancelDIDTelephoneNumbers (
/* in */ const char *npanxx,
/* in */ const char *confirmationNumber,
/* in */ xstaTelephoneNumberAssignment::TelephoneNumberRangeList&didRangeList,
/* inout */ xstaString &clli,
/* out */ xstaCommon::MsgHeader &messageHeader
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

### Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*didRangeList:*

TelephoneNumberRangeList is a list of TelephoneNumberRange:

```
struct TelephoneNumberRange {
xstaCommon::TelephoneNumber lowValue;
xstaCommon::TelephoneNumber highValue;
};
```

*messageHeader:*

See description in xstaTelephoneNumberAssignment::getAvailableTelephoneNumbers.

### Return Value

See description in xstaTelephoneNumberAssignment::getAvailableTelephoneNumbers.

### Remarks

- Cancels a reservation of a list of DID telephone numbers.

### Example

See "Appendix B- Sample Code".

## xstaTelephoneNumberAssignment:: cancelMLHTelephoneNumbers

```
xstaCommon::Status cancelMLHTelephoneNumbers (
/* in */ const char *npanxx,
/* in */ const char *return1,
/* in */ const char *return2,
/* inout */ xstaString &clli,
/* out */ xstaCommon::MsgHeader &messageHeader
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
```

```
                xstaCommon::BackendResourceLimitation,
                xstaCommon::GatewayTimeout
        );
```

Data Structures

(For typedefs and minor structures see "Appendix A- C++ Header Files" on page 97.)

*messageHeader:*

See description in xstaTelephoneNumberAssignment::getAvailableTelephoneNumbers.

Return Value

See description in xstaTelephoneNumberAssignment::getAvailableTelephoneNumbers.

Remarks

- Cancels reservation of a multi-line hunt group number.

Example

See "Appendix B - Sample Code".

### 3.5.2.7 xstaLoop

Inheritance

> xstaServerConnection

> xstaLoop

Description

The xstaLoop class is for Loop Makeup Queries

- Working Loop Query
- Spare Loop Query
- Loop Reservation
- Loop Cancel

Include Files

```
#include "xstaLoop.h"
```

See Also

xstaAddressValidation

xstaAppointmentScheduling

xstaCustomerRecord

xstaServiceAvailaility

xstaTelephoneNumberAssignment

## Methods

### xstaLoop::xstaLoop

```
xstaLoop (
        const char *applid,
        const char *applpass,
        const char *userid
        )
throw ();
```

### Remarks

- Creates an instance of xstaLoop.
- Sets initial values from provided strings.
- If any string is empty or zero, the corresponding set method must be called before issuing the first query. For example,

  xstaServerConnection::userId.
- applid and applpass must match values stored in BellSouth security system.
- userid is an arbitrary value logged by the TAG Gateway.

### xstaLoop::~xstaLoop

```
~xstaLoop ()
        throw (
                xstaCommon::SecurityException,
                xstaCommon::BackendResourceLimitation,
                xstaCommon::GatewayTimeout
        );
```

### Remarks

- Destroys an instance of xstaLoop.

### xstaLoop::makeupQueryWorkingLoops

```
xstaCommon::Status makeupQueryWorkingLoops (
        /* in */ const xstaCommon::TestProdIndicator testProdIndicator,
        /* in */ const char        *cc,    // Company Code
        /* in */ const char        *cktId,
        /* in */ const char        *tn,
        /* in */ const xstaLoop::Address &address,
        /* out */ xstaCommon::MsgHeader &messageHeader,
        /* out */ xstaLoop::LoopList  &loopList
        ) throw (
        xstaCommon::SecurityException,
                xstaCommon::BackendResourceLimitation,
                xstaCommon::InvalidData,
                xstaCommon::GatewayTimeout
        );
```

## Data Structures

(For typedefs and minor structures see "Appendix A – C++ Header Files" on page 97).

address:

```
struct Address {
        HouseNumber houseNumber;
        HouseNumberSuffix houseNumberSuffix;
        StreetName streetName;
```

```
        StreetDirectional streetDirectional;
        StreetThoroughfare streetThoroughfare;
        StreetSuffix streetSuffix;
        Building building;
        Floor floor;
        Room room;
        City city;
        State state;
        ZipCode zipCode;
        UnnumberedHouse unnumberedHouseIndicator;
};
```

messageHeader:

```
struct MsgHeader {
xstaString inquiryNumber;
  DateTime dateSent;



};
```

loopList:

```
struct Loop {
        xstaString lpstat; // Status of assembled facility
        xstaString ssc; // Single Subscriber Carrier Indicator;
        xstaString rtf; // Receive/Transmit Indicator
        xstaString rz; // Resistance Zone
        xstaString cz; // Carrier Zone
        FnList FnLIst; // Segment Aggregate List. 1-9 times per loop
};
```

to add the two new fields rz and cz.

## Return Values

```
struct Status {
   xstaString msgId;
   xstaString msgTxt;
};
```

## Remarks

- Retreives a list of Loop aggregates.

## Example

See "Appendix B – Sample Code" on page 151.

### xstaLoop::makeupQuerySpare

```
xstaCommon::Status makeupQuerySpare (
        /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
        /* in  */ const char        *cc,    // Company Code
        /* in  */ xstaLoop::Quantity  numberRequested,
        /* in  */ const char        *nc,
        /* in  */ const char        *nci,
        /* in  */ const char        *secnci,
        /* in  */ const xstaLoop::Address &address,
        /* out */ xstaCommon::MsgHeader &messageHeader,
        /* out */ xstaLoop::LoopList  &loopList
        ) throw (
                xstaCommon::SecurityException,
                xstaCommon::BackendResourceLimitation,
                xstaCommon::InvalidData,
                xstaCommon::GatewayTimeout
        );
```

## Data Structures

(For typedefs and minor structures see "Appendix A –C++ Header Files" on page 97).

address:

See description in xstaLoop::makeupQueryWorkingLoops.

messageHeader:

See description in xstaLoop::makeupQueryWorkingLoops.

loopList:

See description in xstaLoop::makeupQueryWorkingLoops.

Return Values

```
struct Status {
  xstaString msgId;
  xstaString msgTxt;
};
```

Remarks

- Retreives a list of Loop aggregates.

Example

See "Appendix B – Sample Code" on page 151.

## xstaLoop::reservationSpare

```
xstaCommon::Status reservationSpare (
        /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
        /* in  */ const char        *cc,    // Company Code
        /* in  */ xstaLoop::Quantity  numberRequested,
        /* in  */ const char        *nc,
        /* in  */ const char        *nci,
        /* in  */ const char        *secnci,
        /* in  */ const xstaLoop::Address &address,
        /* out */ xstaCommon::MsgHeader &messageHeader,  /* out */
xstaLoop::Quantity  &numberReserved,
        /* out */ xstaString        &frn,
        /* out */ xstaLoop::LoopList  &loopList
        ) throw (
                xstaCommon::SecurityException,
                xstaCommon::BackendResourceLimitation,
                xstaCommon::InvalidData,
                xstaCommon::GatewayTimeout
        );
```

Data Structures

(For typedefs and minor structures see "Appendix A –C++ Header Files" on page 97).

address:

See description in xstaLoop::makeupQueryWorkingLoops.

messageHeader:

See description in xstaLoop::makeupQueryWorkingLoops.

loopList:

See description in xstaLoop::makeupQueryWorkingLoops.

Return Values

```
struct Status {
  xstaString msgId;
  xstaString msgTxt;
};
```

Remarks

- Retreives a list of Loop aggregates.

Example

See "Appendix B – Sample Code" on page 151.

xstaLoop::reservationCancel

```
xstaCommon::Status reservationCancel (
        /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
        /* in  */ const char        *cc,    // Company Code
        /* in  */ const char        *frn,
        /* in  */ const xstaLoop::Address &address,
        /* out */ xstaCommon::MsgHeader &messageHeader
        ) throw (
        xstaCommon::SecurityException,
                xstaCommon::BackendResourceLimitation,
                xstaCommon::InvalidData,
                xstaCommon::GatewayTimeout
        );
```

#### Data Structures

(For typedefs and minor structures see "Appendix A –C++ Header Files" on page 97).

address:

See description in xstaLoop::makeupQueryWorkingLoops.

messageHeader:

See description in xstaLoop::makeupQueryWorkingLoops.

loopList:

See description in xstaLoop::makeupQueryWorkingLoops.

#### Return Values

```
struct Status {
  xstaString msgId;
  xstaString msgTxt;
};
```

#### Remarks

- Success or failure of the reservation cancel is returned in the message id and message text field of the Status.

#### Example

See "Appendix B – Sample Code" on page 151.

### 3.5.2.8 xstaDueDate

#### Inheritance

> xstaPreOrder

> xstaDueDate

#### Description

The xstaDueDate class is used to perform Pre Order Due Date Calculations with a partial LSR.

#### Include Files

```
#include "xstaDueDate.h"
```

#### Methods

xstaDueDate::xstaDueDate

```
xstaDueDate (
        const char *applid,
        const char *applpass,
```

```
                const char *userid
                )
        throw ();
```

Remarks

- Creates an instance of xstaDueDate

xstaDueDate::~xstaDueDate

```
~xstaDueDate ()
        throw (
                xstaCommon::SecurityException,
                xstaCommon::BackendResourceLimitation,
                xstaCommon::GatewayTimeout
        );
```

Remarks

- Destroyes an instance of xstaDueDate

xstaDueDate::estimatedServiceDate

```
xstaCommon::Status estimatedServiceDate (
                /* in */ const xstaDueDate::TypeRecord &typeRecord,
                /* in */ const xstaDueDate::OrderInfo &orderInfo,
                /* in */ const xstaDueDate::ServiceAddressList
&serviceAddressList,
                /* out */ xstaCommon::MsgHeader &messageHeader,
                /* out */ xstaDueDate::DueDateResponse &dueDate
                ) throw (
                        xstaCommon::SecurityException,
                        xstaCommon::BackendResourceLimitation,
                        xstaCommon::InvalidData,
                        xstaCommon::GatewayTimeout
                );
```

Remarks

- Performs a Pre Order calculated Due Date.

### 3.5.3 Firm Order Functionality Classes

This section describes the two classes that make up the Firm Order business rules; ***xstaOrder***, and ***xstaOrderNotification***. Both of these classes directly inherit from the ***xstaServerConnection*** class.

#### 3.5.3.1 xstaOrder

Inheritance

> xstaServerConnection

> xstaOrder

Description

The xstaOrder class is for submitting the Ordering operations:

- LOOP Services Order Request
- LOOP Services with Number Portability (NP) Order Request
- Number Portability (NP) Order Request
- Resale Order Request
- PORT Services Order Request
- Directory Listing and Assistance Order Request
- LOOP/PORT Combination Services Order Request
- Pre-Order Due Date Calculation
- Purchase Order Number (PON) List Query
- Service Order Status Query
- View LSR

### Include Files

```
#include "xstaOrder.h"
```

### See Also

xstaOrderNotification

### Methods

#### xstaOrder::xstaOrder

```
xstaOrder (
/* in */ const char *applid,
/* in */ const char *applpass,
/* in */ const char *userid
) throw ( );
```

### Remarks

- Creates an instance of xstaOrder.
- Sets initial values from provided strings.
- If any string is empty or zero, the corresponding set method must be called before issuing the first query. For example: xstaServerConnection::userid.
- *applid* and *applpass* must match values stored in the BellSouth security system.
- *userid* is an arbitrary value logged by the TAG Gateway.

#### xstaOrder::~xstaOrder

```
~xstaOrder( )
throw (
        xstaCommon::SecurityException,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
```

```
);
```

Remarks

- Destroys an instance of xstaOrder.
- Internally calls xstaServerConnection::surrenderCredentials.

## xstaOrder::orderLoop

```
xstaCommon::Status orderLoop (
/* in */ const xstaCommon::TestProdIndicator testProdIndicator,
/* in */ const xstaOrder::LSR &lsr,
/* in */ const xstaOrder::EU &eu,

/* in */const xstaOrder::DSDL &dsdl,
/* in */const xstaOrder::LOOP &LOOP,
/* out */ xstaCommon::MsgHeader &messageHeader,
/* out */ xstaOrder::DueDateResponse &dueDate
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*testProdIndicator*

```
enum TestProdIndicator {
Test, Production
};
```

*lsr:*

```
struct LSR {
    xstaString          ccna; // Customer Carrier Name
    Abbreviation
    xstaString          pon; // Purchase Order Number
    Version             version;     // Version Number
    xstaString          lsrNumber;   // Local Service
    Request Number
    xstaString          locqty;      // Location Quantity
    xstaString          htqty;       // Hunting Group
    Quantity
    xstaString          an;  // Account Number
    TelephoneNumber     atn; // Account Telephone Number
    xstaString          serviceCenter;      // Service
    Center
    Date                dateSent;    // Date/Time Sent
    Date                ddd; // Desired Due Date
    Date                apptimeDdd;  // Appointment Time
    Date                dddo;// Desired Due Date Out
    Time                dfdt;// Desired Frame Due Time
    xstaString          project;     // Project
    Identification
    xstaString          chc; // Coordinated Hot Cut
    xstaString          reqtyp;      // Request Type
```

```
xstaString          act; // Activity Type
xstaString          sup; // Supplement Type
xstaString          exp; // Expedite
xstaString          cc;  // Company Code
xstaString          albr;// Additional Labor
xstaString          sca; // Special Construction
Authorization
xstaString          agauth;      // Agency
Authorization Status
Date                dated;       // Date of Agency
Authorization
xstaString          authName;    // Authorization Name
xstaString          porttyp;     // Port Type
xstaString          actl;// Access Customer Terminal
Location
xstaString          ai;  // Additional Point of
Termination Indicator
xstaString          apot;// Additional Point of
Termination
xstaString          lst; // Local Service Termination
NpaNxx              lso; // Local Service Office
xstaString          tos; // Type of Service
xstaString          spec;// Service and Product
Enhancement Code
xstaString          nc;  // Network Channel Code
xstaString          nci; // Network Channel Interface
Code
xstaString          secnci;      // Secondary Network
Channel Interface Code
xstaString          rpon;// Related Purchase Order
Number
xstaString          rord;// Related Order Number
xstaString          lspAuth;     // Local Service
Provider Authorization
Date                lspAuthDate; // Local Service
Provider Authorization Date
xstaString          lspAuthName; // Local Service
Provider Authorization Name
xstaString          cic; // Carrier Identification Code
xstaString          cust;// Customer Name
xstaString          bi1; // Billing Account Number
Identifier 1
xstaString          ban1;// Billing Account Number 1
xstaString          bi2; // Billing Account Number
Identifier 2
xstaString          ban2;// Billing Account Number 2
xstaString          acna;// Access Customer Name
Abbreviation
Date                ebd; // Effective Bill Date
xstaString          billName;    // Billing Name
xstaString          sbillName;   // Secondary Bill Name
xstaString          billNameStreet;      // Billing Name
Street Address
```

```
xstaString          billNameFloor;      // Billing Name
Floor
xstaString          billNameRoom;       // Billing Name
Room
xstaString          billNameCity;       // Billing Name
City
xstaString          billNameState;      // Billing Name
State/Province
xstaString          billNameZipCode;    // Billing Name
Zip Code
xstaString          billCon;     // Billing Contact
xstaString          billConTelNo;       // Billing
Contact Telephone Number
xstaString          vta; // Variable Term Agreement
xstaString          init;// Initiator Identification
xstaString          initTelNo;   // Initiator Telephone
Number
xstaString          initFaxNo;   // Initiator Facsimile
Number
xstaString          initStreet;  // Initiator Street
Address
xstaString          initFloor;   // Initiator Floor
xstaString          initRoomMailStop;    // Initiator
Room/Mail Stop
xstaString          initCity;    // Initiator City
xstaString          initState;   // Initiator
State/Province
xstaString          initZipCode; // Initiator Zip Code
xstaString          impCon;      // Implementation
Contact
xstaString          impConTelNo; // Implementation
Contact Telephone Number
xstaString          impConPager; // Implementation
Contact Pager Number
xstaString          altImpCon;   // Alternate
Implementation Contact
xstaString          altImpConTelNo;     // Alternate
Implementation Contact Telephone Number
xstaString          altImpConPager;     // Alternate
Implementation Contact Pager
xstaString          dsgCon;      // Design/Engineering
Contact
xstaString          drc; // Design Routing Code
xstaString          dsgConTelNo; // Design/Engineering
Contact Telephone Number
xstaString          dsgConFaxNo; // Design/Engineering
Contact Facsimile Number
xstaString          dsgConStreet;       //
Design/Engineering Contact Street Address
xstaString          dsgConFloor; // Design/Engineering
Contact Floor
xstaString          dsgConRoomMailStop; //
Design/Engineering Contact Room/Mail Stop
```

```
xstaString          dsgConCity;  // Design/Engineering
Contact City
xstaString          dsgConState; // Design/Engineering
Contact State/Province
xstaString          dsgConZipCode;      //
Design/Engineering Contact Zip Code
xstaString          remarks;     // Remarks
xstaString          pbt; // POT Bay Type
xstaString          bcs; // Basic Class of Service
xstaString          uncommon; // Uncommon Dispatch Flag
xstaString          resid;  // Reservation Identifier
LSRHuntGroupInformationList huntGroupInformationList;
    // Hunt Group Information
LSR ();
};
```

eu:

```
struct EU {
        EUHeader            header;  // Header Section
        EUDetail            detail;  // Detail Section
    };
```

*dsdl:*

```
struct DSDL {
        DSDLListingList      listingList;     // Listing List
        DSDLDeliveryAddress  deliveryAddress; // Delivery
    Address Section
    };
```

*loop:*

```
struct LOOP {
        xstaString           lqty;    // Loop Quantity
        LOOPServiceDetailList serviceDetailList;     //
    Service Detail List
    };
```

*messageHeader:*

```
struct MsgHeader {
xxstaString inquiryNumber;
DateTime  dateSent;
};
```

*dueDate:*

```
struct XST_DECLSPEC DueDateResponse {
  xstaString dueDate;
  xstaString msgId;
  xstaString msgTxt;
};
```

## Return Value

```
struct Status {
xstaString msgId;
xstaString msgTxt;
};
```

## Remarks

- Submits a Loop Order request message.

## Example

See "Appendix B - Sample Code".

## xstaOrder::orderLoopWithNp

```
xstaCommon::Status orderLoopWithnp (
/* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
/* in  */ const xstaOrder::LSR &lsr,
/* in  */ const xstaOrder::EU  &eu,
/* in  */ const xstaOrder::DSDL &dsdl,
/* in  */ const xstaOrder::LSNP &lsnp,
/* out */ xstaCommon::MsgHeader &messageHeader,
/* out */ xstaOrder::DueDateResponse &dueDate
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

## Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*testProdIndicator:*

See description in xstaOrder::orderLoop.

*lsr:*

See description in xstaOrder::orderLoop.

eu:

See description in xstaOrder::orderLoop.

*dsdl:*

See description in xstaOrder::orderLoop.

*lsnp:*

```
struct LSNP {
        xstaString          lqty;    // Loop Quantity
        xstaString          npqty;   // Number Portability
    Quantity
        LSNPServiceDetailList serviceDetailList;     //
    Service Detail List
        };
```

*messageHeader:*

See description in xstaOrder::orderLoop.

*dueDate:*

See description in xstaOrder::orderLoop.

## Return Value

See description in xstaOrder::orderLoop.

## Remarks

- Submits a Loop Order with Number Portability request message.

Example

See "Appendix B - Sample Code".

## xstaOrder::orderNp

```
xstaCommon::Status  ordernp (
/* in */ const xstaCommon::TestProdIndicator testProdIndicator,
/* in */ const xstaOrder::LSR &lsr,
/* in */ const xstaOrder::EU  &eu,
/* in */ const xstaOrder::DSDL &dsdl,
/* in */ const xstaOrder::NP  &np,
/* out */ xstaCommon::MsgHeader &messageHeader,
/* out */ xstaOrder::DueDateResponse &dueDate
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*testProdIndicator:*

See description in xstaOrder::orderLoop.

*lsr:*

See description in xstaOrder::orderLoop.

eu:

See description in xstaOrder::orderLoop.

*dsdl:*

See description in xstaOrder::orderLoop.

*Np:*

```
struct NP {
        xstaString          npqty;   // Number Portability
    Quantity
        NPServiceDetailList  serviceDetailList;      //
    Service Detail List
        };
```

*messageHeader:*

See description in xstaOrder::orderLoop.

*dueDate:*

See description in xstaOrder::orderLoop.

Return Value

See description in xstaOrder::orderLoop.

Remarks

- Submits a Number Portability (NP) Order Request message.

Example

See "Appendix B - Sample Code".

## xstaOrder::orderResale

```
xstaCommon::Status orderResale (
/* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
/* in  */ const xstaOrder::LSR &lsr,
/* in  */ const xstaOrder::EU  &eu,
/* in  */ const xstaOrder::DSDL &dsdl,
/* in  */ const xstaOrder::Resale &resale,
/* inout */ xstaOrder::MsgHeader &messageHeader
/* out */ xstaOrder::DueDateResponse &dueDate
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*testProdIndicator:*

See description in xstaOrder::orderLoop.

*lsr:*

See description in xstaOrder::orderLoop.

eu:

See description in xstaOrder::orderLoop.

*dsdl:*

See description in xstaOrder::orderLoopWithnp.

*resale:*

```
struct Resale {
        xstaString           ord;      // Order Number
        xstaString           rsqty;    // Resale Quantity
        RSServiceDetailList  serviceDetailList;      //
    Service Detail List
        };:
```

See description in xstaOrder::orderRetailBundled.

*messageHeader:*

See description in xstaOrder::orderLoop.

*dueDate:*

See description in xstaOrder::orderLoop.

Return Value

See description in xstaOrder::orderLoop.

Remarks

- Submits an Resale Order Request message.

Example

See "Appendix B - Sample Code".

## xstaOrder::orderPort

```
xstaCommon::Status orderPort (
/* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
/* in  */ const xstaOrder::LSR &lsr,
/* in  */ const xstaOrder::EU  &eu,
/* in  */ const xstaOrder::DSDL &dsdl,
/* in  */ const xstaOrder::PORT &port,
/* out */ xstaCommon::MsgHeader &messageHeader,
/* out */ xstaOrder::DueDateResponse &dueDate
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*testProdIndicator:*

See description in xstaOrder::orderLoop.

*lsr:*

See description in xstaOrder::orderLoop.

eu:

See description in xstaOrder::orderLoop.

*dsdl:*

See description in xstaOrder::orderLoop.

*port:*

```
struct PORT {
        xstaString          ord;     // Order Number
        xstaString          pqty;    // Port Quantity
        PORTServiceDetailList serviceDetailList;     //
    Service Detail List
        };
```

*messageHeader:*

See description in xstaOrder::orderLoop.

*dueDate:*

See description in xstaOrder::orderLoop.

Return Value

See description in xstaOrder::orderLoop.

Remarks

- Submits an Port Order request message.

Example

See "Appendix B - Sample Code".

xstaOrder::orderDirectoryListingAndAssistance

```
xstaCommon::Status orderDirectoryListingAndAssistance (
/* in */ const xstaCommon::TestProdIndicator testProdIndicator,
/* in */ const xstaOrder::LSR &lsr,
/* in */ const xstaOrder::EU &eu,
/* in */ const xstaOrder::DSDL &dsdl,
/* out */ xstaCommon::MsgHeader &messageHeader,
/* out */ xstaOrder::DueDateResponse &dueDate
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

Data Structures

(For typedefs and minor structures see "Appendix A- C++ Header Files" on page 97.)

*testProdIndicator:*

See description in xstaOrder::orderLoop.

*lsr:*

See description in xstaOrder::orderLoop.

eu:

See description in xstaOrder::orderLoop.

*dsdl:*

See description in xstaOrder::orderLoop

*messageHeader:*

See description in xstaOrder::orderLoop.

*dueDate:*

See description in xstaOrder::orderLoop.

Return Value

See description in xstaOrder::orderLoop.

Remarks

- Submits Directory Listing and Assistance Order request message.

Example

See "Appendix B - Sample Code".

## xstaOrder::orderLoopPort

```
xstaCommon::Status orderLoopPort (
/* in */ const xstaCommon::TestProdIndicator testProdIndicator,
/* in */ const xstaOrder::LSR &lsr,
/* in */ const xstaOrder::EU &eu,
/* in */ const xstaOrder::DSDL &dsdl,
/* in */ const xstaOrder::PORT &port,
/* in */ const xstaOrder::Resale &resale,
/* inout */ xstaOrder::MsgHeader &messageHeader
/* out */ xstaOrder::DueDateResponse &dueDate
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

### Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*testProdIndicator:*

See description in xstaOrder::orderLoop.

*lsr:*

See description in xstaOrder::orderLoop.

eu:

See description in xstaOrder::orderLoop.

*port:*

See description in xstaOrder::orderPort.

*dsdl:*

See description in xstaOrder::orderPort.

*resale:*

See description in xstaOrder::orderResale.

*messageHeader:*

See description in xstaOrder::orderLoop.

*dueDate:*

See description in xstaOrder::orderLoop.

### Return Value

See description in xstaOrder::orderLoop.

### Remarks

- Submits a LOOP/PORT Combination Services Order Request message.

### Example

See "Appendix B - Sample Code".

## xstaOrder::getPurchaseOrderList

```
xstaCommon::Status getPurchaseOrderList (
/* in */ const xstaCommon::TestProdIndicator testProdIndicator,
/* in */ const xstaOrder::PONSpec &ponSpec,
/* out */ xstaOrder::MsgHeader &messageHeader,
/* out */ xstaOrder::PONResponseList &ponResponseList
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

### Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

*testProdIndicator:*

See description in xstaOrder::orderLoop.

*ponSpec:*

```
struct PONSpec {
xstaString cc;      // Company Code
ProcessingStatus processingStatus;
        // LEO Status Code
Date fromDate;      // From Date
Date toDate;
        // To Date must not be more than 7 calendar// days from fromDate

PONSpec ();
};
```

*messageHeader:*

See description in xstaOrder::orderLoop.

*ponResponseList:*

PONResponseList is a list of PONResponse:

```
struct PONResponse {
xstaString pon; // CLEC's Purchase Order Number
Version ver;   // LSR Version Number
Date  isaDate; // Date CLEC sent LSR to LEO
Time  isaTime; // Time CLEC sent LSR to LEO
};
```

### Return Value

See description in xstaOrder::orderLoop.

### Remarks

- Submits a PON List Query message.

### Example

See "Appendix B - Sample Code".

## xstaOrder::getServiceOrderStatus

```
xstaCommon::Status getServiceOrderStatus (
```

```
/* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
/* in  */ const char *cc,
/* in  */ const char *pon,
/* out */ xstaCommon::MsgHeader &messageHeader,
/* out */ xstaOrder::ServiceOrderResponse &serviceOrderResponse
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

Data Structures

(For typedefs and minor structures see "Appendix A- C++ Header Files" on page 97.)

*testProdIndicator:*

See description in xstaOrder::orderLoop.

*messageHeader:*

See description in xstaOrder::orderLoop.

*serviceOrderResponse:*

```
struct ServiceOrderResponse {
xstaString cc;  // Company Code
xstaString pon; // CLEC's Purchase Order Number
Version ver;    // LSR Version Number
ServiceOrderStatus status;      // SOCS order status.
};
```

Return Value

See description in xstaOrder::orderLoop.

Remarks

- Submits a Service Order Status Query message.

Example

See "Appendix B - Sample Code".

## xstaOrder::order

```
        xstaCommon::Status order (
/* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
/* in  */ const xstaOrder::LSR &lsr,
/* in  */ const xstaOrder::EU  &eu,
/* in  */ const xstaOrder::DSDL &dsdl,
/* in  */ const xstaOrder::LOOP &loop,
/* in  */ const xstaOrder::LSNP &lsnp,
/* in  */ const xstaOrder::NP  &np,
/* in  */ const xstaOrder::PORT &port,
/* in  */ const xstaOrder::Resale &resale,
/* in  */ const xstaOrder::DID &did,
/* out */ xstaCommon::MsgHeader &messageHeader
/* out */ xstaOrder::DueDateResponse &dueDate
) throw (
   xstaCommon::SecurityException,
   xstaCommon::BackendResourceLimitation,
   xstaCommon::InvalidData,
   xstaCommon::GatewayTimeout
        );
```

Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

Return Value

```
struct Status {
xstaString msgId;
xstaString msgTxt;
};
```

Remarks

- Submits any one of the seven Order types and includes the necessary code for calculating due date.

See Also

getDueDate

Example

See "Appendix B – Sample Code"

## xstaOrder::getDueDate

```
xstaCommon::Status getDueDate (
/* in */ const xstaCommon::TestProdIndicator testProdIndicator,
/* in */ const xstaOrder::LSR &lsr,
/* in */ const xstaOrder::EU &eu,
/* in */ const xstaOrder::DSDL &dsdl,
/* in */ const xstaOrder::LOOP &loop,
/* in */ const xstaOrder::LSNP &lsnp,
/* in */ const xstaOrder::NP &np,
/* in */ const xstaOrder::PORT &port,
/* in */ const xstaOrder::Resale &resale,
/* in */ const xstaOrder::DID &did,
/* out */ xstaCommon::MsgHeader &messageHeader,
/* out */ xstaOrder::DueDateResponse &dueDate
) throw (
        xstaCommon::SecurityException,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::InvalidData,
        xstaCommon::GatewayTimeout
);
```

Data Structures

(For typedefs and minor structures see "Appendix A - C++ Header Files" on page 97.)

```
struct DueDateResponse {
        xstaString dueDate; // Due Date
        xstaString msgId; // Due Date Message ID
        xstaString msgTxt; // Due Date Message Text
};
```

Return Value

```
struct Status {
xstaString msgId;
xstaString msgTxt;
};
```

Remarks

- Use this method to calculate Due Date for a PreOrder submission.

See Also

order

Example

See "Appendix B – Sample Code"

xstaOrder::getFormattedLsr

```
xstaCommon::Status getFormattedLsr (
/* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
/* in  */ const char *      *cc,     // Company Code
/* in  */ const char        *pon,
/* out */ xstaCommon::MsgHeader &messageHeader,
/* out */ xstaString        &flatLsr
            ) throw (
xstaCommon::SecurityException,
xstaCommon::BackendResourceLimitation,
xstaCommon::InvalidData,
xstaCommon::GatewayTimeout
            );
```

Data Structures

(For typedefs and minor structures see ``Appendix A - C++ Header Files" on page 6-906-84.)

Return Value

```
struct Status {
xstaString msgId;
xstaString msgTxt;
            };
```

Remarks

- Used to submit retrieve one formatted LSR.

See Also

getServiceOrderStatus

Example

See ``Appendix B - Sample Code"

### 3.5.3.2 xstaOrderNotification

Inheritance

> xstaServerConnection

> xstaOrderNotification

Description

The xstaOrderNotification class provides a way to receive notifications from the TAG Gateway.

## Include Files

```
#include "xstaOrderNotification.h"
```

## See Also

xstaOrder

## Methods

### xstaOrderNotification::xstaOrderNotification

```
xstaOrderNotification (
/* in */ const char *applid,
/* in */ const char *applpass,
/* in */ const char *userid
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

### Remarks

- Creates an instance of xstaOrderNotification.
- Sets initial values from provided strings.
- *applid* and *applpass* must match values stored in the BellSouth security system.
- *userid* is an arbitrary value logged by the TAG Gateway.
- An exception thrown from the constructor means calls to readNotification will fail.

### xstaOrderNotification::~xstaOrderNotification

```
~xstaOrderNotification ( )
throw (
        xstaCommon::SecurityException,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

### Remarks

- Destroys an instance of xstaOrderNotification.
- Internally calls xstaServerConnection::surrenderCredentials.

### xstaOrderNotification::poll_interval

```
void poll_interval( unsigned long seconds );
```

### Remarks

- When *blocking* is used by readNotification, the TAG Client API polls internally for incoming notifications. Use poll_interval to specify the polling retry interval in seconds.
- The default polling interval is 60 seconds.

## xstaOrderNotification::readNotification

```
unsigned char readNotification(
/* in */ int            blocking,
        /* out */ xstaOrderNotification::Notification &notification
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout);
```

### Data Structures

NotificationList is a sequence of Notification as above.

### Return Value

Returns zero if no notification is returned, non-zero if a notification is present.

### Remarks

- Retrieves the next available notification from the TAG Gateway.
- If blocking is nonzero, readNotification will not return to the application until a notification has been received.
- If blocking is zero, readNotification will return immediately to the application even if a notification is not available.
- readNotification sends an automatic acknowledgement back to the TAG Server upon successful receipt of a notification.This can result in lost notifications. Application programmers should instead use readNotificationNoAck and acknowledgeNotification, or, better, readNotificationList and acknowledgeNotificationList.
- readNotification may receive notifications prior to the application submitting the initial order request as the result of previous requests from the same Application ID
- If applId() has been called with a Group ID, readNotificationList() will return available notifications for all the Application Ids in that Group.

## xstaOrderNotification::readNotificationNoAck

```
int readNotificationNoAck (
/* out */ xstaOrderNotification::Notification &notification
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
```

```
);
```

Data Structures

Notification is as above.

Return Values

Returns zero if no notification is returned, non-zero if a notification is present.

Remarks

- Retrieves the next available notification from the TAG Gateway.
- readNotificationNoAck will return immediately even if a notification is not available. Check the return code to see if one was returned.
- readNotificationNoAck may receive notifications prior to the application submitting the initial order request as the result of previous requests from the same Application ID.
- readNotificationNoAck does not send an automatic acknowledgement back to the TAG Server. As soon as possible after the application has successfully received the notification, it must call acknowledgeNotification. Failure to do so will result in duplicate notifications being received by the application.

xstaOrderNotification::acknowledgeNotification

```
void acknowledgeNotification (
/* in */ const xstaOrderNotification::Notification & notification
) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData
                xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
);
```

Data Structures

Notification as above

Return Value

None

Remarks

- acknowledgeNotification must be called with each notification received from readNotificationNoAck.

xstaOrderNotification::readNotificationList

```
unsigned char readNotificationList (
        /* out */ xstaOrderNotification::NotificationList &notificationList
        ) throw (
        xstaCommon::SecurityException,
        xstaCommon::InvalidData,
        xstaCommon::BackendResourceLimitation,
        xstaCommon::GatewayTimeout
```

```
);
```

Data Structures

NotificationList is a sequence of Notification as above.

Return Values

Returns zero if no notification is returned, non-zero if one or more notification is present.

Remarks

- Retrieves the next available notifications from the TAG Gateway.
- readNotificationList will return immediately even if a notification is not available. Check the return code to see if any were returned. readNotificationList may receive notifications prior to the application submitting the initial order request as the result of previous requests from the same Application ID. readNotificationList does not send an automatic acknowledgement back to the TAG Server. As soon as possible after the application has successfully received the notification, it must call acknowledgeNotificationList. Failure to do so will result in duplicate notifications being received by the application.

## xstaOrderNotification::acknowledgeNotificationList

```
void acknowledgeNotificationList (
/* in */ const xstaOrderNotification::NotificationList &notificationList
        ) throw (
                xstaCommon::SecurityException,
                xstaCommon::InvalidData,
                xstaCommon::BackendResourceLimitation,
                xstaCommon::GatewayTimeout
        );
```

Data Structures

NotificationList is a sequence of Notification as above.

Return Values

None

Remarks

- acknowledgeNotificationList must be called with the notification list received from readNotificationList. The API will send an acknowledgement for each of the notifications in the list. Alternatively, the application may call acknowledgeNotification individually for each notification in the list.

## 3.6 Exceptions

This section describes exceptions raised by the TAG Client API. The ***xstaCommon*** class contains 4 exceptions described in this section.

### 3.6.1 xstaCommon::BackendResourceLimitation

#### Description

BackendResourceLimitation exception is thrown when invoked operation could not be forwarded to the BellSouth OSS.

#### Data Structures

```
struct BackendResourceLimitation {
xstaString code;
xstaString description;
xstaString errTrackNum;
};
```

#### Handling

```
catch (xstaCommon::BackendResourceLimitation &exc) {
cerr << "Resource limitation "
<< exc.code
<< ", "
<< exc.description
<< ", "
<< exc.errTrackNum
<< endl;
};
```

### 3.6.2 xstaCommon::GatewayTimeout

#### Description

GatewayTimeout exception is thrown when no response is received after a query is sent to the TAG Gateway.

#### Data Structures

```
struct GatewayTimeout {
xstaString code;
xstaString description;
xstaString errTrackNum;
};
```

#### Handling

```
catch (xstaCommon::GatewayTimeout &exc) {
cerr << "Gateway timeout "
<< exc.code
<< ", "
<< exc.description
<< ", "
<< exc.errTrackNum
<< endl;
}
```

### 3.6.3 xstaCommon::InvalidData

#### Description

InvalidData exception is thrown when query data is invalid, or when TAG cannot map a query or response.

### Data Structures

```
struct InvalidData {
InvalidDataElementList invalidDataElementList ;
xstaString inquiryNumber;        // If available.
};
```

### Handling

```
catch (xstaCommon::InvalidData &exc) {
        // Some combination of data was invalid or missing
        unsigned long i;
        for (i=0; i < exc.invalidDataElementList.length(); i++) {
                cerr << "Invalid data element "
                << exc.invalidDataElementList[i].dataElementName
                << ", error code "
                << exc.invalidDataElementList[i].status.msgId
                << ", "
                << exc.invalidDataElementList[i].status.msgTxt
                << endl;
                }
        }
```

## 3.6.4 xstaCommon::SecurityException

### Description

SecurityException occurs when the TAG Gateway rejects a request due to a failure in authentication, authorization, throttle control, or other security related criteria.

### Data Structures

```
enum SecurityExceptionType {
AuthenticationFailed,
AuthorizationFailed,
AccountDisabled,
ThrottleControlFailed,
NoPriorAuthentication,
CookieExpired,
SecurityViolationDetected,
CookieInvalid,
InactivityDetected,
ResourceLimitation
};
struct SecurityException {
SecurityExceptionType exc;
xstaString message;
xstaString inquiryNumber;      // If available.

};
```

### Handling

```
catch (xstaCommon::SecurityException &exc) {
cerr << "Security exception: "
<< exc.message << endl;
switch (exc.exc) {
        case xstaCommon::AuthenticationFailed:
                break;
        case xstaCommon::AuthorizationFailed:
                break;
        case xstaCommon::AccountDisabled:
                break;
        case xstaCommon::ThrottleControlFailed:
```

```
            break;
case xstaCommon::NoPriorAuthentication:
            break;
case xstaCommon::CookieExpired:
            break;
case xstaCommon::SecurityViolationDetected:
            break;
case xstaCommon::CookieInvalid:
            break;
case xstaCommon::InactivityDetected:
            break;
case xstaCommon::ResourceLimitation:
            break;
            }
}
```

# 4. Configuration

This section describes the two main configuration files for the TAG Client API software; ***xst_Client***, and ***xst_Key***. It also describes how to turn on and off program execution tracing and how to load the Program Validation (PV) data. For information on configuring the CLEC Notification Server see Section 5.

## 4.1 Configuration File

This section describes the file that contains the configuration data for the TAG Client API. This file must exist and contain the appropriate values before running the client application with the TAG Client API.

This file has the following formatting: one record per line, lines beginning with hash mark (#) are ignored, and case is important. The format for each record is:

*Field=value;*

White space around any of the elements is ignored. If a value needs to include white space or punctuation, enclose value in double quotes.

The configuration data on UNIX clients is located in the file *SXST_CONFIG/xst_Client*.

To configure the CLEC API, modify the following values.

```
ClecId=LOCALCLECID;
EncryptionFile=xst_Key;        # file name
Security = {
   Server=TAPSECURITYSERVER;
   Host=TAPGATEWAYHOSTNAME;
};

SGGateway = {
   ConnectionMethod = CONNECTMETHOD;
   CBRelayServerName = CBRELAYNAME;
   CBRelayHostName = CBRELAYHOST;

   LdapIf = {
      CBIORFile=TAPCBIORFILE;
      LdapHost=TAPLDAPHOST;
      LdapPort=TAPLDAPPORT;
      CBLdapFilterPart=TAPLDAPFILTER;
      CBLdapSearchBasePart=TAPLDAPBASE;
      CBLdapRootDN=TAPLDAPROOTDN;
        };
                    };
```

*security_server* is the name of the TAG Security Server on the TAG Gateway.

*security_server_host* is the name of the TAG Gateway host running the TAG Security Server.

```
EncryptionFile = xst_Key;
```

*xst_Key* is the name of the file containing the encryption keys to use during Client API communications with TAG servers. This value should not be changed.

*CoGIORFile* is the name of a file in the config directory where CoG IOR information is kept.
*ldap_host* is the hostname of the LDAP server.
*ldap_port* is the port number of the LDAP server.
*LDAP Filter Part* – the name of the Corporate Gateway Bridge Server (see the instructions for installing COG)
*LDAP Base Part* –the name of the LDAP Server
*LDAP Root Domain* – the domain name of your system to be used by LDAP

Example:

```
ClecId=HP_CLIENT;

EncryptionFile=xst_Key;        # file name

Security = {

        Server=TagSecurityServer7107;

        Host=oregon.mk.telcordia.com;

};


SGGateway = {


   ConnectionMethod = BIND;

   CBRelayServerName = CogRelayServer7107;

   CBRelayHostName = oregon.mk.telcordia.com;

        LdapIf = {

                CBIORFile=CoGIORFile;

                LdapHost=cherek.mk.telcordia.com;

                LdapPort=28493;

                CBLdapFilterPart=CogBridgeServer1008;

                CBLdapSearchBasePart=1.0_PT3;

                CBLdapRootDN=telcordia.com;

        };
```

```
};

ClecId = ''Ma & Pa Phone'';
Security = {
       Server = TagFactoryServer;
       ServerHost = taggateway.bellsouth.com;
};
EncryptionFile = xst_Key;
```

The key file is located in $XST_CONFIG/xst_Key and is used by both the client API and the CLEC Notification Server.

To configure the key file, modify it with the fields and values provided by BellSouth.

```
hostname = key;
```

hostname is the name of the host running the TAG Gateway. Key is the encryption key used for all TAG queries from the client machine to the TAG Gateway host. If there are multiple hosts running parts of the TAG Gateway, you will need a line in the config file for each.

Example:

Taggateway.bellsouth.com = secret;

Example

```
SGGateway = {
LdapIf = {
CBIORFile=CoGIORFile;
LdapHost= cogldap.bellsouth.com;
LdapPort= 5975;
CBLdapFilterPart= CogBridgeServer10;
CBLdapSearchBasePart= 1.0_PT3;
CBLdapRootDN= bellsouth.com;
        .};

};
```

## 4.2 Client API Trace Configuration

The TAG Client API has the ability to produce program execution trace information. (For information on tracing in the CLEC Notification Server see "CLEC Notification Server" on page 88.)

The TAG Client API trace facility is controlled by the environment variable `XSTA_TRACE. On UNIX platforms, XSTA_TRACE` can be set in the user .profile or can be exported from the command line. The client application must be restarted in order for changes to take effect.

To turn TAG Client API tracing on:

```
export XSTA_TRACE=d:f:o,filename
```

"d:" turns on data tracing

"f:" turns on function call tracing

"o,*filename*" sets the output filename. If *filename* is omitted, output goes to stderr.

To turn TAG Client API tracing off:

```
unset XSTA_TRACE
```

(On Windows platforms, use **set** rather than **export**, and **set XSTA_TRACE=** rather than **unset**.)

## 4.3 PV Configuration

You will only need to load the Programmable Validation (PV) file once per distribution of new PV Tables by BellSouth. To load the PV files, execute the following command.

```
xswPvLoad xstPvData
```

xstPvData is the file containing the Programmable Validation data. This command installs the PV's under the *$XST_CONFIG/pv* directory.

# 5. CLEC Notification Server

The CLEC Notification Server collects unsolicited notifications from the TAG Gateway Notification Server and provides them to the CLEC application through the TAG Client API.

## 5.1 Configuration

To configure the CLEC Notification Server, modify the following variables in the *SXST_CONFIG/**xst_ClecNotification*** file.

Syntax:

```
ServerName = TAPCLECNOTIFYSERVER;

Security = {
   Server = TAPSECURITYSERVER;
  Host = TAPGATEWAYHOSTNAME;
   Parameters={
      # Since the security server treats this like just
      # another login attempt, we need to reflect this
      # in the password file too...
      ApplId = tapNotifySrv_hp;
      ApplPassword = tapdev;
   };

   RegisterAttributes = {
      ApplIdList = ( tapdev_hp1,
tapdev_hp2,tapdev_hp3,tapdev_hp4,tapdev_hp6 );
      GroupIdList = {
         Chandra = (tapdev_hp1,tapdev_hp2,tapdev_hp3);
         tapdev_hp  =
(tapdev_hp1,tapdev_hp2,tapdev_hp3,tapdev_hp4);
      };
   };
};

SGGateway = {

   ConnectionMethod = CONNECTMETHOD;

   CBRelayServerName = CBRELAYNAME;

   CBRelayHostName = CBRELAYHOST;

   LdapIf = {
      CBIORFile=TAPCBIORFILE;
      LdapHost=TAPLDAPHOST;
      LdapPort=TAPLDAPPORT;
      CBLdapFilterPart=TAPLDAPFILTER;
      CBLdapSearchBasePart=TAPLDAPBASE;
      CBLdapRootDN=TAPLDAPROOTDN;
   };
```

```
};

EncryptionFile=xst_Key;        # file name
TransactionLog = { SizeLimit = 2000000; };

ThreadPoolSize = 10;

# time in minutes, 5 minutes minimum
CogPollInterval = 10;
```

#TransactionLog = {SizeLimit = *logSize*;};

*LogSize* is the size limit in bytes of the CLEC Notification Server's transaction log file. When the size limit is exceeding, the CLEC Notification Server will start a new transaction log file. See Section 5.6.2.

*GroupIdList* specifies Group names and the Application IDs which are in each group. A particular Application ID may be appear within multiple Groups. Calling xstaOrderNotification::readNotificationList() with Application ID set to a Group ID will return a list containing all available notifications for all Application Ids in that Group. Thus, if a notification is available for Application_id_1, it will be returned to the next client query using an Application ID of Application_id_1, Group1, or Group2.

ThreadPoolSize – specifies the maximum number of threads to use to service Notifications. The default is 10 and any number less than 10 will be ignored.

*CoGIORFile* is the name of a file in the config directory where CoG IOR information is kept.
*LDAP Host Name* – the name of the host where the LDAP Server will reside
*LDAP Port Number* – the port number of the LDAP Server
*LDAP Filter Part* – the name of the SG Gateway Bridge Server (see the instructions for installing COG)
*LDAP Base Part* –the name of the LDAP Server
*LDAP Root Domain* – the domain name of your system to be used by LDAP

*Encryption_Configuration_file* - is the name of the file containing the encryptionkeys to use during Client API communications with TAGservers. This value should not be changed.

Example:

```
ServerName = xst_ClecNotificationServer_Hp;

Security = {

   Server = xstSecurityServer_2_1;
   Host = host.domain;
   Parameters={
```

```
        #
        # Since the security server treats this like just
        # another login attempt, we need to reflect this
        # in the password file too...
        #
        ApplId = xst_ClecNotificationServer_Hp_login;
};
        ApplPassword = KillroyWasHere;
    };

    RegisterAttributes = {
        #
        # This ClecNotification server attempts to
        # service the following application ids.
        #
        ApplIdList = ( Chandra, tapdev_hp );
        GroupIdList={
            Regular=(Chandra);
             tapdev= (tapdev_hp, tapdev_hp1, tapdev_hp2);

    };
};

EncryptionFile=xst_Key;        # file name

TransactionLog = { SizeLimit = 1000000; };
ThreadPoolSize = 10;

CorporateGateway = {
   Server = cog_server_10;
   Host = domain.name;
};
SGGateway = {
LdapIf = {
CBIORFile=CoGIORFile;
LdapHost= cogldap.bellsouth.com;
LdapPort= 5975;
CBLdapFilterPart= CogBridgeServer10;
CBLdapSearchBasePart= 1.0_PT3;
CBLdapRootDN= bellsouth.com;
        };
];
EncryptionFile=xst_Key;        # file name
TransactionLog = { SizeLimit = 2000000; };
```

## 5.2 Startup

On the UNIX client platform the command **xsta_start** is used to start the CLEC Notification server.

```
xsta_start
```

On the MS-Windows client platform start the CLEC Notification Server from a DOS window by executing the following command.

***xstNotificationSrv***

**Note:** The ORBIX Daemon must be running before starting the CLEC Notification Server. The IT_DAEMON_PORT, IT_DAEMON_SERVER_BASE AND IT_DAEMON_SERVER_RANGE should match the BellSouth configuration.

**Note:** The **xsta_start** command do not work on the Windows platform.

## 5.3 Shutdown

On the UNIX client platform use the **xsta_stop** command to stop the CLEC Notification Server.

```
xsta_stop
```

On the MS-Windows client platform stop the CLEC Notification Server from a DOS windows prompt by executing the following command.

```
killit server_name
```

where *server_name* is the name of the CLEC Notification Server displayed by the following command:

```
psit
```

(TagCLECNotificationServr is the default).

These command do not accept command line arguments.

**Note: xsta_stop** does not work on the Windows platform.

## 5.4 Status

To check for status updates on the Client Notification Server, use the ORBIX ***psit*** command to display currently running servers. The output should show at least one instance of the CLEC Notification Server, for example:

```
psit

Active servers at node clyde.vin.bellsouth.com are :

Name            Marker  Code    Comms   Port    Launch  PerClient
OS-pid
-------------------------------------------------------------------
-----------TagCLECNotificationServer*      xdr     tcp     1612
manual  ---       1085
```

## 5.5 Trace Facility

The CLEC Notification Server has the ability to produce program execution trace information. This allows more debugging statements to be logged for better monitoring.

### 5.5.1 Trace File Naming

Trace files are located in the directory specified by the environment variable $XST_TRACE_DIR (by default, *$XST_HOME/trace*). When tracing is turned on, the CLEC Notification Server creates and writes to a trace file. Trace file names are created using the server name as follows:

```
xst_servername_trace_out
```

For example, if the CLEC Notification Server is named TagCLECNotificationServer, the trace file name is *xst_TagCLECNotificationServer_trace_out.*

When a server restarts, a new instance of the trace file is created. If a trace file already exists, it is saved and renamed by appending the suffix ".bak". For example, an existing is *xst_TagCLECNotificationServer_trace_out* becomes is *xst_TagCLECNotificationServer_trace_out.bak.*

### 5.5.2 Trace Configuration

The trace facility is controlled by the following environment variables. The environment variables can be set in the xst_adm user .profile file or can be exported from the command line.

Servers must be restarted in order for changes to take effect. (On Windows platforms, use **set** rather than **export**, and **set *variable*=** rather than **unset**.)

- To turn tracing on:

  ```
  export XST_TRACE_ON=YES
  ```

- To turn tracing off:

  ```
  unset XST_TRACE_ON
  ```

- To change the directory to which trace files are written, provide the pathname of an existing directory. (The default is *$XST_HOME/trace*.) For example:

  ```
  export TRACE_DIR=$XST_HOME/some_directory
  ```

- To change the type of information produced modify the trace level. For example:

  ```
  export XST_TRACE_LEVEL=10
  ```

  The following table lists the possible trace level values. (Trace levels are hierarchical, setting a given level sets all levels numerically below it.)

**Table 14- Trace Levels**

| Trace Type | Value | Description |
|---|---|---|

| Trace Type | Value | Description |
|---|---|---|
| MAJOR | 10 | Major events and errors. Examples include changes in network connection status, missing or incorrect configuration messages, and process failures. |
| AUDIT | 20 | Events and errors which might be useful in creating an audit trail of who is doing what with a TAG server. Examples include the creation/deletion of TAG objects, security steps. |
| FLOW | 30 | Events at significant points of processing flow. Examples include the receiving of message in the server, invocation of the mapper, sending a message from the client. |
| DATA | 40 | Messages showing the value of key pieces of data. Examples include the return values of functions, filenames, important function parameters, or temporary data. |
| DEBUG_FLOW | 50 | Like the trace type of FLOW but used for a finer grain of detail. |
| DEBUG_DATA | 60 | Like the trace type of DATA but used for a finer grain of detail. |
| FUNCTION | 70 | Function entry/exit messages. |
| LOOPING | 100 | At this level, even when the process is at an "idle" state, the trace file will grow. An example would be to trace a polling loop. |

## 5.6 Transaction Log

The Transaction Log captures administrative information about each message that passes through the CLEC Notification Server.

### 5.6.1 Transaction Log Naming

Transaction Log files are located in the directory specified by the environment variable $XST_LOG_DIR (by default, *$XST_HOME/log*). Transaction Log file names are created using the following format:

```
xstTLOG.servername.n
```

where *servername* is the name of the server writing to the log, *n* is a sequence number starting at 1.

For example, if the CLEC Notification Server is named TagCLECNotificationServer, the initial CLEC Notification Server Transaction Log file name is:

```
xstTLOG.TagCLECNotificationServer.1
```

When a Transaction Log file reaches the size limit specified by *TransactionLogSizeLimit* in the particular server's configuration file, a new log file is created with the *n* value incremented. For example:

```
xstTLOG.TagCLECNotificationServer.2
```

**Note:** Older transaction log files can be deleted at your discretion.

### 5.6.2 Transaction Log Size Limit

The TAG Transaction Log consists of a set of Transaction Log files. To set the maximum size of a Transaction Log file, modify the SizeLimit parameter in the configuration file. See Section 5.1.

Syntax:

```
TransactionLog = {SizeLimit = logSize;}
```

*logSize* is the maximum number of bytes for a Transaction Log file. *logSize* should be greater than 4,000 bytes and less than *ulimit* bytes, (*ulimit* is the UNIX file size limit for the process).

Default:

2,000,000 bytes

Example:

```
TransactionLog = {SizeLimit= 80000;};
```

### 5.6.3 Viewing the Transaction Log

The Transaction Log is comprised of several log files written to separate servers. A utility, *SXST_BIN/xst_tlog,* allows the merging of disparate files into a single view.

To run the **xst_tlog**, use the following command:

**xst_tlog** [*options*]

Options:

```
-r  view the log in reverse order
```

**xst_tlog** writes to Standard Output, this will redirect or pipe output which is ordered by Date-Timestamp, Transaction ID and Log type. This will allow a customized view of the Transaction Log.

**Note: xst_tlog** does not work on the Windows platform, use any text editor to view the transaction logs.

### 5.6.4 Log Record Format

The following is the CLEC Notification Server log record format. The fields in each record are delimited by bars ( | ). Field content is subject to data availability (i.e. The format shown below applies to all the records. However, some field contents might be empty).

```
log type|CLEC id|appl id|user id|transaction id|transaction
name|transaction class|date-time read|API Version|CLEC
inquiry number|CCNA|PON|LSR Version|Notification msgid|
```

where:

- *log type* is a text description of the event being logged.

  T8 - CLEC application (via API) sends an acknowledgement
  T9 - CLEC application picks up notification
  T10 - notification arrives from TAG.

- *CLEC id* is the company ID.
- *app id* is the application ID as configured in the BellSouth security system.
- *user id* is the user ID provided by the CLEC client application.
- *transaction id* is the unique number returned from the OSS in reply to a query.
- *transaction name* is "NOTIFY".
- *transaction class* is "O" .
- *date-time read* is the date and time the record was logged. The Timestamp entry is now logged in hundreds of a second resolution.
- *CLEC inquiry number* is a unique number generated by the CLEC Notification Server for the identified transaction. The *inquiry number* is contained in all messages that make up a given transaction.
- *CLEC API Version*
- *company code* (CC)
- *purchase order number* (PON)
- *LSR version* (VER)
- *Notification msgid* contains the Notification message id.

Example:

```
T8 -
RECEIVE-NOTIFICATION|XYZ|tapdev_sol|krohn|CLEC1-PO-NOTIFY-
000001-00|NOTIFY|O|
19990702-08:45:03|||CLEC1|PO NOTIFY-000001|01|00
T9 -
READ-NOTIFICATION|XYZ|tapdev_sol|krohn|CLEC1-PO-NOTIFY-
000001-00|NOTIFY|O|199
90702-08:45:27|3.1|622fb2f600000001|CLEC1|PO NOTIFY-
000001|01|00
T10 -
SEND-ACKNOWLEDGEMENT|XYZ|tapdev_sol|krohn|CLEC1-PO-NOTIFY-
000001-00|ACKNOTIF|
O|19990702-08:45:27|3.1|622fb2f600000001|CLEC1|||
```

# 6. Appendix A - C++ Header Files

## 6.1 xstaCommon.h

```
/*-------------------------------------------------------------------------
|                                                                         |
|(C) BellSouth Telecommunications, Inc. 1998                              |
|                                                                         |
|                       PRIVATE/PROPRIETARY/LOCK                          |
|          CONTAINS PRIVATE AND/OR PROPRIETARY INFORMATION.               |
|     MAY NOT BE USED OR DISCLOSED OUTSIDE THE BELLSOUTH COMPANIES        |
|               EXCEPT PURSUANT TO A WRITTEN AGREEMENT.                   |
|            MUST BE STORED IN LOCKED FILES WHEN NOT IN USE.              |
|                                                                         |
-------------------------------------------------------------------------*/

/*
 * @(#)xstaCommon.h 71.1
 * created on 00/06/14 at 14:53:56
 */


#ifndef xstaCommon_h
#define xstaCommon_h

#include "xstaString.h"

/*
 * Lots of common data structures used by xstaTAP.h.  These shadow the data
 * structures provided through xstCommon.idl.  Why the shadowing?  Mostly
 * because the IDL exposes String_mgr all over and String_mgr is Orbix
 * specific and String_mgr is aggressive in controlling memory (easy to
 * have it free non-heap memory accidentally).  Instead, use xstaString.
 */

/* This macro declares a list class.  The provided member functions are
 * similar to those provided by the IDL lists.
 * The default constructor builds an empty list (both length and maximum
 * are 0).
 * maximum() returns the current maximum list size.
 * length() returns the current list size.
 * length(len) changes the list size.  If len is greater than the previous
 * length, more memory is allocated and the list elements are copied.  If len
 * is less than the previous length, no memory is deallocated.
 * operator[](i) returns the i-th element of the list.  It is the programmer's
 * responsibility to keep i < length.
 */
#define xstaLIST_DECLARATION(element_type_name,list_type_name) \
      class XST_DECLSPEC list_type_name { \
        private: \
            unsigned long _maximum; \
            unsigned long _length; \
            element_type_name *_buffer; \
        public: \
            list_type_name (); \
            ~list_type_name (); \
            list_type_name (const list_type_name &from); \
            list_type_name &operator= (const list_type_name &from); \
            unsigned long maximum () const { return _maximum; } \
```

```
            unsigned long length () const { return _length; } \
            void length (unsigned long len); \
            element_type_name &operator[] (unsigned long i) { return _buffer[i]; } \
            const element_type_name &operator[] (unsigned long i) const { return
_buffer[i]; } \
      }

struct xstaCommon    /* Name scoping only */
{

      typedef xstaString TransactionType;

      enum SecurityExceptionType {
            AuthenticationFailed,
            AuthorizationFailed,
            AccountDisabled,
            ThrottleControlFailed,
            NoPriorAuthentication,
            CookieExpired,
            SecurityViolationDetected,
            CookieInvalid,
            InactivityDetected,
            ResourceLimitation
      };

      struct SecurityException
      {
            SecurityExceptionType exc;
            xstaString message;
            xstaString inquiryNumber;     // If available.
      };

      struct Status {
            xstaString msgId;
            xstaString msgTxt;
      };

      typedef xstaString DateTime;

      struct MsgHeader {
            xstaString inquiryNumber;
            DateTime   dateSent;
      };

      typedef xstaString DataElementName;

      struct InvalidDataElement {
            DataElementName dataElementName;
            Status          status;
      };

      xstaLIST_DECLARATION (InvalidDataElement, InvalidDataElementList);

      struct InvalidData {
            InvalidDataElementList invalidDataElementList;
            xstaString inquiryNumber;     // If available.
      };

      struct BackendResourceLimitation {
```

```
        xstaString code;
        xstaString description;
        xstaString errTrackNum;
        xstaString inquiryNumber;     // If available.
};

struct GatewayTimeout {
        xstaString code;
        xstaString description;
        xstaString errTrackNum;
        xstaString inquiryNumber;     // If available.
};

enum Directional {
        NoDirection,
        East,
        West,
        North,
        South,
        NorthEast,
        NorthWest,
        SouthWest,
        SouthEast
};

typedef xstaString ReservationNumber;

typedef xstaString LocalServiceTermination;
typedef xstaString Clli;
typedef xstaString NpaNxx;

xstaLIST_DECLARATION (NpaNxx, NpaNxxList);

typedef short InterlataPic;

xstaLIST_DECLARATION (InterlataPic, InterlataPicList);

typedef xstaString TypeOfService;
typedef xstaString TelephoneNumber;
xstaLIST_DECLARATION (TelephoneNumber, TelephoneNumberList);


typedef unsigned short  Quantity;
typedef unsigned short  Version;

enum TestProdIndicator {
        Test, Production
};

enum OrderTransactionType
{
        OrderSubmit = 1,
        OrderPON = 2,
        OrderServiceStatus = 5,
        OrderUnsolicitedResponse = 90,
        OrderUnsolicitedResponseAck = 91,
        PreOrderDueDate = 181
};
```

```
        struct Attribute {
                xstaString   tag;
                xstaString   value;
        };

        xstaLIST_DECLARATION (Attribute, AttributeList);

};

#endif
```

## 6.2 xstaTAP.h

```
/*-------------------------------------------------------------------------
|                                                                         |
|(C) BellSouth Telecommunications, Inc. 1998                              |
|                                                                         |
|                        PRIVATE/PROPRIETARY/LOCK                         |
|          CONTAINS PRIVATE AND/OR PROPRIETARY INFORMATION.               |
|     MAY NOT BE USED OR DISCLOSED OUTSIDE THE BELLSOUTH COMPANIES        |
|               EXCEPT PURSUANT TO A WRITTEN AGREEMENT.                   |
|             MUST BE STORED IN LOCKED FILES WHEN NOT IN USE.             |
|                                                                         |
-------------------------------------------------------------------------*/

/*
 * @(#)xstaTAP.h    36.1
 * created on 00/03/03 at 12:11:49
 */

#ifndef xstaTAP_h
#define xstaTAP_h

#include "xstaString.h"
#include "xstaCommon.h"

#include "xstaServerConnection.h"
#include "xstaPreOrder.h"
#include "xstaAddressValidation.h"
#include "xstaAppointmentScheduling.h"
#include "xstaCustomerRecord.h"
#include "xstaServiceAvailability.h"
#include "xstaTelephoneNumberAssignment.h"
#include "xstaOrder.h"
#include "xstaOrderNotification.h"

#endif
```

## 6.3 xstaServerConnection.h

```
/*-------------------------------------------------------------------------
|                                                                         |
|(C) BellSouth Telecommunications, Inc. 1998                              |
|                                                                         |
|                       PRIVATE/PROPRIETARY/LOCK                          |
|          CONTAINS PRIVATE AND/OR PROPRIETARY INFORMATION.               |
|     MAY NOT BE USED OR DISCLOSED OUTSIDE THE BELLSOUTH COMPANIES        |
|               EXCEPT PURSUANT TO A WRITTEN AGREEMENT.                   |
|            MUST BE STORED IN LOCKED FILES WHEN NOT IN USE.              |
|                                                                         |
-------------------------------------------------------------------------*/

/*
 * @(#)xstaServerConnection.h 71.6
 * created on 00/06/07 at 16:10:23
 */

#ifndef xstaServerConnection_h
#define xstaServerConnection_h

#ifdef WIN32
#pragma warning(disable : 4290)
#endif

// undefine conflicting symbols
#ifdef connect
#undef connect
#endif

// No CORBA types should leak through the Client API.

#include "xstaCommon.h"

class xstTapFactory1;
typedef xstTapFactory1 *xstTapFactory1_ptr;
class xscCogConnector;

class XST_DECLSPEC xstaServerConnection
{
  protected:
      xstaServerConnection (
            const char *applid,
            const char *applpass,
            const char *userid)
            throw ();

      ~xstaServerConnection ()
            throw (
                  xstaCommon::SecurityException,
                  xstaCommon::BackendResourceLimitation,
                  xstaCommon::GatewayTimeout
            );

  public:
      void applicationId (const char *applid);

      const char *applicationId ();

      void applicationPassword (const char *applpass);

      const char *applicationPassword ();

      const char *getApplicationPassword () const;

      void userId (const char *userid);

      const char *userId ();
```

```
    void clecId (const char *clecid);

    static const char *apiVersion ();

    static void changePassword (
          const char *applid,
          const char *oldPassword,
          const char *newPassword
          ) throw (
                xstaCommon::SecurityException,
                xstaCommon::InvalidData,
                xstaCommon::BackendResourceLimitation,
                xstaCommon::GatewayTimeout
          );

    void routingInfo (const xstaCommon::AttributeList &routing_list);

    static void finalize ();

public:
    static xscCogConnector *cogConnector_;

protected:
    void acquireCredentials ()
          throw (
                xstaCommon::BackendResourceLimitation,
                xstaCommon::GatewayTimeout,
                xstaCommon::InvalidData,
                xstaCommon::SecurityException
          );

public:
    void surrenderCredentials ()
          throw (
                xstaCommon::SecurityException,
                xstaCommon::BackendResourceLimitation,
                xstaCommon::GatewayTimeout
          );

    static void readConfig ()
          throw (
                xstaCommon::InvalidData
          );

protected:

    static void readConfigOnce ()
          throw (
                xstaCommon::InvalidData
          );

    void transactionType (xstaCommon::TransactionType ttype);
    void contractType (const char *contype);

    static void connect ()
          throw (
                xstaCommon::GatewayTimeout,
                xstaCommon::BackendResourceLimitation,
                xstaCommon::InvalidData
          );

    void initClecId ();

    void validate (xstaCommon::InvalidData &err)
          throw (
                xstaCommon::InvalidData
          );

    static int isBlank (const char *str);

    void makeHeader (void *header, const char *inquiryNumber);
```

```
    static void exceptionInvalidData (
          const char *elementName,
          const char *msgId, const char *msgTxt
          ) throw (xstaCommon::InvalidData);

    static void buildErrList (const char *elementName,
          const char *msgId,
          const char *msgTxt,
          xstaCommon::InvalidData &err);


protected:
    static unsigned long nextEvent ();
private:
    static unsigned long event_counter_;

    unsigned long event_clecId_;

    unsigned long event_inquiry_;

    unsigned long event_routingInfo_;

protected:
    unsigned long event_query_server_;

    unsigned long event_credentials_;

    static unsigned long event_connect_;

    static unsigned long event_config_;

    void *serverObjectReference ();

    void initInquiryNumber (const char *inquiryNumberSeed);

    const char *incrementInquiryNumber ();

private:
    // State of security server connection.
    static xstTapFactoryl_ptr tapFactoryPtr_;

    // Authentication information and credentials.
    xstaString applId_;
    xstaString applPassword_;
    xstaString transactionType_;
    xstaString contractType_;
    void *tapCookie_ptr_;

    enum { inquiryNumberLength_ = 16 };
    // won't compile on HP CC: const unsigned int inquiryNumberLength_ = 16;
    char inquiryNumber_[inquiryNumberLength_+1];

    xstaCommon::AttributeList routingInfo_;

    unsigned long connectionMarker_;

    // These are set by readConfig().
    static xstaString securityServerName_;
    static xstaString securityServerHostName_;
    static xstaString initial_clecId_;

    static xstaString cogServerName_;
    static xstaString cogHostName_;

    // And this checks that they are set.
    static void checkConfigStatics () throw (xstaCommon::InvalidData);

    // Things that will go into the MsgHeader.
    xstaString userId_;
    xstaString clecId_;
```

```
};

#endif
```

## 6.4 xstaPreOrder.h

```
/*
 * @(#)xstaPreOrder.h      15.1
 * created on 98/12/15 at 19:00:12
 */

#ifndef xstaPreOrder_h
#define xstaPreOrder_h
#include "xstaServerConnection.h"

//. Common functionality for Pre-Ordering.
class XST_DECLSPEC xstaPreOrder : public xstaServerConnection
{
  protected:
     xstaPreOrder (
          const char *applid,
          const char *applpass,
          const char *userid)
          throw ();
     //. Passes arguments through to the base class.

     ~xstaPreOrder ()
          throw (
               xstaCommon::SecurityException,
               xstaCommon::BackendResourceLimitation,
               xstaCommon::GatewayTimeout
          );
     //. The base class destructor calls surrenderCredentials().

     void validate (xstaCommon::InvalidData &err)
          throw (
               xstaCommon::InvalidData
          );
     //. Perform validations common to all pre-ordering query types.
     //. Calls base class's validate() for validations common to both
     //. pre-ordering and ordering.
     //. Throws xstaCommon::InvalidData for any validation problems
     //. (missing data, invalid data, inconsistent data).
     //. InvalidData contains a list of InvalidDataElements so one
     //. InvalidDataElement is added to the list for each problem found.

  protected:
     void lst (const char *clli,
          xstaCommon::InvalidData &err);
     //. Utility function used by validation code to check LST.

     void npaNxx (const char *npannx,
          xstaCommon::InvalidData &err);
     //. Utility function used by validation code to check NPANXX.

     int alphaNumeric (const char *str);
     //. Utility function used by validation code to check alphanumeric.

     int numeric (const char *str);
     //. Utility function used by validation code to check numeric.

};

#endif
```

# 6.5 xstaAddressValidation.h

```
/*
 * This file was automatically generated by xstAddressValidation.pl.
 * Do not edit this file directly!
 * @(#)xstAddressValidation.pl        73.4
 */

/*------------------------------------------------------------------------
|                                                                        |
|(C) BellSouth Telecommunications, Inc. 1999                             |
|                                                                        |
|                       PRIVATE/PROPRIETARY/LOCK                         |
|           CONTAINS PRIVATE AND/OR PROPRIETARY INFORMATION.             |
|      MAY NOT BE USED OR DISCLOSED OUTSIDE THE BELLSOUTH COMPANIES      |
|                EXCEPT PURSUANT TO A WRITTEN AGREEMENT.                 |
|             MUST BE STORED IN LOCKED FILES WHEN NOT IN USE.            |
|                                                                        |
------------------------------------------------------------------------*/

#ifndef xstaAddressValidation_h
#define xstaAddressValidation_h

#include "xstaPreOrder.h"

class xstAddressValidation3;

/*
 * RSAG -- Address Validation
 */
class XST_DECLSPEC xstaAddressValidation : public xstaPreOrder
{
  public:
      xstaAddressValidation (
            const char *applid,
            const char *applpass,
            const char *userid
            )
            throw ();
      //. Sets initial values from provided strings.  If any string is
      //. empty or 0, the corresponding set method must be called before
      //. first query.

      ~xstaAddressValidation ()
            throw (
                  xstaCommon::SecurityException,
                  xstaCommon::BackendResourceLimitation,
                  xstaCommon::GatewayTimeout
            );
      //. The base class destructor calls surrenderCredentials().

      typedef xstaString HouseNumberPrefix;  // SAPR
      typedef xstaString HouseNumber; // SANO
      typedef xstaString HouseNumberSuffix;  // SASF
      typedef xstaCommon::Directional StreetDirectional;    // SASD
      typedef xstaString StreetThoroughfare; // SATH
      typedef xstaString StreetName;  // SASN
      typedef xstaString StreetSuffix;       // SASS
      typedef xstaString Room;        // ROOM
      typedef xstaString Building;    // BLDG
      typedef xstaString Floor;       // FLOOR
      typedef xstaString DescriptiveLocation;        // SADLO
      typedef xstaString City;        // SALOC
      typedef xstaString State;       // SAST
      typedef xstaString ZipCode;     // SAZC
      typedef unsigned char UnnumberedHouse;
      typedef xstaString CrossBoundary;
      typedef xstaString PostalBox;
      typedef xstaString Route;
      typedef xstaString RateZone;
```

```
typedef xstaString DriveInstructions;
typedef xstaString CompanyIndicator;
typedef xstaString TariffExchangeCode;

struct XST_DECLSPEC AddressPattern {
      xstaString           structurePattern;   // TAG-DSA-STRUC-TYPEPAT1
      xstaString           elevationPattern;   // TAG-DSA-ELEV-TYPEPAT1
      xstaString           unitPattern;        // TAG-DSA-UNIT-TYPEPAT1
};
xstaLIST_DECLARATION (AddressPattern, AddressPatternList);

struct XST_DECLSPEC Address {
      HouseNumber          houseNumber;
      HouseNumberSuffix    houseNumberSuffix;
      StreetDirectional    streetDirectional;
      StreetThoroughfare   streetThoroughfare;
      StreetName           streetName;
      StreetSuffix         streetSuffix;
      Room                 room;
      Building             building;
      Floor                floor;
      DescriptiveLocation  descriptiveLocation;
      City                 city;
      State                state;
      ZipCode              zipCode;
      UnnumberedHouse      unnumberedHouseIndicator;  // Added for TAG
      CrossBoundary        crossBoundary;      // Added for TAG
      PostalBox            postalBox;  // Added for TAG
      Route                route;      // Added for TAG
      RateZone             rateZone;
      DriveInstructions    driveInstructions;
      CompanyIndicator     companyIndicator;
      AddressPatternList   addressPattern;     // Added for TAG - response only

      Address ();
};
xstaLIST_DECLARATION (Address, AddressList);

struct XST_DECLSPEC HouseNumberRange {
      HouseNumber          fromHouseNumber;
      HouseNumber          toHouseNumber;
};

struct XST_DECLSPEC AddressWithRange {
      HouseNumber          houseNumber;        // ZP-98160-34
      HouseNumberRange     houseNumberRange;
      HouseNumberSuffix    houseNumberSuffix;
      xstaString           oddEvenIndicator;   // TAG-DSA-ODD-EVEN-IND
      xstaString           assignedHouseNumberStatus; // TAG-DSA-AHN-STATUS
      StreetDirectional    streetDirectional;
      StreetThoroughfare   streetThoroughfare;
      StreetName           streetName;
      StreetSuffix         streetSuffix;
      Room                 room;
      Building             building;
      Floor                floor;
      DescriptiveLocation  descriptiveLocation;
      City                 city;
      State                state;
      ZipCode              zipCode;
      CrossBoundary        crossBoundary;      // ZP-98156-10
      PostalBox            postalBox;  // ZP-98160-34
      Route                route;      // ZP-98160-34
      RateZone             rateZone;
      DriveInstructions    driveInstructions;
      CompanyIndicator     companyIndicator;

      AddressWithRange ();
};

enum AddressType {
```

```
        UnfieldedChoice,
        RangedChoice,
        FieldedChoice
    };

    /* IDL union */
    struct AlternativeAddress {
      private:
        AddressType __d;
        int _isSet;
        union {
            AddressWithRange * _addresswithrange_;
            Address * _fieldedaddress_;
        };
      public:

        AddressType _d() const { return __d; }


            AddressWithRange &addresswithrange () { return *_addresswithrange_; }
            const AddressWithRange &addresswithrange () const { return *_addresswithrange_;
}
            void addresswithrange (const AddressWithRange &from);

            Address &fieldedaddress () { return *_fieldedaddress_; }
            const Address &fieldedaddress () const { return *_fieldedaddress_; }
            void fieldedaddress (const Address &from);

        AlternativeAddress ();
        AlternativeAddress (const AlternativeAddress &);
        ~AlternativeAddress ();
        AlternativeAddress &operator= (const AlternativeAddress &);
    };

    struct XST_DECLSPEC ServiceAddressTelephoneInfo {
        xstaCommon::TelephoneNumber telephoneNumber;     // TAG-DTN-TN
        xstaString          quickServeIndicator;       // TAG-DTN-QUICK-SERV-IND
        xstaString          addressStatus;      // TAG-DTN-ADDR-STATUS
        xstaString          availableFacilitiesIndicator;      // TAG-DTN-FACAVAIL
    };
    xstaLIST_DECLARATION (ServiceAddressTelephoneInfo, ServiceAddressTelephoneInfoList);

    struct XST_DECLSPEC AreaTransferInfo {
        xstaCommon::DateTime areaTransferCutDate;
        xstaCommon::DateTime areaTransferNumChgDate;
        xstaCommon::NpaNxx   areaTransferNpaNxx;
        TariffExchangeCode   tariffExchangeCode;
    };

    struct XST_DECLSPEC AlternativeAddressInfo {
        AlternativeAddress   alternativeaddress;
        xstaCommon::NpaNxx   npanxx;
        xstaCommon::LocalServiceTermination localServiceTermination;
        ServiceAddressTelephoneInfoList telephoneInfo;
        AreaTransferInfo     areaTransferInfo;
    };
    xstaLIST_DECLARATION (AlternativeAddressInfo, AlternativeAddressList);

    struct XST_DECLSPEC ServiceRestrictionInfo {
        xstaCommon::DateTime estServiceDate;
        xstaString          restrictionCode;
        xstaString          restrictionText;
    };
    xstaLIST_DECLARATION (ServiceRestrictionInfo, ServiceRestrictionInfoList);

    struct XST_DECLSPEC AddressInfoHeader {
        ServiceRestrictionInfoList serviceRestrictionInfoList;
        xstaString          serviceInstructionText;
    };

    xstaCommon::Status verifyWithAddress (
```

```
            /* in  */ const xstaAddressValidation::Address &address,
            /* in  */ const char           *inquiryNumber,
            /* out */ xstaCommon::MsgHeader &messageHeader,
            /* out */ xstaAddressValidation::AlternativeAddressList
&alternativeaddressInformation,
            /* out */ xstaAddressValidation::AddressInfoHeader &addressInfoHeader
            ) throw (
                  xstaCommon::SecurityException,
                  xstaCommon::BackendResourceLimitation,
                  xstaCommon::InvalidData,
                  xstaCommon::GatewayTimeout
            );

      xstaCommon::Status verifyWithTelephone (
            /* in  */ const char           *queryTelephone,
            /* out */ xstaCommon::MsgHeader &messageHeader,
            /* out */ xstaAddressValidation::AlternativeAddressList
&alternativeaddressInformation,
            /* out */ xstaAddressValidation::AddressInfoHeader &addressInfoHeader
            ) throw (
                  xstaCommon::SecurityException,
                  xstaCommon::BackendResourceLimitation,
                  xstaCommon::InvalidData,
                  xstaCommon::GatewayTimeout
            );

  private:
      void getTransactionObject ()
            throw (
                  xstaCommon::BackendResourceLimitation,
                  xstaCommon::GatewayTimeout,
                  xstaCommon::InvalidData
            );
      //. Get the server object pointer to handle address validation
      //. requests.

      void validateAVQ_ADDR (
            const xstaAddressValidation::Address &queryAddress,
            const char *inquiryNumber
            ) throw (
                  xstaCommon::InvalidData
            );
      //. Does data validations for verifyWithAddress.
      //. Calls base class validate() for common validations.

      void validateAVQ_TN (
            const char *queryTelephone
            ) throw (
                  xstaCommon::InvalidData
            );
      //. Does data validations for verifyWithTelephone.
      //. Calls base class validate() for common validations.

  private:
      xstAddressValidation3 *server_ptr_;

};

#endif
```

# 6.6 xstaAppointmentScheduling.h

```
/*-------------------------------------------------------------------------
|                                                                         |
|(C) BellSouth Telecommunications, Inc. 1998                              |
|                                                                         |
|     PRIVATE/PROPRIETARY/LOCK CONTAINS PRIVATE AND/OR PROPRIETARY INFORMATION
|     MAY NOT BE USED OR DISCLOSED OUTSIDE THE BELLSOUTH COMPANIES        |
|     EXCEPT PURSUANT TO A WRITTEN AGREEMENT.                             |
|     MUST BE STORED IN LOCKED FILES WHEN NOT IN USE.                     |
|                                                                         |
-------------------------------------------------------------------------*/

/*
 * This file was automatically generated by xstAppointmentScheduling.pl.
 * Do not edit this file directly!
 * @(#)xstAppointmentScheduling.pl    73.3
 */

/*-------------------------------------------------------------------------
|                                                                         |
|(C) BellSouth Telecommunications, Inc. 1999                              |
|                                                                         |
|                       PRIVATE/PROPRIETARY/LOCK                          |
|          CONTAINS PRIVATE AND/OR PROPRIETARY INFORMATION.               |
|     MAY NOT BE USED OR DISCLOSED OUTSIDE THE BELLSOUTH COMPANIES        |
|               EXCEPT PURSUANT TO A WRITTEN AGREEMENT.                   |
|            MUST BE STORED IN LOCKED FILES WHEN NOT IN USE.              |
|                                                                         |
-------------------------------------------------------------------------*/

#ifndef xstaAppointmentScheduling_h
#define xstaAppointmentScheduling_h

#include "xstaPreOrder.h"

class xstAppointmentScheduling1;

/*
 * DSAP -- Appointment Scheduling
 *
 * This varies significantly from ECIC which only provides a list
 * of date/times as output.
 */
class XST_DECLSPEC xstaAppointmentScheduling : public xstaPreOrder
{
  public:
       xstaAppointmentScheduling (
              const char *applid,
              const char *applpass,
              const char *userid
              )
              throw ();
       //. Sets initial values from provided strings.  If any string is
       //. empty or 0, the corresponding set method must be called before
       //. first query.

       ~xstaAppointmentScheduling ()
              throw (
                     xstaCommon::SecurityException,
                     xstaCommon::BackendResourceLimitation,
                     xstaCommon::GatewayTimeout
              );
       //. The base class destructor calls surrenderCredentials().

       typedef xstaCommon::DateTime DateTime;       // TAG-DAU-DATE
```

```
        struct AvailableDays {
                char days[7];  // Monday thru Sunday, Y or N.
                        // TAG-DAA-CO-MON1 .. TAG-DAA-CO-SUN1 or
                        // TAG-DAA-PV-MON1 .. TAG-DAA-PV-SUN1
                AvailableDays ();
        };
        xstaLIST_DECLARATION (AvailableDays, AvailableDaysList);
        xstaLIST_DECLARATION (DateTime, HolidayList);

        struct XST_DECLSPEC CloseDate {
                DateTime            closeDate;       // TAG-DAU-DATE
                xstaString          reasonCode1;     // TAG-DAU-CLOSE-REASCD1
                xstaString          reasonCode2;     // TAG-DAU-CLOSE-REASCD2
        };
        xstaLIST_DECLARATION (CloseDate, CloseDateList);

        struct XST_DECLSPEC Interval {
                xstaString          pvResidentialReinstall; // TAG-DAI-PV-REINST
                xstaString          reinstall3;      // TAG-DAI-REINST-3
                xstaString          residentialNewinstall1_2;       // TAG-DAI-NEWINST1-2
                xstaString          newinstall3;     // TAG-DAI-NEWINST3
                xstaString          newinstall4;     // TAG-DAI-NEWINST4
                xstaString          newinstall5;     // TAG-DAI-NEWINST5
                xstaString          newinstall6_10;  // TAG-DAI-NEWINST6-10
                xstaString          newinstall11_15; // TAG-DAI-NEWINST11-15
                xstaString          addLine; // TAG-DAI-ADDLINE
                xstaString          residentialInsideWire;  // TAG-DAI-RES-IW
                xstaString          pvBusinessReinstall;    // TAG-DAI-PV-REINST-BUS
                xstaString          businessNewinstall1_2;  // TAG-DAI-NEWINST1-2-BUS
                xstaString          businessInsideWire;     // TAG-DAI-BUS-IW
                xstaString          quickService;    // TAG-DAI-QUICK-SERVICE
        };
        xstaLIST_DECLARATION (Interval, IntervalList);

        struct XST_DECLSPEC AppointmentData {
                AvailableDaysList   coAvailability;  // 0 or 1 entries
                AvailableDaysList   pvAvailability;  // 0 or 1 entries
                IntervalList        interval;// 0 or 1 entries
                HolidayList         holidays;// 0 or more
                CloseDateList       closeDates;      // 0 or more
        };

        xstaCommon::Status verifyAvailAppts (
                /* inout */ xstaString          &npanxx,
                /* out */ xstaCommon::MsgHeader &messageHeader,
                /* out */ xstaAppointmentScheduling::AppointmentData &availableAppointments
                ) throw (
                        xstaCommon::SecurityException,
                        xstaCommon::BackendResourceLimitation,
                        xstaCommon::InvalidData,
                        xstaCommon::GatewayTimeout
                );

private:
        void getTransactionObject ()
                throw (
                        xstaCommon::BackendResourceLimitation,
                        xstaCommon::GatewayTimeout,
                        xstaCommon::InvalidData
                );
        //. Get the server object pointer to handle appointment scheduling
        //. requests.

        void validateAAQ (
                const char *npanxx
                ) throw (
                        xstaCommon::InvalidData
```

```
            );
        //. Does data validations for verifyAvailAppts.
        //. Calls base class validate() for common validations.

        xstAppointmentScheduling1 *server_ptr_;

};

#endif
```

# 6.7 xstaCustomerRecord.h

```
/*
 * This file was automatically generated by xstCustomerRecord.pl.
 * Do not edit this file directly!
 * @(#)xstCustomerRecord.pl   73.3
 */

/*-------------------------------------------------------------------------
|                                                                         |
|(C) BellSouth Telecommunications, Inc. 1999                              |
|                                                                         |
|                       PRIVATE/PROPRIETARY/LOCK                          |
|          CONTAINS PRIVATE AND/OR PROPRIETARY INFORMATION.               |
|     MAY NOT BE USED OR DISCLOSED OUTSIDE THE BELLSOUTH COMPANIES        |
|               EXCEPT PURSUANT TO A WRITTEN AGREEMENT.                   |
|            MUST BE STORED IN LOCKED FILES WHEN NOT IN USE.              |
|                                                                         |
-------------------------------------------------------------------------*/

#ifndef xstaCustomerRecord_h
#define xstaCustomerRecord_h

#include "xstaPreOrder.h"

class xstCustomerRecord1;

/*
 * CSR -- Customer Service Records
 */
class XST_DECLSPEC xstaCustomerRecord : public xstaPreOrder
{
  public:
        xstaCustomerRecord (
                const char *applid,
                const char *applpass,
                const char *userid
                )
                throw ();
        //. Sets initial values from provided strings.  If any string is
        //. empty or 0, the corresponding set method must be called before
        //. first query.

        ~xstaCustomerRecord ()
                throw (
                        xstaCommon::SecurityException,
                        xstaCommon::BackendResourceLimitation,
                        xstaCommon::GatewayTimeout
                );
        //. The base class destructor calls surrenderCredentials().


        struct XST_DECLSPEC CustomerRequest {
                xstaString           stateCode;
                xstaString           customerCode;
                xstaString           exchangeCompanyCircuitId;
                xstaCommon::TelephoneNumber accountTelephoneNumber;
                xstaString           agencyAuthorizationStatus;
                xstaString           authorizationName;
                xstaCommon::DateTime authorizationDate;
                unsigned char        lsiIndicator;

                CustomerRequest ();
        };
        typedef xstaString CustomerText;
        xstaLIST_DECLARATION (CustomerText, CustomerTextList);
```

```
        xstaCommon::Status retrieveCustomerRecord (
                /* in  */ const xstaCustomerRecord::CustomerRequest &customerRequest,
                /* out */ xstaCommon::MsgHeader &messageHeader,
                /* out */ xstaCustomerRecord::CustomerTextList &customerTextList
                ) throw (
                        xstaCommon::SecurityException,
                        xstaCommon::BackendResourceLimitation,
                        xstaCommon::InvalidData,
                        xstaCommon::GatewayTimeout
                );

    private:
        void getTransactionObject ()
                throw (
                        xstaCommon::BackendResourceLimitation,
                        xstaCommon::GatewayTimeout,
                        xstaCommon::InvalidData
                );
        //. Get the server object pointer to handle customer record
        //. requests.

        void validateCSRQ (
                const xstaCustomerRecord::CustomerRequest &customerRequest
                ) throw (
                        xstaCommon::InvalidData
                );
        //. Does data validations for retrieveCustomerRecord.
        //. Calls base class validate() for common validations.

    private:
        xstCustomerRecord1 *server_ptr_;

};

#endif
```

# 6.8 xstaServiceAvailability.h

```
/*
 * This file was automatically generated by xstServiceAvailability.pl.
 * Do not edit this file directly!
 * @(#)xstServiceAvailability.pl      73.3
 */

/*-------------------------------------------------------------------------
|                                                                         |
|(C) BellSouth Telecommunications, Inc. 1999                              |
|                                                                         |
|                      PRIVATE/PROPRIETARY/LOCK                           |
|          CONTAINS PRIVATE AND/OR PROPRIETARY INFORMATION.               |
|     MAY NOT BE USED OR DISCLOSED OUTSIDE THE BELLSOUTH COMPANIES        |
|               EXCEPT PURSUANT TO A WRITTEN AGREEMENT.                   |
|            MUST BE STORED IN LOCKED FILES WHEN NOT IN USE.              |
|                                                                         |
-------------------------------------------------------------------------*/

#ifndef xstaServiceAvailability_h
#define xstaServiceAvailability_h

#include "xstaPreOrder.h"

class xstServiceAvailability1;

/*
 * PSIMS -- Service Availability
 */
class XST_DECLSPEC xstaServiceAvailability : public xstaPreOrder
{
  public:
        xstaServiceAvailability (
                const char *applid,
                const char *applpass,
                const char *userid
                )
                throw ();
        //. Sets initial values from provided strings.  If any string is
        //. empty or 0, the corresponding set method must be called before
        //. first query.

        ~xstaServiceAvailability ()
                throw (
                        xstaCommon::SecurityException,
                        xstaCommon::BackendResourceLimitation,
                        xstaCommon::GatewayTimeout
                );
        //. The base class destructor calls surrenderCredentials().


        struct XST_DECLSPEC ProductFeatureUsoc {
                xstaString           featureName;
                xstaString           tariffUsoc;
        };
        xstaLIST_DECLARATION (ProductFeatureUsoc, ProductFeatureUsocList);

        struct XST_DECLSPEC ProductFeatureInfo {
                xstaString           productId;
                xstaString           productName;
                xstaString           featureTitle;
                xstaString           centralOfficeFeatureAvail;
                xstaCommon::DateTime featureEffectiveDate;
                xstaString           accessNumber;
                xstaString           tariffNotes;
                xstaString           tariffStatus;
                xstaCommon::DateTime tariffEffectiveDate;
```

```
            ProductFeatureUsocList productFeatureUsocList;
        };
        xstaLIST_DECLARATION (ProductFeatureInfo, ProductFeatureInfoList);

        struct XST_DECLSPEC CarrierInfo {
                xstaString          cic;
                xstaString          serviceOfferingType;
                xstaString          accessCarrierName;
                xstaString          accessCarrierNameAbbreviation;
                xstaString          accessCarrierTelephoneNumber;
        };
        xstaLIST_DECLARATION (CarrierInfo, CarrierInfoList);
        typedef xstaString ServiceAbbreviation;
        xstaLIST_DECLARATION (ServiceAbbreviation, ServiceAbbreviationList);
        typedef xstaString SwitchNpaNxx;
        xstaLIST_DECLARATION (SwitchNpaNxx, SwitchNpaNxxList);
        typedef xstaString SwitchClli;
        xstaLIST_DECLARATION (SwitchClli, SwitchCllilist);

        struct XST_DECLSPEC SwitchInfo {
                xstaString          switchType;
                xstaString          isdnIndicator;
                SwitchNpaNxxList    switchNpaNxxList;
                xstaString          eightHundredServingOffice;
                xstaString          watsServingOffice;
                SwitchCllilist      switchCllilist;
        };

        xstaCommon::Status verifyService (
                /* inout */ xstaString          &clli,
                /* in  */ const char           *npanxx,
                /* in  */ const char           *picServiceOffering,
                /* in  */ const xstaServiceAvailability::ServiceAbbreviationList
&serviceAbbreviationList,
                /* out */ xstaCommon::MsgHeader &messageHeader,
                /* out */ xstaServiceAvailability::SwitchInfo &switchInfo,
                /* out */ xstaServiceAvailability::ProductFeatureInfoList &productFeatureInfoList,
                /* out */ xstaServiceAvailability::CarrierInfoList &carrierInfoList
                ) throw (
                        xstaCommon::SecurityException,
                        xstaCommon::BackendResourceLimitation,
                        xstaCommon::InvalidData,
                        xstaCommon::GatewayTimeout
                );

    private:
        void getTransactionObject ()
                throw (
                        xstaCommon::BackendResourceLimitation,
                        xstaCommon::GatewayTimeout,
                        xstaCommon::InvalidData
                );
        //. Get the server object pointer to handle service availability
        //. requests.

        void validateSAQ (
                const char *clli,
                const char *npanxx,
                const char *picServiceOffering,
                const xstaServiceAvailability::ServiceAbbreviationList &serviceAbbreviationList
                ) throw (
                        xstaCommon::InvalidData
                );
        //. Does data validations for verifyService.
        //. Calls base class validate() for common validations.

        xstServiceAvailability1 *server_ptr_;
```

```
};

#endif
```

# 6.9 xstaTelephoneNumberAssignment.h

```
/*
 * This file was automatically generated by xstTelephoneNumberAssignment.pl.
 * Do not edit this file directly!
 * @(#)xstTelephoneNumberAssignment.pl       73.2
 */

/*-------------------------------------------------------------------------
|                                                                         |
|(C) BellSouth Telecommunications, Inc. 1999                              |
|                                                                         |
|                       PRIVATE/PROPRIETARY/LOCK                          |
|          CONTAINS PRIVATE AND/OR PROPRIETARY INFORMATION.               |
|     MAY NOT BE USED OR DISCLOSED OUTSIDE THE BELLSOUTH COMPANIES        |
|               EXCEPT PURSUANT TO A WRITTEN AGREEMENT.                   |
|            MUST BE STORED IN LOCKED FILES WHEN NOT IN USE.              |
|                                                                         |
-------------------------------------------------------------------------*/

#ifndef xstaTelephoneNumberAssignment_h
#define xstaTelephoneNumberAssignment_h

#include "xstaPreOrder.h"

class xstTelephoneNumberAssignment2;

/*
 * ATLAS -- Telephone Number Availability
 */
class XST_DECLSPEC xstaTelephoneNumberAssignment : public xstaPreOrder
{
  public:
      xstaTelephoneNumberAssignment (
            const char *applid,
            const char *applpass,
            const char *userid
            )
            throw ();
      //. Sets initial values from provided strings.  If any string is
      //. empty or 0, the corresponding set method must be called before
      //. first query.

      ~xstaTelephoneNumberAssignment ()
            throw (
                  xstaCommon::SecurityException,
                  xstaCommon::BackendResourceLimitation,
                  xstaCommon::GatewayTimeout
            );
      //. The base class destructor calls surrenderCredentials().

      typedef xstaCommon::DateTime DateTime; // TAG-DAU-DATE

      struct XST_DECLSPEC TelephoneNumberRange {
            xstaCommon::TelephoneNumber lowValue;
            xstaCommon::TelephoneNumber highValue;
      };
      xstaLIST_DECLARATION (TelephoneNumberRange, TelephoneNumberRangeList);
      typedef xstaCommon::TelephoneNumber TelephoneNumber;
      xstaLIST_DECLARATION (TelephoneNumber, TelephoneNumberList);
      typedef xstaString ExceptionCharacter;

      enum TNOptions {
            None,
            Easy,
            Coin,
            SeqLine,
            AscendingLineDigits,
            DescendingLineDigits,
            IdenticalLineDigits
```

```
};

enum TelephoneOption {
      TN,
      DID
};
typedef xstaString ConfirmationNumber;

xstaCommon::Status getAvailableTelephoneNumbers (
      /* in  */ const char           *npanxx,
      /* in  */ unsigned short       quantityRequested,
      /* in  */ const char           *exceptionChar,
      /* in  */ xstaTelephoneNumberAssignment::TNOptions options,
      /* in  */ const char           *requestedNumberLow,     // TAG-TA-REQNUM-LOW
      /* in  */ const char           *typeOfService,  // TAG-SW-TOS-PIC-SVC-OFNG
      /* inout */ xstaString          &clli,  // TAG-SW-LST
      /* out */ xstaCommon::MsgHeader &messageHeader,
      /* out */ xstaString           &confirmationNumber,     // TAG-TA-CONFIRM-NUM
      /* out */ xstaTelephoneNumberAssignment::TelephoneNumberList &telephoneNumbers
      ) throw (
            xstaCommon::SecurityException,
            xstaCommon::BackendResourceLimitation,
            xstaCommon::InvalidData,
            xstaCommon::GatewayTimeout
      );
typedef xstaString MLHNumber;
typedef xstaString TerminalCode;

struct XST_DECLSPEC TerminalRange {
      TerminalCode         lowValue;
      TerminalCode         highValue;
};
xstaLIST_DECLARATION (TerminalCode, TerminalCodeList);

struct XST_DECLSPEC HuntGroupNumber {
      MLHNumber            huntGroupNumber;    // TAG-TA-HUNT-GRP-NUM
      xstaCommon::TelephoneNumber leadTelephoneNumber;        // TAG-TA-MLGH-LEAD-TN
      TerminalRange        inTerminalRange;    // TAG-TA-IN-TER-RANGE
      TerminalRange        outTerminalRange;   // TAG-TA-OUT-TER-RANGE
};
xstaLIST_DECLARATION (HuntGroupNumber, HuntGroupNumberList);

xstaCommon::Status getAvailableMLHTelephoneNumbers (
      /* in  */ const char           *npanxx,
      /* in  */ const char           *leadTelephoneNumber,
      /* in  */ unsigned short       quantityInTerminals,
      /* in  */ unsigned short       quantityOutTerminals,
      /* inout */ xstaTelephoneNumberAssignment::TerminalCodeList &existingMLHNumberList,
      /* in  */ const char           *lastInTerminal,
      /* in  */ const char           *lastOutTerminal,
      /* inout */ xstaString          &clli,  // TAG-SW-LST
      /* out */ xstaCommon::MsgHeader &messageHeader,
      /* out */ xstaTelephoneNumberAssignment::HuntGroupNumberList &huntGroupNumberList
      ) throw (
            xstaCommon::SecurityException,
            xstaCommon::BackendResourceLimitation,
            xstaCommon::InvalidData,
            xstaCommon::GatewayTimeout
      );

xstaCommon::Status getAvailableDIDTelephoneNumbers (
      /* in  */ const char           *npanxx,
      /* in  */ const char           *telephoneNumber,
      /* in  */ const char           *exceptionChar,
      /* in  */ unsigned short       quantityRequested,
      /* in  */ long                 routeIndex,      // TAG-ROUTE-INDEX
      /* inout */ xstaString          &clli,  // TAG-SW-LST
      /* out */ xstaCommon::MsgHeader &messageHeader,
      /* out */ xstaString           &confirmationNumber,     // TAG-TA-CONFIRM-NUM
      /* out */ unsigned short       &quantityProvided,
```

```
            /* out */ xstaTelephoneNumberAssignment::TelephoneNumberRangeList
&telephoneNumberRangeList
            ) throw (
                  xstaCommon::SecurityException,
                  xstaCommon::BackendResourceLimitation,
                  xstaCommon::InvalidData,
                  xstaCommon::GatewayTimeout
            );

      xstaCommon::Status getAvailableMiscTelephoneNumbers (
            /* in  */ const char           *npanxx,
            /* in  */ const char           *city,
            /* in  */ const char           *state,
            /* in  */ unsigned short       quantityRequested,
            /* out */ xstaCommon::MsgHeader &messageHeader,
            /* out */ unsigned short       &quantityProvided,
            /* out */ xstaCommon::NpaNxxList &npanxxList,
            /* out */ xstaTelephoneNumberAssignment::TelephoneNumberList &telephoneNumberList
            ) throw (
                  xstaCommon::SecurityException,
                  xstaCommon::BackendResourceLimitation,
                  xstaCommon::InvalidData,
                  xstaCommon::GatewayTimeout
            );

      xstaCommon::Status selectAvailableTelephoneNumbers (
            /* in  */ const char           *npanxx,
            /* in  */ const xstaTelephoneNumberAssignment::TelephoneNumberList
&telephoneNumberList,
            /* in  */ xstaTelephoneNumberAssignment::TelephoneOption option,
            /* in  */ const char           *confirmationNumber,     // TAG-TA-CONFIRM-NUM
            /* in  */ const char           *reserveUntilDate,
            /* inout */ xstaString           &clli,  // TAG-SW-LST
            /* out */ xstaCommon::MsgHeader &messageHeader
            ) throw (
                  xstaCommon::SecurityException,
                  xstaCommon::BackendResourceLimitation,
                  xstaCommon::InvalidData,
                  xstaCommon::GatewayTimeout
            );

      xstaCommon::Status cancelTelephoneNumbers (
            /* in  */ const char           *npanxx,
            /* in  */ const char           *confirmationNumber,     // TAG-TA-CONFIRM-NUM
            /* in  */ const xstaTelephoneNumberAssignment::TelephoneNumberList
&telephoneNumberList,
            /* inout */ xstaString           &clli,  // TAG-SW-LST
            /* out */ xstaCommon::MsgHeader &messageHeader
            ) throw (
                  xstaCommon::SecurityException,
                  xstaCommon::BackendResourceLimitation,
                  xstaCommon::InvalidData,
                  xstaCommon::GatewayTimeout
            );

      xstaCommon::Status cancelDIDTelephoneNumbers (
            /* in  */ const char           *npanxx,
            /* in  */ const char           *confirmationNumber,     // TAG-TA-CONFIRM-NUM
            /* in  */ const xstaTelephoneNumberAssignment::TelephoneNumberRangeList
&didRangeList,    // TAG-DTL-DID-RANGE
            /* inout */ xstaString           &clli,  // TAG-SW-LST
            /* out */ xstaCommon::MsgHeader &messageHeader
            ) throw (
                  xstaCommon::SecurityException,
                  xstaCommon::BackendResourceLimitation,
                  xstaCommon::InvalidData,
                  xstaCommon::GatewayTimeout
            );

      xstaCommon::Status cancelMLHTelephoneNumbers (
            /* in  */ const char           *npanxx,
```

```
            /* in  */ const char           *return1,
            /* in  */ const char           *return2,
            /* inout */ xstaString           &clli, // TAG-SW-LST
            /* out */ xstaCommon::MsgHeader &messageHeader
            ) throw (
                  xstaCommon::SecurityException,
                  xstaCommon::BackendResourceLimitation,
                  xstaCommon::InvalidData,
                  xstaCommon::GatewayTimeout
            );

  private:
      void getTransactionObject ()
            throw (
                  xstaCommon::BackendResourceLimitation,
                  xstaCommon::GatewayTimeout,
                  xstaCommon::InvalidData
            );
      //. Get the server object pointer to handle telephone number
      //. requests.

      void lstNpaNxx (
            const char *npanxx,
            const char *clli,
            xstaCommon::InvalidData &err);
      //. Utility function used by validation code to check LST and NPANXX.

      void confirmNum(const char *confirmationNumber,
            unsigned long telephoneNumberListLength,
            xstaCommon::InvalidData &err);
      //. Utility function used by validation code to check confirmation
      //. number.

      void validateTNAQ (
            const char *npanxx,
            unsigned short quantityRequested,
            const char *exceptionChar,
            xstaTelephoneNumberAssignment::TNOptions options,
            const char *clli
            ) throw (
                  xstaCommon::InvalidData
            );
      //. Does data validations for getAvailableTelephoneNumbers.
      //. Calls base class validate() for common validations.

      void validateTNAQ_MLH (
            const char *npanxx,
            const char *leadTelephoneNumber,
            unsigned short quantityInTerminals,
            unsigned short quantityOutTerminals,
            const xstaTelephoneNumberAssignment::TerminalCodeList &existingMLHNumberList,
            const char *lastInTerminal,
            const char *lastOutTerminal,
            const char *clli
            ) throw (
                  xstaCommon::InvalidData
            );
      //. Does data validations for getAvailableMLHTelephoneNumbers.
      //. Calls base class validate() for common validations.

      void validateTNAQ_DID (
            const char *npanxx,
            const char *telephoneNumber,
            const char *exceptionChar,
            unsigned short quantityRequested,
            long routeIndex,
            const char *clli
            ) throw (
                  xstaCommon::InvalidData
            );
      //. Does data validations for getAvailableDIDTelephoneNumbers.
```

```
        //. Calls base class validate() for common validations.

        void validateTNAQ_MISC (
              const char *npanxx,
              const char *city,
              const char *state,
              unsigned short quantityRequested
              ) throw (
                    xstaCommon::InvalidData
              );
        //. Does data validations for getAvailableMiscTelephoneNumbers.
        //. Calls base class validate() for common validations.

        void validateTNSQ (
              const char *npanxx,
              const xstaTelephoneNumberAssignment::TelephoneNumberList &telephoneNumbers,
              xstaTelephoneNumberAssignment::TelephoneOption option,
              const char *confirmationNumber,
              const char *reserveUntilDate,
              const char *clli
              ) throw (
                    xstaCommon::InvalidData
              );
        //. Does data validations for selectAvailableTelephoneNumbers.
        //. Calls base class validate() for common validations.

        void validateTNCAN_TN (
              const char *npanxx,
              const char *confirmationNumber,
              const xstaTelephoneNumberAssignment::TelephoneNumberList &telephoneNumberList,
              const char *clli
              ) throw (
                    xstaCommon::InvalidData
              );
        //. Does data validations for cancelTelephoneNumbers.
        //. Calls base class validate() for common validations.

        void validateTNCAN_DID (
              const char *npanxx,
              const char *confirmationNumber,
              const xstaTelephoneNumberAssignment::TelephoneNumberRangeList &didRangeList,
              const char *clli
              ) throw (
                    xstaCommon::InvalidData
              );
        //. Does data validations for cancelTelephoneNumbers.
        //. Calls base class validate() for common validations.

        void validateTNCAN_MLH (
              const char *npanxx,
              const char *return1,
              const char *return2,
              const char *clli
              ) throw (
                    xstaCommon::InvalidData
              );
        //. Does data validations for cancelTelephoneNumbers.
        //. Calls base class validate() for common validations.

        xstTelephoneNumberAssignment2 *server_ptr_;

};

#endif
```

# 6.10 xstaLoop.h

```
/*
 * This file was automatically generated by xstLoop.pl.
 * Do not edit this file directly!
 * @(#)xstLoop.pl      77.1
```

```
 */

/*-------------------------------------------------------------------------
|                                                                         |
|(C) BellSouth Telecommunications, Inc. 1999                              |
|                                                                         |
|                       PRIVATE/PROPRIETARY/LOCK                          |
|          CONTAINS PRIVATE AND/OR PROPRIETARY INFORMATION.               |
|     MAY NOT BE USED OR DISCLOSED OUTSIDE THE BELLSOUTH COMPANIES        |
|               EXCEPT PURSUANT TO A WRITTEN AGREEMENT.                   |
|            MUST BE STORED IN LOCKED FILES WHEN NOT IN USE.              |
|                                                                         |
-------------------------------------------------------------------------*/

#ifndef xstaLoop_h
#define xstaLoop_h

#include "xstaPreOrder.h"

class xstLoop2;

class xstaPvInterface;
class xscaLoop;

/*
 * LOOP -- CLEC Loop Makeup and Reservation
 */
class XST_DECLSPEC xstaLoop : public xstaServerConnection
{
  public:
        xstaLoop (
                const char *applid,
                const char *applpass,
                const char *userid
                )
                throw ();
        //. Sets initial values from provided strings.  If any string is
        //. empty or 0, the corresponding set method must be called before
        //. first query.

        ~xstaLoop ()
                throw (
                        xstaCommon::SecurityException,
                        xstaCommon::BackendResourceLimitation,
                        xstaCommon::GatewayTimeout
                );
        //. The base class destructor calls surrenderCredentials().

        typedef xstaString HouseNumber;       // SANO
        typedef xstaString HouseNumberSuffix; // SASF
        typedef xstaCommon::Directional StreetDirectional;   // SASD
        typedef xstaString StreetThoroughfare;       // SATH
        typedef xstaString StreetName;// SASN
        typedef xstaString StreetSuffix;      // SASS
        typedef xstaString Room;      // ROOM
        typedef xstaString Building;  // BLDG
        typedef xstaString Floor;     // FLOOR
        typedef xstaString City;      // SALOC
        typedef xstaString State;     // SAST
        typedef xstaString ZipCode;   // SAZC
        typedef xstaString UnnumberedHouse;
        typedef unsigned long Quantity;

        struct XST_DECLSPEC Address {
                HouseNumber           houseNumber;
                HouseNumberSuffix     houseNumberSuffix;
                StreetName            streetName;
```

```
        StreetDirectional    streetDirectional;
        StreetThoroughfare   streetThoroughfare;
        StreetSuffix         streetSuffix;
        Building             building;
        Floor                floor;
        Room                 room;
        City                 city;
        State                state;
        ZipCode              zipCode;
        UnnumberedHouse      unnumberedHouseIndicator;      // Added for TAG
};

struct XST_DECLSPEC Spl {
        xstaString           ga;      // Gauge
        xstaString           lgth;    // Length
        xstaString           uba;     // Type of Cable
        xstaString           capac;   // Capacitance
        xstaString           btoff;   // Bridge Tap Offset
};
xstaLIST_DECLARATION (Spl, SplList);

struct XST_DECLSPEC Bo {
        xstaString           boCap;   // Build Out Capacity
        xstaString           boRes;   // Build Out Resistance
        xstaString           boOff;   // Build Out Offset
};
xstaLIST_DECLARATION (Bo, BoList);

struct XST_DECLSPEC Lmu {
        xstaString           lmstat;  // Loop Makeup Status
        xstaString           luint;   // Length Unit
        xstaString           nld;     // Load Point Number, Null if Non-loaded
        xstaString           coil;    // Load Coil Type
        xstaString           es;      // End Section
        xstaString           ldsp;    // Load Spacing
        BoList               boList;  // Build Out Aggregate List, 1-2 times per LMU
        SplList              splList; // Splice Section Aggregate List, 1-10 times per LMU
};
xstaLIST_DECLARATION (Lmu, LmuList);

struct XST_DECLSPEC Fn {
        xstaString           ca;      // Cable Identifier
        xstaString           pr;      // Pair Identifier
        xstaString           abp;     // Assignable Binding Post
        xstaString           tea;     // Terminal Identifier
        xstaString           trmed;   // Transmission Media Type
        xstaString           tlm;     // Telemetry Indicator
        xstaString           rla;     // Remote Location Address
        xstaString           rlc;     // Remote Term CLLI Code
        xstaString           lts;     // Line Term Status
        xstaString           rloe;    // Remote Loop Origination Equipment
        xstaString           onutype; // Optical Network Unit Type
        LmuList              lmuList; // Loop Makedup Aggregate List, 1 time per Fn
};
xstaLIST_DECLARATION (Fn, FnList);

struct XST_DECLSPEC Loop {
        xstaString           lpstat;  // Status of assembled facility
        xstaString           ssc;     // Single Subscriber Carrier Indicator
        xstaString           rtf;     // Receive/Transmit Indicator
        Quantity             rz;      // Resistance Zone
        Quantity             cz;      // Carrier Zone
        FnList               fnList;  // Segment Aggregate List, 1-9 times per Loop
};
xstaLIST_DECLARATION (Loop, LoopList);

xstaCommon::Status makeupQueryWorkingLoops (
```

```
            /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
            /* in  */ const char           *cc,   // Company Code
            /* in  */ const char           *cktId,
            /* in  */ const char           *tn,
            /* in  */ const xstaLoop::Address &address,
            /* out */ xstaCommon::MsgHeader &messageHeader,
            /* out */ xstaLoop::LoopList   &loopList
            ) throw (
                    xstaCommon::SecurityException,
                    xstaCommon::BackendResourceLimitation,
                    xstaCommon::InvalidData,
                    xstaCommon::GatewayTimeout
            );

    xstaCommon::Status makeupQuerySpare (
            /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
            /* in  */ const char           *cc,   // Company Code
            /* in  */ xstaLoop::Quantity   numberRequested,
            /* in  */ const char           *nc,
            /* in  */ const char           *nci,
            /* in  */ const char           *secnci,
            /* in  */ const xstaLoop::Address &address,
            /* out */ xstaCommon::MsgHeader &messageHeader,
            /* out */ xstaLoop::LoopList   &loopList
            ) throw (
                    xstaCommon::SecurityException,
                    xstaCommon::BackendResourceLimitation,
                    xstaCommon::InvalidData,
                    xstaCommon::GatewayTimeout
            );

    xstaCommon::Status reservationSpare (
            /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
            /* in  */ const char           *cc,   // Company Code
            /* in  */ xstaLoop::Quantity   numberRequested,
            /* in  */ const char           *nc,
            /* in  */ const char           *nci,
            /* in  */ const char           *secnci,
            /* in  */ const xstaLoop::Address &address,
            /* out */ xstaCommon::MsgHeader &messageHeader,
            /* out */ xstaLoop::Quantity   &numberReserved,
            /* out */ xstaString           &resid,
            /* out */ xstaLoop::LoopList   &loopList
            ) throw (
                    xstaCommon::SecurityException,
                    xstaCommon::BackendResourceLimitation,
                    xstaCommon::InvalidData,
                    xstaCommon::GatewayTimeout
            );

    xstaCommon::Status reservationCancel (
            /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
            /* in  */ const char           *cc,   // Company Code
            /* in  */ const char           *resid,
            /* in  */ const xstaLoop::Address &address,
            /* out */ xstaCommon::MsgHeader &messageHeader
            ) throw (
                    xstaCommon::SecurityException,
                    xstaCommon::BackendResourceLimitation,
                    xstaCommon::InvalidData,
                    xstaCommon::GatewayTimeout
            );

private:
    void getTransactionObject ()
            throw (
                    xstaCommon::BackendResourceLimitation,
```

```
                xstaCommon::GatewayTimeout,
                xstaCommon::InvalidData
        );
//. Get the server object pointer to handle address validation
//. requests.

xstaString validate (xstaPvInterface &pv)
        throw (
                xstaCommon::InvalidData
        );
//. Perform validations common to all preorder loop query types.
//. Calls base class's validate() for validations common to both
//. pre-ordering and ordering.
//. Throws xstaCommon::InvalidData for any validation problems
//. (missing data, invalid data, inconsistent data).
//. InvalidData contains a list of InvalidDataElements so one
//. InvalidDataElement is added to the list for each problem found.
//. Returns a string used for routing.

xstaString validateLOOP_Makeup_Working (
        const char           *cc,
        const char           *cktId,
        const char           *tn,
        const xstaLoop::Address &address
        ) throw (
                xstaCommon::InvalidData
        );
//. Does data validations for makeupQueryWorkingLoops.
//. Calls base class validate() for common validations.

xstaString validateLOOP_Makeup_Spare (
        const char           *cc,
        xstaLoop::Quantity   numberRequested,
        const char           *nc,
        const char           *nci,
        const char           *secnci,
        const xstaLoop::Address &address
        ) throw (
                xstaCommon::InvalidData
        );
//. Does data validations for makeupQuerySpare.
//. Calls base class validate() for common validations.

xstaString validateLOOP_Reservation_Spare (
        const char           *cc,
        xstaLoop::Quantity   numberRequested,
        const char           *nc,
        const char           *nci,
        const char           *secnci,
        const xstaLoop::Address &address
        ) throw (
                xstaCommon::InvalidData
        );
//. Does data validations for reservationSpare.
//. Calls base class validate() for common validations.

xstaString validateLOOP_Reservation_Cancel (
        const char           *cc,
        const char           *resid,
        const xstaLoop::Address &address
        ) throw (
                xstaCommon::InvalidData
        );
//. Does data validations for verifyWithTelephone.
//. Calls base class validate() for common validations.

void makeHeader (void *real_standardHeader,
```

```
                void *real_orderHeader,
                const char *cc,
                const xstaCommon::TestProdIndicator testProdIndicator,
                xstaCommon::MsgHeader &messageHeader
                );
        //. Populates xstCommon::OrderStandardHeader and OrderHeader
        //. objects for ordering class and xstaCommon::MsgHeader for caller.
        //. Assumes that cc and testProdIndicator are already set.

  private:
        xstLoop2 *server_ptr_;
        xscaLoop *contract_ptr_;

};

#endif
```

## 6.11 xstaDueDate.h

```
/*
 * This file was automatically generated by xstDueDate.pl.
 * Do not edit this file directly!
 * @(#)xstDueDate.pl   75.1
 */

/*-------------------------------------------------------------------------
|                                                                         |
|(C) BellSouth Telecommunications, Inc. 1999                              |
|                                                                         |
|                       PRIVATE/PROPRIETARY/LOCK                          |
|          CONTAINS PRIVATE AND/OR PROPRIETARY INFORMATION.               |
|     MAY NOT BE USED OR DISCLOSED OUTSIDE THE BELLSOUTH COMPANIES        |
|               EXCEPT PURSUANT TO A WRITTEN AGREEMENT.                   |
|            MUST BE STORED IN LOCKED FILES WHEN NOT IN USE.              |
|                                                                         |
-------------------------------------------------------------------------*/

#ifndef xstaDueDate_h
#define xstaDueDate_h

#include "xstaPreOrder.h"

class xstDueDate3;

class xstaPvInterface;

/*
 * CDD -- Due Date Calculation
 */
class XST_DECLSPEC xstaDueDate : public xstaPreOrder
{
  public:
        xstaDueDate (
                const char *applid,
                const char *applpass,
                const char *userid
                )
                throw ();
        //. Sets initial values from provided strings.  If any string is
        //. empty or 0, the corresponding set method must be called before
        //. first query.

        ~xstaDueDate ()
                throw (
                        xstaCommon::SecurityException,
                        xstaCommon::BackendResourceLimitation,
```

```
                xstaCommon::GatewayTimeout
        );
//. The base class destructor calls surrenderCredentials().

typedef xstaString Date;

struct XST_DECLSPEC TypeRecord {
        xstaString          reqtyp;  // Request Type
        xstaString          act;     // Activity Type
        xstaString          tos;     // Type of Service
        xstaString          nc;      // Network Channel Code
        xstaString          secnci;  // Secondary Network Channel Interface Code
};

struct XST_DECLSPEC OrderInfo {
        xstaString          uncommon;// Uncommon Dispatch Flag
        Date                ddd;     // Desired Due Date
        xstaString          lqty;    // Loop Quantity
        xstaString          npqty;   // Number Portability Quantity
        xstaString          rsqty;   // Resale Quantity
};

struct XST_DECLSPEC Feature {
        xstaString          fa;      // Feature Activity
        xstaString          feature; // Feature
        xstaString          featureDetail;   // Feature Detail
};
xstaLIST_DECLARATION (Feature, FeatureList);

struct XST_DECLSPEC LineInfo {
        xstaString          npt;     // Number Portability Type
        xstaString          lna;     // Line Activity
        xstaString          lneclssvc;       // Line Level Class of Service
        FeatureList         featureList;     // Feature Section
};
xstaLIST_DECLARATION (LineInfo, LineInfoList);

struct XST_DECLSPEC ServiceAddress {
        xstaString          houseNumber;
        xstaString          houseNumberSuffix;
        xstaString          streetDirectional;
        xstaString          streetName;
        xstaString          streetThoroughfare;
        xstaString          streetSuffix;
        xstaString          room;
        xstaString          building;
        xstaString          floor;
        xstaString          city;
        xstaString          state;
        xstaString          zipCode;
        xstaString          hunting;
        LineInfoList        lineInfoList;    // Line Info Section
};
xstaLIST_DECLARATION (ServiceAddress, ServiceAddressList);

struct XST_DECLSPEC DueDateResponse {
        xstaString          dueDate; // Due Date
        xstaString          msgId;   // Due Date Message ID
        xstaString          msgTxt;  // Due Date Message Text
        xstaString          pvIndicator;     // Premise Visit Indicator
};

xstaCommon::Status estimatedServiceDate (
        /* in  */ const xstaDueDate::TypeRecord &typeRecord,
        /* in  */ const xstaDueDate::OrderInfo &orderInfo,
        /* in  */ const xstaDueDate::ServiceAddressList &serviceAddressList,
        /* out */ xstaCommon::MsgHeader &messageHeader,
```

```
                /* out */ xstaDueDate::DueDateResponse &dueDate
                ) throw (
                        xstaCommon::SecurityException,
                        xstaCommon::BackendResourceLimitation,
                        xstaCommon::InvalidData,
                        xstaCommon::GatewayTimeout
                );

  private:
        void getTransactionObject ()
                throw (
                        xstaCommon::BackendResourceLimitation,
                        xstaCommon::GatewayTimeout,
                        xstaCommon::InvalidData
                );
        //. Get the server object pointer to handle due date calculation
        //. requests.

        void validate (xstaPvInterface &pv)
                throw (
                        xstaCommon::InvalidData
                );
        //. Perform validations common to all due date query types.
        //. Calls base class's validate() for validations common to both
        //. pre-ordering and ordering.
        //. Throws xstaCommon::InvalidData for any validation problems
        //. (missing data, invalid data, inconsistent data).
        //. InvalidData contains a list of InvalidDataElements so one
        //. InvalidDataElement is added to the list for each problem found.

        void validateESD (
                const xstaDueDate::TypeRecord &typeRecord,
                const xstaDueDate::OrderInfo &orderInfo,
                const xstaDueDate::ServiceAddressList &serviceAddressList
                ) throw (
                        xstaCommon::InvalidData
                );
        //. Does data validations for estimatedServiceDate.
        //. Calls current class validate() for common validations.

  private:
        xstDueDate3 *server_ptr_;

};

#endif
```

## 6.12 6.12 xstaOrder.h

```
/*
 * This file was automatically generated by xstOrder.pl.
 * Do not edit this file directly!
 * @(#)xstOrder.pl     87.1
 */

/*-----------------------------------------------------------------------
|                                                                       |
|(C) BellSouth Telecommunications, Inc. 1999                            |
|                                                                       |
|                       PRIVATE/PROPRIETARY/LOCK                        |
|           CONTAINS PRIVATE AND/OR PROPRIETARY INFORMATION.            |
|      MAY NOT BE USED OR DISCLOSED OUTSIDE THE BELLSOUTH COMPANIES     |
|                EXCEPT PURSUANT TO A WRITTEN AGREEMENT.                |
|             MUST BE STORED IN LOCKED FILES WHEN NOT IN USE.           |
|                                                                       |
```

```
-----------------------------------------------------------------------*/

#ifndef xstaOrder_h
#define xstaOrder_h

#include "xstaServerConnection.h"

class xstOrder9;

class xstaPvInterface;
class xscaOrder;

class XST_DECLSPEC xstaOrder : public xstaServerConnection
{
  public:
       xstaOrder (
              const char *applid,
              const char *applpass,
              const char *userid
              )
              throw ();
       //. Passes arguments through to the base class.

       ~xstaOrder ()
              throw (
                     xstaCommon::SecurityException,
                     xstaCommon::BackendResourceLimitation,
                     xstaCommon::GatewayTimeout
              );
       //. The base class destructor calls surrenderCredentials().

       typedef xstaString Date;       // CCYYMMDD
       typedef xstaString Time;       // HHMM
       typedef xstaString NpaNxx;
       typedef xstaString TelephoneNumber;

       struct XST_DECLSPEC LSRHuntGroupIdentification {
              xstaString           locnum;  // Location Number  (HUNTING)
              xstaString           hnum;    // Hunt Number
              xstaString           ha;      // Hunt Group Activity
              xstaString           hid;     // Hunting Group Identifier
              xstaString           tli;     // Telephone Line Identifier
              xstaString           hntyp;   // Hunting Type Code
       };

       struct XST_DECLSPEC LSRHuntDetail {
              xstaString           hla;     // Line Hunt Group Activity
              xstaString           htseq;   // Hunting Sequence
              xstaString           notyp;   // Number Type
              xstaString           ht;      // Hunting Telephone Number
       };
       xstaLIST_DECLARATION (LSRHuntDetail, LSRHuntDetailList);

       struct XST_DECLSPEC LSRHuntGroupInformation {
              LSRHuntGroupIdentification identification;   // Hunting Group Identification
              LSRHuntDetailList    detailList;      // Hunting Detail List
       };
       xstaLIST_DECLARATION (LSRHuntGroupInformation, LSRHuntGroupInformationList);

       struct XST_DECLSPEC LSR {
              xstaString           ccna;    // Customer Carrier Name Abbreviation
              xstaString           pon;     // Purchase Order Number
              xstaString           version; // Version Number
              xstaString           lsrNumber;       // Local Service Request Number
              xstaString           somanSomecInd;   // SOMAN/SOMEC Indicator
              xstaString           locqty;  // Location Quantity
              xstaString           htqty;   // Hunting Group Quantity
```

```
xstaString          an;       // Account Number
TelephoneNumber     atn;      // Account Telephone Number
xstaString          serviceCenter;   // Service Center
Date                dateSent;// Date/Time Sent
Date                ddd;      // Desired Due Date
Date                apptimeDdd;      // Appointment Time
Date                dddo;     // Desired Due Date Out
Time                dfdt;     // Desired Frame Due Time
xstaString          project; // Project Identification
xstaString          chc;      // Coordinated Hot Cut
xstaString          reqtyp;  // Request Type
xstaString          act;      // Activity Type
xstaString          sup;      // Supplement Type
xstaString          exp;      // Expedite
xstaString          cc;       // Company Code
xstaString          albr;     // Additional Labor
xstaString          sca;      // Special Construction Authorization
xstaString          agauth;  // Agency Authorization Status
Date                dated;   // Date of Agency Authorization
xstaString          authName;// Authorization Name
xstaString          porttyp; // Port Type
xstaString          actl;     // Access Customer Terminal Location
xstaString          ai;       // Additional Point of Termination Indicator
xstaString          apot;     // Additional Point of Termination
xstaString          lst;      // Local Service Termination
NpaNxx              lso;      // Local Service Office
xstaString          tos;      // Type of Service
xstaString          spec;     // Service and Product Enhancement Code
xstaString          nc;       // Network Channel Code
xstaString          nci;      // Network Channel Interface Code
xstaString          secnci;  // Secondary Network Channel Interface Code
xstaString          rpon;     // Related Purchase Order Number
xstaString          rord;     // Related Order Number
xstaString          lspAuth; // Local Service Provider Authorization
Date                lspAuthDate;     // Local Service Provider Authorization Date
xstaString          lspAuthName;     // Local Service Provider Authorization Name
xstaString          cic;      // Carrier Identification Code
xstaString          cust;     // Customer Name
xstaString          bil;      // Billing Account Number Identifier 1
xstaString          ban1;     // Billing Account Number 1
xstaString          bi2;      // Billing Account Number Identifier 2
xstaString          ban2;     // Billing Account Number 2
xstaString          acna;     // Access Customer Name Abbreviation
Date                ebd;      // Effective Bill Date
xstaString          billName;// Billing Name
xstaString          sbillName;       // Secondary Bill Name
xstaString          billNameStreet;  // Billing Name Street Address
xstaString          billNameFloor;   // Billing Name Floor
xstaString          billNameRoom;    // Billing Name Room
xstaString          billNameCity;    // Billing Name City
xstaString          billNameState;   // Billing Name State/Province
xstaString          billNameZipCode; // Billing Name Zip Code
xstaString          billCon; // Billing Contact
xstaString          billConTelNo;    // Billing Contact Telephone Number
xstaString          vta;     // Variable Term Agreement
xstaString          init;    // Initiator Identification
xstaString          initTelNo;       // Initiator Telephone Number
xstaString          initFaxNo;       // Initiator Facsimile Number
xstaString          initStreet;      // Initiator Street Address
xstaString          initFloor;       // Initiator Floor
xstaString          initRoomMailStop;       // Initiator Room/Mail Stop
xstaString          initCity;// Initiator City
xstaString          initState;       // Initiator State/Province
xstaString          initZipCode;     // Initiator Zip Code
xstaString          impCon;  // Implementation Contact
xstaString          impConTelNo;     // Implementation Contact Telephone Number
xstaString          impConPager;     // Implementation Contact Pager Number
```

```
                xstaString          altImpCon;      // Alternate Implementation Contact
                xstaString          altImpConTelNo; // Alternate Implementation Contact Telephone
Number
                xstaString          altImpConPager; // Alternate Implementation Contact Pager
                xstaString          dsgCon;  // Design/Engineering Contact
                xstaString          drc;     // Design Routing Code
                xstaString          dsgConTelNo;     // Design/Engineering Contact Telephone
Number
                xstaString          dsgConFaxNo;     // Design/Engineering Contact Facsimile
Number
                xstaString          dsgConStreet;    // Design/Engineering Contact Street Address
                xstaString          dsgConFloor;     // Design/Engineering Contact Floor
                xstaString          dsgConRoomMailStop;     // Design/Engineering Contact
Room/Mail Stop
                xstaString          dsgConCity;      // Design/Engineering Contact City
                xstaString          dsgConState;     // Design/Engineering Contact State/Province
                xstaString          dsgConZipCode;   // Design/Engineering Contact Zip Code
                xstaString          remarks; // Remarks
                xstaString          nnsp;    // Network Service Provider Identification
                xstaString          pbt;     // POT Bay Type
                xstaString          bcs;     // Basic Class of Service
                xstaString          uncommon;// Uncommon Dispatch Flag
                xstaString          resid;   // Response Identifier
                LSRHuntGroupInformationList huntGroupInformationList;       // Hunt Group
Information
        };

        struct XST_DECLSPEC TC {
                xstaString          tcid;    // Transfer of Calls To Identifier
                xstaString          tcName;  // Transfer of Calls To Name
        };

        struct XST_DECLSPEC EUAdministrative {
                xstaString          dqty;    // Disconnect Quantity
                xstaString          ibt;     // ISDN-BRI Type
        };

        struct XST_DECLSPEC EULocationAndAccessHeader {
                xstaString          aact;    // Address Activity
                xstaString          locnum;  // Location Number
                xstaString          euName;  // End User Name
                xstaString          sano;    // Service Address House Number
                xstaString          sasf;    // Service Address House Number Suffix
                xstaString          sasd;    // Service Address Street Directional
                xstaString          sasn;    // Service Address Street Name
                xstaString          sath;    // Service Address Thoroughfare
                xstaString          sass;    // Service Address Street Suffix
                xstaString          sadlo;   // Service Address Descriptive Location
                xstaString          euFloor; // Floor-EU
                xstaString          euRoom;  // Room-EU
                xstaString          euBldg;  // Building
                xstaString          euCity;  // City-EU
                xstaString          euState; // State-EU
                xstaString          euZipCode;      // Zip Code-EU
                xstaString          lconName;// Local Contact
                xstaString          lconTelNo;      // Telephone Number-LCON
                xstaString          eumi;    // End User Moving Indicator
                xstaString          acc;     // Access Information
                xstaString          wsop;    // Working Service on Premises
                xstaString          erl;     // End User Retaining Listing
        };

        struct XST_DECLSPEC EUInsideWire {
                xstaString          iwOptions;      // Inside Wire Options
                xstaString          iwConName;      // Inside Wire Contact
                xstaString          iwConTelNo;     // Inside Wire Contact Telephone Number
        };
```

```
struct XST_DECLSPEC EUBill {
        xstaString          ean;     // Existing Account Number
        xstaString          eatn;    // Existing Account Telephone Number
        xstaString          fbi;     // Final Bill Information Indicator
        xstaString          fbBillName;      // Bill Name
        xstaString          fbSbillName;     // Secondary Bill Name
        xstaString          fbStreet;// Street Address-FB
        xstaString          fbFloor; // Floor BILLNM-FB
        xstaString          fbRoom;  // Room BILLNM-FB
        xstaString          fbCity;  // City BILLNM-FB
        xstaString          fbState; // State/PROVINCE-FB
        xstaString          fbZipCode;       // Zip Code-FB
        xstaString          fbBillCon;       // Billing Contact Final Bill
        xstaString          fbConTelNo;      // Telephone Number-FB
};

struct XST_DECLSPEC EUHeader {
        EUAdministrative    administrative; // Administrative Section
        EULocationAndAccessHeader locationAndAccess; // Location and Access Header Section
        EUInsideWire        insideWire;      // Inside Wire Section
        EUBill              bill;    // Bill Section
};

struct XST_DECLSPEC EULocationAndAccess {
        xstaString          locnum;  // Location Number
        xstaString          locact;  // Location Activity
        xstaString          euName;  // End User Name
        xstaString          sano;    // Service Address House Number
        xstaString          sasf;    // Service Address House Number Suffix
        xstaString          sasd;    // Service Address Street Directional
        xstaString          sasn;    // Service Address Street Name
        xstaString          sath;    // Service Address Thoroughfare
        xstaString          sass;    // Service Address Street Suffix
        xstaString          sadlo;   // Service Address Descriptive Location
        xstaString          euFloor; // Floor-EU
        xstaString          euRoom;  // Room-EU
        xstaString          euBldg;  // Building
        xstaString          euCity;  // City-EU
        xstaString          euState; // State-EU
        xstaString          euZipCode;       // Zip Code-EU
        xstaString          lconName;// Local Contact
        xstaString          lconTelNo;       // Telephone Number-LCON
        xstaString          acc;     // Access Information
};
xstaLIST_DECLARATION (EULocationAndAccess, EULocationAndAccessList);

struct XST_DECLSPEC EUDisconnectInformation {
        xstaString          dnum;    // Disconnect Line Number
        xstaString          discNbr; // Disconnect Telephone Number
        xstaString          ter;     // Terminal Number
        xstaString          tcOpt;   // Transfer of Call Options
        xstaString          tcPer;   // Transfer of Calls Period
        TelephoneNumber     tcToPrimary;     // Transfer of Calls to Primary Number
        TelephoneNumber     tcToSecondary;   // Transfer of Calls to Secondary Number
        TC                  primary; // Transfer of Calls Primary
        TC                  secondary;       // Transfer of Calls Secondary
};
xstaLIST_DECLARATION (EUDisconnectInformation, EUDisconnectInformationList);

struct XST_DECLSPEC EUDetail {
        EULocationAndAccessList locationAndAccessList;      // Location And Access List
        EUDisconnectInformationList disconnectInformationList;      // Disconnect
Information List
};

struct XST_DECLSPEC EU {
```

```
        EUHeader            header;  // Header Section
        EUDetail            detail;  // Detail Section
};

struct XST_DECLSPEC Feature {
        xstaString          fa;      // Feature Activity
        xstaString          feature; // Feature
        xstaString          featureDetail;   // Feature Detail
};
xstaLIST_DECLARATION (Feature, FeatureList);

struct XST_DECLSPEC BA {
        xstaString          ba;      // Blocking Activity
        xstaString          block;   // Block
};
xstaLIST_DECLARATION (BA, BAList);

struct XST_DECLSPEC InsideWire {
        xstaString          iwjk;    // Inside Wire Jack Code
        xstaString          iwjq;    // Inside Wire Jack Quantity
};
xstaLIST_DECLARATION (InsideWire, InsideWireList);

struct XST_DECLSPEC RSServiceDetail {
        BAList              baList;  // Blocking Activity List
        xstaString          cfa;     // Connecting Facility Assignment
        xstaString          ckr;     // Customer Circuit Reference
        xstaString          cnam;    // Calling Name
        xstaString          ecckt;   // Exchange Company Circuit ID
        xstaString          fpi;     // Freeze PIC Indicator
        xstaString          ispid;   // ISDN Service Profile Identification
        InsideWireList      insideWireList; // Inside Wire List
        xstaString          jkCode;  // Jack Code
        xstaString          jkNum;   // Jack Number
        xstaString          jkPos;   // Jack Position
        xstaString          jr;      // Jack Request
        xstaString          lean;    // Line Existing Account Number
        xstaString          leatn;   // Line Existing Account Telephone Number
        xstaString          lna;     // Line Activity
        xstaString          lneclssvc;       // Line Level Class of Service
        xstaString          lnex;    // Line Number Extension
        xstaString          lnum;    // Line Number
        xstaString          locnum;  // Location Number
        xstaString          lpic;    // IntraLATA Presubscription Indicator Code
        xstaString          matn;    // Main/Alternate Telephone Number
        xstaString          nidr;    // Network Interface Device Request
        xstaString          npi;     // Number Portability Indicator
        xstaString          otn;     // Out Telephone Number
        xstaString          pic;     // InterLATA Presubscription Indicator Code
        xstaString          ptktyp;  // PBX Trunk Type
        xstaString          ptkcon;  // PBX Trunk Configuration
        xstaString          pulse;   // Type of Pulsing
        xstaString          san;     // Subscriber Authorization Number
        xstaString          sdi;     // Switched Data Identifier
        xstaString          sgnl;    // Signaling
        xstaString          ssig;    // Start Signaling
        xstaString          tcFr;    // TC From Telephone Number
        xstaString          tcOpt;   // Transfer of Call Options
        xstaString          tcPer;   // Transfer of Calls Period
        TelephoneNumber     tcToPrimary;     // Transfer of Calls To Primary Number
        TelephoneNumber     tcToSecondary;   // Transfer of Calls To Secondary Number
        TC                  primary; // Transfer of Calls Primary
        TC                  secondary;       // Transfer of Calls Secondary
        xstaString          ters;    // Terminal Number
        xstaString          tli;     // Telephone Line Identifier
        xstaString          tns;     // Telephone Numbers
        xstaString          tsp;     // Telecommunications Service Priority
```

```
        FeatureList          featureList;     // Feature Section
};
xstaLIST_DECLARATION (RSServiceDetail, RSServiceDetailList);

struct XST_DECLSPEC Resale {
        xstaString           ord;     // Order Number
        xstaString           rsqty;   // Resale Quantity
        RSServiceDetailList  serviceDetailList;      // Service Detail List
};

struct XST_DECLSPEC DSDLListingControl {
        xstaString           dlnum;   // Directory Listing Number
        xstaString           lact;    // Listing Activity Indicator
        xstaString           ali;     // Alpha/Numeric Listing Identifier Code
        xstaString           rty;     // Record Type
        xstaString           lty;     // Listing Type
        xstaString           tt;      // TDD Indicator
        xstaString           styc;    // Style Code
        xstaString           toa;     // Type of Account
};

struct XST_DECLSPEC DSDLListingText {
        xstaString           ltext;   // Line of Text
        xstaString           ltxty;   // Listing Text Type
        xstaString           ltxnum;  // Line of Text Reference Number
};
xstaLIST_DECLARATION (DSDLListingText, DSDLListingTextList);

struct XST_DECLSPEC DSDLListingInstruction {
        xstaString           wpp;     // White Page Products
        xstaString           doi;     // Degree of Indent
        xstaString           ltn;     // Listing Telephone Number
        xstaString           nstn;    // Non Standard Telephone Number
        xstaString           lnln;    // Listed Name Last
        xstaString           lnfn;    // Listed Name First
        xstaString           lnpl;    // Letter Name Placement
        xstaString           pla;     // Place Listing As
        xstaString           des;     // Designation
        xstaString           tl;      // Title of Lineage
        xstaString           title1;  // Title of Address 1
        xstaString           title2;  // Title of Address 2
        xstaString           nick;    // Nickname
        xstaString           lapr;    // Listed Address Prefix
        xstaString           lano;    // Listed Address House Number
        xstaString           lasf;    // Listed Address House Number Suffix
        xstaString           lasd;    // Listed Address Street Directional
        xstaString           lasn;    // Listed Address Street Name
        xstaString           lath;    // Listed Address Thoroughfare
        xstaString           lass;    // Listed Address Street Suffix
        xstaString           laloc;   // Listed Address Locality
        xstaString           last;    // Listed Address State/Province
        DSDLListingTextList  textList;// Text Loop
        xstaString           yph;     // Yellow Page Heading Code
        xstaString           sic;     // Standard Industry Classification
};

struct XST_DECLSPEC DSDLListingIndicators {
        xstaString           adi;     // Address Indicator
        xstaString           dirname; // Directory Name
        xstaString           dirsub;  // Directory Subsection
        xstaString           adv;     // Advance to Directory Indicator
        xstaString           dml;     // Direct Mail List
        xstaString           dlnm;    // Dual Name Listing
        xstaString           bro;     // Business/Residence Placement Override
};

struct XST_DECLSPEC DSCRALIRSequencing {
```

```
        xstaString          so;      // Sequence Override
        xstaString          seqtext; // Sequence Text
        xstaString          seqaddr; // Sequence Address
        xstaString          seqtn;   // Sequence  Telephone Number
        xstaString          lvl;     // Level of Indent (Level = 0)
        xstaString          hs;      // Header Status
        xstaString          htn;     // Header Telephone Number
};

struct XST_DECLSPEC DSCRLevelDetail {
        xstaString          lvl;     // Level of Indent (Level = 1-6)
        xstaString          ins1;    // Indent Level 1 Status
        xstaString          so1;     // Sequence Override Level 1
        xstaString          seqtext1;// Sequence Text Level 1
        xstaString          seqaddr1;// Sequence Address Level 1
        xstaString          seqtn1;  // Sequence Telephone Number  1
        xstaString          intn;    // Indent Level Telephone Number
        xstaString          innstn;  // Indent Level Non Standard Telephone Number
        xstaString          intext;  // Indent Level Text
        xstaString          inaddr;  // Indent Level Address
};
xstaLIST_DECLARATION (DSCRLevelDetail, DSCRLevelDetailList);

struct XST_DECLSPEC DSCR {
        DSCRALIRSequencing   alirSequencing; // ALIR Sequencing Section
        DSCRLevelDetailList  levelDetailList; // Level Detail List
};

struct XST_DECLSPEC DSDLListing {
        DSDLListingControl   listingControl; // Listing Control Section
        DSDLListingInstruction listingInstruction;   // Listing Instruction Section
        DSDLListingIndicators listingIndicators;     // Listing Indicators Section
        DSCR                 dscr;    // Caption Request
};
xstaLIST_DECLARATION (DSDLListing, DSDLListingList);

struct XST_DECLSPEC DSDLDirectoryType {
        xstaString          dirtyp;  // Directory ID Type
        xstaString          dirqtya; // Number of Directories for Annual Delivery
        xstaString          dirqtync;// Number of Directories Delivered on New Connect
};
xstaLIST_DECLARATION (DSDLDirectoryType, DSDLDirectoryTypeList);

struct XST_DECLSPEC DSDLDeliveryAddress {
        xstaString          dact;    // Delivery Activity
        xstaString          nameDel; // Delivery Name
        xstaString          ddapr;   // Delivery Address House Prefix
        xstaString          ddano;   // Delivery Address House Number
        xstaString          ddasf;   // Delivery Address House Number Suffix
        xstaString          ddasd;   // Delivery Address Street Directional
        xstaString          ddasn;   // Delivery Address Street Name
        xstaString          ddath;   // Delivery Address Thoroughfare
        xstaString          ddass;   // Delivery Address Street Suffix
        xstaString          ddalo;   // Delivery Address Location
        xstaString          ddadlo;  // Delivery Descriptive Location
        xstaString          ddaloc;  // Delivery Address Locality
        xstaString          ddast;   // Delivery Address State/Province
        xstaString          ddazc;   // Delivery Address Zip Code
        DSDLDirectoryTypeList directoryTypeList;     // Directory ID Type
};

struct XST_DECLSPEC DSDL {
        DSDLListingList      listingList;     // Listing List
        DSDLDeliveryAddress  deliveryAddress; // Delivery Address Section
};

struct XST_DECLSPEC LOOPServiceDetail {
```

```
        xstaString          cableId; // Cable Identification
        xstaString          cfa;     // Connecting Facility Assignment
        xstaString          chanPair;// Channel/Pair
        xstaString          chanPair2;       // 2nd Channel/Pair
        xstaString          ckr;     // Customer Circuit Reference
        xstaString          discNbr; // Disconnect Telephone Number
        xstaString          ecckt;   // Exchange Company Circuit ID
        InsideWireList      insideWireList; // Inside Wire List
        xstaString          jkCode;  // Jack Code
        xstaString          jkNum;   // Jack Number
        xstaString          jkPos;   // Jack Position
        xstaString          jr;      // Jack Request
        xstaString          lean;    // Line Existing Account Number
        xstaString          leatn;   // Line Existing Account Telephone Number
        xstaString          lna;     // Line Activity
        xstaString          lnum;    // Line Number
        xstaString          locnum;  // Location Number
        xstaString          nidr;    // Network Interface Device Request
        xstaString          relayRack;       // Relay Rack
        xstaString          san;     // Subscriber Authorization Number
        xstaString          shelf;   // Shelf
        xstaString          slot;    // Slot
        xstaString          sltn;    // Line Sharing Telephone Number
        xstaString          systemId;// System Identification
        xstaString          tcFr;    // TC From Telephone Number
        xstaString          tcOpt;   // Transfer of Call Options
        xstaString          tcPer;   // Transfer of Calls Period
        TelephoneNumber     tcToPrimary;     // Transfer of Calls To Primary Number
        TelephoneNumber     tcToSecondary;   // Transfer of Calls To Secondary Number
        TC                  primary; // Transfer of Calls Primary
        TC                  secondary;       // Transfer of Calls Secondary
        xstaString          ters;    // Terminal Number
        xstaString          tsp;     // Telecommunications Service Priority
};
xstaLIST_DECLARATION (LOOPServiceDetail, LOOPServiceDetailList);

struct XST_DECLSPEC LOOP {
        xstaString          lqty;    // Loop Quantity
        LOOPServiceDetailList serviceDetailList;     // Service Detail List
};

struct XST_DECLSPEC LSNPServiceDetail {
        BAList              baList;  // Blocking Activity List
        xstaString          cableId; // Cable Identification
        xstaString          cfa;     // Connecting Facility Assignment
        xstaString          cftn;    // Call Forwarding to Number
        xstaString          chanPair;// Channel/Pair
        xstaString          chanPair2;       // 2nd Channel/Pair
        xstaString          ckr;     // Customer Circuit Reference
        xstaString          ecckt;   // Exchange Company Circuit ID
        xstaString          fpi;     // Freeze PIC Indicator
        InsideWireList      insideWireList; // Inside Wire List
        xstaString          jkCode;  // Jack Code
        xstaString          jkNum;   // Jack Number
        xstaString          jkPos;   // Jack Position
        xstaString          jr;      // Jack Request
        xstaString          lean;    // Line Existing Account Number
        xstaString          leatn;   // Line Existing Account Telephone Number
        xstaString          lna;     // Line Activity
        xstaString          lnum;    // Line Number
        xstaString          locnum;  // Location Number
        xstaString          lpic;    // IntraLATA Presubscription Indicator Code
        xstaString          nidr;    // Network Interface Device Request
        xstaString          npi;     // Number Portability Indicator
        xstaString          npt;     // Number Portability Type
        xstaString          nptg;    // Number Portability Trunk Group
        xstaString          portedNbr;       // Ported Telephone Number(s)
```

```
        xstaString          relayRack;      // Relay Rack
        xstaString          rti;    // Route Index
        xstaString          san;    // Subscriber Authorization Number
        xstaString          shelf;  // Shelf
        xstaString          slot;   // Slot
        xstaString          systemId;// System Identification
        xstaString          tcFr;   // TC From Telephone Number
        xstaString          tcOpt;  // Transfer of Call Options
        xstaString          tcPer;  // Transfer of Calls Period
        TelephoneNumber     tcToPrimary;    // Transfer of Calls To Primary Number
        TelephoneNumber     tcToSecondary;  // Transfer of Calls To Secondary Number
        TC                  primary; // Transfer of Calls Primary
        TC                  secondary;      // Transfer of Calls Secondary
        xstaString          tnp;    // Total Number of Paths
        xstaString          tsp;    // Telecommunications Service Priority
};
xstaLIST_DECLARATION (LSNPServiceDetail, LSNPServiceDetailList);

struct XST_DECLSPEC LSNP {
        xstaString          lqty;   // Loop Quantity
        xstaString          npqty;  // Number Portability Quantity
        LSNPServiceDetailList serviceDetailList;    // Service Detail List
};

struct XST_DECLSPEC NPServiceDetail {
        BAList              baList; // Blocking Activity List
        xstaString          cftn;   // Call Forwarding to Number
        xstaString          ckr;    // Customer Circuit Reference
        xstaString          ecckt;  // Exchange Company Circuit ID
        xstaString          fpi;    // Freeze PIC Indicator
        xstaString          lean;   // Line Existing Account Number
        xstaString          leatn;  // Line Existing Account Telephone Number
        xstaString          lna;    // Line Activity
        xstaString          lnum;   // Line Number
        xstaString          locnum; // Location Number
        xstaString          lpic;   // IntraLATA Presubscription Indicator Code
        xstaString          npi;    // Number Portability Indicator
        xstaString          npt;    // Number Portability Type
        xstaString          nptg;   // Number Portability Trunk Group
        xstaString          portedNbr;      // Ported Telephone Number(s)
        xstaString          rti;    // Route Index
        xstaString          tcFr;   // TC From Telephone Number
        xstaString          tcOpt;  // Transfer of Call Options
        xstaString          tcPer;  // Transfer of Calls Period
        TelephoneNumber     tcToPrimary;    // Transfer of Calls To Primary Number
        TelephoneNumber     tcToSecondary;  // Transfer of Calls To Secondary Number
        TC                  primary; // Transfer of Calls Primary
        TC                  secondary;      // Transfer of Calls Secondary
        xstaString          tnp;    // Total Number of Paths
};
xstaLIST_DECLARATION (NPServiceDetail, NPServiceDetailList);

struct XST_DECLSPEC NP {
        xstaString          npqty;  // Number Portability Quantity
        NPServiceDetailList serviceDetailList;      // Service Detail List
};

struct XST_DECLSPEC PORTServiceDetail {
        BAList              baList; // Blocking Activity List
        xstaString          cableId; // Cable Identification
        xstaString          cfa;    // Connecting Facility Assignment
        xstaString          chanPair;// Channel/Pair
        xstaString          ckr;    // Customer Circuit Reference
        xstaString          ecckt;  // Exchange Company Circuit ID
        xstaString          fpi;    // Freeze PIC Indicator
        xstaString          lean;   // Line Existing Account Number
        xstaString          leatn;  // Line Existing Account Telephone Number
```

```
        xstaString          lna;     // Line Activity
        xstaString          lneclssvc;       // Line Level Class of Service
        xstaString          lnex;    // Line Number Extension
        xstaString          lnum;    // Line Number
        xstaString          locnum;  // Location Number
        xstaString          lpic;    // IntraLATA Presubscription Indicator Code
        xstaString          matn;    // Main/Alternate Telephone Number
        xstaString          npi;     // Number Portability Indicator
        xstaString          otn;     // Out Telephone Number
        xstaString          pic;     // InterLATA Presubscription Indicator Code
        xstaString          pulse;   // Type of Pulsing
        xstaString          relayRack;       // Relay Rack
        xstaString          san;     // Subscriber Authorization Number
        xstaString          sdi;     // Switched Data Identifier
        xstaString          sgnl;    // Signaling
        xstaString          shelf;   // Shelf
        xstaString          slot;    // Slot
        xstaString          ssig;    // Start Signaling
        xstaString          systemId;// System Identification
        xstaString          tcFr;    // TC From Telephone Number
        xstaString          tcOpt;   // Transfer of Call Options
        xstaString          tcPer;   // Transfer of Calls Period
        TelephoneNumber     tcToPrimary;     // Transfer of Calls To Primary Number
        TelephoneNumber     tcToSecondary;   // Transfer of Calls To Secondary Number
        TC                  primary; // Transfer of Calls Primary
        TC                  secondary;       // Transfer of Calls Secondary
        xstaString          ters;    // Terminal Number
        xstaString          tli;     // Telephone Line Identifier
        xstaString          tns;     // Telephone Numbers
        xstaString          tsp;     // Telecommunications Service Priority
        FeatureList         featureList;     // Feature Section
};
xstaLIST_DECLARATION (PORTServiceDetail, PORTServiceDetailList);

struct XST_DECLSPEC PORT {
        xstaString          ord;     // Order Number
        xstaString          pqty;    // Port Quantity
        PORTServiceDetailList serviceDetailList;     // Service Detail List
};

struct XST_DECLSPEC DueDateResponse {
        xstaString          dueDate; // Due Date
        xstaString          msgId;   // Due Date Message ID
        xstaString          msgTxt;  // Due Date Message Text
        xstaString          pvIndicator;     // Premise Visit Indicator
};

struct XST_DECLSPEC DIDServiceDetail {
        xstaString          locnum;  // Location Number
        xstaString          didnum;  // Line Number DID Number
        xstaString          ckr;     // Customer Circuit Reference
        xstaString          dtnract; // DID Telephone Number Activity
        xstaString          dtnrq;   // DID Telephone Number
        xstaString          dtnr;    // DID Telephone Number Range
        xstaString          dtkact;  // Line Activity-DID
        xstaString          dtk;     // DID Trunk Quantity
        xstaString          dtkid;   // DID Trunk ID
        xstaString          dtgn;    // Trunk Group Number
        xstaString          drti;    // Route Index Number-DID
        xstaString          dtli;    // Telephone Line Identifier DID
        xstaString          dgout;   // DID Digits Out
        xstaString          dpulse;  // Type Of Pulsing-DID
        xstaString          dsgnl;   // Start Signaling-DID
        xstaString          ba;      // Blocking Activity
        xstaString          blocking;// Blocking Activity
        xstaString          leatn;   // Line Existing Account Telephone Number
        xstaString          lean;    // Line Existing Account Number
```

```
};
xstaLIST_DECLARATION (DIDServiceDetail, DIDServiceDetailList);

struct XST_DECLSPEC DID {
       DIDServiceDetailList serviceDetailList;      // Service Detail List
};

xstaCommon::Status order (
       /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
       /* in  */ const xstaOrder::LSR &lsr,
       /* in  */ const xstaOrder::EU  &eu,
       /* in  */ const xstaOrder::DSDL &dsdl,
       /* in  */ const xstaOrder::LOOP &loop,
       /* in  */ const xstaOrder::LSNP &lsnp,
       /* in  */ const xstaOrder::NP  &np,
       /* in  */ const xstaOrder::PORT &port,
       /* in  */ const xstaOrder::Resale &resale,
       /* in  */ const xstaOrder::DID &did,
       /* out */ xstaCommon::MsgHeader &messageHeader,
       /* out */ xstaOrder::DueDateResponse &dueDate
       ) throw (
              xstaCommon::SecurityException,
              xstaCommon::BackendResourceLimitation,
              xstaCommon::InvalidData,
              xstaCommon::GatewayTimeout
       );

xstaCommon::Status getDueDate (
       /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
       /* in  */ const xstaOrder::LSR &lsr,
       /* in  */ const xstaOrder::EU  &eu,
       /* in  */ const xstaOrder::DSDL &dsdl,
       /* in  */ const xstaOrder::LOOP &loop,
       /* in  */ const xstaOrder::LSNP &lsnp,
       /* in  */ const xstaOrder::NP  &np,
       /* in  */ const xstaOrder::PORT &port,
       /* in  */ const xstaOrder::Resale &resale,
       /* in  */ const xstaOrder::DID &did,
       /* out */ xstaCommon::MsgHeader &messageHeader,
       /* out */ xstaOrder::DueDateResponse &dueDate
       ) throw (
              xstaCommon::SecurityException,
              xstaCommon::BackendResourceLimitation,
              xstaCommon::InvalidData,
              xstaCommon::GatewayTimeout
       );

xstaCommon::Status orderLoop (
       /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
       /* in  */ const xstaOrder::LSR &lsr,
       /* in  */ const xstaOrder::EU  &eu,
       /* in  */ const xstaOrder::DSDL &dsdl,
       /* in  */ const xstaOrder::LOOP &loop,
       /* out */ xstaCommon::MsgHeader &messageHeader,
       /* out */ xstaOrder::DueDateResponse &dueDate
       ) throw (
              xstaCommon::SecurityException,
              xstaCommon::BackendResourceLimitation,
              xstaCommon::InvalidData,
              xstaCommon::GatewayTimeout
       );

xstaCommon::Status orderLoopWithNp (
       /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
       /* in  */ const xstaOrder::LSR &lsr,
       /* in  */ const xstaOrder::EU  &eu,
       /* in  */ const xstaOrder::DSDL &dsdl,
```

```
        /* in  */ const xstaOrder::LSNP &lsnp,
        /* out */ xstaCommon::MsgHeader &messageHeader,
        /* out */ xstaOrder::DueDateResponse &dueDate
        ) throw (
                xstaCommon::SecurityException,
                xstaCommon::BackendResourceLimitation,
                xstaCommon::InvalidData,
                xstaCommon::GatewayTimeout
        );

xstaCommon::Status orderNp (
        /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
        /* in  */ const xstaOrder::LSR &lsr,
        /* in  */ const xstaOrder::EU  &eu,
        /* in  */ const xstaOrder::DSDL &dsdl,
        /* in  */ const xstaOrder::NP  &np,
        /* out */ xstaCommon::MsgHeader &messageHeader,
        /* out */ xstaOrder::DueDateResponse &dueDate
        ) throw (
                xstaCommon::SecurityException,
                xstaCommon::BackendResourceLimitation,
                xstaCommon::InvalidData,
                xstaCommon::GatewayTimeout
        );

xstaCommon::Status orderResale (
        /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
        /* in  */ const xstaOrder::LSR &lsr,
        /* in  */ const xstaOrder::EU  &eu,
        /* in  */ const xstaOrder::DSDL &dsdl,
        /* in  */ const xstaOrder::Resale &resale,
        /* out */ xstaCommon::MsgHeader &messageHeader,
        /* out */ xstaOrder::DueDateResponse &dueDate
        ) throw (
                xstaCommon::SecurityException,
                xstaCommon::BackendResourceLimitation,
                xstaCommon::InvalidData,
                xstaCommon::GatewayTimeout
        );

xstaCommon::Status orderPort (
        /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
        /* in  */ const xstaOrder::LSR &lsr,
        /* in  */ const xstaOrder::EU  &eu,
        /* in  */ const xstaOrder::DSDL &dsdl,
        /* in  */ const xstaOrder::PORT &port,
        /* out */ xstaCommon::MsgHeader &messageHeader,
        /* out */ xstaOrder::DueDateResponse &dueDate
        ) throw (
                xstaCommon::SecurityException,
                xstaCommon::BackendResourceLimitation,
                xstaCommon::InvalidData,
                xstaCommon::GatewayTimeout
        );

xstaCommon::Status orderDirectoryListingAndAssistance (
        /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
        /* in  */ const xstaOrder::LSR &lsr,
        /* in  */ const xstaOrder::EU  &eu,
        /* in  */ const xstaOrder::DSDL &dsdl,
        /* out */ xstaCommon::MsgHeader &messageHeader,
        /* out */ xstaOrder::DueDateResponse &dueDate
        ) throw (
                xstaCommon::SecurityException,
                xstaCommon::BackendResourceLimitation,
                xstaCommon::InvalidData,
                xstaCommon::GatewayTimeout
```

```
        );

    xstaCommon::Status orderLoopPort (
        /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
        /* in  */ const xstaOrder::LSR &lsr,
        /* in  */ const xstaOrder::EU  &eu,
        /* in  */ const xstaOrder::DSDL &dsdl,
        /* in  */ const xstaOrder::PORT &port,
        /* in  */ const xstaOrder::Resale &resale,
        /* out */ xstaCommon::MsgHeader &messageHeader,
        /* out */ xstaOrder::DueDateResponse &dueDate
        ) throw (
            xstaCommon::SecurityException,
            xstaCommon::BackendResourceLimitation,
            xstaCommon::InvalidData,
            xstaCommon::GatewayTimeout
        );

    xstaCommon::Status getFormattedLsr (
        /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
        /* in  */ const char           *cc,   //  Company Code
        /* in  */ const char           *pon,
        /* out */ xstaCommon::MsgHeader &messageHeader,
        /* out */ xstaString           &flatLsr
        ) throw (
            xstaCommon::SecurityException,
            xstaCommon::BackendResourceLimitation,
            xstaCommon::InvalidData,
            xstaCommon::GatewayTimeout
        );

    enum ProcessingStatus {
        Reject,
        Confirmed,
        Completed,
        Clarification
    };

    struct XST_DECLSPEC PONSpec {
        xstaString          cc;      // Company Code
        ProcessingStatus    processingStatus;       // LEO Status Code
        Date                fromDate;// From Date
        Date                toDate;  // To Date must not be more than 7 calendar days
from fromDate

        PONSpec ();
    };

    struct XST_DECLSPEC PONResponse {
        xstaString          pon;     // CLEC's Purchase Order Number
        xstaString          ver;     // LSR Version Number
        Date                isaDate; // Date CLEC sent LSR to LEO
        Time                isaTime; // Time CLEC sent LSR to LEO
    };
    xstaLIST_DECLARATION (PONResponse, PONResponseList);

    xstaCommon::Status getPurchaseOrderList (
        /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
        /* in  */ const xstaOrder::PONSpec &ponSpec,
        /* out */ xstaCommon::MsgHeader &messageHeader,
        /* out */ xstaOrder::PONResponseList &ponResponseList
        ) throw (
            xstaCommon::SecurityException,
            xstaCommon::BackendResourceLimitation,
            xstaCommon::InvalidData,
            xstaCommon::GatewayTimeout
        );
```

```
        enum ServiceOrderStatus {
                AssignableOrder,
                PendingOrder,
                PendingFacilities,
                CompletedOrder,
                CanceledOrder,
                MissedAppointment,
                PendingCompletion,
                NoOrderStatusFound
        };

        struct XST_DECLSPEC ServiceOrderResponse {
                xstaString          cc;      // Company Code
                xstaString          pon;     // CLEC's Purchase Order Number
                xstaString          ver;     // LSR Version Number
                ServiceOrderStatus  status;  // SOCS order status.
        };

        xstaCommon::Status getServiceOrderStatus (
                /* in  */ const xstaCommon::TestProdIndicator testProdIndicator,
                /* in  */ const char           *cc,   //  Company Code
                /* in  */ const char           *pon,
                /* out */ xstaCommon::MsgHeader &messageHeader,
                /* out */ xstaOrder::ServiceOrderResponse &serviceOrderResponse
                ) throw (
                        xstaCommon::SecurityException,
                        xstaCommon::BackendResourceLimitation,
                        xstaCommon::InvalidData,
                        xstaCommon::GatewayTimeout
                );

    private:
        void getTransactionObject ()
                throw (
                        xstaCommon::BackendResourceLimitation,
                        xstaCommon::GatewayTimeout,
                        xstaCommon::InvalidData
                );
        //. Get the server object pointer to handle order
        //. requests.

        void validate (xstaPvInterface &pv)
                throw (
                        xstaCommon::InvalidData
                );
        //. Perform validations common to all ordering query types.
        //. Calls base class's validate() for validations common to both
        //. pre-ordering and ordering.
        //. Throws xstaCommon::InvalidData for any validation problems
        //. (missing data, invalid data, inconsistent data).
        //. InvalidData contains a list of InvalidDataElements so one
        //. InvalidDataElement is added to the list for each problem found.

        xstaString validateOrder (
                const xstaOrder::LSR &lsr,
                const xstaOrder::EU  &eu,
                const xstaOrder::DSDL  &dsdl,
                const xstaOrder::LOOP  &loop,
                const xstaOrder::LSNP  &lsnp,
                const xstaOrder::NP  &np,
                const xstaOrder::PORT  &port,
                const xstaOrder::Resale  &resale,
                const xstaOrder::DID  &did
                ) throw (
                        xstaCommon::InvalidData
                );
```

```
        //. Does data validations for order.
        //. Calls base class validate() for common validations.
        //. Returns a string used internally for routing.

        void validateCDD (
                const xstaOrder::LSR &lsr,
                const xstaOrder::EU  &eu,
                const xstaOrder::DSDL  &dsdl,
                const xstaOrder::LOOP  &loop,
                const xstaOrder::LSNP  &lsnp,
                const xstaOrder::NP  &np,
                const xstaOrder::PORT  &port,
                const xstaOrder::Resale  &resale,
                const xstaOrder::DID  &did
                ) throw (
                        xstaCommon::InvalidData
                );
        //. Does data validations for PreOrder due date.
        //. Calls base class validate() for common validations.

        void validateFormattedLsr (
                const char *cc,
                const char *pon
                ) throw (
                        xstaCommon::InvalidData
                );
        //. Does data validations for getFormattedLsr.
        //. Calls base class validate() for common validations.

        void validatePON (
                const xstaOrder::PONSpec &ponSpec
                ) throw (
                        xstaCommon::InvalidData
                );
        //. Does data validations for getPurchaseOrderList.
        //. Calls base class validate() for common validations.

        void validateSOS (
                const char *cc,
                const char *pon
                ) throw (
                        xstaCommon::InvalidData
                );
        //. Does data validations for getServiceOrderStatus.
        //. Calls base class validate() for common validations.

        void makeHeader (void *real_standardHeader,
                void *real_orderHeader,
                const char *cc,
                const char *pon,
                const char *version,
                const xstaCommon::TestProdIndicator testProdIndicator,
                xstaCommon::MsgHeader &messageHeader
                );
        //. Populates xstCommon::OrderStandardHeader and OrderHeader
        //. objects for ordering class and xstaCommon::MsgHeader for caller.
        //. Assumes that orderHeader.cc, .pon, .ver, and .testProdIndicator
        //. are already set and that standardHeader.subTransaction is
        //. already set.

  private:
        xstOrder9 *server_ptr_;
        xscaOrder *cog_ptr_;

};

#endif
```

# 6.13 Order Notification

```
/*
 * This file was automatically generated by xstOrderNotification.pl.
 * Do not edit this file directly!
 * @(#)xstOrderNotification.pl       75.1
 */

/*-------------------------------------------------------------------------
|                                                                         |
|(C) BellSouth Telecommunications, Inc. 1999                              |
|                                                                         |
|                      PRIVATE/PROPRIETARY/LOCK                           |
|          CONTAINS PRIVATE AND/OR PROPRIETARY INFORMATION.               |
|     MAY NOT BE USED OR DISCLOSED OUTSIDE THE BELLSOUTH COMPANIES        |
|               EXCEPT PURSUANT TO A WRITTEN AGREEMENT.                   |
|             MUST BE STORED IN LOCKED FILES WHEN NOT IN USE.             |
|                                                                         |
-------------------------------------------------------------------------*/

#ifndef xstaOrderNotification_h
#define xstaOrderNotification_h

#include "xstaServerConnection.h"

class xstOrderNotification6;

class XST_DECLSPEC xstaOrderNotification : public xstaServerConnection
{
  public:
      xstaOrderNotification (
            const char *applid,
            const char *applpass,
            const char *userid
            )
            throw (
                  xstaCommon::BackendResourceLimitation,
                  xstaCommon::GatewayTimeout,
                  xstaCommon::InvalidData,
                  xstaCommon::SecurityException
            );
      //. Connects to security server and acquires credentials.
      //. An exception thrown from the constructor means calls to
      //. readNotification will fail!

      ~xstaOrderNotification ()
            throw (
                  xstaCommon::SecurityException,
                  xstaCommon::BackendResourceLimitation,
                  xstaCommon::GatewayTimeout
            );
      //. The base class destructor calls surrenderCredentials().

      void poll_interval (unsigned long seconds);
      //. Specify the polling retry interval in seconds when calling
      //. readNotification() with blocking on.
      //. Default interval is 60 seconds.

      typedef xstaString DateTime;    // CCYYMMDD-HH:MM:SS
      typedef xstaString Date;        // CCYYMMDD
      typedef unsigned long Count;
      typedef xstaString TelephoneNumber;

      enum TransSetPurposeCd {
            OrderConfirmation,
            OrderFatalError,
```

```
            OrderReject,
            OrderClarification,
            OrderJeopardy,
            OrderCompletion,
            OrderStatus,
            OrderUnknownPurpose
      };

      struct XST_DECLSPEC StandardHeader {
            xstaString          msgId;
            xstaString          msgTxt;
            xstaString          leoMessage;
            xstaString          applId;
            xstaString          userId;
            xstaString          trxId;
            xstaString          tagCogRouting;
      };

      struct XST_DECLSPEC FocCnHdr {
            xstaString          dataSource;
            xstaCommon::TestProdIndicator testProdIndicator;
            xstaString          pon; // Purchase Order Number
            TransSetPurposeCd   tranSetPurposeCd;
            xstaString          tranAckType;
            Date                cdtsent;    // Date and Time Confirmation Sent
            xstaString          version;    // Version Identification
            xstaString          ecVer;      // Exchange Carrier Version
            xstaString          cno; // LSOG Inquiry Case Number
            xstaString          lsrnoDsrno; // Local/Directory Service Request Number
            xstaString          ean; // Existing Account Number
            TelephoneNumber     eatn;       // Existing Account Telephone Number
            xstaString          svcOrd;     // Order Number (ORD)
            xstaString          bi1; // Billing Account Nbr ID 1
            xstaString          ban1;       // Billing Account Nbr 1
            xstaString          bi2; // Billing Account Nbr ID 2
            xstaString          ban2;       // Billing Account Nbr 2
            xstaString          dbi1;       // Directory Billing Account Nbr ID 1
            xstaString          dban1;      // Directory Billing Account Nbr 1
            xstaString          dbi2;       // Directory Billing Account Nbr ID 2
            xstaString          dban2;      // Directory Billing Account Nbr 2
            xstaString          rt; // Response Type for Directory
            xstaString          dlord;      // Directory Listing Order Number
            xstaString          daord;      // Directory Assistance Order Number
            xstaString          an; // Account Number
            TelephoneNumber     atn; // Account Telephone Number
            xstaString          rep; // Provider Contact Representative
            TelephoneNumber     repTelNo;   // Provider Contact Telephone Number
            xstaString          dlcontInit; // Directory Contact at Providing Company
            xstaString          dlcontTn;   // Directory Contact Telephone Number
            xstaString          dsgConNsp;  // NSP Design/Engineering Contact
            xstaString          dsgConTelNoNsp;     // NSP Design/Engineering Contact Telephone
Number
            xstaString          dacontInit; // Directory Assistance Contact at Providing Company
            xstaString          dacontTn;   // Directory Assistance Contact Telephone Number
            xstaString          dlqtyr;     // Number of Listings Received
            xstaString          sqtyr;      // Service Address Quantity Received
            xstaString          ddqtyr;     // Number of Delivery Segments Received
            Date                dor; // Date of Reciept
            Date                ddCd;       // Due Date/Completion Date
            Date                ebd; // Effective Bill Date
            Date                dda; // Date of Availability in DA
            Date                esdd;       // Estimated Due Date
            xstaString          fdtDadt;    // Frame Due Time - Directory Assistance Due Time
            xstaString          ibt; // ISDN BRI Type
            xstaString          sc1; // Service Center
            xstaString          sc2; // Service Center
            xstaString          st; // Switch Type
            xstaString          chcDchc;    // Coordinated Hot Cut - Directory
            xstaString          afvr;       // Additional Field Visit Required
            xstaString          cc; // Company Code
            xstaString          dinit;      // Directory Request Initiator
```

```
        xstaString          dsgcon;     // Design/Engineering Contact
        xstaString          nnsp;       // New Network Service Provider Identification
        xstaString          ccna;       // Customer Carrier Name Abbreviation
        xstaString          init;       // Initiator Identification
        TelephoneNumber     initTelNo;  // Initiator Telephone Number
        xstaString          resid;      // Response Identifier
        xstaString          leoRetfdSeq;        // LEO Return feed sequence number
        xstaString          gsPartnerId;
        xstaString          gsVersion;
};

struct XST_DECLSPEC RemarksHdr {
        xstaString          remarks;    // Remarks
        xstaString          remarksCnf; // Remarks - Confirmation
        xstaString          remarksDsred;       // Remarks - Directory Service Error Detail
        xstaString          remarksDscn;        // Remarks - Directory Service Completion
};

struct XST_DECLSPEC StatusHdr {
        xstaString          statusCode; // Status Code
        xstaString          statusMsg;  // Status Messege
};

struct XST_DECLSPEC Heading {
        FocCnHdr            foc;
        RemarksHdr          remarks;
        StatusHdr           status;
};

struct XST_DECLSPEC ErrorHdr {
        xstaString          errorCode;  // Error Code
        xstaString          errorMsg;   // Error Messege
};
xstaLIST_DECLARATION (ErrorHdr, ErrorList);

struct XST_DECLSPEC DetailDid {
        TelephoneNumber     dstn;       // Disassociated TN - DID
        xstaString          dtnr;       // DID Telephone Number Range
};
xstaLIST_DECLARATION (DetailDid, DetailDidList);

struct XST_DECLSPEC DirectoryText {
        xstaString          ltxnum;     // Line of Text Reference Number
        xstaString          ltxty;      // Listing Text Type
        xstaString          listtext;   // Listing Text
};
xstaLIST_DECLARATION (DirectoryText, DirectoryTextList);

struct XST_DECLSPEC DirectoryListing {
        xstaString          wpp; // White Page Products
        xstaString          listnm;     // Listed Name
        xstaString          listadr;    // Listed Address
        DirectoryTextList   directoryTextList;
};
xstaLIST_DECLARATION (DirectoryListing, DirectoryListingList);

struct XST_DECLSPEC HuntLine {
        xstaString          htseq;      // Hunting Sequence
        xstaString          ht;  // Hunt Telephone Number
};
xstaLIST_DECLARATION (HuntLine, HuntLineList);

struct XST_DECLSPEC Detail {
        xstaString          tns; // Telephone Number(s)
        xstaString          terS;       // Terminal Number(s)
        xstaString          matn;       // Main/AlternateTelephone Number
        xstaString          ltc; // Line Treatment Code
        xstaString          toa; // Type of Account
        xstaString          ckr; // Customer Circuit Reference
        xstaString          ecckt;      // Exchange Company Circuit ID
        xstaString          book;       // Book of Listing
```

```
        xstaString          dlci;       // Data Link Connection Identifier
        xstaString          sectionX;   // Section of Directory for Listing
        xstaString          doi; // Degree of Indent
        xstaString          dsn; // Dialable Station Number
        xstaString          recckt;     // Related ECCKT
        xstaString          portedNbr;  // Ported Number
        xstaString          demarc;     // Demarc Designation
        xstaString          systemId;   // System ID
        xstaString          cableId;    // Cable ID
        xstaString          shelf;      // Shelf
        xstaString          slot;       // Slot
        xstaString          relayRack;  // Relay Rack
        xstaString          chanPair;   // Channel/Pair
        xstaString          chanPair2;  // 2nd Channel/Pair
        xstaString          unit;       // Unit Number
        xstaString          lact;       // Listing Activity Type
        xstaString          lty; // Listing Type
        xstaString          listtnda;   // Listed Telephone Number DA
        TelephoneNumber     ltn; // Listed Telephone Number
        xstaString          lst; // Local Service Termination
        TelephoneNumber     discNbr;    // Disconnect Telephone Number
        xstaString          nstn;       // Non Standard Telephone Number
        TelephoneNumber     otn; // Old Telephone Number
        xstaString          rtiDrti;    // Route Index
        xstaString          ha;  // Hunt Group Activity
        xstaString          hid; // Hunt Group Identifier
        xstaString          rdlci;      // Related Data Link Connection Identifier
        xstaString          ispid;      // ISDN Service Profile Identification
        xstaString          dgout;      // DID Digits Out
        xstaString          dtgn;       // DID Trunk Group Number
        xstaString          dtkid;      // DID Trunk ID
        xstaString          tliDtli;    // Telephone Line Identifier - DID TLI
        xstaString          styc;       // Style Code
        xstaString          notyp;      // Number Type
        xstaString          sltn;       // Line Sharing Telephone Number
        xstaString          cfa; // Connecting Facility Assignment
        xstaString          pgi; // Pair Gain Indicator
        xstaString          oor; // Out of Range Indicator
        xstaString          nid; // Network Interface Device
        xstaString          lOrd;       // Loop Order Number
        xstaString          npord;      // Number Portability Order Number
        xstaString          discOrd;    // Disconnect Order Number
        xstaString          ali; // Alpha/Numeric Listing Identifier Code
        xstaString          locnumDtl;  // Location Number
        xstaString          lnum;       // Line Number
        xstaString          lnex;       // Line Number Extension
        xstaString          dnum;       // Disconnect Reference Number
        xstaString          hnum;       // Hunting Number
        xstaString          dlnum;      // Directory Listing Reference Number
        xstaString          delnum;     // Delivery Reference Number
        xstaString          ernum;      // Error Detail Reference Number
        xstaString          didnum;     // Line Number - DID number
        xstaString          lean;       // Line Existing Account Number
        xstaString          leatn;      // Line Existing Account Telephone Number
        xstaString          dtk; // DID Trunk Quantity
        DetailDidList       detailDidList;
        DirectoryListingList directoryListingList;
        HuntLineList        huntLineList;
};
xstaLIST_DECLARATION (Detail, DetailList);

struct XST_DECLSPEC Trailer {
        xstaString          transactionNumber;
        Count               trailerCount;       // Record Count
};

struct XST_DECLSPEC Notification {
        StandardHeader      orderHeader;
        // xstaCommon::OrderStandardHeader orderHeader;
        Heading             heading;
        ErrorList           errorList;
```

```
        DetailList          detailList;
        Trailer             trailer;
    };
    xstaLIST_DECLARATION (Notification, NotificationList);

    unsigned char readNotification (
        /* in  */ int                  blocking,
        /* out */ xstaOrderNotification::Notification &notification
        ) throw (
            xstaCommon::SecurityException,
            xstaCommon::BackendResourceLimitation,
            xstaCommon::InvalidData,
            xstaCommon::GatewayTimeout
        );
    // CLEC client application calls this to fetch a Notification
    // queued via sendNotification.  Returns true if one was
    // retrieved and false if not.
    // Automatically sends an acknowledgement if a notification was received.

    unsigned char readNotificationNoAck (
        /* out */ xstaOrderNotification::Notification &notification
        ) throw (
            xstaCommon::SecurityException,
            xstaCommon::BackendResourceLimitation,
            xstaCommon::InvalidData,
            xstaCommon::GatewayTimeout
        );
    // Retrieves one notification from the server.  Does not block.
    // Internally, calls
    // readNotification.  Does not call
    // acknowledgeNotification after a
    // notification has been received.

    unsigned char readNotificationList (
        /* out */ xstaOrderNotification::NotificationList &notificationList
        ) throw (
            xstaCommon::SecurityException,
            xstaCommon::BackendResourceLimitation,
            xstaCommon::InvalidData,
            xstaCommon::GatewayTimeout
        );
    // CLEC client application calls this to fetch a list of Notification
    // queued via sendNotification.  Returns true if one or more were
    // retrieved and false if not.
    // Does not automatically send an acknowledgement.

    void acknowledgeNotification (
        /* in  */ const xstaOrderNotification::Notification &notification
        ) throw (
            xstaCommon::SecurityException,
            xstaCommon::BackendResourceLimitation,
            xstaCommon::InvalidData,
            xstaCommon::GatewayTimeout
        );
    // CLEC client application calls this to acknowledge
    // receipt of a Notification back to TAG.

    void acknowledgeNotificationList (
        /* in  */ const xstaOrderNotification::NotificationList &notificationList
        ) throw (
            xstaCommon::SecurityException,
            xstaCommon::BackendResourceLimitation,
            xstaCommon::InvalidData,
            xstaCommon::GatewayTimeout
        );
    // CLEC client application calls this to acknowledge
    // receipt of a Notification back to TAG.

private:
    void getTransactionObject ()
        throw (
```

```
                xstaCommon::BackendResourceLimitation,
                xstaCommon::GatewayTimeout,
                xstaCommon::InvalidData
            );
      //. Get the server object pointer to handle order notification
      //. requests.

  private:
      xstOrderNotification6 *server_ptr_;
      unsigned long  poll_interval_;

};

#endif
```

# 7. Appendix B - Sample Code xstTestClient.C

```
/*-------------------------------------------------------------------------
|                                                                         |
|(C) BellSouth Telecommunications, Inc. 1998                              |
|                                                                         |
|                      PRIVATE/PROPRIETARY/LOCK                           |
|          CONTAINS PRIVATE AND/OR PROPRIETARY INFORMATION.               |
|     MAY NOT BE USED OR DISCLOSED OUTSIDE THE BELLSOUTH COMPANIES        |
|               EXCEPT PURSUANT TO A WRITTEN AGREEMENT.                   |
|            MUST BE STORED IN LOCKED FILES WHEN NOT IN USE.              |
|                                                                         |
-------------------------------------------------------------------------*/

/*
 * xstTestClient.C
 * created on 00/10/11 at 16:26:58
 */

static char
sccsId[]        = "@(#)xstTestClient.C 87.3",
copyright[]     = "COPYRIGHT (c) 1998, BellSouth Telecommunications, Inc.";
inline void sccsId_copyright() { if (sccsId==copyright); }


//      MR List - Changes
//-------------------------------------------------------------------------
//  MR#           Date      Who    What
//  BS-99123-02 07/29/1999 EJK    Complete rewrite to change I/O syntax for
//                                 OSS99.
//  ZP-99241-02 09/01/1999 EJK    Guard xsttDump of Directional so it does
//                                 not index out of bounds when value is bogus.
//  BS-99274-02 09/28/1999 EJK    New CDD.
//  BS-99301-02 11/19/1999 EJK    CC79 Add tnaq_misc.
//  BS-99327-02 11/22/1999 EJK    CC105 Add routingInfo.
//  BS-99327-02 12/06/1999 EJK    Make RoutingInfo same format as in xst_Client.
//  ZD-99342-02 12/08/1999 EJK    Fix DSAP translation of AvailableDays.
//  xx-xxxxx-xx 12/10/1999 EJK    Change tag defn for NT compiler.
//  ZD-00011-03 01/12/2000 EJK    On file load exception, do not exit. Instead
//                                 rethrow to surrender credentials.
//  BS-00098-03 04/10/2000 EJK    Add xDSL Loop queries.
//  BS-00173-01 06/21/2000 EJK    Add VIEWLSR.
//  ZD-00171-04 06/26/2000 EJK    Add ESDQ.

#include <stdlib.h>

#ifndef WIN32
#include <unistd.h>
#else
#include <windows.h>
#include <direct.h>
#endif

#include <iostream.h>
```

```
#include <iomanip.h>
#include <fstream.h>

#ifdef WIN32
        #include <rw/compiler.h>
        #undef RW_NO_BOOL
        #include <rw/defs.h>
#endif

#include <rw/cstring.h>
#include <rw/tvslist.h>
#include <ctype.h>
#include "xstaAddressValidation.h"
#include "xstaAppointmentScheduling.h"
#include "xstaCustomerRecord.h"
#include "xstaDueDate.h"
#include "xstaLoop.h"
#include "xstaOrder.h"
#include "xstaServiceAvailability.h"
#include "xstaTelephoneNumberAssignment.h"
#include "xstConfig.h"
#include "xsttDumpCommon.h"
#include "xsttDumpHashMarks.h"
#include "xsttDumpListMarks.h"

#include "xstaTrace.h"

#define DUMP(what)      xsttDump (#what, obj.what, indent, out)

static void Usage (const char *progname);

struct QueryObject
{
        xstaAddressValidation          &addrval;
        xstaServiceAvailability        &servavail;
        xstaAppointmentScheduling      &apptavail;
        xstaCustomerRecord             &custrecord;
        xstaTelephoneNumberAssignment &tnavail;
        xstaOrder                      &order;
        xstaLoop                       &loop;
        xstaDueDate                    &duedate;

        QueryObject (
                xstaAddressValidation          &addr,
                xstaServiceAvailability        &servav,
                xstaAppointmentScheduling      &apptav,
                xstaCustomerRecord             &custrec,
                xstaTelephoneNumberAssignment &tnav,
                xstaOrder                      &ord,
                xstaLoop                       &loop,
                xstaDueDate                    &duedate
        );
};

QueryObject::QueryObject (
                xstaAddressValidation          &addr,
                xstaServiceAvailability        &servav,
                xstaAppointmentScheduling      &apptav,
                xstaCustomerRecord             &custrec,
                xstaTelephoneNumberAssignment &tnav,
                xstaOrder                      &ord,
                xstaLoop                       &lp,
                xstaDueDate                    &dd
        )
        : addrval (addr)
        , servavail (servav)
        , apptavail (apptav)
        , custrecord (custrec)
        , tnavail (tnav)
        , order (ord)
        , loop (lp)
```

```
        , duedate (dd)
{
}

static void do_item (const RWCString &filename, xstValue &input, unsigned int delay, QueryObject
&object, ostream &out);

static void do_AVQ_ADDR (xstValue &input, xstaAddressValidation &addrval, ostream &out);
static void do_AVQ_TN (xstValue &input, xstaAddressValidation &addrval, ostream &out);
static void do_SAQ (xstValue &input, xstaServiceAvailability &servavail, ostream &out);
static void do_AAQ (xstValue &input, xstaAppointmentScheduling &apptavail, ostream &out);
static void do_CSRQ (xstValue &input, xstaCustomerRecord &custrecord, ostream &out);
static void do_TNAQ (xstValue &input, xstaTelephoneNumberAssignment &tnavail, ostream &out);
static void do_TNAQ_MLH (xstValue &input, xstaTelephoneNumberAssignment &tnavail, ostream &out);
static void do_TNAQ_DID (xstValue &input, xstaTelephoneNumberAssignment &tnavail, ostream &out);
static void do_TNAQ_MISC (xstValue &input, xstaTelephoneNumberAssignment &tnavail, ostream &out);
static void do_TNSQ (xstValue &input, xstaTelephoneNumberAssignment &tnavail, ostream &out);
static void do_TNCAN_TN (xstValue &input, xstaTelephoneNumberAssignment &tnavail, ostream &out);
static void do_TNCAN_MLH (xstValue &input, xstaTelephoneNumberAssignment &tnavail, ostream &out);
static void do_TNCAN_DID (xstValue &input, xstaTelephoneNumberAssignment &tnavail, ostream &out);
static void do_CDD (xstValue &input, xstaOrder &order, ostream &out);
static void do_ORDER (xstValue &input, xstaOrder &order, ostream &out);
static void do_LOOP (xstValue &input, xstaOrder &order, ostream &out);
static void do_LOOPNP (xstValue &input, xstaOrder &order, ostream &out);
static void do_NP (xstValue &input, xstaOrder &order, ostream &out);
static void do_RESALE (xstValue &input, xstaOrder &order, ostream &out);
static void do_PORT (xstValue &input, xstaOrder &order, ostream &out);
static void do_DIRLIST (xstValue &input, xstaOrder &order, ostream &out);
static void do_LOOPPORT (xstValue &input, xstaOrder &order, ostream &out);
static void do_VIEWLSR (xstValue &input, xstaOrder &order, ostream &out);
static void do_PONLIST (xstValue &input, xstaOrder &order, ostream &out);
static void do_SOS (xstValue &input, xstaOrder &order, ostream &out);
static void do_LOOP_MAKEUP_WORKING (xstValue &input, xstaLoop &loop, ostream &out);
static void do_LOOP_MAKEUP_SPARE (xstValue &input, xstaLoop &loop, ostream &out);
static void do_LOOP_RESERVATION_SPARE (xstValue &input, xstaLoop &loop, ostream &out);
static void do_LOOP_RESERVATION_CANCEL (xstValue &input, xstaLoop &loop, ostream &out);
static void do_ESDQ (xstValue &input, xstaDueDate &duedate, ostream &out);

int main(int argc, char** argv)
{
        Trace ("main xstTestClient");
        RWCString appId, appPasswd, userId, inFile;
        RWCString outputFile;
        ostream *outStream = &cout;

        unsigned int delay = 0;
        int repeat = 0;
        const char *Progname = argv[0];
        Tout(Tvar(Progname));
        Tout(Tvar(xstaServerConnection::apiVersion()));

#ifdef WIN32
        atexit(xstaServerConnection::finalize);
#endif

        // simulate getopt() for benefit of NT.
        --argc;
        ++argv;
        while (argc > 0 && **argv == '-') {
                switch (*++*argv) {
                case 'd':
                        if (*++*argv == '\0') { --argc; ++argv; }
                        delay = atoi (*argv);
                        Tout(Tvar(delay));
                        break;
                case 'o':
                        {
                        if (*++*argv == '\0') { --argc; ++argv; }
                        outputFile = *argv;
                        Tout(Tvar(outputFile));
                        ostream *os = new ofstream (outputFile, ios::out);
```

```
                if (os == 0 || os->fail()) {
                        cerr << "Cannot open " << outputFile << endl;
                        exit (1);
                }
                outStream = os;
                }
                break;
        case 'r':
                if (*++*argv == '\0') { --argc; ++argv; }
                repeat = atoi (*argv);
                Tout(Tvar(repeat));
                break;
        case '?':
                Usage (Progname);
                break;
        default:
                cerr << "Unknown option -" << char(**argv) << endl;
                Usage (Progname);
                break;
        }
        ++argv;
        --argc;
}

// make sure we have enough arguments
if (argc < 4) {
        cerr << "Only have " << argc << " arguments" << endl;
        Usage (Progname);
}

// get argv list
appId = argv[0];
appPasswd = argv[1];
userId = argv[2];
Tout(Tvar(appId) Tvar(appPasswd) Tvar(userId) );

argc -= 3;
argv += 3;

try {
// object definitions for each of the query types.
xstaAddressValidation         addrval (appId, appPasswd, userId);
xstaServiceAvailability       servavail (appId, appPasswd, userId);
xstaAppointmentScheduling     apptavail (appId, appPasswd, userId);
xstaCustomerRecord            custrecord (appId, appPasswd, userId);
xstaTelephoneNumberAssignment tnavail (appId, appPasswd, userId);
xstaOrder                     order (appId, appPasswd, userId);
xstaLoop                      loop (appId, appPasswd, userId);
xstaDueDate                   duedate (appId, appPasswd, userId);

QueryObject query_object (
        addrval,
        servavail,
        apptavail,
        custrecord,
        tnavail,
        order,
        loop,
        duedate
);

while (1) {     // repeat loop
        int old_argc = argc;
        while (argc > 0) {
                inFile = *argv;
                Tout(Tvar(inFile) );
                xstConfig input;
                try {
                        // must not force read from XST_CONFIG.
                        input.load (inFile, 0);
                }
```

```
                        catch (const RWCString &exc) {
                                RWCString new_exc = "Cannot load ";
                                new_exc += inFile;
                                new_exc += ": ";
                                new_exc += exc;
                                throw new_exc;
                        }
                        do_item (inFile, input.top(), delay, query_object, *outStream);
                        --argc;
                        ++argv;
                }
                if (repeat-- <= 0) {
                        break;
                }
                argc += old_argc;
                argv -= old_argc;
        }
        }
        catch (xstaCommon::SecurityException& excp) {
                cerr << "FAILURE: security error" << endl;
                cerr << "    type: " << excp.exc << endl;
                cerr << "    message: " << (const char*) excp.message << endl;
                if (!excp.inquiryNumber.isEmpty())
                        cerr << "    inquiry number: " << excp.inquiryNumber << endl;
        }
        catch (xstaCommon::BackendResourceLimitation& excp) {
                cerr << "FAILURE: backend resource limitation" << endl;
                cerr << "    code: " << (const char*) excp.code << endl
                << "    description: " << (const char*) excp.description << endl;
                if (!excp.inquiryNumber.isEmpty())
                        cerr << "    inquiry number: " << excp.inquiryNumber << endl;
        }
        catch (xstaCommon::GatewayTimeout& excp) {
                cerr << "TIMEOUT: gateway timeout" << endl;
                cerr << "    code: " << (const char*) excp.code << endl
                << "    description: " << (const char*) excp.description << endl;
                if (!excp.inquiryNumber.isEmpty())
                        cerr << "    inquiry number: " << excp.inquiryNumber << endl;
        }
        catch (RWCString &excp) {
                cerr << "FAILURE: " << excp << endl;
        }

        delete outStream;

        return 0;
}

/* Case insensitive string comparison.  Return 0 if different, 1 if identical
 * modulo case.
 */
int streq (const char *a, const char *b)
{
        while (*a && *b) {
                if (*a == *b) {}
                else if ((*a ^ *b) == 32 && isalpha (*a)) {}
                else { return 0; }
                ++a;
                ++b;
        }
        if (*a == '\0' && *b == '\0') return 1;
        return 0;
}

void do_item (const RWCString &filename, xstValue &input, unsigned int delay, QueryObject
&object, ostream &out)
{
        Trace ("do_item");
        if (input.hash("RequestType").type() == xstValue::String) {
                RWCString tag = input["RequestType"].string();
                Tout(Tvar(tag) );
```

```
tag.toLower ();
try {
if (0) {}
else if (tag == "avq") {
       do_AVQ_ADDR (input, object.addrval, out);
}
else if (tag == "avq_tn") {
       do_AVQ_TN (input, object.addrval, out);
}
else if (tag == "saq") {
       do_SAQ (input, object.servavail, out);
}
else if (tag == "aaq") {
       do_AAQ (input, object.apptavail, out);
}
else if (tag == "csrq") {
       do_CSRQ (input, object.custrecord, out);
}
else if (tag == "tnaq") {
       do_TNAQ (input, object.tnavail, out);
}
else if (tag == "tnaq_mlh") {
       do_TNAQ_MLH (input, object.tnavail, out);
}
else if (tag == "tnaq_did") {
       do_TNAQ_DID (input, object.tnavail, out);
}
else if (tag == "tnaq_misc") {
       do_TNAQ_MISC (input, object.tnavail, out);
}
else if (tag == "tnsq") {
       do_TNSQ (input, object.tnavail, out);
}
else if (tag == "tncan_tn") {
       do_TNCAN_TN (input, object.tnavail, out);
}
else if (tag == "tncan_mlh") {
       do_TNCAN_MLH (input, object.tnavail, out);
}
else if (tag == "tncan_did") {
       do_TNCAN_DID (input, object.tnavail, out);
}
else if (tag == "esdq") {
       do_ESDQ (input, object.duedate, out);
}
// Order
else if (tag == "cdd") {
       do_CDD (input, object.order, out);
}
else if (tag == "order") {
       do_ORDER (input, object.order, out);
}
else if (tag == "loop") {
       do_LOOP (input, object.order, out);
}
else if (tag == "loopnp") {
       do_LOOPNP (input, object.order, out);
}
else if (tag == "np") {
       do_NP (input, object.order, out);
}
else if (tag == "resale") {
       do_RESALE (input, object.order, out);
}
else if (tag == "port") {
       do_PORT (input, object.order, out);
}
else if (tag == "dirlist") {
       do_DIRLIST (input, object.order, out);
}
else if (tag == "loopport") {
```

```
                do_LOOPPORT (input, object.order, out);
        }
        else if (tag == "viewlsr") {
                do_VIEWLSR (input, object.order, out);
        }
        else if (tag == "ponlist") {
                do_PONLIST (input, object.order, out);
        }
        else if (tag == "sos") {
                do_SOS (input, object.order, out);
        }
        // xDSL preorder
        else if (tag == "loop_makeup_working") {
                do_LOOP_MAKEUP_WORKING (input, object.loop, out);
        }
        else if (tag == "loop_makeup_spare") {
                do_LOOP_MAKEUP_SPARE (input, object.loop, out);
        }
        else if (tag == "loop_reservation_spare") {
                do_LOOP_RESERVATION_SPARE (input, object.loop, out);
        }
        else if (tag == "loop_reservation_cancel") {
                do_LOOP_RESERVATION_CANCEL (input, object.loop, out);
        }
        else {
                RWCString message ("unknown query type ");
                message += tag;
                throw message;
        }
        out << flush;
        if (delay > 0) {
#ifndef WIN32
                sleep (delay);
#else
                Sleep (delay*1000);
#endif
        }
        }
        catch (xstaCommon::SecurityException& excp) {
                cerr << "FAILURE: security error" << endl;
                cerr << "       type: " << excp.exc << endl;
                cerr << "       message: " << (const char*) excp.message << endl;
                if (!excp.inquiryNumber.isEmpty())
                        cerr << "       inquiry number: " << excp.inquiryNumber << endl;
        }
        catch (xstaCommon::InvalidData& excp) {
                cerr << "FAILURE: invalid data for " << filename << endl;
                if (!excp.inquiryNumber.isEmpty())
                        cerr << "       inquiry number: " << excp.inquiryNumber << endl;
                for (unsigned int i=0; i<excp.invalidDataElementList.length(); i++) {
                        const xstaCommon::InvalidDataElement& elem =
excp.invalidDataElementList[i];
                        cerr << "       data element: " << (const char*)
elem.dataElementName << endl
                                << "    msgId: " << (const char*) elem.status.msgId << endl
                                << "    msgTxt: " << (const char*) elem.status.msgTxt <<
endl;
                }
        }
        catch (xstaCommon::BackendResourceLimitation& excp) {
                cerr << "FAILURE: backend resource limitation" << endl;
                cerr << "       code: " << (const char*) excp.code << endl
                << "    description: " << (const char*) excp.description << endl;
                if (!excp.inquiryNumber.isEmpty())
                        cerr << "       inquiry number: " << excp.inquiryNumber << endl;
        }
        catch (xstaCommon::GatewayTimeout& excp) {
                cerr << "TIMEOUT: gateway timeout" << endl;
                cerr << "       code: " << (const char*) excp.code << endl
                << "    description: " << (const char*) excp.description << endl;
                if (!excp.inquiryNumber.isEmpty())
```

```
                        cerr << "       inquiry number: " << excp.inquiryNumber << endl;
                }
                catch (RWCString& excp) {
                        cerr << "FAILURE: client error" << endl;
                        cerr << excp << endl;
                }
        }
        else if (input.hash("RequestList").type() == xstValue::List) {
                xstValueListIterator list_iter (input["RequestList"]);
                while (list_iter()) {
                        xstValue value = list_iter.value();
                        if (value.type() != xstValue::Hash) {
                                RWCString message ("values inside RequestList must be hashes;
instead found ");
                                switch (value.type()) {
                                case xstValue::List:   message += "List";     break;
                                case xstValue::Hash:   message += "Hash";     break;
                                case xstValue::String: message += "String";   break;
                                case xstValue::Invalid: message += "Invalid";        break;
                                default:               message += "Unknown";  break;
                                }
                                throw message;
                        }
                        do_item (filename, value, delay, object, out);
                }
        }
        else {
                cerr << "FAILURE: client error" << endl;
                cerr << "No RequestType tag found to determine the type of this query" << endl;
        }
}

void must_be (xstValue::Type found, xstValue::Type expected, const char *where)
{
        if (found == expected) return;
        RWCString message ("attempt to ");
        switch (expected) {
        case xstValue::List:   message += "List";     break;
        case xstValue::Hash:   message += "Hash";     break;
        case xstValue::String: message += "String";   break;
        case xstValue::Invalid: message += "Invalid";        break;
        default:               message += "Unknown";  break;
        }
        message += " iterate over ";
        switch (found) {
        case xstValue::List:   message += "List";     break;
        case xstValue::Hash:   message += "Hash";     break;
        case xstValue::String: message += "String";   break;
        case xstValue::Invalid: message += "Invalid";        break;
        default:               message += "Unknown";  break;
        }
        message += " while parsing ";
        message += where;
        throw message;
}

/*
 * Common assignment and dump functions used by the generated code.
 */

void
xsttAssign (xstValue &input, const char *, xstaString &target)
{
        must_be (input.type(), xstValue::String, "String");
        target = input.string();
}

// Convenience function to throw a single InvalidData.
static void
throwInvalidData (
        const char *elementName, const char *msgId, const char *msgTxt)
```

```
        throw (xstaCommon::InvalidData)
{
        Trace ("throwInvalidData");
        xstaCommon::InvalidData err;
        err.invalidDataElementList.length (1);
        err.invalidDataElementList[0].dataElementName = elementName;
        err.invalidDataElementList[0].status.msgId = msgId;
        err.invalidDataElementList[0].status.msgTxt = msgTxt;
        Tout (Tvar(elementName) Tvar(msgId) Tvar(msgTxt) );
        throw err;
}

static unsigned long
atoul (const RWCString &str, const char *tag)
        throw (xstaCommon::InvalidData)
{
        const char *nptr = str.data();
        const char *eptr = nptr + str.length();
        unsigned long value = 0;
        while (nptr < eptr) {
                if (isdigit (*nptr)) {
                        value = value*10 + *nptr-'0';
                }
                else {
                        RWCString error_message = "Invalid value (";
                        error_message += str;
                        error_message += ") provided to test client for integer field, must be all
digits";
                        throwInvalidData (tag, "TAG0001TEST", error_message);
                }
                ++nptr;
        }
        return value;
}

void
xsttAssign (xstValue &input, const char *tag, unsigned short &target)
        throw (xstaCommon::InvalidData)
{
        must_be (input.type(), xstValue::String, "String");
        target = (unsigned short) atoul (input.string (), tag);
}

void
xsttAssign (xstValue &input, const char *tag, unsigned long &target)
        throw (xstaCommon::InvalidData)
{
        must_be (input.type(), xstValue::String, "String");
        target = atoul (input.string (), tag);
}

void
xsttAssign (xstValue &input, const char *tag, long &target)
{
        must_be (input.type(), xstValue::String, "String");
        target = atoul (input.string (), tag);
}

// This is for booleans.
void
xsttAssign (xstValue &input, const char *, unsigned char &target)
{
        must_be (input.type(), xstValue::String, "boolean");
        RWCString str (input.string());

        str.toUpper();
        if (str.data()[0] == '0' || str.data()[0] == 'N') target = 0;
        else if (str.data()[0] == '1' || str.data()[0] == 'Y') target = 1;
        else target = 0;
}
```

```
// This is for booleans.
void xsttDump (const char *name,
        unsigned char obj, int indent, ostream &out)
{
        xsttDump (name, obj ? "Y" : "N", indent, out);
}

// This is for xstaAppointmentScheduling::AvailableDays
void xsttDump (const char *name,
        char obj, int indent, ostream &out)
{
        char buffer[2];
        buffer[0] = obj;
        buffer[1] = '\0';
        xsttDump (name, buffer, indent, out);
}

void xsttDump (const char *name,
        const xstaCommon::Status &obj, int indent, ostream &out)
{
        xsttDumpHashMarks marks (name, indent, out);
        indent++;
        DUMP (msgId);
        DUMP (msgTxt);
}

void xsttDump (const char *name,
        const xstaCommon::MsgHeader &obj, int indent, ostream &out)
{
        xsttDumpHashMarks marks (name, indent, out);
        indent++;
        DUMP (inquiryNumber);
        DUMP (dateSent);
}

void xsttDump (const char *name,
        const xstaCommon::Directional &obj, int indent, ostream &out)
{
        out << setw(indent)<<"";
        if (name) {
                out << name << "=";
        }
        switch (obj) {
        case xstaCommon::NoDirection: out << "\"\""; break;
        case xstaCommon::East: out << "E";    break;
        case xstaCommon::West: out << "W";    break;
        case xstaCommon::North:       out << "N";    break;
        case xstaCommon::South:       out << "S";    break;
        case xstaCommon::NorthEast:   out << "NE";   break;
        case xstaCommon::NorthWest:   out << "NW";   break;
        case xstaCommon::SouthWest:   out << "SW";   break;
        case xstaCommon::SouthEast:   out << "SE";   break;
        default:
                out << (unsigned int) obj;
                break;
        }
        out << ";" << endl;
}

void xsttAssign (xstValue &input, const char *, xstaCommon::Directional &dest)
{
        must_be (input.type(), xstValue::String, "Directional");
        RWCString str (input.string());

        // use upper case and strip space
        str.toUpper();
        str.strip(RWCString::both);

        if (str == "EAST" || str == "E") dest = xstaCommon::East;
        else if (str == "WEST" || str == "W") dest = xstaCommon::West;
        else if (str == "NORTH" || str == "N") dest = xstaCommon::North;
```

```
        else if (str == "SOUTH" || str == "S") dest = xstaCommon::South;
        else if (str == "NORTHEAST" || str == "NE") dest = xstaCommon::NorthEast;
        else if (str == "NORTHWEST" || str == "NW") dest = xstaCommon::NorthWest;
        else if (str == "SOUTHEAST" || str == "SE") dest = xstaCommon::SouthEast;
        else if (str == "SOUTHWEST" || str == "SW") dest = xstaCommon::SouthWest;
        else dest = xstaCommon::NoDirection;
}

void xsttAssign (xstValue &input, const char *, xstaCommon::AttributeList &dest)
{
        must_be (input.type(), xstValue::Hash, "AttributeList");
        xstValueHashIterator iter (input);
        while (iter()) {
                unsigned long i = dest.length ();
                dest.length (i+1);
                xstValue tag = iter.key();
                xstValue value = iter.value();
                xsttAssign (tag, "", dest[i].tag);
                xsttAssign (value, "", dest[i].value);
        }
}

void routingInfo (xstaServerConnection &obj, xstValue &input)
{
        must_be (input.type(), xstValue::Hash, "RoutingInfo");
        xstaCommon::AttributeList routingInfo;
        xsttAssign (input, "", routingInfo);
        obj.routingInfo (routingInfo);
}

/*
 * Pull in the generated code.  These files have function definitions for
 * contract specific xsttAssign and xsttDump as well as a function (e.g.,
 * do_SAQ) to set up and make the call to the corresponding TAG API function.
 */

#include "gen_xsttReqAddressValidation.hh"

#include "gen_xsttReqServiceAvailability.hh"

#include "gen_xsttReqAppointmentScheduling.hh"

#include "gen_xsttReqCustomerRecord.hh"

#include "gen_xsttReqDueDate.hh"

void xsttDump (const char *name,
        const xstaCommon::NpaNxxList &obj,
        int indent, ostream &out)
{
        xsttDumpListMarks marks (name, indent, out);
        indent++;
        unsigned long i;
        for (i=0; i < obj.length(); i++) {
                xsttDump (0, obj[i], indent, out);
        }
}

#include "gen_xsttReqTelephoneNumberAssignment.hh"

void xsttAssign (xstValue &input, const char *, xstaCommon::TestProdIndicator &dest)
{
        must_be (input.type(), xstValue::String, "TestProdIndicator");
        RWCString str (input.string());
        str.toUpper();
        if (str == "T") dest = xstaCommon::Test;
        else if (str == "P") dest = xstaCommon::Production;
        else dest = (xstaCommon::TestProdIndicator)-1;
}

#include "gen_xsttReqOrder.hh"
```

```
#include "gen_xsttReqLoop.hh"


static void Usage (const char *progname)
{
        cerr << "Usage: " << progname << " [options] app_id app_passwd user_id input_file ..." <<
endl;
        cerr << "Options:" << endl;
        cerr << "       -d s  delay s seconds between repeats (see -r)" << endl;
        cerr << "       -r n  repeat query n more times after first query" << endl;
        cerr << "       -o out write output to file out instead of stdout." << endl;
        exit(1);
}
```

# 8. Appendix C - Errors

## 8.1 API Errors

API errors are returned to inform the end-user of an improper configuration in their source code that interfaces with the API.

**Table 15- API Errors**

| Message Id | Exception | Message Text |
|---|---|---|
| TAGT0001API | InvalidData | Environment variable XST_CONFIG must be set to path of directory containing x |
| TAGT0002API | InvalidData | Failed to read config file <filename> |
| TAGT0003API | InvalidData | xst_client: <line>: unrecognized field name '<name>' |
| TAGT0004API | InvalidData | <field> was not set in config file |
| TAGT0006API | Backend Resource Limitation | Unexpected CORBA system exception: <message> |
| TAGT0007API | Backend Resource Limitation | Unexpected CORBA exception: <action> |
| TAGT0010API | InvalidData | Missing Transaction Type |
| TAGT0012API | Backend Resource Limitation | Unexpected CORBA system exception: <message> |
| TAGT0013API | InvalidData | Invalid object pointer could not be narrowed to type <typename> |
| TAGT0014API | Backend Resource Limitation | Unexpected CORBA exception: <action> |
| TAGT0060API | | LSR.REQTYP=<preqtyp> must match expected value of <ereqtyp> for function <function> |
| TGWT0067CB | Backend | TAG API input message sent to wrong |

| Message Id | Exception | Message Text |
|---|---|---|
| 1 | Resource Limitation | TAG release. |

## 8.2 Pre-Order Validation Error Messages

Pre-Order Validation Error Messages are returned to the end-user when they have submitted improper information in their Pre-Order query.

**Table 16- Pre-Order Errors Messages**

| Message Id | Field Name | Message Text |
|---|---|---|
| TAGT0001VAL | TAP-CLEC-ID | NUM Inquiry number is required for the resend option on the Address Validation Query function and cannot be greater than sixteen (16) characters. |
| TAGT0002VAL | TAP-CLEC-ID | CLEC Identifier is required |
| TAGT0003VAL | TP-SW-LSO-NPA-TTA | NPA/NXX is required and must be 6 numeric characters |
| TAGT0004VAL | TAP-CLEC-APPL-ID | CLEC Application Identifier is required |
| TAGT0005VAL | TAP-CLEC-APPL-PSWD | CLEC Application Password is required |
| TAGT0006VAL | TAP-CLEC-USER-ID | CLEC User Identifier is required |
| TAGT0008VAL | TAP-SW-LST | Local Service Termination is required and must be 8 or 11 alpha and/or numeric characters |
| TAGT0009VAL | TAP-TA-QTY-REQUESTED | Quantity Requested is required and must be between 1 and 25 |
| TAGT0011VAL | TAP-TA-QTY-IN-TER | In Terminal Quantity or Out Terminal Quantity is required on Hunt Group requests |
| TAGT0012VAL | TAP-TA-LEAD-TN | Lead TelephoneNumber is required on Hunt Group requests |
| TAGT0013VAL | TAP-TA-QTY-IN-TER | In Terminal Quantity must be greater than 2, if specified |
| TAGT0014VAL | TAP-TA-QTY-OUT-TER | Out Terminal Quantity must be less than 25, if specified |
| TAGT0015VAL | TAP-TA-EXIST-MLH-NO | Last Terminal In or Last Terminal Out is required when adding to existing Hunt Group |
| TAGT0017VAL | TAP-TA-QTY-REQESTED | Quantity Requested is required and must be between 1 and 500 |

| Message Id | Field Name | Message Text |
|---|---|---|
| TAGT0018VAL | TAP-TA-ROUTE-INDEX | Route Index is required on DID requests |
| TAGT0019VAL | TP-SW-LSO-NPA-TTA | NPA/NXX must be 6 numeric characters, if provided. |
| TAGT0020VAL | TAP-SW-LST | Either the LST or the NPA/NXX is required |
| TAGT0021VAL | TAP-TA-CONFIRM-NUM | Either the Confirmation Number or the Requested Number is required |
| TAGT0023VAL | TAP-SA-TN | Telephone Number is required |
| TAGT0024VAL | TAP-SA-STATE | Either the City/Community or the ZipCode is required |
| TAGT0025VAL | TAP-TN-ATN | Either the Account Telephone Number or the Circuit ID must be provided |
| TAGT0026VAL | TAP-CSR-STATE-CODE | State Code is required, if the Circuit ID is provided |
| TAGT0027VAL | TAP-CSR-AG-AUTH-ST | Agency authorization Status is required and must be Y |
| TAGT0028VAL | TAP-CSR-AUTH-DATE | Authorization date is required and must be less than or equal to the current date |
| TAGT0029VAL | TAP-SW-LST | LocalService Termination must be 8 or 11 alpha and/or numeric characters, if specified |
| TAGT0030VAL | TAP-TN-OPTION | OPTION TN Option is required and must be "None", "Easy", "Coin", "SeqLine", "AscendingLineDigits", "DescendingLineDigits", or "IdenticalLineDigits" |
| TAGT0031VAL | TAP-TNSQ-OPTION | TNSQ Option is required and must be "TN" or "DID" |
| TAGT0032VAL | TAP-TA-EXPIRE-DT | The Expiration Date must be a valid date greater than or equal to current date, if provided |
| TAGT0034VAL | TAP-SA-STATE | State is required |
| TAGT0034VAL | TAP-SW-PIC-SVC—OFNG PIC | Service Offering is required and must be 'B' (business) 'C' (coin), 'R' (residence), or 'W' (WATS) |
| TAGT0035VAL | TAP-SA-XBOUND-STATE | The City/Community is required when Cross Boundary State is provided. |
| TAGT0040VAL | TAP-TA-CONFIRM-NUM | Either the Confirmation Number or the Requested Number/Begin Range Number is required. |

| Message Id | Field Name | Message Text |
|---|---|---|
| TAGT0041VAL | TAP-TA-CONFIRM-NUM | Either the Confirmation Number or the Requested Number/Begin Range Number is required. |
| TAGT0050VAL | TAP-TA-MLH-RETURN-NUM1 | The MLH Return NUM1 is required. |
| TAGT0066VAL | TAP-SW-SERVICE-ABBREV | Maximum Number Of Service Abbreviations On A Query Is 10. |
| TAGT0200VAL | SOMAN-SOMEC-IND | Maximum field length exceeded |

## 8.3 Firm Order Validation Errors

Firm Order Validation Errors are returned to the end-user when they have submitted improper information in their Firm Order query.

**Table 17- Firm Order Validation Errors**

| Message ID | Message Text |
|---|---|
| TAGD4000VAL | DL DATA ELEMENTS REQUIRED |
| TAGD4005VAL | DL DATA ELEMENTS PROHIBITED |
| TAGD4010VAL | DL DATA PROHIBITED WHEN ERL Y IS POPULATED |
| TAGD4015VAL | VALID DLNUM REQUIRED |
| TAGD4020VAL | DLNUM=&DLNM LTN=&LTN DLNUM MUST BE UNIQUE |
| TAGD4022VAL | DLNUM=&DLNM LTN=&LTN DLNUM MUST BE 4 NUMERICS |
| TAGD4025VAL | DLNUM=&DLNM LTN=&LTN VALID LACT REQUIRED |
| TAGD4030VAL | DLNUM=&DLNM LTN=&LTN LACT REQUIRED |
| TAGD4035VAL | DLNUM=&DLNM LTN=&LTN ALI CODE PROHIBITED WHEN THE RTY 2ND AND 3RD CHARACTERS ARE ML |
| TAGD4040VAL | DLNUM=&DLNM LTN=&LTN ALI DATA MUST BE 1 - 3 ALPHAS |
| TAGD4045VAL | DLNUM=&DLNM LTN=&LTN ASSOCIATED LACT COMBINATION I OR O MISSING |
| TAGD4050VAL | DLNUM=&DLNM LTN=&LTN ALI CODE REQUIRED |
| TAGD4055VAL | DLNUM=&DLNM LTN=&LTN ALI MUST BE UNIQUE |
| TAGD4060VAL | DLNUM=&DLNM LTN=&LTN VALID RTY REQUIRED |
| TAGD4061VAL | DLNUM=$DLNM LTN=$LTN LASN, ADI OR LALOC DATA MUST BE PRESENT WHEN REQTYP IS J AND RTY IS LML AND LACT IS N OR I |

| Message ID | Message Text |
|---|---|
| TAGD4065VAL | DLNM=&DLNM LTN=&LTN ASSOCIATED LACT COMBINATION I AND O IS MISSING |
| TAGD4070VAL | DLNUM=&DLNM LTN=&LTN LISTING ACTIVITY INDICATOR (LACT) D PROHIBITED ON MAIN LISTING |
| TAGD4075VAL | MAIN LISTING REQUIRED |
| TAGD4080VAL | DLNUM=&DLNM LTN=&LTN INVALID FOREIGN LISTING TYPE |
| TAGD4085VAL | DLNUM=&DLNM LTN=&LTN INVALID SECONDARY LISTING TYPE |
| TAGD4090VAL | DLNUM=&DLNM LTN=&LTN VALID LTY REQUIRED |
| TAGD4095VAL | DLNUM=&DLNM LTN=&LTN LTY AND RTY COMBINATION INVALID |
| TAGD4097VAL | DLNUM=&DLNM LTN=&LTN LTY PROHIBITED WITH LACT Z |
| TAGD4100VAL | DLNUM=&DLNM LTN=&LTN TT VALUE MUST BE 1 OR 8 |
| TAGD4105VAL | DLNUM=&DLNM LTN=&LTN TT PROHIBITED WITH LACT Z |
| TAGD4110VAL | DLNUM=&DLNM LTN=&LTN VALID STYC CI, SH, SI, OR SL REQUIRED |
| TAGD4115VAL | DLNUM=&DLNM LTN=&LTN STYC PROHIBITED WITH LACT OF Z |
| TAGD4120VAL | DLNUM=&DLNM LTN=&LTN TOA B, R, RP OR BP REQUIRED |
| TAGD4125VAL | DLNUM=&DLNM LTN=&LTN TOA PROHIBITED WITH LACT OF Z |
| TAGD4130VAL | DLNUM=$DLNM LTN=$LTN WPP REQUIRES TOA OF R |
| TAGD4135VAL | DLNUM=$DLNM LTN=$LTN TOA DATA MUST BE BP |
| TAGD4140VAL | DLNUM=$DLNM LTN=$LTN WPP PROHIBITED WITH LACT OF Z |
| TAGD4145VAL | DLNUM=$DLNM LTN=$LTN WPP REQUIRES AN RTY FIRST CHARACTER OF L |
| TAGD4150VAL | DLNUM=$DLNM LTN=$LTN VALID VALUES OF WPP ARE DB, DBP, DS OR DSP |
| TAGD4155VAL | DLNUM=$DLNM LTN=$LTN WPP PROHIBITED FOR STATE |
| TAGD4160VAL | DLNUM=$DLNM LTN=$LTN DOI REQUIRED VALUE MUST BE 0 - 6 |
| TAGD4165VAL | DLNUM=$DLNM LTN=$LTN DOI PROHIBITED WITH LACT Z |
| TAGD4170VAL | DLNUM=$DLNM LTN=$LTN DOI MUST BE 1 |

| Message ID | Message Text |
|---|---|
| TAGD4175VAL | DLNUM=$DLNM LTN=$LTN DOI MUST BE GREATER THAN ZERO |
| TAGD4180VAL | DLNUM=$DLNM LTN=$LTN DOI VALUE MUST BE ZERO |
| TAGD4185VAL | DLNUM=$DLNM LTN=$LTN DOI DATA INVALID WITH LTY 3 |
| TAGD4190VAL | DLNUM=$DLNM LTN=$LTN DOI VALUE INVALID FOR STYLE CODE |
| TAGD4195VAL | DLNUM=$DLNM LTN=$LTN LTN PROHIBITED WITH RTY FCR OR LCR |
| TAGD4200VAL | DLNUM=$DLNM LTN=$LTN LTN MUST BE 10 NUMERICS |
| TAGD4205VAL | DLNUM=$DLNM LTN=$LTN LTN REQUIRED |
| TAGD4210VAL | DLNUM=$DLNM LTN=$LTN NSTN PROHIBITED WITH LACT Z |
| TAGD4215VAL | DLNUM=$DLNM LTN=$LTN INVALID CHARACTERS FOR NSTN |
| TAGD4220VAL | DLNUM=$DLNM LTN=$LTN LNLN REQUIRED |
| TAGD4225VAL | DLNUM=$DLNM LTN=$LTN LNLN PROHIBITED WITH LACT Z |
| TAGD4230VAL | DLNUM=$DLNM LTN=$LTN LNFN PROHIBITED WITH LACT Z |
| TAGD4235VAL | DLNUM=$DLNM LTN=$LTN LNPL CHARACTER MUST BE Y |
| TAGD4240VAL | DLNUM=$DLNM LTN=$LTN LNPL PROHIBITED WITH LACT Z |
| TAGD4245VAL | DLNUM=$DLNM LTN=$LTN LNPL PROHIBITED WHEN PLA IS POPULATED |
| TAGD4250VAL | DLNUM=$DLNM LTN=$LTN PLA PROHIBITED WITH LACT Z |
| TAGD4255VAL | DLNUM=$DLNM LTN=$LTN DES PROHIBITED WITH LACT Z |
| TAGD4265VAL | DLNUM=$DLNM LTN=$LTN TITLE OF LINEAGE INVALID |
| TAGD4270VAL | DLNUM=$DLNM LTN=$LTN TL PROHIBITED WITH LACT Z |
| TAGD4275VAL | DLNUM=$DLNM LTN=$LTN TITLE1 PROHIBITED WITH LACT Z |
| TAGD4280VAL | DLNUM=$DLNM LTN=$LTN TITLE1 DATA INVALID |
| TAGD4285VAL | DLNUM=$DLNM LTN=$LTN TITLE2 PROHIBITED WITH LACT Z |
| TAGD4290VAL | DLNUM=$DLNM LTN=$LTN TITLE2 DATA INVALID |
| TAGD4295VAL | DLNUM=$DLNM LTN=$LTN NICK PROHIBITED WITH LACT Z |

| Message ID | Message Text |
|---|---|
| TAGD4300VAL | DLNUM=$DLNM LTN=$LTN LAPR PROHIBITED WITHOUT LANO |
| TAGD4305VAL | DLNUM=$DLNM LTN=$LTN LAPR PROHIBITED WITH LACT Z |
| TAGD4310VAL | DLNUM=$DLNM LTN=$LTN LANO PROHIBITED WITHOUT LASN |
| TAGD4315VAL | DLNUM=$DLNM LTN=$LTN LANO PROHIBITED WITH LACT Z |
| TAGD4320VAL | DLNUM=$DLNM LTN=$LTN LASF PROHIBITED WITHOUT LANO |
| TAGD4325VAL | DLNUM=$DLNM LTN=$LTN LASF PROHIBITED WITH LACT Z |
| TAGD4330VAL | DLNUM=$DLNM LTN=$LTN LASD PROHIBITED WITH LACT Z |
| TAGD4335VAL | DLNUM=$DLNM LTN=$LTN LASD ENTRY INVALID |
| TAGD4340VAL | DLNUM=$DLNM LTN=$LTN LASD PROHIBITED WITHOUT LASN |
| TAGD4345VAL | DLNUM=$DLNM LTN=$LTN LASN PROHIBITED WITH LACT Z |
| TAGD4350VAL | DLNUM=$DLNM LTN=$LTN LATH PROHIBITED WITH LACT Z |
| TAGD4355VAL | DLNUM=$DLNM LTN=$LTN LATH PROHIBITED WITHOUT LASN |
| TAGD4360VAL | DLNUM=$DLNM LTN=$LTN LASS PROHIBITED WITH LACT Z |
| TAGD4365VAL | DLNUM=$DLNM LTN=$LTN LASS ENTRY INVALID |
| TAGD4370VAL | DLNUM=$DLNM LTN=$LTN LASS PROHIBITED WITHOUT LASN |
| TAGD4375VAL | DLNUM=$DLNM LTN=$LTN LALOC PROHIBITED WITH LACT Z |
| TAGD4380VAL | DLNUM=$DLNM LTN=$LTN LALOC REQUIRED WITH FOREIGN LISTING |
| TAGD4385VAL | DLNUM=$DLNM LTN=$LTN INVALID LAST ENTRY |
| TAGD4390VAL | DLNUM=$DLNM LTN=$LTN LAST PROHIBITED WITH LACT Z |
| TAGD4395VAL | DLNUM=$DLNM LTN=$LTN LAST PROHIBITED WITH ADI |
| TAGD4400VAL | DLNUM=$DLNM LTN=$LTN LTEXT PROHIBITED WITH LACT Z |
| TAGD4405VAL | DLNUM=$DLNM LTN=$LTN LTEXT REQUIRED |
| TAGD4410VAL | DLNUM=$DLNM LTN=$LTN LTEXT NUMERIC INVALID |
| TAGD4415VAL | DLNUM=$DLNM LTN=$LTN LTEXT MUST NOT EXCEED 90 CHARACTERS FOR SPECIAL TEXT |

| Message ID | Message Text |
|---|---|
| TAGD4420VAL | DLNUM=$DLNM LTN=$LTN LTXTY PROHIBITED WITH LACT Z |
| TAGD4425VAL | DLNUM=$DLNM LTN=$LTN LTXTY REQUIRED |
| TAGD4430VAL | DLNUM=$DLNM LTN=$LTN NVALID LTXTY DATA |
| TAGD4435VAL | DLNUM=$DLNM LTN=$LTN LTXTY AND RTY VALUES MUST MATCH |
| TAGD4445VAL | DLNUM=$DLNM LTN=$LTN RTY VALUE MUST BE AC WHEN LTXTY IS AC OR WPP |
| TAGD4450VAL | DLNUM=$DLNM LTN=$LTN LTXTY INVALID FOR STATE |
| TAGD4455VAL | DLNUM=$DLNM LTN=$LTN LTXNUM VALUE MUST BE 4 NUMERICS |
| TAGD4460VAL | DLNUM=$DLNM LTN=$LTN LTXNUM PROHIBITED WITH LACT Z |
| TAGD4465VAL | DLNUM=$DLNM LTN=$LTN LTXNUM IS REQUIRED |
| TAGD4470VAL | DLNUM=$DLNM LTN=$LTN LTXNUM MUST BE CONSECUTIVE AND UNIQUE WITHIN THE DLNUM |
| TAGD4475VAL | DLNUM=$DLNM LTN=$LTN INVALID YPH ENTRY |
| TAGD4480VAL | DLNUM=$DLNM LTN=$LTN YPH PROHIBITED WITH LACT Z |
| TAGD4485VAL | DLNUM=$DLNM LTN=$LTN YPH REQUIRED WHEN THE TOS IS 1 OR 3 AND RTY IS ML, AM OR CM |
| TAGD4490VAL | DLNUM=$DLNM LTN=$LTN YPH PROHIBITED WITH THIS RTY |
| TAGD4495VAL | DLNUM=$DLNM LTN=$LTN SIC ENTRY MUST BE 4 OR 5 NUMERICS |
| TAGD4500VAL | DLNUM=$DLNM LTN=$LTN SIC PROHIBITED WITH LACT Z |
| TAGD4505VAL | DLNUM=$DLNM LTN=$LTN SIC REQUIRED WHEN ACT IS N, V OR P |
| TAGD4510VAL | DLNUM=$DLNM LTN=$LTN ONLY ONE SIC ALLOWED PER ACCOUNT |
| TAGD4515VAL | DLNUM=$DLNM LTN=$LTN SIC IS PROHIBITED WITH RESIDENCE |
| TAGD4520VAL | DLNUM=$DLNM LTN=$LTN ADI VALID ENTRY IS Y |
| TAGD4525VAL | DLNUM=$DLNM LTN=$LTN ADI PROHIBITED WITH LACT Z |
| TAGD4530VAL | DLNUM=$DLNM LTN=$LTN ADI PROHIBITED WHEN LASN OR LALOC IS POPULATED |
| TAGD4535VAL | DLNUM=$DLNM LTN=$LTN ADI PROHIBITED WITH CROSS REFERENCE |
| TAGD4540VAL | DLNUM=$DLNM LTN=$LTN ADI PROHIBITED WITH FOREIGN LISTINGS |

| Message ID | Message Text |
|---|---|
| TAGD4545VAL | DLNUM=$DLNM LTN=$LTN DIRNAME PROHIBITED WITH LACT Z |
| TAGD4550VAL | DLNUM=$DLNM LTN=$LTN DIRNAME REQUIRED ON FOREIGN OR SECONDARY LISTING |
| TAGD4555VAL | DLNUM=$DLNM LTN=$LTN DIRSUB PROHIBITED WITH LACT Z |
| TAGD4560VAL | DLNUM=$DLNM LTN=$LTN ADV VALID ENTRY IS Y |
| TAGD4565VAL | DLNUM=$DLNM LTN=$LTN ADV PROHIBITED WITH LACT Z |
| TAGD4570VAL | DLNUM=$DLNM LTN=$LTN ADV IS PROHIBITED WHEN WPP IS POPULATED |
| TAGD4575VAL | DLNUM=$DLNM LTN=$LTN ADV IS PROHIBITED WITH DESIGNER LISTINGS |
| TAGD4580VAL | DLNUM=$DLNM LTN=$LTN DML VALUE MUST BE Y |
| TAGD4585VAL | DLNUM=$DLNM LTN=$LTN DML PROHIBITED WITH LACT Z |
| TAGD4590VAL | DLNUM=$DLNM LTN=$LTN DLNM VALID ENTRY IS Y |
| TAGD4595VAL | DLNUM=$DLNM LTN=$LTN DLNM PROHIBITED WITH LACT Z |
| TAGD4600VAL | DLNUM=$DLNM LTN=$LTN AMPERSAND REQUIRED WITH DLNM |
| TAGD4605VAL | DLNUM=$DLNM LTN=$LTN BRO MUST BE B OR R |
| TAGD4610VAL | DLNUM=$DLNM LTN=$LTN BRO PROHIBITED WITH LACT OF Z |
| TAGD4615VAL | DLNUM=$DLNM LTN=$LTN DSCR FIELDS PROHIBITED WHEN STYC IS NOT CI, SH OR SI |
| TAGD4620VAL | DLNUM=$DLNM LTN=$LTN SO VALUE MUST BE A OR F |
| TAGD4625VAL | DLNUM=$DLNM LTN=$LTN SO PROHIBITED WITH LACT OF Z |
| TAGD4630VAL | DLNUM=$DLNM LTN=$LTN SEQTEXT PROHIBITED WITHOUT THE SO FIELD OF A |
| TAGD4635VAL | DLNUM=$DLNM LTN=$LTN SEQTEXT PROHIBITED WITH LACT OF Z |
| TAGD4640VAL | DLNUM=$DLNM LTN=$LTN SEQADDR PROHIBITED WHEN THE SO FIELD IS NOT A |
| TAGD4645VAL | DLNUM=$DLNM LTN=$LTN SEQADDR PROHIBITED WITH LACT OF Z |
| TAGD4650VAL | DLNUM=$DLNM LTN=$LTN SEQTN PROHIBITED WHEN THE SEQTEXT OR SEQADDR IS NOT POPULATED |
| TAGD4655VAL | DLNUM=$DLNM LTN=$LTN SEQTN PROHIBITED WITH LACT OF Z |

| Message ID | Message Text |
|---|---|
| TAGD4660VAL | DLNUM=$DLNM LTN=$LTN SEQTN MUST BE 3 OR 10 NUMERICS |
| TAGD4665VAL | DLNUM=$DLNM LTN=$LTN SEQTN 2ND AND 3RD NUMERICS MUST BE 1 1 |
| TAGD4670VAL | DLNUM=$DLNM LTN=$LTN LVL REQUIRED WHEN STYC IS CI |
| TAGD4675VAL | DLNUM=$DLNM LTN=$LTN INVALID LVL VALUE |
| TAGD4680VAL | DLNUM=$DLNM LTN=$LTN LVL PROHIBITED WITH LACT OF Z |
| TAGD4685VAL | DLNUM=$DLNM LTN=$LTN LVL ENTRIES MUST BE SEQUENTIAL AND THE SAME LVL VALUE CANNOT APPEAR MORE THAN TWICE |
| TAGD4690VAL | DLNUM=$DLNM LTN=$LTN HS PROHIBITED WHEN THE STYC IS NOT CI, SH OR SI |
| TAGD4695VAL | DLNUM=$DLNM LTN=$LTN HS VALUE MUST BE E OR N |
| TAGD4700VAL | DLNUM=$DLNM LTN=$LTN HS PROHIBITED WITH LACT OF Z |
| TAGD4710VAL | DLNUM=$DLNM LTN=$LTN INVALID HTN DATA |
| TAGD4715VAL | DLNUM=$DLNM LTN=$LTN HTN PROHIBITED WHEN THE STYC IS NOT SI OR SH |
| TAGD4720VAL | DLNUM=$DLNM LTN=$LTN HTN PROHIBITED WITH LACT OF Z |
| TAGD4725VAL | DLNUM=&DLNUM LTN=&LTN INS1 VALUE MUST BE E OR N |
| TAGD4730VAL | DLNUM=&DLNUM LTN=&LTN INS1 REQUIRED WHEN DOI IS GREATER THAN 1 |
| TAGD4735VAL | DLNUM=&DLNUM LTN=&LTN INS1 REQUIRED WHEN LVL IS GREATER THAN 1 |
| TAGD4740VAL | DLNUM=&DLNUM LTN=&LTN INS1 REQUIRED WHEN INTEXT OR INADDR IS POPULATED |
| TAGD4745VAL | DLNUM=&DLNUM LTN=&LTN INS1 PROHIBITED WITH LACT OF Z |
| TAGD4750VAL | DLNUM=&DLNUM LTN=&LTN SO1 VALUE MUST BE A OR F |
| TAGD4755VAL | DLNUM=&DLNUM LTN=&LTN SOI PROHIBITED WITH LACT OF Z |
| TAGD4760VAL | DLNUM=&DLNUM LTN=&LTN SEQTEXT1 PROHIBITED WITH LACT OF Z |
| TAGD4765VAL | DLNUM=&DLNUM LTN=&LTN SEQADDR1 REQUIRES SO1 |
| TAGD4770VAL | DLNUM=&DLNUM LTN=&LTN SEQADDR1 PROHIBITED WITH LACT OF Z |
| TAGD4775VAL | DLNUM=&DLNUM LTN=&LTN SEQTN1 REQUIRES SO1 |

| Message ID | Message Text |
|---|---|
| TAGD4780VAL | DLNUM=&DLNUM LTN=&LTN SEQTN1 PROHIBITED WITH LACT OF Z |
| TAGD4785VAL | DLNUM=&DLNUM LTN=&LTN SEQTN1 MUST BE 3 OR 10 NUMERICS |
| TAGD4790VAL | DLNUM=&DLNUM LTN=&LTN SEQTN1 2ND AND 3RD NUMERICS MUST BE 1 1 |
| TAGD4795VAL | DLNUM=&DLNUM LTN=&LTN INTN REQUIRES INADDR OR INTEXT |
| TAGD4800VAL | DLNUM=&DLNUM LTN=&LTN INTN PROHIBITED WITH LACT OF Z |
| TAGD4805VAL | DLNUM=&DLNUM LTN=&LTN INNSTN PROHIBITED WITH LACT OF Z |
| TAGD4810VAL | DLNUM=&DLNUM LTN=&LTN INS1 REQUIRED WHEN INTEXT IS POPULATED |
| TAGD4815VAL | DLNUM=&DLNUM LTN=&LTN INTEXT PROHIBITED WITH LACT OF Z |
| TAGD4820VAL | DLNUM=&DLNUM LTN=&LTN INADDR PROHIBITED WITH LACT OF Z |
| TAGD4825VAL | DLNUM=&DLNUM LTN=&LTN INS1 REQUIRED WHEN INADDR IS POPULATED |
| TAGD4830VAL | ONLY ONE DACT PER LSR |
| TAGD4835VAL | DACT ENTRY MUST BE N |
| TAGD4837VAL | DACT REQUIRED |
| TAGD4840VAL | NAME REQUIRED |
| TAGD4845VAL | DDAPR PROHIBITED |
| TAGD4850VAL | DDANO PROHIBITED |
| TAGD4855VAL | DDASF PROHIBITED |
| TAGD4860VAL | DDASD PROHIBITED |
| TAGD4865VAL | DDASD INVALID ENTRY |
| TAGD4870VAL | DDASN IS REQUIRED |
| TAGD4875VAL | DDATH PROHIBITED |
| TAGD4880VAL | DDASS PROHIBITED |
| TAGD4885VAL | DDALO PROHIBITED |
| TAGD4890VAL | DDADLO IS PROHIBITED |
| TAGD4895VAL | DDALOC REQUIRED |
| TAGD4900VAL | DDAST REQUIRED |
| TAGD4905VAL | DDAZC REQUIRED |
| TAGD4910VAL | VALID DIRTYPE ENTRY IS W, Y, B OR O |
| TAGD4915VAL | DIRTYP PROHIBITED WITHOUT DIRQTY A OR DIRQTY NC |
| TAGD4920VAL | DIRTYP A PROHIBITED WITHOUT DIRTYP |
| TAGD4925VAL | DIRTYP NC PROHIBITED WITHOUT DIRTYP |

| Message ID | Message Text |
|---|---|
| TAGD9496VAL | CAPTION DATA EXPECTED WHEN LVL IS 0 |
| TAGD9497VAL | MULTIPLE ENTRIES OF ADV ARE PROHIBITED ON LSR |
| TAGD9498VAL | RTY COMBINATION REQUIRES AN ADI OF Y |
| TAGD9499VAL | LAST PROHIBITED WITH RTY COMBINATION |
| TAGD9500VAL | THE NSTN ASSOCIATED WITH A MAIN LISTING IS PROHIBITED |
| TAGD9501VAL | DACT IS REQUIRED |
| TAGD9502VAL | HS REQUIRED |
| TAGD9503VAL | WHEN BOTH THE DIRTYP A IS 0 AND DIRQTY = 0 OTHER OCCURRENCES OF THESE FIELDS ARE PROHIBITED |
| TAGD9504VAL | DLNUM=$DLNM LTN=$LTN INTN MUST HAVE 10 NUMBERS |
| TAGD9505VAL | DLNUM=$DLNM LTN=$LTN LNPL CHARACTER MUST BE Y IF EDI AND L IF FAX |
| TAGD9506VAL | LTY OF 2 OR 3 PROHIBITED WHEN THE WPP FIELD IS POPULATED |
| TAGD9510VAL | DLNUM=$DLNM LTN=$LTN LTEXT, LTXTY, AND LTXNUM ARE ALL REQUIRED WHEN ANY ONE IS POPULATED |
| TAGD9511VAL | ONLY 3 OCCURRENCES OF THE COMBINED FIELDS OF LTXNUM, LTXTY AND LTEXT ARE ALLOWED PER DLNUM |
| TAGD9512VAL | DLNUM=$DLNM LTN=$LTN MORE THAN ONE MAIN LISTING PROHIBITED |
| TAGD9513VAL | DLNUM=$DLNM LTN=$LTN PLA REQUIRED WHEN LNLN IS NUMERIC |
| TAGD9515VAL | DLNUM=$DLNM LTN=$LTN SIC ENTRY MUST BE MIN OF 4 AND MAX OF 5 NUMERIC |
| TAGD9517VAL | DLNUM=$DLNM LTN=$LTN SIC REQUIRED FOR TOS |
| TAGD9555VAL | INVALID CAPITALIZATION RULES |
| TAGD9613VAL | UNABLE TO DETERMINE USOC FOR LISTING BEING RE-ESTABLISHED |
| TAGD9614VAL | INVALID YPH CODE |
| TAGD9618VAL | INCORRECT YPH PROVIDED FOR NON-LISTED OR NON-PUBLISHED SERVICES |
| TAGD9619VAL | HANDLE MANUALLY-MULTIPLE OCCURRENCES OF DIRTYP AND DIRQTY A |
| TAGD9620VAL | ALI VALUE INVALID |
| TAGD9621VAL | AUTONO PAGE DIRECTORIES FOUND FOR THIS NPA NXX |

| Message ID | Message Text |
|---|---|
| TAGD9622VAL | NO WHITE PAGE DIRECTORIES FOUND FOR THIS NPA NXX |
| TAGD9626VAL | REQUEST FOR ADVANCE DIRECTORY LISTING PASS DIRECTORY CLOSE |
| TAGD9754VAL | HANDLE DIRECTORY ORDER MANUALLY |
| TAGD9764VAL | DIRECTORY REQUEST HANDLED BY LCSC |
| TAGD9772VAL | MANUAL HANDLING REQUIRED FOR TITLE2 FIELD |
| TAGD9773VAL | DUAL NAME LISTING PROHIBITED WITH TOS 1 OR 3 |
| TAGD9774VAL | MANUAL HANDLING REQUIRED FOR NICK / DLNM COMBINATION |
| TAGD9782VAL | ADDRESS SHOWN FOR ADDITIONAL LISTING DOES NOT MATCH SERVICE ADDRESS |
| TAGD9784VAL | HANDLE MANUALLY - COMPLEX LISTING |
| TAGD9786VAL | INVALID CAPITALIZATION RULES |
| TAGD9788VAL | EXTRA LINE LISTING EXCEEDS 32 CHARACTERS |
| TAGD9790VAL | EXTRA LINE LISTING HAS INVALID CHARACTERS |
| TAGD9791VAL | THE FIRST 3 CHARACTERS OF NSTN MUST BE NUMERIC |
| TAGD9792VAL | HANDLE MANUALLY: LESS THAN 10 CHARACTERS IN NSTN FIELD |
| TAGD9793VAL | MANUAL HANDLING REQUIRED FOR NSTN |
| TAGD9794VAL | INVALID CHARACTERS FOR STYLIST LISTING |
| TAGD9796VAL | DDA EXCEEDS 5 LINES |
| TAGD9798VAL | HANDLE REQUEST FOR INTERIM DIRECTORIES MANUALLY AND RESUBMIT TO SERVICE ORDER GENERATION |
| TAGD9800VAL | UNABLE TO LOCATE Q ACCOUNT IN CRIS |
| TAGD9803VAL | INVALID END USER ADDRESS |
| TAGD9804VAL | EXISTING ACCOUNT NUMBER FOR ACCOUNT NOT PROVIDED |
| TAGD9806VAL | LTXTY OF CR REQUIRES SEE AS FIRST WORD IN LTEXT |
| TAGD9810VAL | HANDLE MANUALLY - UNABLE TO FIND THE LTN COMBINATION FOR LACT Z ASSOCIATED WITH THIS RTY |
| TAGD9814VAL | UNABLE TO ASSOCIATE LISTING FOR CR WITH AN ACCOUNT |
| TAGD9815VAL | CLASS OF SERVICE OR OBSOLETE OFFERING NOT ELIGIBLE FOR PORTING |
| TAGE2000VAL | DQTY REQUIRED TO BE GREATER THAN ZERO WHEN DISC NBR IS POPULATED |
| TAGE2015VAL | LOCNUM FOR MAIN LOCATION MUST BE 000 |

| Message ID | Message Text |
|---|---|
| TAGE2020VAL | LOCNUM REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION AT THIS LOCATION |
| TAGE2025VAL | LOCNUM MUST BE 3 NUMERICS AND GREATER THAN 000 AT THIS LOCATION |
| TAGE2030VAL | LOCNUM MUST BE UNIQUE FOR EACH LOCATION |
| TAGE2035VAL | LOCNUM=&LCNM NAME EU REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION AT THIS LOCATION |
| TAGE2040VAL | LOCNUM=&LCNM SANO PROHIBITED WHEN SASN IS NOT POPULATED AT THIS LOCATION |
| TAGE2045VAL | LOCNUM=&LCNM SASF PROHIBITED WHEN SASN AND SANO ARE NOT POPULATED AT THIS LOCATION |
| TAGE2050VAL | LOCNUM=&LCNM SASD PROHIBITED WHEN SASN IS NOT POPULATED AT THIS LOCATION |
| TAGE2055VAL | LOCNUM=&LCNM SASD VALID ENTRY IS E, W, N, S, NE, NW, SE, OR SW AT THIS LOCATION |
| TAGE2060VAL | LOCNUM=&LCNM SASN REQUIRED WITH THIS REQTYP/ACT TYP COMBINATION AT THIS LOCATION |
| TAGE2065VAL | LOCNUM=&LCNM SASN WITHOUT SANO MUST HAVE AN @ SYMBOL IN THE FIRST POSITION AT THIS LOCATION |
| TAGE2070VAL | LOCNUM=&LCNM SATH PROHIBITED WHEN SASN IS NOT POPULATED AT THIS LOCATION |
| TAGE2075VAL | LOCNUM=&LCNM SASS PROHIBITED WHEN SASN IS NOT POPULATED AT THIS LOCATION |
| TAGE2080VAL | LOCNUM=&LCNM SADLO REQUIRED WHEN SANO IS NOT POPULATED AT THIS LOCATION |
| TAGE2085VAL | LOCNUM=&LCNM FLOOR-EU MUST NOT BE POPULATED WITH FLR IN ANY POSITION AT THIS LOCATION |
| TAGE2090VAL | LOCNUM=&LCNM ROOM-EU MUST NOT BE POPULATED WITH RM OR ROOM IN ANY POSITION AT THIS LOCATION |
| TAGE2095VAL | LOCNUM=&LCNM BLDG-EU MUST NOT BE POPULATED WITH BLDG IN ANY POSITION AT THIS LOCATION |
| TAGE2100VAL | LOCNUM=&LCNM CITY-EU REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION AT THIS LOCATION |
| TAGE2105VAL | LOCNUM=&LCNM STATE-EU REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION AT THIS LOCATION |

| Message ID | Message Text |
|---|---|
| TAGE2110VAL | LOCNUM=&LCNM ZIP CODE-EU REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION AT THIS LOCATION |
| TAGE2115VAL | LOCNUM=&LCNM ZIP CODE-EU MUST BE 5 OR 9 NUMERICS AT THIS LOCATION |
| TAGE2120VAL | LOCNUM=&LCNM LCON-NAME MUST BE UP TO 15 ALPHANUMERICS WITH EMBEDDED BLANKS AT THIS LOCATION |
| TAGE2130VAL | LOCNUM=&LCNM TEL NO-LCON MUST BE 10 NUMERICS AT THIS LOCATION |
| TAGE2135VAL | LOCNUM=&LCNM EUMI VALID ENTRY IS Y OR N AT THIS LOCATION |
| TAGE2140VAL | IBT VALID ENTRY IS 1, 2 OR 3 |
| TAGE2145VAL | IWO VALID ENTRY IS S, U, OR W |
| TAGE2150VAL | IWO PROHIBITED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGE2155VAL | IWO VALID ENTRY FOR REQTYP A, B, OR M IS W |
| TAGE2160VAL | IWCON REQUIRED WHEN IWO IS POPULATED WITH U OR W |
| TAGE2165VAL | IWCON PROHIBITED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGE2170VAL | IWCON-TEL NO REQUIRED WHEN IWCON IS POPULATED |
| TAGE2175VAL | IWCON-TEL NO PROHIBITED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGE2180VAL | EAN REQUIRED WHEN EATN IS NOT POPULATED AND ACT TYPE IS V, P, OR Q |
| TAGE2185VAL | EAN MUST BE 10 NUMERICS OR 13 ALPHANUMERICS |
| TAGE2190VAL | EAN PROHIBITED WHEN EATN, LEATN, OR LEAN IS POPULATED |
| TAGE2195VAL | EATN REQUIRED WHEN EAN IS NOT POPULATED AND THE ACT IS V, P, OR Q |
| TAGE2200VAL | EATN MUST BE 10 NUMERICS |
| TAGE2205VAL | EATN PROHIBITED WHEN EAN, LEATN, OR LEAN IS POPULATED |
| TAGE2210VAL | FBI VALID ENTRY IS Y OR N |
| TAGE2215VAL | BILLNM-FB REQUIRED WHEN THE FBI FIELD IS POPULATED WITH Y |
| TAGE2220VAL | SBILLNM-FB MUST BE UP TO 25 ALPHANUMERICS WITH EMBEDDED BLANKS |
| TAGE2225VAL | STREET BILLNM-FB REQUIRED WHEN FBI FIELD IS POPULATED WITH Y |

| Message ID | Message Text |
|---|---|
| TAGE2230VAL | FLOOR BILLNM-FB MUST BE UP TO 12 ALPHANUMERICS |
| TAGE2235VAL | ROOM BILLNM-FB MUST BE UP TO 9 ALPHANUMERICS |
| TAGE2240VAL | CITY BILLNM-FB REQUIRED WHEN THE FBI IS POPULATED WITH Y |
| TAGE2245VAL | STATE BILLNM-FB REQUIRED WHEN FBI IS POPULATED WITH Y |
| TAGE2250VAL | STATE BILLNM-FB MUST BE 2 ALPHAS |
| TAGE2255VAL | ZIP CODE BILLNM-FB REQUIRED WHEN FBI IS POPULATED WITH Y |
| TAGE2260VAL | ZIP CODE BILLNM-FB MUST BE 5 OR 9 NUMERICS |
| TAGE2265VAL | BILLCON-FB REQUIRED WHEN FBI IS POPULATED WITH Y |
| TAGE2270VAL | TEL NO-FBCON MUST BE 10 NUMERICS IN THE FIRST TEN POSITIONS |
| TAGE2275VAL | TEL NO-FBCON REQUIRED WHEN FBI IS POPULATED WITH Y |
| TAGE2280VAL | DNUM REQUIRED WHEN DISC NBR IS POPULATED |
| TAGE2285VAL | DNUM MUST BE 5 ALPHANUMERICS |
| TAGE2290VAL | DNUM PROHIBITED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGE2295VAL | DNUM MUST BE GREATER THAN THE PREVIOUS DNUM |
| TAGE2300VAL | DNUM REQUIRED WHEN TC OPT IS POPULATED |
| TAGE2305VAL | DISC NBR MUST BE 10 NUMERICS |
| TAGE2310VAL | DISC NBR PROHIBITED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGE2315VAL | DISC NBR REQUIRED IF DNUM IS POPULATED |
| TAGE2320VAL | DISC NBR REQUIRED IF TER IS POPULATED |
| TAGE2325VAL | TER MUST BE UP TO 10 ALPHANUMERICS |
| TAGE2330VAL | TER PROHIBITED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGE2335VAL | AACT VALID ENTRY IS N |
| TAGE2340VAL | ERL WITH THE DATA OF Y PROHIBITED WHEN LEATN IS POPULATED |
| TAGE2345VAL | AACT PROHIBITED FOR THIS REQTYP/ACT TYPE COMBINATION |
| TAGE2350VAL | ERL REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGE2355VAL | ERL PROHIBITED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGE2360VAL | ERL VALID ENTRY IS Y OR N |

| Message ID | Message Text |
|---|---|
| TAGE2365VAL | LOCACT VALID ENTRY IS N, D, OR E |
| TAGE2370VAL | LOCACT REQUIRED WHEN LOCNUM IS POPULATED |
| TAGE2375VAL | LOCNUM=&LCNM WSOP MUST BE V OR BLANK |
| TAGE9100VAL | ERL WITH THE DATA OF Y PROHIBITED WHEN LEAN OR LEATN IS POPULATED |
| TAGE9101VAL | BILLCON-FB REQUIRED WHEN THE FBI FIELD IS POPULATED WITH Y IF EDI OR D IF FAX |
| TAGE9102VAL | BILLNM-FB REQUIRED WHEN THE FBI FIELD IS POPULATED WITH D FOR FAX OR Y FOR EDI |
| TAGE9105VAL | THE ONLY VALID TEXT ENTRIES FOR THE BLDG-EU ARE WNG OR PIER FOLLOWED BY A SPACE AT EACH SECONDARY LOCATION |
| TAGE9106VAL | CITY-FB REQUIRED WHEN THE FBI FIELD FOR EDI IS Y OR FOR FAX IS A D |
| TAGE9110VAL | DQTY INVALID DATA |
| TAGE9112VAL | EAN REQUIRED WHEN LEAN, LEATN AND EATN ARE NOT POPULATED AND ACT = V, P, OR Q |
| TAGE9114VAL | EATN REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION WHEN EAN OR LEAN OR LEATN IS NOT POPULATED AND THE ACT IS V, P, OR Q |
| TAGE9120VAL | ERL MUST BE N WHEN EUMI IS Y |
| TAGE9121VAL | LOCNUM=$LCNM EUMI CANNOT BE Y IF ERL FIELD IS Y |
| TAGE9123VAL | FBI - VALID ENTRIES ARE EDI Y OR N OR FOR FAX D OR E |
| TAGE9128VAL | THE ONLY VALID TEXT ENTRIES FOR THE ROOM FIELD ARE SLIP, LOT, UNIT, APT, SUIT FOLLOWED BY A SPACE AT EACH SECONDARY LOCATION |
| TAGE9130VAL | STATE-FB REQUIRED WHEN THE FBI FIELD IS POPULATED WITH Y IF EDI OR D IF FAX |
| TAGE9135VAL | STREET BILLNM-FB-REQUIRED WHEN THE FBI FIELD IS POPULATED WITH D FOR FAX OR Y FOR EDI |
| TAGE9140VAL | TEL NO-FBCON REQUIRED WHEN FBI FIELD IS POPULATED WITH Y IF EDI OR D IF FAX |
| TAGE9145VAL | ZIP CODE-FB REQUIRED WHEN THE FBI FIELD IS POPULATED WITH Y IF EDIT OR D IF FAX |
| TAGE9305VAL | LNUM=$LNUM TELNO=$TNUM LOCACT VALUE MUST BE E FOR REQTYP B OR C WHEN NPT IS D |
| TAGH5000VAL | HUNTING PROHIBITED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGH5005VAL | LOCNUM=&LCNM THE FOLLOWING FIELDS ARE REQUIRED: HNUM, HA AND HID |

| Message ID | Message Text |
|---|---|
| TAGH5010VAL | HTQTY MUST BE 2 NUMERICS, 01 THRU 99 |
| TAGH5015VAL | HTQTY MUST EQUAL TOTAL NUMBER OF HNUM ON THIS REQUEST |
| TAGH5025VAL | LOCNUM=&LCNM HNUM=&HNUM HA=&HA HA MUST BE N, E, C OR D |
| TAGH5030VAL | LOCNUM=&LCNM HNUM=&HNUM HA OF E PROHIBITED ON ACCT ACTIVITY TYPE N, T, P OR Q |
| TAGH5035VAL | LOCNUM=&LCNM HNUM=&HNUM HA OF C PROHIBITED ON ACCT ACTIVITY TYPE N, T, P OR Q |
| TAGH5040VAL | LOCNUM=&LCNM HNUM=&HNUM HA OF D PROHIBITED ON ACCT ACTIVITY TYPE N OR T |
| TAGH5045VAL | LOCNUM=&LCNM LOCNUM MUST BE 3 NUMERICS |
| TAGH5050VAL | LOCNUM=&LCNM DOES NOT MATCH AN END USER LOCNUM ON THIS LSR |
| TAGH5055VAL | LOCNUM=&LCNM HNUM=&HNUM HNUM MUST BE 5 NUMERICS |
| TAGH5060VAL | LOCNUM=&LCNM HNUM=&HNUM HNUM MUST BE UNIQUE WITHIN EACH LOCNUM |
| TAGH5065VAL | LOCNUM=&LCNM HNUM=&HNUM HID ENTRY FOR HNTYP 1, 2, 3 OR 4 MUST BE AN N OR UP TO 3 ALPHAS OR 4 NUMERICS |
| TAGH5070VAL | LOCNUM=&LCNM HNUM=&HNUM HID MUST BE N WHEN HA IS N AND HNTYP IS 1, 2, 3 OR 4 |
| TAGH5075VAL | LOCNUM=&LCNM HNUM=&HNUM HID MUST BE N OR HID NUMBER IF HA IS N AND HNTYP IS 5 OR 6 |
| TAGH5080VAL | LOCNUM=&LCNM HNUM=&HNUM HID MUST BE AN HID NUMBER WHEN HA IS C, D, OR E, AND HNTYP IS 5 OR 6 |
| TAGH5085VAL | LOCNUM=&LCNM HNUM=&HNUM TLI MUST BE 10 NUMERICS |
| TAGH5090VAL | LOCNUM=&LCNM HNUM=&HNUM TLI REQUIRED IF HNTYP IS 5 OR 6 |
| TAGH5095VAL | LOCNUM=&LCNM HNUM=&HNUM TLI PROHIBITED WHEN HNTYP IS 1, 2, 3 OR 4 AND NOTYP IS T |
| TAGH5098VAL | LOCNUM=&LCNM HNUM=&HNUM HNTYP REQUIRED FOR THIS ACT TYPE/HA COMBINATION |
| TAGH5100VAL | LOCNUM=&LCNM HNUM=&HNUM HNTYP VALID ENTRIES ARE: 1, 2, 3, 4, 5 OR 6 |
| TAGH5103VAL | LOCNUM=&LCNM HNUM=&HNUM HLA REQUIRED FOR THIS ACT TYPE/HA COMBINATION |
| TAGH5105VAL | LOCNUM=&LCNM HNUM=&HNUM HLA=&HLA HLA VALID ENTRIES ARE: N, E OR D |
| TAGH5110VAL | LOCNUM=&LCNM HNUM=&HNUM HLA=&HLA HLA OF N PROHIBITED WHEN HUNT GROUP ACTIVITY IS E |

| Message ID | Message Text |
|---|---|
| TAGH5115VAL | LOCNUM=&LCNM HNUM=&HNUM HLA=&HLA HLA OF E PROHIBITED WHEN HUNT GROUP ACTIVITY IS N |
| TAGH5120VAL | LOCNUM=&LCNM HNUM=&HNUM HLA=&HLA HLA OF D PROHIBITED WHEN HUNT GROUP ACTIVITY IS N OR E |
| TAGH5123VAL | LOCNUM=&LCNM HNUM=&HNUM HTSEQ REQUIRED FOR THIS HA/HLA COMBINATION |
| TAGH5125VAL | LOCNUM=&LCNM HNUM=&HNUM HTSEQ=&HTSQ SAME HTSEQ NOT ALLOWED FOR MORE THAN ONE HT WHEN HLA IS N OR E |
| TAGH5130VAL | LOCNUM=&LCNM HNUM=&HNUM HTSEQ=&HTSQ HTSEQ MUST BE 4 NUMERICS |
| TAGH5135VAL | LOCNUM=&LCNM HNUM=&HNUM HTSEQ=&HTSQ SAME HT NOT ALLOWED IN MORE THAN ONE HTSEQ WHEN HLA IS N OR E |
| TAGH5138VAL | LOCNUM=&LCNM HNUM=&HNUM NOTYP REQUIRED FOR THIS HA/HLA COMBINATION |
| TAGH5140VAL | LOCNUM=&LCNM HNUM=&HNUM NOTYP=&NTYP VALID ENTRIES FOR NOTYP ARE T OR L |
| TAGH5145VAL | LOCNUM=&LCNM HNUM=&HNUM NOTYP=&NTYP NOTYP MUST BE L IF HNTYP IS 5 OR 6 |
| TAGH5150VAL | LOCNUM=&LCNM HNUM=&HNUM NOTYP=&NTYP NOTYP MUST BE T IF HNTYP IS 1, 2, 3 OR 4 |
| TAGH5153VAL | LOCNUM=&LCNM HNUM=&HNUM HT REQUIRED FOR THIS HA/HLA COMBINATION |
| TAGH5155VAL | LOCNUM=&LCNM HNUM=&HNUM HT CANNOT EXCEED 16 IN AN HID WHEN HNTYP IS 1, 2, 3 OR 4 AND HA IS N |
| TAGH5160VAL | LOCNUM=$LCNM HNUM=$HNUM HT WITH HLA OF E OR N, CANNOT EXCEED 3 IN AN HID WHEN TOS IS 2A-- IN $STAT |
| TAGH5165VAL | LOCNUM=&LCNM HNUM=&HNUM HT WITH HLA OF E OR N, CANNOT EXCEED 5 IN AN HID WHEN TOS IS 2A-- IN FL |
| TAGH5170VAL | LOCNUM=&LCNM HNUM=&HNUM HT WITH HLA OF E OR N, CANNOT EXCEED 10 IN AN HID WHEN TOS IS 2A-- IN LA |
| TAGH5175VAL | HNUM=&HNUM HT=&HT HT MUST BE 10 NUMERICS OR 14 NUMERICS WITH A HYPHEN IF HNTYP 1-4 |
| TAGH5180VAL | LOCNUM=&LCNM HNUM=&HNUM HT=&HT TN OR TER SCOPED NUMBER RANGES MUST BE IN ASCENDING ORDER |
| TAGH5185VAL | LOCNUM=&LCNM HNUM=&HNUM HT=&HT FOR HNTYP 5 OR 6, HT MUST BE 5 OR 10 ALPHANUMERICS |
| TAGP7000VAL | EAN OR EATN OR LEATN ON LINES OR LEAN ON LINES IS REQUIRED WHEN ACT IS P, Q OR V |

| Message ID | Message Text |
|---|---|
| TAGP7005VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM EAN, EATN, LEATN, LEAN ARE MUTUALLY EXCLUSIVE. LEAN PROHIBITED WHEN LEATN IS POPULATED |
| TAGP7010VAL | EATN OR EAN REQUIRED WHEN ACT IS P, Q OR V |
| TAGP7015VAL | LEAN AND LEATN PROHIBITED WHEN 2ND CHARACTER OF TOS IS NOT A OR B OR FOR REQTYP N ON ACT P, Q OR V |
| TAGP7040VAL | LNUM=&LNUM TELNO=&TNUM FA PROHIBITED FOR REQTYP |
| TAGP7050VAL | LNUM=&LNUM TELNO=&TNUM LNA CANNOT BE N WHEN TNS IS POPULATED |
| TAGP7055VAL | LNUM=$LNUM TELNO=$TNUM LNA MUST BE V FOR REQTYP C |
| TAGP7075VAL | EATN, AN AND ATN ARE REQUIRED FOR REQTYP B |
| TAGP7080VAL | EATN AND AN ARE REQUIRED FOR REQTYP |
| TAGP9556VAL | INVALID ACT TYPE FOR PARTIAL MIGRATION |
| TAGP9557VAL | INVALID ACT TYPE FOR FULL MIGRATION |
| TAGP9702VAL | PARTIAL MIGRATION HANDLE MANUALLY |
| TAGR1001VAL | CCNA MUST BE 3 ALPHAS |
| TAGR1005VAL | CCNA REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGR1010VAL | PON REQUIRED (STOP) |
| TAGR1015VAL | PON DUPLICATE ON INITIAL LSR |
| TAGR1020VAL | PON VALID VALUES ARE ONLY UPPER CASE ALPHA A THRU Z, NUMERIC 0 THRU 9 AND SYMBOLS . , - ' |
| TAGR1025VAL | VER REQUIRED FOR SUPS |
| TAGR1030VAL | VER MUST BE GREATER THAN PREVIOUS VERSION |
| TAGR1035VAL | VER MUST BE TWO NUMERICS - 01 OR GREATER FOR 860 |
| TAGR1040VAL | VER MUST BE SPACES OR ZEROES FOR 850 |
| TAGR1045VAL | LOCQTY REQUIRED, GREATER THAN ZERO, WHEN MORE THAN ONE LOCATION APPEARS ON LSR |
| TAGR1050VAL | LOCQTY MUST BE 3 NUMERICS |
| TAGR1055VAL | AN REQUIRED FOR THIS REQTYP/ACT TYPE COMBINATION WHEN ATN IS NOT POPULATED |
| TAGR1060VAL | AN PROHIBITED WHEN ATN IS POPULATED UNLESS REQTYP IS B |
| TAGR1065VAL | AN MUST BE 10 OR 13 ALPHANUMERICS |
| TAGR1070VAL | AN OR ATN REQUIRED WHEN EAN IS POPULATED |
| TAGR1075VAL | ATN REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION WHEN AN IS NOT POPULATED |

| Message ID | Message Text |
|---|---|
| | COMBINATION WHEN AN IS NOT POPULATED |
| TAGR1080VAL | ATN PROHIBITED WHEN AN IS POPULATED UNLESS REQTYP IS B |
| TAGR1085VAL | ATN MUST BE 10 NUMERICS |
| TAGR1090VAL | ATN OR AN REQUIRED WHEN EATN IS POPULATED |
| TAGR1095VAL | SERVICE CENTER REQUIRED |
| TAGR1100VAL | SERVICE CENTER MUST BE LCSC |
| TAGR1105VAL | D/TSENT REQUIRED |
| TAGR1110VAL | D/TSENT MUST BE CURRENT DATE OR A FUTURE DATE |
| TAGR1115VAL | D/TSENT MUST BE A VALID DATE |
| TAGR1120VAL | DDD REQUIRED |
| TAGR1125VAL | DDD MUST BE GREATER THAN OR EQUAL TO D/TSENT |
| TAGR1130VAL | DDD MUST BE A VALID DATE |
| TAGR1135VAL | APPTIME-DDD MUST BE HHMM-HHMM (MILITARY TIME) COVERING A SPAN OF TIME OF ONE HOUR OR GREATER |
| TAGR1140VAL | DDDO REQUIRED WHEN ACT IS T AND REQTYP IS A, E, M, OR N |
| TAGR1145VAL | INTERVAL BETWEEN DDD AND DDDO MUST BE 30 CALENDAR DAYS OR LESS |
| TAGR1150VAL | DDDO MUST BE A VALID DATE |
| TAGR1155VAL | DFDT MUST BE POPULATED WITH A SINGLE (HHMM) TIME WHEN CHC IS Y |
| TAGR1165VAL | CHC ONLY VALID ENTRY IS Y OR N |
| TAGR1170VAL | CHC REQUIRED WHEN REQTYP IS A OR B AND DFDT IS POPULATED |
| TAGR1175VAL | REQTYP REQUIRED (STOP) |
| TAGR1180VAL | INVALID REQTYP/ACT TYPE COMBINATION (STOP) |
| TAGR1185VAL | REQTYP VALID ENTRIES MUST BE: AB, BB, CB, EB, FB, JB, MB, NB (STOP) |
| TAGR1190VAL | ACTIVITY TYPE REQUIRED (STOP) |
| TAGR1195VAL | ACTIVITY TYPE VALID ENTRY MUST BE N, C, D, T, R, V, S, B, W, L, Y, P OR Q (STOP) |
| TAGR1200VAL | SUP REQUIRED WHEN VER IS GREATER THAN 00 |
| TAGR1205VAL | SUP VALID ENTRIES ARE 01, 04, OR 05 |
| TAGR1210VAL | SUP PROHIBITED ON 850 |
| TAGR1215VAL | SUP PROHIBITED WHEN 1ST CHARACTER OF REQTYP CHANGES |
| TAGR1220VAL | EXPEDITE VALID ENTRY IS Y OR N |

| Message ID | Message Text |
|---|---|
| TAGR1225VAL | CC REQUIRED ON THIS REQTYP/ACT TYPE COMBINATION (STOP) |
| TAGR1230VAL | CC MUST BE 4 ALPHANUMERICS (STOP) |
| TAGR1235VAL | ALBR ONLY VALID ENTRY IS Y OR N |
| TAGR1240VAL | SCA MUST BE Y OR N |
| TAGR1245VAL | AGAUTH MUST BE Y OR N |
| TAGR1250VAL | DATED REQUIRED WHEN AGAUTH IS POPULATED WITH Y |
| TAGR1255VAL | DATED MUST BE A VALID DATE |
| TAGR1260VAL | DATED PROHIBITED WHEN AGAUTH IS NOT POPULATED |
| TAGR1265VAL | AUTHNM MUST BE 1 TO 15 ALPHANUMERICS |
| TAGR1270VAL | PORTTYP REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGR1275VAL | PORTTYP PROHIBITED ON THIS REQTYP/ACT TYPE COMBINATION |
| TAGR1280VAL | PORTTYP MUST BE L OR T |
| TAGR1285VAL | ACTL REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGR1290VAL | ACTL MUST BE 11 ALPHANUMERICS |
| TAGR1295VAL | AI PROHIBITED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGR1300VAL | AI REQUIRED WHEN THE APOT FIELD IS POPULATED |
| TAGR1305VAL | AI VALID ENTRY IS Y OR N |
| TAGR1310VAL | APOT REQUIRED IF AI IS POPULATED |
| TAGR1315VAL | APOT MUST BE 11 ALPHANUMERICS IF AI IS Y |
| TAGR1320VAL | APOT PROHIBITED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGR1325VAL | LST MUST BE 11 ALPHANUMERICS |
| TAGR1330VAL | LST REQUIRED IF REQTYP IS F |
| TAGR1335VAL | LSO REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGR1340VAL | LSO MUST BE 6 NUMERICS |
| TAGR1345VAL | TOS REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION (STOP) |
| TAGR1350VAL | TOS MUST BE 3 ALPHANUMERICS |
| TAGR1355VAL | TOS FIRST CHARACTER MUST BE 1, 2, 3, OR 4 |
| TAGR1360VAL | TOS SECOND CHARACTER MUST BE A, B, C, D, H, J, R OR - (HYPHEN) (STOP) |

| Message ID | Message Text |
|---|---|
| TAGR1365VAL | TOS THIRD CHARACTER MUST BE M, F, G, OR - (HYPHEN) |
| TAGR1370VAL | TOS THIRD CHARACTER MUST NOT BE F IF REQTYP IS FB |
| TAGR1375VAL | TOS FIRST CHARACTER IS 2, SECOND CHARACTER MUST BE A, B, H, J, R OR (HYPHEN) |
| TAGR1380VAL | TOS FIRST CHARACTER IS 1 OR 3, SECOND CHARACTER MUST NOT BE C (STOP) |
| TAGR1385VAL | TOS FIRST CHARACTER IS 4, SECOND CHARACTER MUST BE C |
| TAGR1390VAL | TOS SECOND CHARACTER MUST BE - (HYPHEN) IF REQTYP IS JB |
| TAGR1392VAL | TOS 2ND CHARACTER OF J IS PROHIBITED ON REQTYP A, B, C, F OR J |
| TAGR1395VAL | TOS THIRD CHARACTER MUST BE - (HYPHEN) IF REQTYPE IS JB, BB, OR CB |
| TAGR1400VAL | SPEC MUST BE 5 OR 7 ALPHANUMERICS |
| TAGR1405VAL | RPON VALID VALUES ARE ONLY UPPER CASE ALPHA A THRU Z, NUMERIC 0 THRU 9, AND SYMBOLS . , - ' |
| TAGR1410VAL | LSP AUTH MUST BE 4 ALPHANUMERICS |
| TAGR1415VAL | LSP AUTH DATE REQUIRED WHEN LSP AUTH IS POPULATED |
| TAGR1420VAL | LSP AUTH DATE MUST BE A VALID DATE |
| TAGR1425VAL | LSP AUTH NAME REQUIRED WHEN LSP AUTH IS POPULATED |
| TAGR1430VAL | CIC REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGR1435VAL | CIC MUST BE 4 NUMERICS |
| TAGR1440VAL | CUST REQUIRED WHEN CCNA IS CUS |
| TAGR1445VAL | BI1 REQUIRED WHEN BAN1 AND BAN2 ARE POPULATED |
| TAGR1450VAL | BI1 VALID ENTRIES ARE A, D, L, M, N, P, R, U, V, W OR Z |
| TAGR1453VAL | BAN1 REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGR1455VAL | BAN1 VALID ENTRY MUST BE VALID BILLING ACCOUNT NUMBER OR E WITH TRAILING BLANKS |
| TAGR1460VAL | BAN1 REQUIRED ON REQTYP A WHEN NC IS TY, OTHERWISE NOT EDITED |
| TAGR1470VAL | BI2 REQUIRED WHEN BAN1 AND BAN2 ARE POPULATED |
| TAGR1475VAL | BI2 VALID ENTRIES ARE A, D, L, M, N, P, R, U, V, W OR Z |

| Message ID | Message Text |
|---|---|
| TAGR1480VAL | BAN2 VALID ENTRY MUST BE VALID BILLING ACCOUNT NUMBER OR E WITH TRAILING BLANKS |
| TAGR1485VAL | BAN2 REQUIRED WHEN BI2 IS POPULATED |
| TAGR1488VAL | BAN2 REQUIRED ON REQTYP B WHEN NC IS TY, OTHERWISE NOT EDITED |
| TAGR1490VAL | ACNA REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGR1495VAL | ACNA MUST BE 3 ALPHAS |
| TAGR1500VAL | VTA MUST BE ALPHANUMERICS |
| TAGR1505VAL | INIT REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGR1510VAL | TEL NO-INIT REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGR1515VAL | TEL NO-INIT FORMAT MUST BE 10 NUMERICS OR UP TO 15 ALPHANUMERICS |
| TAGR1520VAL | FAX NO-INIT REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGR1525VAL | FAX NO-INIT MUST BE 10 NUMERICS |
| TAGR1530VAL | IMPCON REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGR1535VAL | TEL NO IMPCON REQUIRED WHEN IMPCON IS POPULATED |
| TAGR1540VAL | TEL NO IMPCON FORMAT MUST BE 10 NUMERICS IN THE FIRST 10 POSITONS |
| TAGR1545VAL | PAGER-IMPCON MUST BE ALPHANUMERICS |
| TAGR1550VAL | TEL NO ALTIMPCON REQUIRED WHEN ALTIMPCON IS POPULATED |
| TAGR1555VAL | TEL NO ALTIMPCON FORMAT MUST BE 10 NUMERICS IN THE FIRST 10 POSITIONS |
| TAGR1560VAL | DSGCON REQUIRED WHEN DRC IS POPULATED |
| TAGR1565VAL | DRC MUST BE 3 ALPHANUMERICS |
| TAGR1570VAL | TEL NO DSGCON REQUIRED WHEN DSGCON IS POPULATED |
| TAGR1575VAL | TEL NO DSGCON MUST BE 10 NUMERICS IN THE FIRST 10 POSITIONS |
| TAGR1580VAL | FAX NO-DSGCON MUST BE 10 NUMERICS |
| TAGR1585VAL | STREET-DSGCON REQUIRED WHEN DSGCON IS POPULATED |
| TAGR1590VAL | CITY-DSGCON REQUIRED WHEN DSGCON IS POPULATED |
| TAGR1595VAL | STATE-DSGCON REQUIRED WHEN DSGCON IS POPULATED |
| TAGR1600VAL | ZIP CODE-DSGCON REQUIRED WHEN DSGCON IS POPULATED |

| Message ID | Message Text |
|---|---|
| TAGR1605VAL | REMARKS VIRGULES (/) AND ASTERISKS (*) NOT ALLOWED IN THIS FIELD |
| TAGR1610VAL | PBT REQUIRED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGR1615VAL | PBT VALID ENTRY IS A, B, OR C |
| TAGR1620VAL | BCS REQUIRED WITH REQTYP/ACT TYPE/TOS COMBINATION |
| TAGR1625VAL | BCS MUST BE 3 OR 5 ALPHANUMERICS |
| TAGR1630VAL | CANNOT SUP A PREVIOUSLY CANCELED LSR/PON |
| TAGR1635VAL | LSR ORIGINATING SOURCE NOT SAME AS PRIOR VERSION |
| TAGR1640VAL | NO ORIGINAL LSR FOUND FOR THIS SUP |
| TAGR1645VAL | LSR/PON AGED OFF |
| TAGR1650VAL | LSR/PON COMPLETED |
| TAGR1655VAL | LSR ORIGINATING FORMAT (TCIF) NOT SAME AS ORIGINATING FORMAT |
| TAGR1660VAL | SUP NOT ALLOWED ON THIS ACCOUNT ACTIVITY TYPE |
| TAGR9000VAL | ONLY AN ACT OF V, P OR Q IS ALLOWED FOR REQTYP B WHEN NPT IS D |
| TAGR9001VAL | HNUM, DNUM, OR LNUM REQUIRED FOR THIS REQUEST |
| TAGR9003VAL | ACT OF SUP LSR IS DIFFERENT THAN ACT ON ORIGINAL LSR |
| TAGR9004VAL | SUP VALID ENTRIES: 1, 2, OR 3 AND EDI ARE 01, 04, OR 05 |
| TAGR9005VAL | INVALID ACT IF NPT IS D |
| TAGR9010VAL | AN OR ATN REQUIRED WHEN LNLN IS POPULATED |
| TAGR9015VAL | ACTIVITY TYPE VALID ENTRIES MUST BE V, P OR Q WHEN AN IS POPULATED |
| TAGR9020VAL | AN IS PROHIBITED WHEN ATN IS POPULATED FOR REQTYP C |
| TAGR9025VAL | APOT MUST BE 11 ALPHANUMERICS IF AI IS Y FOR EDI OR C FOR FAX |
| TAGR9026VAL | ATN OR AN REQUIRED WHEN LNLN IS POPULATED |
| TAGR9028VAL | ATN IS PROHIBITED WHEN AN IS POPULATED ON REQTYP C |
| TAGR9030VAL | BAN1 REQUIRED WITH THE REQTYP/ACT TYPE COMBINATION |
| TAGR9035VAL | BI1 VALID ENTRIES ARE D, L, N FOR LNP |
| TAGR9040VAL | BI2 VALID ENTRIES = D, L, AND N |
| TAGR9045VAL | INVALID CC CODE FOR LNP |

| Message ID | Message Text |
|---|---|
| TAGR9050VAL | CHC OF Y IS REQUIRED WHEN NC IS LY (IMPLEMENT WITH R4.1) |
| TAGR9055VAL | DFDT MUST BE VALID FORMAT OF HHMM OR HHMM-HHMM |
| TAGR9056VAL | WHEN NC IS LY AND CHC IS Y, DFDT IS REQUIRED. (IMPLEMENT WITH R4.1) |
| TAGR9065VAL | DSGCON IS REQUIRED WHEN NC IS LY--, LY |
| TAGR9071VAL | NNSP MUST BE 4 ALPHANUMERICS |
| TAGR9072VAL | INVALID NNSP CODE FOR LNP |
| TAGR9073VAL | SERVICE CENTER MUST BE LCSC, ATLU, ATLC, COMP OR LCSC |
| TAGR9075VAL | TOS COMBINATION INVALID |
| TAGR9076VAL | TOS THIRD CHARACTER MUST BE (HYPHEN) |
| TAGS3000VAL | LOCNUM=&LCNM LNUM=&LNUM CABLE ID REQUIRED FOR SERVICE TYPE |
| TAGS3005VAL | LOCNUM=&LCNM LNUM=&LNUM CABLE ID PROHIBITED FOR SERVICE TYPE |
| TAGS3010VAL | LOCNUM=&LCNM LNUM=&LNUM CABLE ID REQUIRED FOR NCI SELECTION |
| TAGS3013VAL | LOCNUM=&LCNM LNUM=&LNUM CABLE ID PROHIBITED FOR NCI SELECTION |
| TAGS3015VAL | LOCNUM=&LCNM LNUM=&LNUM CABLE ID MUST BE 5 ALPHANUMERICS |
| TAGS3020VAL | LOCNUM=&LCNM LNUM=$LNUM FIRST CHARACTER OF CABLE ID MUST BE P OR V |
| TAGS3025VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM CABLE ID REQUIRED WHEN CHAN/PAIR IS POPULATED FOR REQTYP F |
| TAGS3027VAL | LOCNUM=&LCNM HNUM=&HNUM DIDNUM=&DIDNUM DNUM=&DNUM LNUM=&LNUM ONLY ONE HNUM, DIDNUM, DNUM, LNUM ALLOWED PER RECORD |
| TAGS3030VAL | LOCNUM=&LCNM LNUM=&LNUM CFA PROHIBITED FOR THIS SERVICE TYPE |
| TAGS3035VAL | LOCNUM=&LCNM LNUM=&LNUM CFA REQUIRED FOR THIS SERVICE TYPE |
| TAGS3040VAL | LOCNUM=&LCNM LNUM=&LNUM CFA REQUIRED FOR NCI SELECTION |
| TAGS3045VAL | LOCNUM=&LCNM LNUM=&LNUM CFA PROHIBITED FOR NCI SELECTION |
| TAGS3050VAL | LOCNUM=&LCNM LNUM=&LNUM CFA FORMAT IS INVALID |

| Message ID | Message Text |
|---|---|
| TAGS3055VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM CFA REQUIRED IF CHAN/PAIR AND CABLE ID ARE NOT POPULATED ON REQTYP F |
| TAGS3060VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM CFA PROHIBITED IF CABLE ID OR CHAN/PAIR IS POPULATED ON REQTYP F |
| TAGS3065VAL | LOCNUM=$LCNM LNUM=$LNUM CFTN REQUIRED FOR REQTYP B OR C WHEN NPT IS **B** |
| TAGS3070VAL | LOCNUM=&LCNM LNUM=&LNUM CFTN MUST BE 10 NUMERICS |
| TAGS3075VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM CFTN REQUIRED WHEN NPT IS B AND ACT IS C, V, P OR Q |
| TAGS3080VAL | LOCNUM=&LCNM LNUM=&LNUM CHAN/PAIR REQUIRED FOR SERVICE TYPE |
| TAGS3085VAL | LOCNUM=&LCNM LNUM=&LNUM CHAN/PAIR PROHIBITED FOR SERVICE TYPE |
| TAGS3090VAL | LOCNUM=&LCNM LNUM=&LNUM CHAN/PAIR REQUIRED FOR NCI SELECTION |
| TAGS3095VAL | LOCNUM=&LCNM LNUM=&LNUM CHAN/PAIR PROHIBITED FOR NCI SELECTION |
| TAGS3100VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM CHAN/PAIR REQUIRED WHEN CABLE ID IS POPULATED |
| TAGS3105VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM CHAN/PAIR MUST BE UP TO 5 ALPHANUMERICS |
| TAGS3110VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM CKR FORMAT INVALID |
| TAGS3115VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM ECCKT IS PROHIBITED WITH REQTYP/ACT/LNA COMBINATION |
| TAGS3120VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM ECCKT IS REQUIRED WITH REQTYP/ACT/LNA COMBINATION |
| TAGS3125VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM ECCKT FORMAT INVALID |
| TAGS3130VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM ECCKT REQUIRED WITH A LAN/NC/NCI/SECNCI COMBINATION |
| TAGS3135VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM ECCKT INVALID FORMAT |
| TAGS3140VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM ECCKT REQUIRED WHEN EAN OR LEAN IS POPULATED |

| Message ID | Message Text |
|---|---|
| TAGS3145VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM ECCKT REQUIRED WITH LINE ACTIVITY OF C, L, B, R OR X FOR REQTYP F |
| TAGS3150VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM ECCKT REQUIRED WITH LINE ACTIVITY OF C FOR REQTYP E |
| TAGS3153VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM ECCKT REQUIRED WHEN LNA IS D |
| TAGS3155VAL | LNUM=&LNUM TELNO=&TNUM FA PROHIBITED IF THE LNA IS D, W, P, L, B OR R |
| TAGS3160VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM FA VALID ENTRY MUST BE N, C OR D |
| TAGS3165VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM FA REQUIRED WHEN FEATURE FIELD IS POPULATED |
| TAGS3170VAL | LNUM=$LNUM TELNO=$TNUM FA MUST BE N WHEN LNA IS G OR N |
| TAGS3180VAL | LNUM=&LNUM TELNO=&TNUM FEATURE REQUIRED WHEN FA FIELD IS POPULATED |
| TAGS3190VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM FEATURE MUST BE 3, 5 OR 6 ALPHANUMERICS |
| TAGS3195VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM FEATURE PROHIBITED IF REQTYP IS B OR C AND ACT IS V, P OR Q |
| TAGS3200VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM FEATURE PROHIBITED WITH LINE ACTIVITY OF W, P, L OR B |
| TAGS3205VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM FEATURE DETAIL REQUIRED WHEN FA IS C |
| TAGS3210VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM FEATURE DETAIL PROHIBITED WITH LINE ACTIVITY OF W, P, L OR B |
| TAGS3215VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM ISPID REQUIRED FOR THIS REQTYP/LNA TYPE COMBINATION |
| TAGS3220VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM IWJK MUST BE 5 ALPHANUMERICS |
| TAGS3225VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM IWJK REQUIRED WHEN IWJQ IS POPULATED |
| TAGS3230VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM IWJK PROHIBITED WITH LINE ACTIVITY OF L OR **B** |
| TAGS3235VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM IWJQ MUST BE 2 NUMERICS |
| TAGS3240VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM IWJQ REQUIRED WHEN IWJK IS POPULATED |

| Message ID | Message Text |
|---|---|
| TAGS3245VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM IWJQ REQUIRED WHEN JR IS Y |
| TAGS3250VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM PROHIBITED WITH LINE ACTIVITY OF L OR B |
| TAGS3255VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM JK CODE MUST BE 5 ALPHANUMERICS |
| TAGS3260VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM JK CODE REQUIRED WHEN NIDR IS POPULATED WITH Y |
| TAGS3265VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM JK CODE PROHIBITED WITH LINE ACTIVITY OF L OR B |
| TAGS3270VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM JK NUM MUST BE 2 ALPHANUMERICS |
| TAGS3275VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM JK NUM REQUIRED WHEN JK CODE IS POPULATED |
| TAGS3280VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM JK NUM PROHIBITED WITH LINE ACTIVITY OF L OR B |
| TAGS3285VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM JK POS REQUIRED WHEN JK CODE IS POPULATED |
| TAGS3290VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM JK POS MUST BE TWO NUMERICS |
| TAGS3295VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM JK POS PROHIBITED WITH LINE ACTIVITY OF L OR B |
| TAGS3300VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM JR MUST BE Y OR N |
| TAGS3305VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM JR PROHIBITED WITH LINE ACTIVITY OF L OR B |
| TAGS3325VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LEAN PROHIBITED WHEN FIRST CHARACTER OF TOS IS NOT 1 OR 2 |
| TAGS3330VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LEAN PROHIBITED WHEN SECOND CHARACTER OF TOS IS NOT A OR B |
| TAGS3335VAL | LNUM=&LNUM TELNO=&TNUM LEAN MUST BE 10 OR 13 ALPHANUMERICS |
| TAGS3360VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LEATN PROHIBITED WHEN FIRST CHARACTER OF TOS IS NOT 1 OR 2 |
| TAGS3365VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM PROHIBITED WHEN SECOND CHARACTER OF TOS IS NOT A OR B |
| TAGS3370VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LEATN MUST BE 10 NUMERICS |

| Message ID | Message Text |
|---|---|
| TAGS3380VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNA MUST BE N IF ACT IS N |
| TAGS3385VAL | LNUM=$LNUM TELNO=$TNUM LNA MUST BE D, G, N, V, W, P, X IF ACT IS V, P OR Q |
| TAGS3390VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNA REQUIRED UNLESS ACT IS N, C, T, R, V, S, P OR Q |
| TAGS3395VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM ASSOCIATED DATA PROHIBITED ON ACT TYPE B, L, W OR Y |
| TAGS3400VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNA MUST BE N OR C IF ACT IS T |
| TAGS3405VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNA MUST BE R IF ACT IS R |
| TAGS3410VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNA MUST BE X IF OTN IS POPULATED |
| TAGS3415VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNA MUST BE G, N, C, D, R, X, W, P, L OR **B** |
| TAGS3420VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNA MUST BE N, C, D, P, X IF ACT IS C |
| TAGS3425VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNA MUST BE L OR B IF ACT TYPE IS S |
| TAGS3430VAL | IF ACT IS P, Q OR V, AT LEAST ONE LNA MUST BE G, V, P, X OR W |
| TAGS3433VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNA PROHIBITED ON THIS REQTYP/ACT TYP/SECNCI COMBINATION |
| TAGS3435VAL | LNUM=&LNUM TELNO=&TNUM LNA MUST BE V, N OR D IF ACT IS V, P, OR Q AND REQTYP IS B OR C |
| TAGS3439VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNA MUST BE D ON ACT TYPE OF D WHEN REQTYP IS A WITH SECNCI DATA POPULATED |
| TAGS3440VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNECLSSVC REQUIRED WHEN BCS IS POPULATED |
| TAGS3445VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNECLSSVC MUST BE 3 OR 5 ALPHANUMERICS |
| TAGS3450VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNEX PROHIBITED WITH THIS REQTYP/LNA TYPE COMBINATION |
| TAGS3455VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNEX MUST BE 5 NUMERICS |

| Message ID | Message Text |
|---|---|
| TAGS3460VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNUM REQUIRED WITH THIS REQTYP/LNA TYPE COMBINATION (STOP) |
| TAGS3465VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNUM MUST BE 5 NUMERICS |
| TAGS3470VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNUM MUST BE UNIQUE WITHIN EACH LOCNUM EXCEPT FOR REQTYP E-ISDN BRI |
| TAGS3475VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNUM INVALID FOR ACT TYPE OF D, L, B OR Y (STOP) |
| TAGS3480VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LOCNUM MUST BE 3 NUMERICS |
| TAGS3485VAL | LOCNUM=&LCNM LNUM=&LNUM LOCNUM DOES NOT MATCH AN END USER LOCNUM FOR THIS LSR |
| TAGS3490VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM MATN VALID ENTRIES MUST BE M OR A |
| TAGS3495VAL | NIDR MUST BE Y OR N |
| TAGS3500VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM NIDR PROHIBITED WITH LINE ACTIVITY L OR B |
| TAGS3505VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM NPI VALID ENTRY MUST BE C OR D FOR REQTYP E, F OR M |
| TAGS3510VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM NPI VALID ENTRY MUST BE A FOR REQTYP C |
| TAGS3515VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM NPI PROHIBITED WITH LINE ACTIVITY OF L OR B |
| TAGS3520VAL | NPQTY REQUIRED ON THIS REQTYP/LNA TYPE COMBINATION |
| TAGS3525VAL | NPQTY MUST BE 5 NUMERICS GREATER THAN ZERO |
| TAGS3530VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM NPT REQUIRED WITH THIS REQTYP/LNA TYPE COMBINATION |
| TAGS3535VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM NPT VALID ENTRY IS A, B OR C |
| TAGS3540VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM NPTG REQUIRED ON REQTYP C OR B WHEN LNA IS C OR V AND NPT IS A |
| TAGS3545VAL | LNUM=&LNUM TELNO=&TNUM OTN REQUIRED WHEN LNA IS X FOR REQTYP E, F, AND M |
| TAGS3550VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM OTN MUST BE 10 NUMERICS |
| TAGS3555VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM OTN PROHIBITED WITH LNA OF L OR B |

| Message ID | Message Text |
|---|---|
| TAGS3560VAL | LOCNUM=$LCNM LNUM=$LNUM PORTED NBR REQUIRED WITH REQTYP B OR C WHEN NPT IS A, B, C OR D AND LNA IS V |
| TAGS3565VAL | LOCNUM=&LCNM LNUM=&LNUM PORTED NBR PROHIBITED WITH REQTYP B WHEN NPT IS A, B, C OR D AND LNA IS N |
| TAGS3570VAL | LOCNUM=&LCNM LNUM=&LNUM PORTED NBR MUST BE 10 NUMERICS |
| TAGS3575VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM PORTED NBR REQUIRED WITH THIS REQTYP/LNA TYPE COMBINATION |
| TAGS3580VAL | PQTY REQUIRED WITH THIS REQTYP/LNA TYPE COMBINATION |
| TAGS3585VAL | PQTY MUST BE 3 NUMERICS GREATER THAN ZERO |
| TAGS3590VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM PULSE VALID ENTRY MUST BE DP, MF OR DTMF |
| TAGS3595VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM PULSE PROHIBITED WITH LNA L OR B |
| TAGS3600VAL | RSQTY REQUIRED WITH THIS REQTYP/LNA TYPE COMBINATION |
| TAGS3605VAL | RSQTY MUST BE 3 NUMERICS GREATER THAN ZERO |
| TAGS3610VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM RTI REQUIRED WHEN LNA IS C OR V AND NPT IS A OR C ON REQTYP C |
| TAGS3613VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM RTI REQUIRED ON REQTYP B WHEN LNA IS V AND NPT IS A OR C |
| TAGS3615VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM SDI VALID ENTRY MUST BE E, F, G, H, I, J, K, L OR M |
| TAGS3620VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM SGNL VALID ENTRY MUST BE LP, E1, E2 OR E3 |
| TAGS3625VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM SGNL PROHIBITED WITH LNA L OR B |
| TAGS3630VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM SHELF REQUIRED ON REQTYP F IF LNA IS N, C OR V |
| TAGS3635VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM SHELF MUST BE 6 ALPHANUMERICS |
| TAGS3640VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM SLOT MUST BE 6 ALPHANUMERICS |
| TAGS3645VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM SSIG VALID ENTRY MUST BE LS, WS, IM, GS OR DD |
| TAGS3650VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM SSIG VALID ENTRY OF WS, IM OR DD IS ALLOWED WHEN SGNL IS E1, E2 OR E3 |

| Message ID | Message Text |
|---|---|
| TAGS3655VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM SSIG VALID ENTRY IS LS OR GS WHEN SGNL IS LP |
| TAGS3660VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM SSIG PROHIBITED WITH LINE ACTIVITY OF L OR B |
| TAGS3665VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM SYSTEM ID MUST BE 5 ALPHANUMERICS |
| TAGS3670VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM TERS REQUIRED WHEN TLI POPULATED |
| TAGS3675VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM TLI MUST BE 10 NUMERICS |
| TAGS3680VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM TLI REQUIRED WHEN TERS IS POPULATED |
| TAGS3685VAL | LOCNUM=&LCNM LNUM=&LNUM TNP REQUIRED FOR REQTYP B OR C, NPT IS B AND LNA IS V |
| TAGS3690VAL | LOCNUM=&LCNM LNUM=&LNUM TNP MUST BE 3 NUMERICS GREATER THAN ZERO |
| TAGS3700VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM TNS REQUIRED WITH THIS REQTYP/LNA TYPE COMBINATION |
| TAGS3705VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM TNS MUST BE A MINIMUM OF 10 OR A MAXIMUM OF 15 ALPHANUMERICS INCLUDING HYPHEN EXCEPT WHEN ACT IS P OR Q |
| TAGS3710VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM TNS MUST BE 10 NUMERICS WHEN ACT IS P OR Q |
| TAGS3715VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM WHEN TNS CONTAINS 15 ALPHANUMERICS THE 11TH CHARACTER MUST BE A HYPHEN |
| TAGS3718VAL | LNUM=&LNUM TELNO=&TNUM TNS RANGE NOT ALLOWED WHEN LNA IS X |
| TAGS3720VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM TSP MUST BE 12 ALPHANUMERICS WITH HYPHEN IN 10TH POSITION |
| TAGS3725VAL | LNUM=&LNUM TELNO=&TNUM FPI MUST BE VALID VALUE FOR REQTYP AND ACTIVITY |
| TAGS3730VAL | LNUM=&LNUM TELNO=&TNUM FPI INVALID ON REQTYP/LNA COMBINATION |
| TAGS3735VAL | LNUM=$LNUM TELNO=$TNUM PIC REQUIRED ON LNA N, P, G OR V |
| TAGS3740VAL | LNUM=&LNUM TELNO=&TNUM PIC VALID ENTRIES ARE NONE, UNDC, NC OR A VALID PIC CODE WHEN LNA IS C, P OR X |

| Message ID | Message Text |
|---|---|
| TAGS3745VAL | LNUM=$LNUM TELNO=$TNUM PIC VALID ENTRIES ARE NONE, UNDC OR A VALID PIC CODE WHEN LNA IS G, N OR V |
| TAGS3750VAL | LNUM=&LNUM TELNO=&TNUM PIC INVALID ON REQTYP/LNA COMBINATION |
| TAGS3755VAL | LNUM=$LNUM TELNO=$TNUM LPIC REQUIRED ON LNA N, P, G OR V |
| TAGS3760VAL | LNUM=&LNUM TELNO=&TNUM LPIC VALID ENTRIES ARE NONE, UNDC, NC OR VALID LPIC CODE WHEN LNA IS C P OR X |
| TAGS3765VAL | LNUM=$LNUM TELNO=$TNUM LPIC VALID ENTRIES ARE NONE, UNDC OR A VALID LPIC CODE WHEN LNA IS G, N OR V |
| TAGS3770VAL | LNUM=&LNUM TELNO=&TNUM LPIC INVALID ON REQTYP/LNA COMBINATION |
| TAGS3775VAL | LNUM=&LNUM TELNO=&TNUM PTKTYP REQUIRED WHEN TLI IS POPULATED |
| TAGS3780VAL | LNUM=&LNUM TELNO=&TNUM PTKTYP VALID ENTRY MUST BE Y |
| TAGS3785VAL | LNUM=&LNUM TELNO=&TNUM PTKCON MUST BE T, I, O |
| TAGS3790VAL | LNUM=$LNUM TELNO=$TNUM PTKCON REQUIRED WHEN THE LNA IS N, V OR G |
| TAGS3795VAL | LOCNUM DOES NOT MATCH AN END USER LOCNUM FOR THIS LSR |
| TAGS3800VAL | TELNO=&DTLI DIDNUM REQUIRED |
| TAGS3805VAL | DIDNUM=&DIDNUM TELNO=&DTLI DIDNUM MUST BE 5 ALPHANUMERICS |
| TAGS3807VAL | DIDNUM=$DIDNUM DIDNUM PROHIBITED WHEN REQTYP IS A, B, C, E, F, J OR M |
| TAGS3810VAL | DIDNUM=&DIDNUM TELNO=&DTLI DTNRACT VALID ENTRY MUST BE N, D, A OR R |
| TAGS3815VAL | DIDNUM=$DIDNUM TELNO=$DTLI DTNRACT REQUIRED WHEN DTNR IS POPULATED |
| TAGS3820VAL | DIDNUM=$DIDNUM TELNO=$DTLI DTNRACT A OR R IS PROHIBITED FOR $STAT |
| TAGS3825VAL | DIDNUM=&DIDNUM TELNO=&DTLI DTNRACT PROHIBITED FOR ACT N, D OR T |
| TAGS3830VAL | DIDNUM=&DIDNUM TELNO=&DTLI DTNRQ VALID ENTRY MUST BE IN INCREMENTS OF 20 AND GREATER THAN 0 |

| Message ID | Message Text |
|---|---|
| TAGS3840VAL | DIDNUM=$DIDNUM TELNO=$DTLI DTNR MUST BE 15 NUMERICS WITH A HYPHEN IN THE 11TH POS |
| TAGS3845VAL | DIDNUM=&DIDNUM TELNO=&DTLI DTNR REQUIRED WHEN DTNRACT IS POPULATED |
| TAGS3846VAL | DIDNUM=$DIDNUM TELNO=$DTLI DTNR RANGE INVALID |
| TAGS3850VAL | DIDNUM=$DIDNUM TELNO=$DTLI DTKACT VALID ENTRIES MUST = N, C, D, V OR W |
| TAGS3855VAL | DIDNUM=&DIDNUM TELNO=&DTLI DTKACT REQUIRED WHEN DTK IS POPULATED |
| TAGS3860VAL | DIDNUM=&DIDNUM TELNO=&DTLI ENTRY OF D INVALID, SERVICE REQUEST REQUIRED |
| TAGS3865VAL | DIDNUM=&DIDNUM TELNO=&DTLI DTKACT PROHIBITED WHEN ACT IS N, D, T OR W |
| TAGS3866VAL | AT LEAST ONE DTKACT OF V OR W REQUIRED WHEN ACT IS V |
| TAGS3870VAL | DIDNUM=$DIDNUM TELNO=$DTLI DTK REQUIRED WHEN DTKACT IS POPULATED WITH N, C, D OR V |
| TAGS3871VAL | DIDNUM=$DIDNUM TELNO=$DTLI DTKID MUST BE UP TO 10 ALPHANUMERICS |
| TAGS3872VAL | DIDNUM=$DIDNUM TELNO=$DTLI DTKID REQUIRED WHEN DTKACT IS C, D OR V |
| TAGS3875VAL | DIDNUM=&DIDNUM TELNO=&DTLI DTGN MUST BE MINIMUM OF 3 OR MAXIMUM OF 4 ALPHANUMERICS |
| TAGS3876VAL | DIDNUM=$DIDNUM TELNO=$DTLI DTGN REQUIRED WHEN DTKACT IS D |
| TAGS3880VAL | DIDNUM=&DIDNUM TELNO=&DTLI DRTI MUST BE MINIMUM OF 3 OR MAXIMUM OF 10 ALPHANUMERICS |
| TAGS3881VAL | DIDNUM=$DIDNUM TELNO=$DTLI DRTI REQUIRED WHEN DTKACT IS D |
| TAGS3885VAL | DIDNUM=&DIDNUM TELNO=&DTLI DTLI MUST BE 10 NUMERICS |
| TAGS3890VAL | DIDNUM=$DIDNUM TELNO=$DTLI DTLI REQUIRED WHEN DTKACT IS N, C, D OR V |
| TAGS3895VAL | DIDNUM=&DIDNUM TELNO=&DTLI DGOUT MUST BE 2 NUMERICS |
| TAGS3900VAL | DIDNUM=&DIDNUM TELNO=&DTLI DGOUT REQUIRED WHEN DTKACT IS N |
| TAGS3905VAL | DIDNUM=&DIDNUM TELNO=&DTLI DPULSE VALID ENTRIES ARE DP, MP OR DTMF |
| TAGS3910VAL | DIDNUM=&DIDNUM TELNO=&DTLI DPULSE REQUIRED WHEN DTKACT IS N |

| Message ID | Message Text |
|---|---|
| TAGS3915VAL | DIDNUM=&DIDNUM TELNO=&DTLI DSGNL VALID ENTRY IS DD, IM OR WS |
| TAGS3920VAL | LNUM=&LNUM TELNO=&TNUM BA VALID VALUES ARE A, D, N, Z OR BLANK |
| TAGS3925VAL | LNUM=&LNUM TELNO=&TNUM BA OF D, N, OR Z PROHIBITED ON LNA OF N |
| TAGS3930VAL | LNUM=&LNUM TELNO=&TNUM BA VALID COMBINATIONS ARE A/D OR A/Z ONLY |
| TAGS3935VAL | LNUM=&LNUM TELNO=&TNUM BA PROHIBITED ON REQTYP/LNA COMBINATIONS |
| TAGS3940VAL | LNUM=&LNUM TELNO=&TNUM BLOCK PROHIBITED WHEN BA IS BLANK |
| TAGS3945VAL | LNUM=&LNUM TELNO=&TNUM BLOCK ENTRY OF A, B, OR C ALLOWED ONLY IN FIRST POSITION IN THIS FIELD |
| TAGS3950VAL | LNUM=&LNUM TELNO=&TNUM BLOCK CANNOT BE A COMBINATION OF SAME ENTRIES |
| TAGS3955VAL | LNUM=&LNUM TELNO=&TNUM BLOCK VALID VALUES ARE A, B, C, H OR BLANK ON REQTYP E, F, OR M |
| TAGS3960VAL | LNUM=&LNUM TELNO=&TNUM BLOCK VALID VALUES ARE A, B, C, OR BLANK ON REQTYP B, C, OR N |
| TAGS3963VAL | LNUM=&LNUM TELNO=&TNUM BLOCK IS REQUIRED WITH BA ENTRY OF A OR D |
| TAGS3965VAL | LNUM=&LNUM TELNO=&TNUM BLOCK INVALID WITH BA ENTRY OF N OR Z |
| TAGS3970VAL | LNUM=&LNUM TELNO=&TNUM BLOCK PROHIBITED ON REQTYP/LNA COMBINATIONS |
| TAGS9239VAL | LOCNUM=$LCNM HNUM=$HNUM DNUM=$DNUM LNUM=$LNUM ONLY ONE HNUM, DNUM, LNUM ALLOWED PER RECORD |
| TAGS9240VAL | LOCNUM=$LCNM LNUM=$LNUM-TELNO=$TNUM BA PROHIBITED WHEN NPT IS D |
| TAGS9242VAL | LOCNUM=$LCNM LNUM=$LNUM-TELNO=$TNUM BLOCK PROHIBITED WHEN NPT IS D |
| TAGS9244VAL | LOCNUM=$LCNM-LNUM=$LNUM TELNO=$TNUM CFTN PROHIBITED WHEN NPT = D |
| TAGS9245VAL | FPI PROHIBITED FOR REQTYP B AND C WHEN NPT = D |
| TAGS9247VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM WHEN ACT IS V, P OR Q ONE OF THE FOLLOWING FIELDS IS REQUIRED: EAN, EATN, LEAN, OR LEATN |

| Message ID | Message Text |
|---|---|
| TAGS9250VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM NO MORE THAN 4 UNIQUE LEANS ARE ALLOWED PER LSR |
| TAGS9252VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM PORTED NBR REQUIRED WITH LEAN FOR LNP |
| TAGS9253VAL | LNUM=$LNUM TELNO=$TNUM LEAN IS REQUIRED WHEN THE ACT IS V, P OR Q AND THE LEATN, EAN OR EATN IS NOT POPULATED |
| TAGS9254VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM NO MORE THAN 4 UNIQUE LEANTN ARE ALLOWED PER LSR |
| TAGS9256VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM PORTED NBR REQUIRED WHEN LEATN OR LEAN IS POPULATED |
| TAGS9257VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LEATN REQUIRED WHEN ACT IS V, P OR Q AND NONE OF THE FOLLOWING FIELDS ARE POPULATED: LEAN, EAN OR EATN |
| TAGS9258VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNA MUST BE V IF ACT IS V, P OR Q FOR LNP |
| TAGS9259VAL | LOCNUM=$LOCNUM LNUM=$LNUM TELNO=$TNUM LNA MUST BE V OR N IF ACT IS V, P, OR Q AND REQTYP IS B |
| TAGS9260VAL | LNA MUST BE C IF THE ACT IS C FOR REQTYP C WHEN NPT IS D |
| TAGS9261VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM AT LEAST ONE LNA MUST BE V IF ACT IS V, P OR Q AND REQTYP IS B |
| TAGS9264VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LPIC PROHIBITED WHEN REQTYP IS B OR C AND NPT = D |
| TAGS9266VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM NPT VALID ENTRY IS D |
| TAGS9268VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM NPTG IS PROHIBITED WHEN NPT IS D |
| TAGS9269VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM PORTED NBR PROHIBITED WHEN LNA IS N ON REQTYP B |
| TAGS9272VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM RTI PROHIBITED WHEN NPT = D |
| TAGS9274VAL | DNUM=$DNUM VALID LNUM TC OPT PROHIBITED WITH THIS REQTYP/ACT TYP COMBINATION |
| TAGS9276VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM TNP PROHIBITED WHEN NPT = D |

| Message ID | Message Text |
|---|---|
| TAGS9280VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM CABLE ID REQUIRED WHEN CFA NOT POPULATED |
| TAGS9284VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM CFA REQUIRED WHEN CHAN/PAIR IS NOT POPULATED |
| TAGS9286VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM CFA PROHIBITED WHEN CABLE ID OR CHAN/PAIR ARE POPULATED |
| TAGS9290VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM CABLE ID AND CHAN/PAIR ARE REQUIRED WHEN NC = TY |
| TAGS9292VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM IWJQ PROHIBITED WHEN JR DOES NOT = Y |
| TAGS9298VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM MULTIPLE INSTANCES OF LEAN MUST APPLY TO SAME SERVICE ADDRESS (LOC) |
| TAGS9469VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM LNUM MUST BE UNIQUE WITHIN EACH LOCNUM |
| TAGS9470VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM A RANGE OF NUMBER IS PROHIBITED FOR LNP |
| TAGS9482VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM CABLE ID AND CHAN/PAIR ARE REQUIRED WHEN NC = TY |
| TAGS9700VAL | MANUAL HANDLING REQUIRED FOR LEAN OR EAN ACCOUNTS |
| TAGS9704VAL | ACCOUNT NUMBER FOUND IN THE EATN OR LEATN IS NOT A NUMBER BEING PORTED OUT |
| TAGS0002VAL | COMPANY CODE IS REQUIRED |
| TAGT0024VAL | EITHER THE CITY/COMMUNITY OR THE ZIP CODE IS REQUIRED |
| TAGT0034VAL | STATE IS REQUIRED |
| TAGT0071VAL | HOUSE NUMBER IS REQUIRED |
| TAGT0072VAL | STREET NAME IS REQUIRED |
| TAGT0073VAL | COMMUNITY NAME IS REQUIRED |
| TAGT0074VAL | NETWORK CHANNEL IS REQUIRED |
| TAGT0075VAL | INVALID NETWORK CHANNEL FOR XDSL |
| TAGT0076VAL | NETWORK CHANNEL INTERFACE IS REQUIRED |
| TAGT0077VAL | INVALID NETWORK CHANNEL INTERFACE FOR XDSL |
| TAGT0078VAL | SECONDARY NETWORK CHANNEL INTERFACE IS REQUIRED |
| TAGT0079VAL | INVALID SECONDARY NETWORK CHANNEL INTERFACE FOR XDSL |
| TAGT0080VAL | TELEPHONE NUMBER OR CIRCUIT ID IS REQUIRED |

| Message ID | Message Text |
|---|---|
| TAGT0081VAL | INVALID TELEPHONE NUMBER FORMAT |
| TAGT0082VAL | INVALID CIRCUIT-ID-FORMAT |
| TAGT0083VAL | NUMBER-REQUESTED IS REQUIRED |
| TAGT0084VAL | NUMBER-REQUESTED MUST BE NUMERIC AND GREATER THAN ZERO |
| TAGT0085VAL | FACILITIES RESERVATION NUMBER (RESID) IS REQUIRED |
| TAGT0086VAL | RESID NOT VALID FOR THIS COMPANY CODE |
| TAGT0087VAL | ONLY ONE OF TELEPHONE NUMBER OR CIRCUIT ID MUST BE POPULATED |
| TAGT0090VAL | PURCHASE ORDER NUMBER (PON) IS REQUIRED |
| TAGT0091VAL | PON VALID VALUES ARE ONLY UPPER CASE ALPHA A THRU Z, NUMERIC 0 THRU 9 AND SYMBOLS . , - ' |
| TAGT0095VAL | NUMBER-REQUESTED CANNOT BE GREATER THAN TEN(10) |
| TAGT0200VAL | MAXIMUM FIELD LENGTH EXCEEDED |
| TAGT0205VAL | NUMERIC DATA EXPECTED |
| TAGT0600VAL | REQTYP/ACT COMBINATION INVALID FOR LNP |
| TAGT0605VAL | INVALID NPT VALUE FOR LNP |
| TAGT0610VAL | RESID REQUIRED WITH EACH LSR REQUEST |
| TAGT0615VAL | RESID MUST BE UP TO 20 ALPHANUMERIC CHARACTERS |
| TAGT0620VAL | RESID POPULATION OF ALL X(S) PROHIBITED WITH LNA OF N |
| TAGT0625VAL | CHC/DFDT PROHIBITED WITH THIS SERVICE TYPE |
| TAGT0630VAL | ONLY ONE ADDRESS HEADER IS ALLOWED FOR THIS SERVICE CODE |
| TAGT0635VAL | ACT/LNA COMBINATION NOT VALID FOR THIS ADSL OR HDSL SERVICE TYPE |
| TAGT0640VAL | LOCNUM=$LCNM LNUM=$LNUM CHAN/PAIR2 REQUIRED FOR SERVICE TYPE |
| TAGT0645VAL | LOCNUM=$LCNM LNUM=$LNUM CHAN/PAIR2 MUST BE UP TO 5 ALHPANUMERICS |
| TAGT0650VAL | LOCNUM=$LCNM LNUM=$LNUM CHAN/PAIR2 IS REQUIRED WITH THIS SERVICE TYPE |
| TAGT0660VAL | DFDT PROHIBITED WITH THIS SERVICE TYPE |
| TAGT0665VAL | SUP VALID ENTRIES ARE 01 OR 04 |
| TAGT0670VAL | LOCNUM=$LCNM LNUM=$LNUM TELNO=$TNUM WHEN LNA = G, THEN ALL OTHER LNA'S MUST BE D, G OR N |
| TAGT0680VAL | FA MUST BE N WHEN LNA IS N |
| TAGT0685VAL | LOCNUM=$LCNM LNUM=$LNUM CHAN/PAIR2 PROHIBITED FOR SERVICE TYPE |
| TAGT0695VAL | LOCNUM=$LCNM LNUM=$LNUM RELAY RACK MUST BE 8AN |

| Message ID | Message Text |
|---|---|
| | BE 8AN |
| TAGT0700VAL | LOCNUM=$LCNM LNUM=$LNUM RELAY RACK REQUIRED WITH THIS REQTYP/LNA |
| TAGT0705VAL | LOCNUM=$LCNM LNUM=$LNUM SHELF MUST BE 2N |
| TAGT0710VAL | LOCNUM=$LCNM LNUM=$LNUM SHELF REQUIRED WITH THIS REQTYP/LNA |
| TAGT0715VAL | LOCNUM=$LCNM LNUM=$LNUM SLOT MUST BE 3N ONLY (REPRESENTS SLOT AND LINE) |
| TAGT0720VAL | LOCNUM=$LCNM LNUM=$LNUM SLOT REQUIRED WITH THIS REQTYP/LNA |
| TAGT0725VAL | CCNA IS REQUIRED FOR LINE SHARING |
| TAGT0730VAL | PON IS REQUIRED FOR LINE SHARING |
| TAGT0735VAL | DDD IS REQUIRED FOR LINE SHARING |
| TAGT0740VAL | ACTL IS REQUIRED FOR LINE SHARING |
| TAGT0745VAL | LSO IS REQUIRED FOR LINE SHARING |
| TAGT0750VAL | BAN1 IS REQUIRED FOR LINE SHARING |
| TAGT0755VAL | ACNA IS REQUIRED FOR LINE SHARING |
| TAGT0760VAL | LOCNUM=$LCNM LNUM=$LNUM SLTN REQUIRED WITH THIS REQTYP/LNA |
| TAGT0765VAL | LOCNUM=$LCNM LNUM=$LNUM SLTN MUST BE 12 AN (WITH 2 HYPHENS) |
| TAGT0770VAL | AN IS REQUIRED FOR LINE SHARING |
| TAGT0785VAL | SC IS REQUIRED FOR LINE SHARING |
| TAGT0790VAL | D/TSENT IS REQUIRED FOR LINE SHARING |
| TAGT0800VAL | LOCNUM=$LCNM LNUM=$LNUM CHAN/PAIR MUST BE 4AN |
| TAGT8000VAL | DNUM=&DNUM TC OPT VALID ENTRY IS ST, NO OR TC |
| TAGT8005VAL | DNUM=&DNUM TC OPT PROHIBITED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGT8010VAL | DNUM=&DNUM TC OPT PROHIBITED WHEN DNUM AND DISC NBR IS NOT POPULATED ON ACT TYPE D |
| TAGT8015VAL | DNUN=&DNUM TC OPT PROHIBITED WHEN ATN IS NOT POPULATED ON ACT TYPE L |
| TAGT8020VAL | DNUM=&DNUM TC OPT OF ST PROHBITIED FOR ACT TYPE L |
| TAGT8025VAL | DNUM=&DNUM TC TO PRIMARY IS REQUIRED WHEN DNUM TC OPT IS TC OR ST |
| TAGT8030VAL | DNUM=&DNUM VALID DNUM TC TO PRIMARY ENTRY MUST BE 10 NUMERICS |
| TAGT8035VAL | DNUM=&DNUM TC TO PRIMARY PROHIBITED WHEN DNUM TC OPT IS NOT TC OR ST |
| TAGT8040VAL | DNUM=&DNUM TC TO PRIMARY CANNOT BE THE SAME AS THE NUMBER BEING REFERRED |

| Message ID | Message Text |
|---|---|
| | SAME AS THE NUMBER BEING REFERRED |
| TAGT8043VAL | DNUM=&DNUM TC TO PRIMARY PROHIBITED WITH THIS REQTYP/ACTTYP COMBINATION |
| TAGT8045VAL | DNUM=&DNUM TC TO SECONDARY REQUIRED WHEN DNUM TC OPT IS ST |
| TAGT8050VAL | DNUM=&DNUM TC TO SECONDARY MUST BE 10 NUMERICS |
| TAGT8055VAL | DNUM=&DNUM TC TO SECONDARY PROHIBITED WHEN DNUM TC OPT IS NOT ST |
| TAGT8060VAL | DNUM=&DNUM TC TO SECONDARY CANNOT BE THE SAME AS THE NUMBER BEING REFERRED |
| TAGT8063VAL | DNUM=&DNUM TC TO SECONDARY PROHIBITED WITH THIS REQTYP/ACTTYP COMBINATION |
| TAGT8065VAL | DNUM=&DNUM TCID VALID ENTRY IS NUMERIC 01 OR 02 |
| TAGT8070VAL | DNUM=&DNUM TCID IS PROHIBITED WHEN DNUM TC OPT DATA IS NOT ST |
| TAGT8075VAL | DNUM=&DNUM TCID 01 AND TCID 02 ARE REQUIRED WHEN DNUM TC OPT IS ST |
| TAGT8080VAL | DNUM=&DNUM TCID (01) AND TCID (02) CANNOT CONTAIN THE SAME VALUE |
| TAGT8083VAL | DNUM=&DNUM TCID PROHIBITED WITH THIS REQTYP/ACTTYP COMBINATION |
| TAGT8085VAL | DNUM=&DNUM BOTH TC NAME (01 AND 02) ARE REQUIRED WHEN DNUM TC OPT IS ST |
| TAGT8090VAL | DNUM=&DNUM TC NAME IS PROHIBITED WHEN DNUM TC OPT IS NOT ST |
| TAGT8095VAL | DNUM=&DNUM TC NAME INVALID |
| TAGT8098VAL | DNUM=&DNUM TC NAME PROHIBITED WITH THIS REQTYP/ACTTYP COMBINATION |
| TAGT8100VAL | DNUM=&DNUM TC PER FORMAT MUST BE CCYYMMDD |
| TAGT8105VAL | DNUM=&DNUM TC PER PROHIBITED WHEN DNUM TC OPT IS NOT ST OR TC |
| TAGT8110VAL | DNUM=&DNUM TC PER DATE INVALID MUST BE LATER THAN THE LSR RECEIPT DATE |
| TAGT8115VAL | LNUM=&LNUM TC OPT PROHIBITED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGT8120VAL | LNUM=&LNUM TC OPT VALID ENTRY IS ST, NO, CA OR TC |
| TAGT8125VAL | LNUM=$LNUM TC OPT OF CA IS INVALID WHEN LNA IS ANYTHING BUT C, G, N OR V |
| TAGT8130VAL | LNUM=&LNUM TC OPT PROHIBITED WITH THIS REQTYP/LNA COMBINATION |

| Message ID | Message Text |
|---|---|
| TAGT8135VAL | LNUM=&LNUM TC OPT OF CA PROHIBITED FOR REQTYP A, B OR C |
| TAGT8140VAL | LNUM=$LNUM TC OPT PROHIBITED IF TC FR IS NOT POPULATED ON REQTYP E, F OR M FOR LNA C, G, N OR V |
| TAGT8145VAL | LNUM=&LNUM TC OPT PROHIBITED IF OTN IS NOT POPULATED ON REQTYP E, F OR M FOR LNA X |
| TAGT8150VAL | LNUM=&LNUM TC OPT PROHIBITED IF LNUM ECCKT IS NOT POPULATED ON REQTYP B OR C |
| TAGT8155VAL | LNUM=&LNUM TC OPT PROHIBITED IF LNUM DISC NBR IS NOT POPULATED ON REQTYP A FOR LNA V |
| TAGT8160VAL | LNUM=&LNUM TC OPT PROHIBITED IF TNS IS NOT POPULATED ON REQTYP E, F OR M FOR LNA D OR L |
| TAGT8165VAL | LNUM=&LNUM TC TO PRIMARY IS REQUIRED WHEN LNUM TC OPT IS TC OR ST |
| TAGT8170VAL | LNUM=&LNUM VALID LNUM TC TO PRIMARY ENTRY MUST CONTAIN 10 NUMERICS |
| TAGT8175VAL | LNUM=&LNUM TC TO PRIMARY PROHIBITED WHEN LNUM TC OPT IS NOT TC OR ST |
| TAGT8180VAL | LNUM=&LNUM TC TO PRIMARY NUMBER MUST BE DIFFERENT FROM NUMBER BEING REFERRED |
| TAGT8183VAL | LNUM=&LNUM TC TO PRIMARY PROHIBITED WITH THIS REQTYP/LNA COMBINATION |
| TAGT8185VAL | LNUM=&LNUM TC TO SECONDARY REQUIRED WHEN LNUM TC OPT IS ST |
| TAGT8190VAL | LNUM=&LNUM TC TO SECONDARY MUST BE 10 NUMERICS |
| TAGT8195VAL | LNUM=&LNUM TC TO SECONDARY PROHIBITED WHEN LNUM TC OPT IS NOT ST |
| TAGT8200VAL | LNUM=&LNUM TC TO SECONDARY NUMBER MUST BE DIFFERENT FROM NUMBER BEING REFERRED. |
| TAGT8203VAL | LNUM=&LNUM TC TO SECONDARY PROHIBITED WITH THIS REQTYP/LNA COMBINATION |
| TAGT8205VAL | LNUM=&LNUM TC PER FORMAT MUST BE CCYYMMDD |
| TAGT8210VAL | LNUM=&LNUM TC PER PROHIBITED WHEN LNUM TC OPT IS NOT ST OR TC |
| TAGT8215VAL | LNUM=&LNUM TC PER DATE INVALID MUST BE LATER THAN THE LSR RECEIPT DATE |
| TAGT8220VAL | LNUM=&LNUM TCID VALID ENTRY IS NUMERIC 01 OR 02 |
| TAGT8225VAL | LNUM=&LNUM TCID (01 OR 02) REQUIRED WHEN LNUM TC OPT IS ST |
| TAGT8230VAL | LNUM=&LNUM TCID (01 OR 02) ARE PROHIBITED WHEN LNUM TC OPT DATA IS NOT ST |

| Message ID | Message Text |
|---|---|
| TAGT8235VAL | LNUM=&LNUM TCID (01) AND TCID (02) CANNOT CONTAIN THE SAME VALUE |
| TAGT8238VAL | LNUM=&LNUM TCID PROHIBITED WITH THIS REQTYP/LNA COMBINATION |
| TAGT8240VAL | LNUM=&LNUM BOTH TC NAME (01 AND 02) ARE REQUIRED WHEN LNUM TC OPT IS ST |
| TAGT8245VAL | LNUM=&LNUM BOTH TC NAME (01 OR 02) ARE PROHIBITED WHEN LNUM TC OPT IS NOT ST |
| TAGT8250VAL | LNUM=&LNUM TC NAME INVALID |
| TAGT8253VAL | LNUM=&LNUM TC NAME PROHIBITED WITH THIS REQTYP/LNA COMBINATION |
| TAGT8255VAL | LNUM=$LNUM TC FR IS REQUIRED WHEN LNUM TC OPT IS POPULATED AND LNA IS C, G, N OR V AND REQTYP IS E, F OR M |
| TAGT8260VAL | LNUM=&LNUM TC FR MUST BE 10 NUMERICS |
| TAGT8265VAL | LNUM=&LNUM TC FR IS PROHIBITED WITH REQTYP/LNA COMBINATION |
| TAGU6000VAL | NC CODE REQUIRED |
| TAGU6005VAL | NC CODE INVALID |
| TAGU6010VAL | NCI REQUIRED FOR NC |
| TAGU6020VAL | NCI PROHIBITED WITH NC |
| TAGU6025VAL | NCI MUST BE 5 TO 12 ALPHANUMERICS OR PERIODS |
| TAGU6030VAL | SECNCI REQUIRED FOR NC |
| TAGU6035VAL | SECNCI PROHIBITED WITH NC |
| TAGU6040VAL | SECNCI MUST BE 5 TO 12 ALPHANUMERICS OR PERIODS |
| TAGU6045VAL | INVALID NC/NCI/SECNCI COMBINATION (STOP) |
| TAGU6050VAL | REQTYP/LOOP TYPE COMBINATION INVALID |
| TAGU6055VAL | LQTY IS REQUIRED FOR REQTYP/ACT COMBINATION |
| TAGU6060VAL | LQTY MUST BE 3 NUMERICS |
| TAGX0002VAL | FORM ($FORM) INVALID FOR REQTYP ($REQTYP) |
| TAGX0003VAL | INVALID REQUEST TYPE FOR COG (CORPORATE GATEWAY) |
| TAGX0004VAL | DIRQTYNC IS PROHIBITED WHEN DIRTYP IS NOT GIVEN |
| TAGX0005VAL | DNUM=$DNUM TC PER PROHIBITED WITH THIS REQTYP/ACT TYPE COMBINATION |
| TAGX0007VAL | REQTYP FIELD MUST EQUAL $REQTYP |
| TAGZ0210VAL | PROCESSING STATUS MUST BE '1', 'F', 'P' OR 'Q' |
| TAGZ0215VAL | FROM DATE REQUIRED |

| Message ID | Message Text |
|---|---|
| TAGZ0220VAL | FROM DATE MUST BE A VALID DATE |
| TAGZ0225VAL | TO DATE IS REQUIRED |
| TAGZ0230VAL | TO DATE MUST BE A VALID DATE |
| TAGZ0235VAL | TO DATE MUST NOT BE GREATER THAN 7 DAYS BEYOND THE FROM DATE |
| TAGZ1170VAL | CC REQUIRED |

## 8.4 TAG Errors

TAG Errors are returned to the end-user due to system, hardware, and/or OSS issues on the BellSouth side of the gateway. Error codes beginning with the prefix "TGW" or "BLP" should be reported to the TAG Gateway System Administrator for resolution. Error Messages with the prefix of "TAGT" are related to problems with Due Date Calculations.

# 9. Appendix D - Pre-Installation Checklist

**UNIX**

- Plan the Environment
- Create volume groups, logical volumes, raw partitions, and file systems
- Set up the File System Layout "Full Duplex is recommended
- Install ORBIX
- Set up TAG User Ids (accounts and password)
  a) TAG User ID= xst_adm; Group ID= xst_grp
  b) Baist User ID; Group ID=xst_grp
- Obtain 3rd Party Licenses
- Set Kernel Parameters
- Get CLEC Information:
  a) CLEC Names
  b) Application IDs
  c) CLEC Notification Server Names
  d) Encryption keys
  e) Host Name

**WINDOWS/NT**

- Plan the Environment
- Get 3rd partry Licenses
- Install ORBIX
- Set up TAG User Ids (accounts and password)
- Get CLEC Information:
  a) CLEC Names
  b) Application IDs
  c) CLEC Notification Server Names
  d) Encryption keys
  e) Host Name

# 10. Appendix E - TAG Client API UNIX Installation

## 10.1 Installing TAG Client API on UNIX

This section describes the procedures used to install TAG Client Server software and configure the TAG Client.

### 10.1.1 Installation Overview

Installing the TAG Client API on the UNIX involves the following major tasks.

1. **Staging.** Move the archive to a permanent location on a file system.
2. **Loading.** Create directories, extract the software from the archive, and place the software into a file system.
3. **Accessing the Menu.** Move through the menu system to access the `Release Functions` menu.
4. **Configuring the TAG Client.** Load the TAG Client API on the client machine(s).

### 10.1.2 Stop the Installation Script

To shut down or stop the installation script.

Stop the installation script and return to the UNIX prompt, from the `Archived Products List`, select option **E** for (Exit).

### 10.1.3 Restart the Installation Script

To restart the installation script, first determine whether you stopped the script before the archive was loaded or after.

If you stopped the script before the archive was successfully loaded, you must begin the installation script from the beginning.

If you stopped the script after the archive was loaded, you can restart the script at the point where you stopped. Then follow the steps below.

1. Complete Phase 1: Staging step 2. (Log in as the TAG Administrator)
2. Complete Phase 1: Staging step 3. (Start the Script)
3. Type the following command:

   **~baist /tools/bcr.flow**

   The `Installed Product List` displays. Continue the installation at the point where you stopped.

### 10.1.4 Task 1: Staging

During Staging, move the archive to a permanent location on a file system. The steps that you complete will depend on how to receive the software.

If you receive the TAG Client API software on a CD-ROM, install the software directly from the CD-ROM. By installing directly from the CD-ROM, you save a large amount of disk space. This disk space would be necessary if the software where copied to a staging area.

For *CD-ROM installation*, complete all the steps noted below.

For TAG software received via the *ftp* process, you will also need to complete the following tasks:

- Create an archive directory called *TAG/Archives* under the ownership of *xst_adm*.
- Copy the installation files to this archive directory.
- Complete only the following steps: 2, 3, and 7.

#### 10.1.4.1 Step 1: Mount the CD-ROM Drive

If you've already mounted the CD-ROM drive on this processor, continue with step 2.

If not, then proceed with the following steps to mount the CD-ROM drive on the processor. First log in as **root** and use the **mount** command to mount the CD-ROM drive (the directory */cdrom* is assumed in this guide). When this task is complete, log out from **root**.

login: **root**
Password: ***password***

**mount** ***device name*** **/cdrom**

#### 10.1.4.2 Step 2: Create TAG Root Directory

Create the root of the TAG directory structure, *tag_root*, with owner xst_adm, group xst_grp, and 755 permissions.

```
mkdir -m 755 tag_root
chown xst_adm:xst_grp tag_root
```

**exit**

#### 10.1.4.3 Step 3: Log in as the TAG Administrator

Log in as the TAG administrator, xst_adm, if you have not done so already.

login: **xst_adm**
Password: ***password***

10.1.4.4 Step 4: Start a Typescript of the Terminal Session

It will be helpful to have a script of the installation and configuration process. This script can provide a complete record of the entries you make and the responses you receive.

To start a typescript, type the commands below:

**cd *tag_root***
**script *script filename***

10.1.4.5 Step 5: Insert the CD-ROM into the Drive

Insert the CD-ROM disk into the CD-ROM drive.

10.1.4.6 Step 6: Change Directory

Change directory to the *cdrom* directory, using the following command:

**cd /cdrom**

10.1.4.7 Step 7: Locate the Archive Name

List the contents of the CD-ROM to locate the archive name of the TAG software, using the following command:

**ls -l**

The archive name has the following format:

```
taprelease[_client].sid.[risc].arch.date
```

*"tap"* is a constant value indicating the product name. (Note: Currently TAP is used rather than TAG for archive naming.)

*release* is the release number in the form *x_x*. For example, the release number for the first release of the product is 3.1.

*client* is an optional value indicating that the archive contains software for a particular client platform only. For example, the archive containing the HP client software uses the value HP_CLIENT in this field.

*sid* is the SID level of the build.

*risc* is an optional value indicating the RISC level (1.x or 2.0) for HP machines.

*"arch"* is a constant value.

*date* is the date of the archive in the form YYMMDD.

Shown here are examples of archives containing the TAG client and software for HP and SUN machine.

HP machine running RISC level 2.0:

```
tap1_0.1.482.R2.arch.980609
```

SUN client machine:

```
tap1_0_SUN_CLIENT.1.482.arch.980609
```

To determine the RISC level of an HP machine, follow these steps:

1. Determine the HP model name by using the **model** command:

```
model
```

The second component of the string returned by **model** is the model name, for example: In "9000/889/K460" K460 is the model name.

2. Use the UNIX command **grep** to search the file */usr/lib/sched.models* for a line containing the model name:

```
grep 889 /usr/lib/sched.models
```

The **grep** output contains the RISC level, for example: In "889 2.0 PA8000", the middle value is the RISC level 2.0.

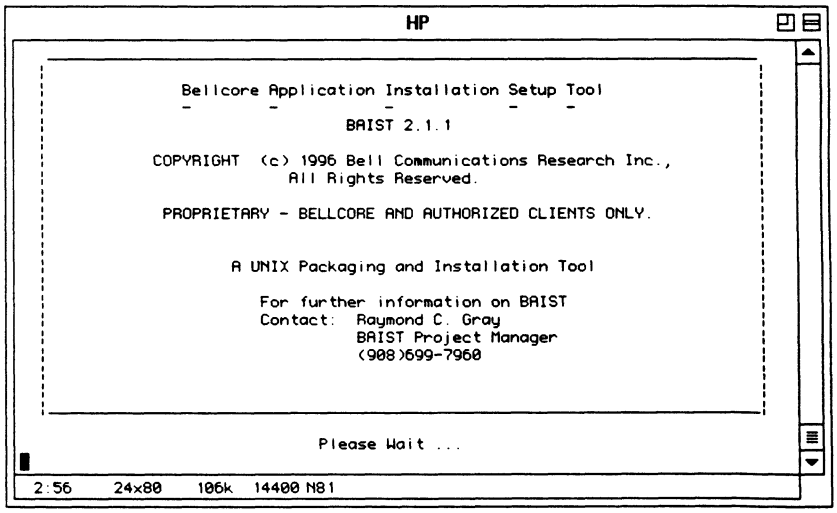
**Note:** If you cannot find the Model name in ***/usr/lib/sched.models*** please contact HP for your RISC level.

10.1.4.8 Step 8: Start the Installation Script

Start the installation script by typing the command line shown below:

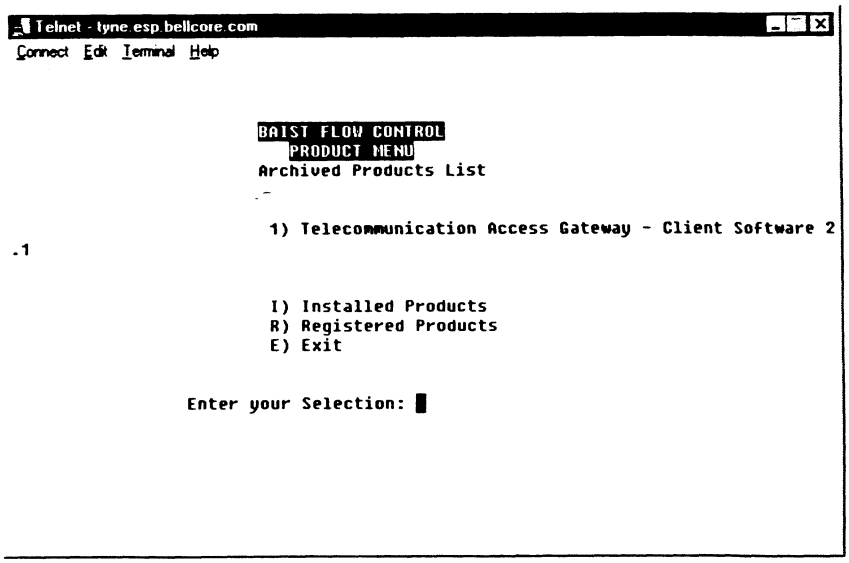
**/bin/sh** ***archive_name***

The installation script begins initializing. A "splash" screen displays while the system initializes. This process takes a few minutes.




When initialization is complete, the `Archived Products List` menu is displayed below. Now you are ready to begin the installation process.

**Note**: The screens shown here may appear slightly different from your installation screen. (i.ethe release number may appear different although the actual screen has not changed.)



### 10.1.5 Task 2: Load

When loading the TAG Client API software, follow the step for creating the directories, extract the software, and place the software into a file system.

#### 10.1.5.1 Step 1: Load TAG Client API

From the `Archived Products List` menu, type **1**.

At the following prompt select **[Enter]** to confirm that you want to load the TAG Client API software:

```
Load Telecommunication Access Gateway - Client Software
7.5.0.10[Y]: [Enter]
```

#### 10.1.5.2 Step 2: Enter the TAG Target Directory

Type the target directory for TAG (*tag_root*).

```
Extracting Application Profile
Where should Telecommunication Access Gateway-7.5.0.10 be
installed?
Enter a full pathname starting with /, or q to Quit processing:
tag_root
```

The installation script confirms the directory in the following message:

```
Running NON_ROOT script for PRELOAD
TAG Release Home is tag_root
Who is the owner of TAG Client application? [xst_adm] :
```

#### 10.1.5.3 Step 3: Enter the TAG Owner

Select **[Enter]** to accept the default owner, the TAG Administrator, xst_adm.

```
Who is the owner of Telecommunication Access Gateway application?
[xst_adm]: [Enter]
```

#### 10.1.5.4 Step 4: Enter the TAG Release

Then select **[Enter]** to accept the default release, 7.5.0.10

```
Enter the Application Release? [7.5.0.10]: [Enter]
```

#### 10.1.5.5 Step 5: Enter Vendor Software Information

At the next prompt, type the vendor software information for the ORBIX products, or select **[Enter]** to accept the default values.

```
Enter the Orbix Home directory? [/opt/Orbix_3.0.1] :
```

After you have entered the vendor software information, the installation script unpacks the TAG archive. Expect this step to take a few minutes. The script displays and updates the following progress message. Unpacking is complete when the indicator reaches 100%.

```
Extracting the Telecommunication Access Gateway Client Software
Please wait.  This may take a while ...
195834 of 195834 blocks loaded 100% COMPLETED
```

10.1.5.6 Step 6: Identify TAG Client

Finally, you must give the TAG software an identity. The script displays the following prompt. Press **[Enter]** to accept the default value.

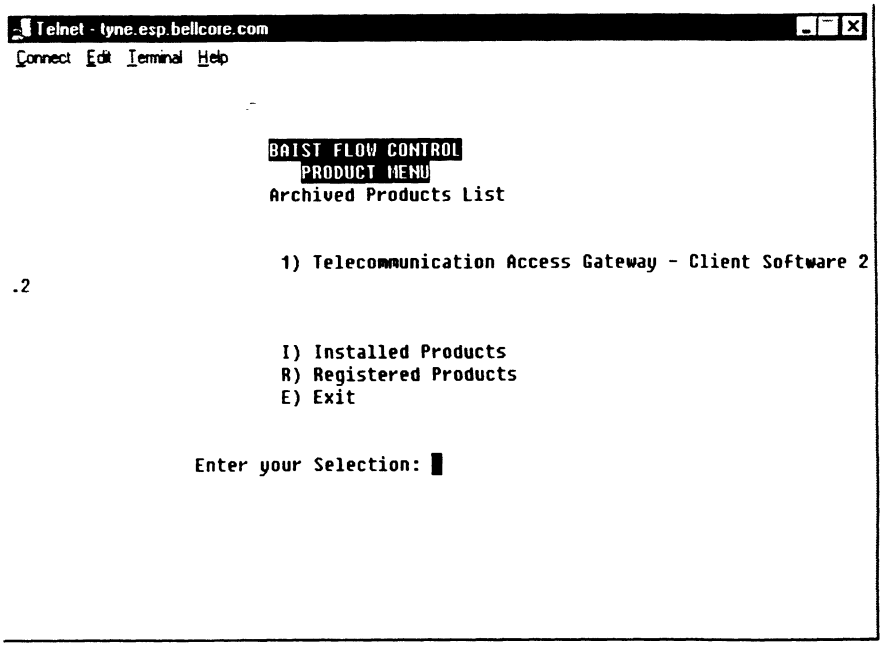
```
Enter a name to identify this copy of Telecommunication Access
Gateway 7.5.0.10 Client Software [Default TAGclient_7.5.0.10]:
[Enter]
```

The script responds with the following messages indicating that TAG is successfully installed.

```
Using TAGCLIENT_ 7.5.0.10 as Release Name
Running NON_ROOT script for POSTLOAD
Creating PV files ...
load successful )loaded into /export/tag/xsta/ 7.5.0.10/config/pv)
Creating PV files completed
saving mzuckerm's .profile to .profile.00000
The release TAGCLIENT_ 7.5.0.10 loaded successfully
```

### 10.1.6 Task 3: Access the Release Functions Menu

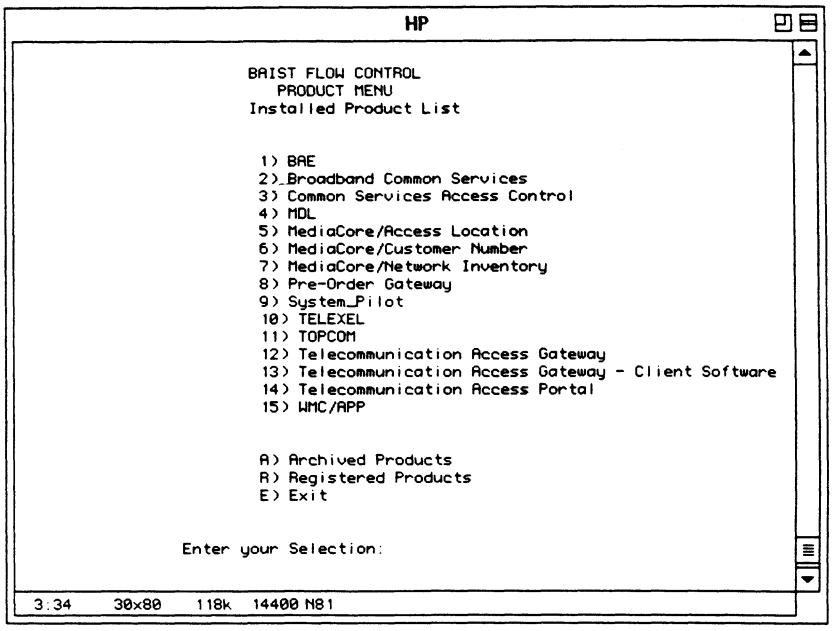
To configure the TAG Client API, the Archived Product List will guide you through the menu path to the `Release Functions Menu`.




From the `Archived Products List` menu, type **I** to select Installed Products.

10.1.6.1 Step 1: Begin the Configuration

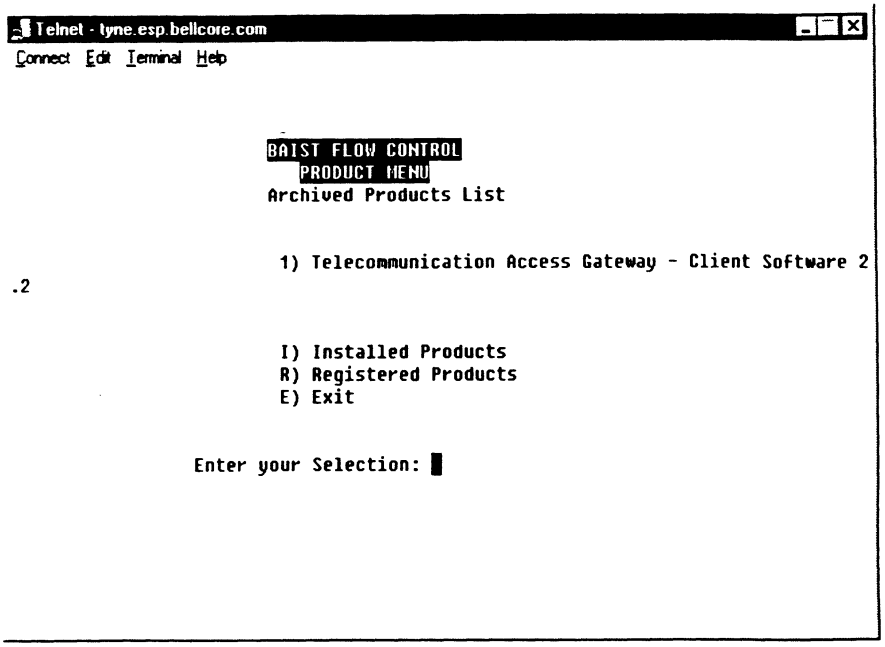
The `Installed Products List` menu appears. (Your screen will appear slightly different from the illustration below.)




From the `Installed Products List` menu, select the menu choice for Telecommunications Access Gateway - Client Software.

### 10.1.6.2 Step 2: Select TAG – Client Software

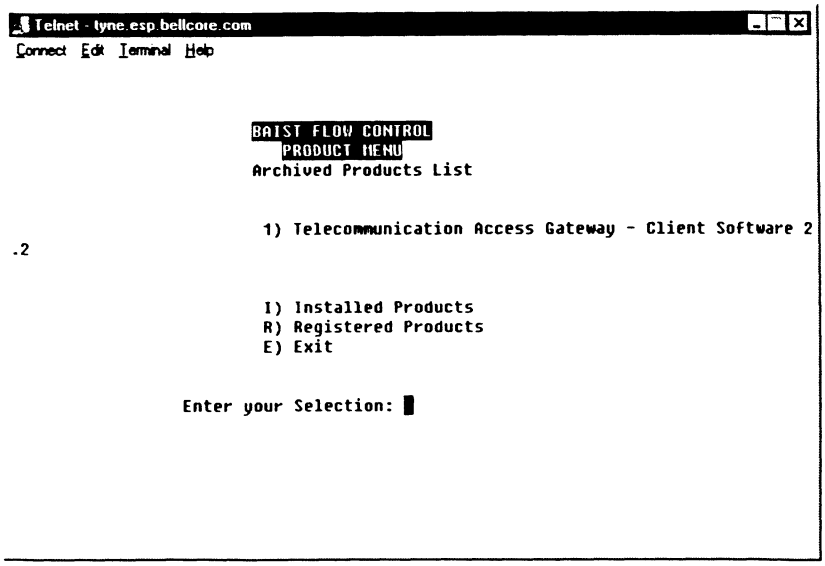
Then the RELEASE MENU appears. (Your screen will appear slightly different from the illustration below.)




From the RELEASE MENU, select the menu choice for TAGCLIENT_X.X.

10.1.6.3 Step 3: Select TAG Client 7.5.0.10

The PRODUCT List menu appears. (Your screen will appear slightly different from the illustration below.)




From the Archived product list menu, type **1** to begin the configuration.

10.1.6.4 Step 4: Configure Client for Release 3.1

The following prompts display. press **[Enter]** to accept the default values. Select **[Exit]** to return to the RELEASE FUNCTIONS menu.

```
Running NON_ROOT script for REL_CONFIG
Enter CLEC ID [clec_id] :
Enter TAG Security Server Name [TagSecurityServer] :
Enter TAG Security Server Host Name [tyne.esp.bellsouth.com] :
Enter CLEC Notification Server Name [TagCLECNotificationServer] :

Creating /export/tag/xsta/7.5.0.10/config/xst_Client file ...
Creating /export/tag/xsta/7.5.0.10/config/xst_ClecNotification
file ...
Creating /export/tag/xsta/7.5.0.10/bin/xst_env file ...
Creating /export/tag/xsta/7.5.0.10/bin/xsta_start file ...
Creating /export/tag/xsta/7.5.0.10/bin/xsta_stop file ...
Press return to continue...
```

10.1.6.5 Step 5: Exit the Client Configuration

To exit the client configuration, from the RELEASE FUNCTIONS menu, type **e**.

# 11. Appendix F - TAG Client API Windows Installation

## 11.1 Overview

This chapter describes steps needed to install the TAG Client software on a PC running Windows NT or Windows 95. This installation loads the TAG Client API and related sample client programs.

## 11.2 Install

InstallShield® is used to perform the TAG Client software installation. Once InstallShield loads the TAG 7.5.0.10 setup program, you are guided through a series of steps. Follow the instructions on each window InstallShield displays.

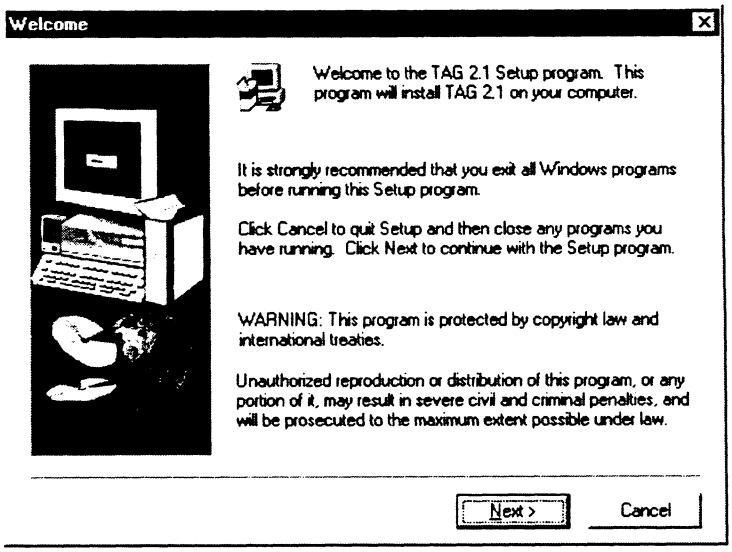
### 11.2.1 Step 1: Locate Setup

Insert the TAG 7.5.0.10 installation CD into the CD-ROM drive of your PC. List the contents of the CD and locate the file named *SETUP.EXE*.

### 11.2.2 Step 2: Load Setup

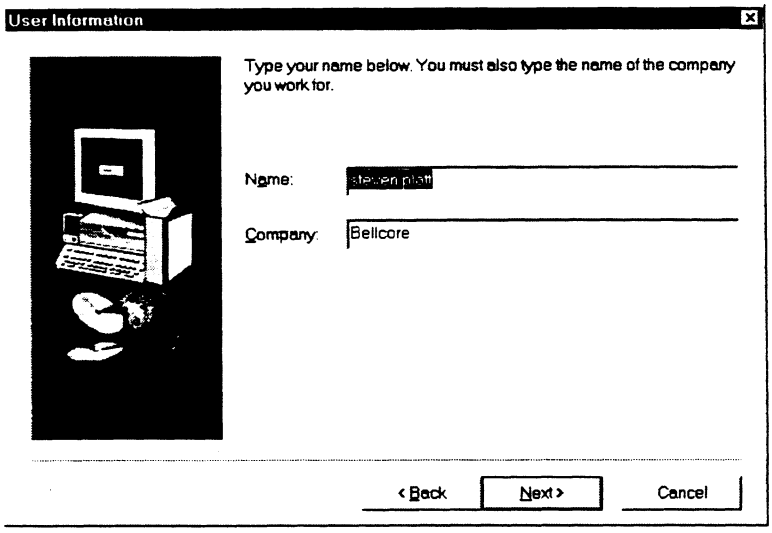
Run *SETUP.EXE* and InstallShield will load the TAG 7.5.0.10 setup program. When the load is complete the `Welcome` window displays. You are now ready to begin the installation process.

**NOTE**: Your screen will appear slightly different from the screen below. (i.e The release number may appear different although the actual screen has not changed.)



### 11.2.3 Step 3: Begin Installation

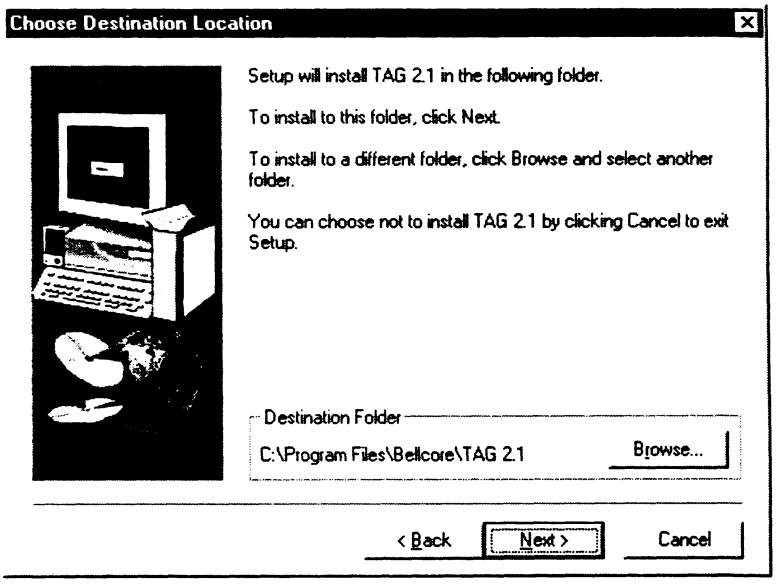
Follow the instructions in the `Welcome` window, then press `Next`. Then the `User Information` window is displayed:




Enter your name and company name, click `Next` to continue.

### 11.2.4 Step 4: Choose Destination

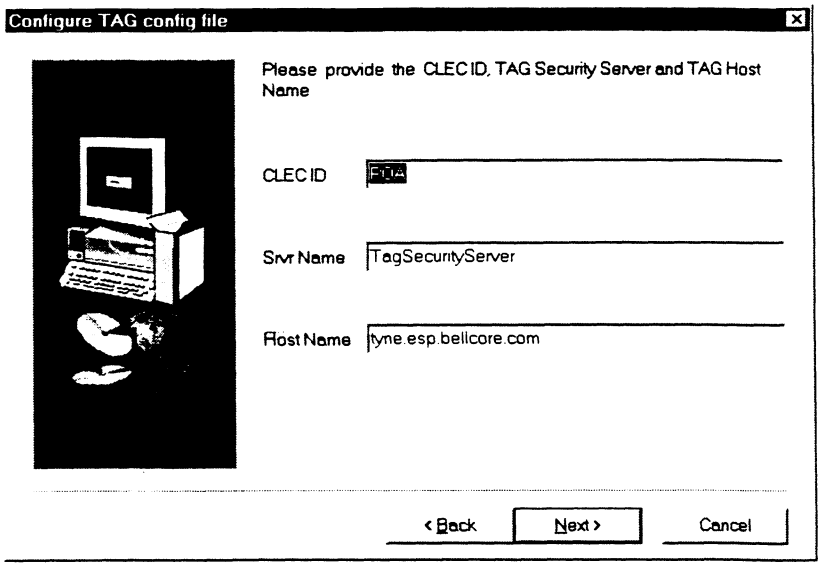
The `Choose Destination Location` window displays:




Follow the instructions in the `Choose Destination Location` window. This window will help you select the folder where TAG will be installed. Press `Next` to continue.

### 11.2.5 Step 5: Configure TAG Config File

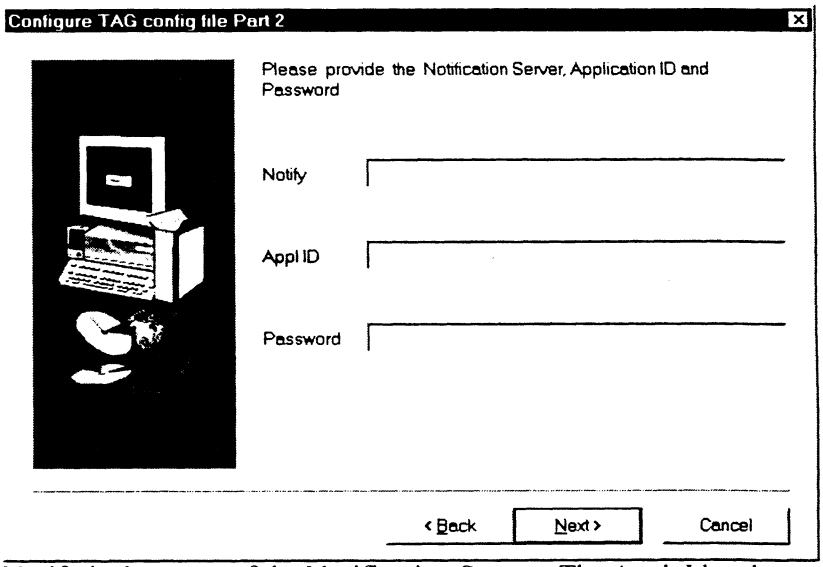
The `Configure TAG config` file window displays:




Enter CLEC ID, TAG Gateway server name, and TAG Gateway host name. Press `Next` to continue.

### 11.2.6 Step 6: Configure TAG Config File Part 2

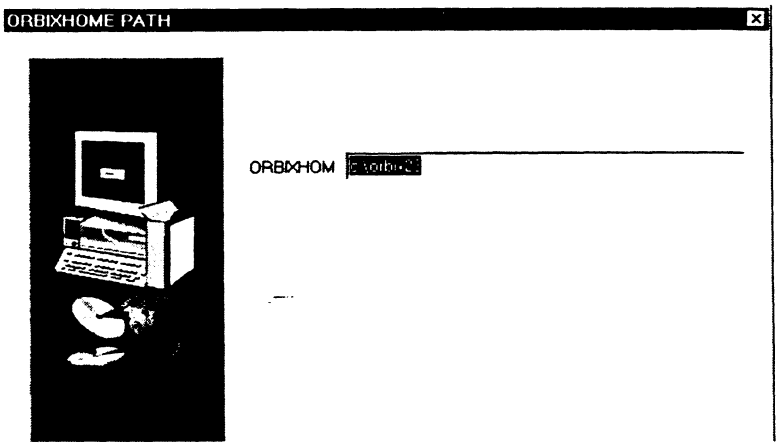
The `Configure TAG config file` Part 2 window displays:




Notify is the name of the Notification Server. The Appl_Id and passwords are the application ID used for notification server to connect to the TAG security server

11.2.7 Step 7: Setup ORBIX

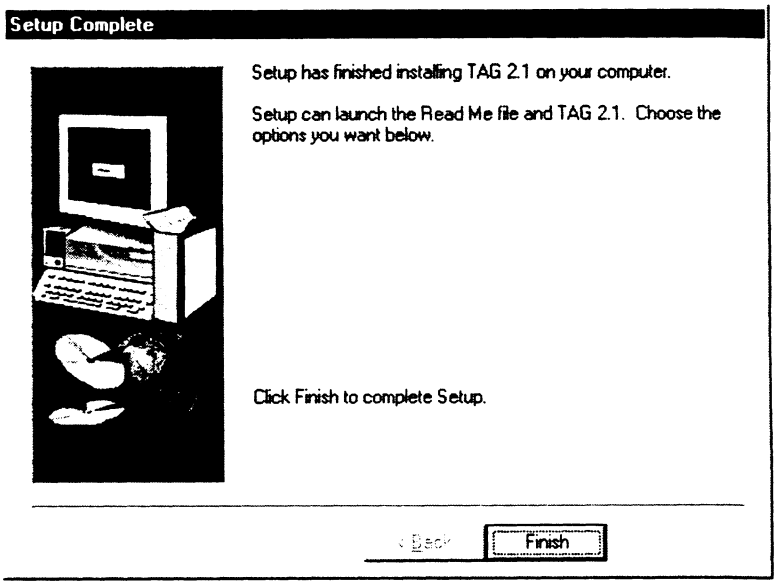
Enter the Notification Server name, application ID and password. Press `Next` to continue. The `ORBIXHOME PATH` window displays:




Enter the pathname of the ORBIX home directory. Press `Next` to continue.

11.2.8 Step 8: Load TAG Client Software

InstallShield will load the TAG Client software onto your PC. This will take a minute, an indicator will display the progress of the load. When installation completes, the `Setup Complete` window displays:




Press `Finish` to exit InstallShield

11.2.9 Step 9: Exit Setup

The TAG 7.5.0.10 Client software is now installed on your PC in the folder you have selected.

# 12. Appendix G - UNIX Kernel Parameters

Shown in the table are recommended UNIX Parameters:

**Table 18- UNIX Kernel Parameters**

| Parameter | Value |
|---|---|
| MAXFILES | 1024 |
| MAXSWAPCHUNKS | 1024 |
| MAXUSERS | 512 |
| MAXUPRC | 512 |
| NPTY | 712 |
| NCALLOUT | 50000 |
| NUM_CLIENTS | 20 |
| NFLOCKS | 400 |
| MAXDSIZ | 0xa000000 |

# 13. Appendix H - Test Client

## 13.1 Test Client

The Test Client simulates a CLEC-developed client application interacting with the TAG Gateway through the TAG Client API. Each Test Client invocation handles one query type, which is specified via a program argument. The test client commands are as follows;

- xstTestClient
- xstTestNotification
- xstChangePwd

### 13.1.1 Overview

When the CLEC application initiates a request, it needs COG's address (IOR) to connect to COG.

It first checks for the COG IOR from a file in the CLEC environment. If the file is found, the CLEC application reads the IOR.

If the IOR file doesn't exist, the CLEC application connects to LDAP, and obtains the COG IOR and writes it to a file.

If the connection to the COG bridge fails 10 attempts, the CLEC application queries LDAP for a new IOR to resume attempts to connect to the bridge.

Due to this process, it may sometimes take up to 60 seconds for either the xstTestClient and /or GUI to be initiated the first time after the server environment has been refreshed.

The Test Client program performs the following functions by invoking the appropriate TAG Client API methods:

- Authenticates using the application ID and password supplied by the user.
- Reads and validates the query data.
- Sends the query and receive a response.

### 13.1.2 Configuration

The TAG installation process configures the *xst_adm* user profile with the necessary information to run the Test Client application. The appropriate environment variables are set when you log in as *xst_adm*. If you do not use the *xst_adm* profile, you must at least set *$XST_CONFIG*.

### 13.1.3 Running the Test Client

Run the Test Client application, use the following command:

**xstTestClient** [*options*] *app_id app_passwd user_id input_file*

Options:

-d s delay s seconds between repeats (see -r)

-r n repeat query n more times after first query

Arguments:

**NOTE**: As of TAG 3.1 the query type is now read from the input file.

RequestType as been added to the input files. This will allow batch of different request types within the same file. The following are values of RequestType:

**Table 19- Test Client REQTYPE Values**

| ***query_type*** | **Query** |
|---|---|
| AAQ | Appointment Availability. |
| AVQ | Address Validation. |
| AVQ_TN | Address Validation with Telephone Number. |
| CDD | Due Date Calculation |
| CSRQ | Customer Record. |
| DIRLIST | Directory Listing and Assistance Order Request |
| NP | Number Portability (NP) Order Request |
| LOOP | LOOP Services Order Request |
| LOOPNP | LOOP Services with Number Portability (NP) Order Request |
| LOOPPORT | LOOP/PORT Combination Services Order Request |
| ORDER | Provide any order type. |
| PONLIST | Purchase Order Number (PON) List Query |
| VIEWLSR | View formatted LSR |
| PORT | PORT Services Order Request |
| RESALE | Resale Order Request |
| SAQ | Service Availability. |
| SOS | Service Order Status Query |
| TNAQ | Telephone Number Assignment. |

| *query_type* | Query |
|---|---|
| TNAQ_DID | Telephone Number Assignment for Direct In Dial. |
| TNAQ_MLH | Telephone Number Assignment for Multi-line Hunt. |
| TNAQ_MISC | Telephone Number Assignment for Miscellaneous |
| TNCAN_DID | Telephone Number Cancellation for Direct In Dial. |
| TNCAN_MLH | Telephone Number Cancellation for Multi-line Hunt. |
| TNCAN_TN | Telephone Number Cancellation. |
| TNSQ | Telephone Number Selection. |
| LOOP_MAKEUP_WORKING | LOOP Makeup Query on Working Loops. |
| LOOP_MAKEUP_SPARE | LOOP Makeup Query on Spare Facilities |
| LOOP_RESERVATION_SPARE | LOOP Reservation Request (Spare Facilities) |
| LOOP_RESERVATION_CANCEL | LOOP Reservation Cancel Request. |

- *app_id*. The Application ID for the Test Client. This value must be configured in the BellSouth security system.
- *app_passwd*. The Application Password for the Test Client. This value must be configured in the BellSouth security system.
- *user_id*. The name of the Test Client user.
- *input_file*. The pathname of the text file containing the values for the query.

### 13.1.4 Running the Notification Test Client

Run the Notification Test Client, use the following command:

**xstTestNotification** [*options*] *app_id* *app_passwd* *user_id*

Options:

**-d s** delay s seconds between fetching notifications from the CLEC Notification Server

**-r n** fetch n notifications before quitting, the default it to run forever

**-a** is used for automatic acknowledgement of notifications using the readNotification() function of the Client API. If this argument is not specified, the readNotificationNoAck() and acknowledgeNotification() functions of the Client API

are used to exercise explicit acknowledgement of notifications.

-g use readNotificationList() and acknowledgeNotificationList() to read and acknowledge notifications.

Arguments:

- *app_id*. The Application ID for the Test Client. This value must be configured in the BellSouth security system.
- *app_passwd*. The Application Password for the Test Client. This value must be configured in the BellSouth security system.
- *user_id*. The name of the Test Client user.

### 13.1.5 Running the Password Reset Test Client

The **xstChangePwd** command will reset the password for an Application ID. **xstChangePwd** takes Application ID as its only argument. You are prompted for the old and new password.

**xstChangePwd** *appl_id*

BELLSOUTH

BELLSOUTH DOCUMENT
BD-TAG-API-R7.5.0.10-28
ISSUE 25 FEBRUARY, 2001
RELEASE 7.5.0.10

# Telecommunications Access Gateway (TAG) API Reference Guide Part B

*For Release 7.5.0.10*

# Proprietary Statement

BellSouth Telecommunications reserves the right to revise this document for any reason, including but not limited to, conformity with standards promulgated by various government or regulatory agencies, utilization of advance in the state of the technical arts, or the reflection of changes in the design of any equipment, techniques, or procedures described or referred to herein.

LIABILITY TO ANYONE ARISING OUT OF USE OR RELIANCE UPON ANY INFORMATION SET FORTH HEREIN IS EXPRESSLY DISCLAIMED, AND NO REPRESENTATIONS OR WARRANTIES, EXPRESSED OR IMPLIED, ARE MADE WITH RESPECT TO THE ACCURACY OR UTILITY OF ANY INFORMATION SET FORTH HEREIN.

This document is not to be construed as a suggestion to any manufacturer to modify or change any of its products, nor does this document represent any commitment by BellSouth Telecommunications to purchase any product whether or not it provides the described characteristics.

Nothing contained herein shall be construed as conferring by implication, estoppel or otherwise, any license or right under any patent, whether or not the use of any information herein necessarily employs an invention of any existing or later issued patent.

## DOCUMENTATION PRODUCTION AND CONTROL
## NOTICE OF DOCUMENTATION UPDATE

Project Name Telecommunications Access Gateway

Type of Update

( ) Initial Release of New Document

( X ) Maintenance Release to Existing Document

| | |
|---|---|
| Documentation Title | Telecommunications Access Gateway (TAG) API Reference Guide |
| Documentation Identification | BD-TAG-API-R7.5.0.10-28 |
| Issue/Revision, Date | Issue 28 FEBRUARY, 2001 |
| First/Current Compatible System Release/Version | Release 7.5.0.10 |
| Date Required at Destination | FEBRUARY, 2001 |

| | |
|---|---|
| Supervisor Initials/Date | Originator Initials/Date |
| Project Approval | Date |

**Reason For Update/Comments; Associated MRs:**

See Section 1.5 for History of changes.

Document Updated by:
Integrated Learning Solutions



Project funding year: 2001

---

# TAG API Reference Guide

## TABLE OF CONTENTS

# 1. About This Guide

## 1.1 Description

This document serves as a programmer's guide for the BellSouth Telecommunications Access Gateway (TAG) Client Application Program Interface (API).

## 1.2 Intended Audience

This guide is intended for developers who need to use the TAG Client API to develop applications that interface with the TAG Gateway.

## 1.3 Assumptions and Caveats

This document provides the reader with the information necessary for writing C++ applications using the TAG Client API.

It is expected that the reader have experience with the C++ language, as well as the knowledge needed to use their Integrated Development Environment (IDE). Specifically, the reader must know how to create and maintain makefiles or project files necessary for compiling and linking C++ programs.

The reader's knowledge of Pre-Order and Firm Order functionality is necessary to use this technology proficiently. This knowledge is also necessary to use the TAG Client APIs. No business rules are described in this guide. It is up to the reader to know and understand the parameters expected for each query and response.

It is helpful, but not essential, that the reader understand the client-server architecture model since the CLEC's application and the CLEC Notification Server will communicate with the TAG Server in a client-server environment.

Although the TAG Client API enforces encryption algorithms, security features, communication protocols, and uses Corba objects, the reader is not expected to have any knowledge of these technologies. The TAG Client API hides these facilities so the developer can focus their attention on Pre-Order and Firm Order tasks without worrying about the underlying architecture.

## 1.4 Organization

This guide is organized as described below. As of Release 7.1, this document is divided into two parts.

**PART A**

1. **About This Guide** provides an overview of this document, its audience, style, organization, and where to go for additional

information. It also provides a summary or running history of changes between releases.

2. **Product Introduction** describes the TAG Client API and its relationship to the client application and TAG Gateway.
3. **TAG Client API** describes the C++ class library that implements the API.
4. **Configuration** explains how to configure the TAG Client API for runtime use.
5. **CLEC Notification Server** describes how to configure, start and stop the CLEC Notification Server.
6. **Appendix A - C++ Header Files** is a listing of the header files included by the client application in order to use the TAG Client API.
7. **Appendix B - Sample Code** contains all the source code for the Test Client. The code will illustrate how the API may be invoked by the client application.
8. **Appendix C - Errors** contains both message ids and error messages generated within the API.
9. **Appendix D - Pre-Installation Checklist** list items to be performed before installation.
10. **Appendix E - TAG Client API UNIX Installation** describes the script used to install the UNIX client software.
11. **Appendix F - TAG Client API Windows Installation** describes the client installation process on Windows.
12. **Appendix G - UNIX Kernel Parameters** lists UNIX kernel parameters.
13. **Appendix H - Test Client** explains how to run the Test Client application.

PART B

14. **Appendix I Test Cases for Pre-Order**
15. **Appendix J Firm Order Test Data**

# 2. Appendix I - Test Cases for Pre-Order Interface Types

**NOTE**: The Test Data provided in this chapter is for example only.

## 2.1 Address Validation

Address Validation test cases will test telephone inputs as well as various forms of address inputs. The number of test cases is needed in order to simulate at least one type of each of the various possible response formats (denoted by a letter in parenthesis following the test title).

### 2.1.1 tag.clec.avq01 Exact Match on WTN (A)

The following sections show API input and output which are what shows up in the trace output files. It is worthwhile to distinguish between test client input/output and API input/output.

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation request with WTN input, which returns an exact address match.

**Interface Type:** xstaAddressValidation.verifyWithTelephone

Special
**Instructions:** None

**API Input:**

```
telephone=4074646476;
```

**API Output:**

```
Status{
msgId=BLP0000ADR;
msgTxt=COMPLETED SUCCESSFULLY;
}
AlternativeAddressList{
0{
alternativeaddress{
fieldedaddress{

houseNumber=2601;
houseNumberSuffix=<null>;
streetDirectional=0;
streetThoroughfare=DR;
streetName=KINGSLEY;
streetSuffix=<null>;
room=<null>;
building=<null>;
floor=<null>;
descriptiveLocation=<null>;
city=FT*P;
state=FL;
zipCode=34946;
unnumberedHouseIndicator=0;
crossBoundary=<null>;
postalBox=<null>;
```

```
route=<null>;
rateZone=<null>;
driveInstructions=<null>;
companyIndicator=<null>;
addressPattern{
0{
structurePattern=<null>;
elevationPattern=<null>;
unitPattern=<null>;
}
}
 }
  }
npanxx=561461;
localServiceTermination=FTPRFLMA;
telephoneInfo{
 0{
telephoneNumber=4074646476;
quickServeIndicator=N;
addressStatus=W;
availableFacilitiesIndicator=N;
}
1{
telephoneNumber=<null>;
quickServeIndicator=N;
addressStatus=<null>;
availableFacilitiesIndicator=N;
}
}
areaTransferInfo{
areaTransferCutDate=<null>;
areaTransferNumChgDate=<null>;
areaTransferNpaNxx=<null>;
tariffExchangeCode=<null>;
}
}
}
AddressInfoHeader{
ServiceRestrictionList{
}
ServiceInstructionText=<null>;
}
```

**xstTestClient Input:**

```
RequestType = avq_tn;
queryTelephone = 4074646476;
```

**xstTest Client Output:**

```
status={
msgId=BLP0000ADR;
msgTxt="COMPLETED SUCCESSFULLY";
};
messageHeader={
inquiryNumber=2ad4852900000001;
dateSent=1999090313314094;
};
```

```
alternativeaddressInformation=(
{
alternativeaddress={
    fieldedaddress={
    houseNumber=2601;
    streetDirectional=;
    streetThoroughfare=DR;
    streetName=KINGSLEY;
    city="FT*P";
    state=FL;
    zipCode=34946;
    unnumberedHouseIndicator=N;
    addressPattern=(
     {
     };
    );
   };
  };
  npanxx=561461;
  localServiceTermination=FTPRFLMA;
  telephoneInfo=(
   {
    telephoneNumber=4074646476;
    quickServeIndicator=N;
    addressStatus=W;
    availableFacilitiesIndicator=N;
   };
   {
    quickServeIndicator=N;
    availableFacilitiesIndicator=N;
   };
  );
  areaTransferInfo={
  };
 };
);
addressInfoHeader={
 serviceRestrictionInfoList=(
 );
};
```

### 2.1.2 tag.clec.avq02 Additional Information Using WTN (B)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Request with WTN input, which returns additional address information.

**Interface Type:** xstaAddressValidation.verifyWithTelephone

**Special Instructions:** None

**API Input:**

```
telephone=4073422500;
```

**API Output:**

```
Status{
 msgId=BLPW907ADR;
 msgTxt=THIS ADDRESS ALSO HAS LIVING UNITS WITH SUPPLEMENTAL ADDRESSES;
}
AlternativeAddressList{
 0{
  alternativeaddress{
   fieldedaddress{

    houseNumber=400;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=<null>;
    streetName=UR 421;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FT*P;
    state=FL;
    zipCode=34951;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=561461;
  localServiceTermination=FTPRFLMA;
  telephoneInfo{
   0{
    telephoneNumber=4073420400;
    quickServeIndicator=N;
    addressStatus=W;
    availableFacilitiesIndicator=N;
   }
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
}
AddressInfoHeader{
 ServiceRestrictionList{
 }
```

```
 ServiceInstructionText=<null>;
}
```

**xstTestClient Input:**

```
RequestType = avq_tn;
queryTelephone = 4073422500;
```

**xstTestClient Output:**

```
status={
 msgId=BLPW907ADR;
 msgTxt="THIS ADDRESS ALSO HAS LIVING UNITS WITH SUPPLEMENTAL ADDRESSES";
};
messageHeader={
 inquiryNumber=0032d01700000001;
 dateSent=1999090313422649;
};
alternativeaddressInformation=(
 {
  alternativeaddress={
   fieldedaddress={
    houseNumber=400;
    streetDirectional=;
    streetName="UR 421";
    city="FT*P";
    state=FL;
    zipCode=34951;
    unnumberedHouseIndicator=N;
    addressPattern=(
     {
     };
    );
   };
  };
  npanxx=561461;
  localServiceTermination=FTPRFLMA;
  telephoneInfo=(
   {
    telephoneNumber=4073420400;
    quickServeIndicator=N;
    addressStatus=W;
    availableFacilitiesIndicator=N;
   };
  );
  areaTransferInfo={
  };
 };
);
addressInfoHeader={
 serviceRestrictionInfoList=(
 );
};
```

2.1.3 tag.clec.avq03 No Match Using WTN (C)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Request with WTN input, which returns no match.

**Interface Type:** xstaAddressValidation.verifyWithTelephone

**Special Instructions:** None

**API Input:**

```
telephone=3057452661;
```

**API Output:**

```
FAILURE: invalid data
         inquiry number: 50009c4800000001
         data element: EDI-POG-HMS-MSG-ID
         msgId: BLPE909ADR
         msgTxt: NO MATCH ON TELEPHONE/CIRCUIT NUMBER. PLEASE ENTER ADDITIONAL
IN
```

**xstTestClient Input:**

```
RequestType = avq_tn
QueryTelephone=3057452661;
```

**xstTestClient Output:**

```
FAILURE: invalid data
         inquiry number: 50009c4800000001
         data element: EDI-POG-HMS-MSG-ID
         msgId: BLPE909ADR
         msgTxt: NO MATCH ON TELEPHONE/CIRCUIT NUMBER. PLEASE ENTER ADDITIONAL
IN
```

## 2.1.4 tag.clec.avq04 Basic Numbered Addresses (L)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Request with numbered address input, which returns a basic numbered address.

**Interface Type:** xstaAddressValidation.verifyWithAddress

**Special Instructions:** None

**API Input:**

```
Address{
 houseNumber=1211;
 houseNumberSuffix=;
 streetDirectional=0;
 streetThoroughfare=DR;
 streetName=SEAWAY;
 streetSuffix=;
 room=APT C;
 building=;
 floor=;
 descriptiveLocation=;
 city=FT PIERCE;
 state=FL;
 zipCode=;
 unnumberedHouseIndicator=0;
 crossBoundary=;
 postalBox=;
 route=;
 rateZone=;
 driveInstructions=;
 companyIndicator=;
 addressPattern{
 }
}
```

**API Output:**

```
Status{
 msgId=BLPE949ADR;
 msgTxt=ADDRESS VALID, NO LIVING UNITS EXIST. SIMILAR HOUSE #'S DISPLAYED;
}
AlternativeAddressList{
 0{
  alternativeaddress{
   fieldedaddress{

    houseNumber=704;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=DR;
    streetName=SKYLARK;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
```

```
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
   0{
    telephoneNumber=<null>;
    quickServeIndicator=<null>;
    addressStatus=<null>;
    availableFacilitiesIndicator=<null>;
   }
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
 1{
  alternativeaddress{
   fieldedaddress{
        houseNumber=704;
    houseNumberSuffix=A;
    streetDirectional=0;
    streetThoroughfare=DR;
    streetName=SKYLARK;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
```

```
driveInstructions=2111111111111111111111111111111111111111111111111111111111111111111
11111112;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
   0{
    telephoneNumber=4074670083;
    quickServeIndicator=<null>;
    addressStatus=W;
    availableFacilitiesIndicator=<null>;
   }
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
}
AddressInfoHeader{
 ServiceRestrictionList{
 }
 ServiceInstructionText=<null>;
}
```

**xstTestClient Input:**

```
RequestType = avq;
address = {
        houseNumber = 1211;
        streetThoroughfare = DR;
        streetName = SEAWAY;
        room = "APT C";
        city = "FT PIERCE";
        state = FL;
        unnumberedHouseIndicator = N;
};
```

**xstTestClient Output:**

```
status={
 msgId=BLPE949ADR;
 msgTxt="ADDRESS VALID, NO LIVING UNITS EXIST. SIMILAR HOUSE #'S DISPLAYED";
};
messageHeader={
 inquiryNumber=6c261f6000000001;
 dateSent=1999090314202688;
};
```

```
alternativeaddressInformation=(
 {
  alternativeaddress={
   fieldedaddress={
    houseNumber=704;
    streetDirectional=;
    streetThoroughfare=DR;
    streetName=SKYLARK;
    city="FORT PIERCE";
    state=FL;
    unnumberedHouseIndicator=N;
    addressPattern=(
     {
     };
    );
   };
  };
  telephoneInfo=(
   {
   };
  );
  areaTransferInfo={
  };
 };
 {
  alternativeaddress={
   fieldedaddress={
    houseNumber=704;
    houseNumberSuffix=A;
    streetDirectional=;
    streetThoroughfare=DR;
    streetName=SKYLARK;
    city="FORT PIERCE";
    state=FL;
    unnumberedHouseIndicator=N;

driveInstructions=2111111111111111111111111111111111111111111111111111111111111111111111111
11111111112;
    addressPattern=(
     {
     };
    );
   };
  };
  telephoneInfo=(
   {
    telephoneNumber=4074670083;
    addressStatus=W;
   };
  );
  areaTransferInfo={
  };
 };
);
addressInfoHeader={
 serviceRestrictionInfoList=(
 );
```

## 2.1.5 tag.clec.avq05 Exact Numbered Address Match (A)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Request with numbered address input, which returns an exact address match.

**Interface Type:** xstaAddressValidation.verifyWithAddress

**Special Instructions:** None

**API Input:**

```
Address{
houseNumber=601;
 houseNumberSuffix=;
 streetDirectional=0;
 streetThoroughfare=DR;
 streetName=SEAWAY;
 streetSuffix=;
 room=RM  1;
 building=;
 floor=;
 descriptiveLocation=;
 city=FT PIERCE;
 state=FL;
 zipCode=;
 unnumberedHouseIndicator=0;
 crossBoundary=;
 postalBox=;
 route=;
 rateZone=;
 driveInstructions=;
 companyIndicator=;
 addressPattern{
 }
}
```

**API Output:**

```
Status{
 msgId=BLP0000ADR;
 msgTxt=COMPLETED SUCCESSFULLY;
}
AlternativeAddressList{
 0{
  alternativeaddress{
   fieldedaddress{

    houseNumber=100;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=LN;
    streetName=BELLS CAMP;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
```

```
    city=BROCKPORT;
    state=NY;
    zipCode=14512;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=IQ;
    driveInstructions=<null>;
    companyIndicator=5;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=716616;
  localServiceTermination=ROCHNY01;
  telephoneInfo{
   0{
    telephoneNumber=4074443333;
    quickServeIndicator=N;
    addressStatus=W;
    availableFacilitiesIndicator=N;
   }
  }
  areaTransferInfo{
   areaTransferCutDate=19990909;
   areaTransferNumChgDate=19990909;
   areaTransferNpaNxx=407745;
   tariffExchangeCode=1234;
  }
 }
}
AddressInfoHeader{
 ServiceRestrictionList{
  0{
   estServiceDate=20000104;
   restrictionCode=NF;
   restrictionText=NO FACILITIES;
  }
 }
 ServiceInstructionText=THIS IS A TEST OF THE NUMBER OF SERVICE INSTRUCTIONS THAT CAN
BE SENT TO CONTRA   CTS VIA RSAGADDR CONTRACT... BASED UPON SERVICE INSTRUCTIONS BEING
ADDED AT THE   GLOBAL LEVELS....THIS ONE IS AT THE GLOBAL LEVEL OF NY1 (EXCHANGE).
IT IS FI   VE LINES LONG, SO THAT THIS TEST CAN BE AS COMPLETE AS POSSIBLE.  SERVICE
INSTR   AT THE BASIC ADDRESS LEVEL ARE ONLY 2 LINES LONG, AT 70-SOME CHAR'S PER LINE.
THIS IS A TEST OF THE NUMBER OF SERVICE INSTRUCTIONS THAT CAN BE SENT TO CONTRA   CTS
VIA RSAGADDR CONTRACT... BASED UPON SERVICE INSTRUCTIONS BEING ADDED AT THE   GLOBAL
LEVELS....THIS ONE IS AT THE GLOBAL LEVEL OF NY1 (EXCHANGE).   IT IS FI   VE LINES
LONG, SO THAT THIS TEST CAN BE AS COMPLETE AS POSSIBLE.  SERVICE INSTR   AT THE BASIC
ADDRESS LEVEL ARE ONLY 2 LINES LONG, AT 70-SOME CHAR'S PER LINE.;
}
```

**xstTestClient Input:**

```
RequestType=avq;
Address ={
       houseNumber=2601;
       streetThoroughfare=DR
       streetName=SKYLARK;
       room=''APT G'';
       city=''FT*P'';
       state=FL;
       unnumberedHouseIndicator=N;
}
```

**xstTestClient Output:**

```
status={
 msgId=BLP0000ADR;
 msgTxt="COMPLETED SUCCESSFULLY";
};
messageHeader={
 inquiryNumber=3610db3900000001;
 dateSent=1999090314324761;
};
alternativeaddressInformation=(
 {
  alternativeaddress={
   fieldedaddress={
    houseNumber=100;
    streetDirectional=;
    streetThoroughfare=LN;
    streetName="BELLS CAMP";
    city=BROCKPORT;
    state=NY;
    zipCode=14512;
    unnumberedHouseIndicator=N;
    rateZone=IQ;
    companyIndicator=5;
    addressPattern=(
     {
     };
    );
   };
  };
  npanxx=716616;
  localServiceTermination=ROCHNY01;
  telephoneInfo=(
   {
    telephoneNumber=4074443333;
    quickServeIndicator=N;
    addressStatus=W;
    availableFacilitiesIndicator=N;
   };
  );
  areaTransferInfo={
   areaTransferCutDate=19990909;
   areaTransferNumChgDate=19990909;
   areaTransferNpaNxx=407745;
   tariffExchangeCode=1234;
  };
 };
```

```
);
addressInfoHeader={
 serviceRestrictionInfoList=(
  {
   estServiceDate=20000104;
   restrictionCode=NF;
   restrictionText="NO FACILITIES";
  };
 );
 serviceInstructionText="THIS IS A TEST OF THE NUMBER OF SERVICE INSTRUCTIONS THAT CAN
BE SENT TO CONTRA   CTS VIA RSAGADDR CONTRACT... BASED UPON SERVICE INSTRUCTIONS BEING
ADDED AT THE   GLOBAL LEVELS....THIS ONE IS AT THE GLOBAL LEVEL OF NY1 (EXCHANGE).
IT IS FI   VE LINES LONG, SO THAT THIS TEST CAN BE AS COMPLETE AS POSSIBLE.  SERVICE
INSTR   AT THE BASIC ADDRESS LEVEL ARE ONLY 2 LINES LONG, AT 70-SOME CHAR'S PER LINE.
THIS IS A TEST OF THE NUMBER OF SERVICE INSTRUCTIONS THAT CAN BE SENT TO CONTRA   CTS
VIA RSAGADDR CONTRACT... BASED UPON SERVICE INSTRUCTIONS BEING ADDED AT THE   GLOBAL
LEVELS....THIS ONE IS AT THE GLOBAL LEVEL OF NY1 (EXCHANGE).   IT IS FI   VE LINES
LONG, SO THAT THIS TEST CAN BE AS COMPLETE AS POSSIBLE.  SERVICE INSTR   AT THE BASIC
ADDRESS LEVEL ARE ONLY 2 LINES LONG, AT 70-SOME CHAR'S PER LINE.";
};
```

### 2.1.6 tag.clec.avq06 Additional Information Returned (B)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Request with numbered address input, which returns additional address information.

**Interface Type:** xstaAddressValidation.verifyWithAddress

**Special Instructions:** None

**Note:** The data format shown is not current, data not available.

**API Input:**

```
Address{
houseNumber=2604;
 houseNumberSuffix=;
 streetDirectional=0;
 streetThoroughfare=DR;
 streetName=KINGSLEY;
 streetSuffix=;
 room=;
 building=;
 floor=;
 descriptiveLocation=;
 city=FT*P;
 state=FL;
 zipCode=;
 unnumberedHouseIndicator=0;
 crossBoundary=;
 postalBox=;
 route=;
 addressPattern{
  0{
   structurePattern=;
   elevationPattern=;
   unitPattern=;
```

```
  }
  1{
   structurePattern=;
   elevationPattern=;
   unitPattern=;
  }
 }
}
```

**API Output:**

```
Status{
 msgId=BLPW904ADR;
 msgTxt=THIS ADDRESS IS VALID, BUT NO LIVING UNIT EXISTS;
}
AlternativeAddressList{
 0{
  alternativeaddress{
   fieldedaddress{
    houseNumber=2604;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=DR;
    streetName=KINGSLEY;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FT*P;
    state=FL;
    zipCode=34946;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=561461;
  localServiceTermination=FTPRFLMA;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
  }
 }
}
AddressInfoHeader{
 ServiceRestrictionList{
```

```
 }
 ServiceInstructionText=<null>;
}
```

**xstTestClient Input:**

```
ADDRESS:|2604|||DR|KINGSLEY||||||FT*P|FL||N|||||||
```

**xstTestClient Output:**

## 2.1.7 tag.clec.avq07 Community Names Returned (E)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Request with numbered address input, which returns community names.

**Interface Type:** xstaAddressValidation.verifyWithAddress

**Special Instructions:** None

**API Input:**

```
Address{
 houseNumber=704;
 houseNumberSuffix=;
 streetDirectional=0;
 streetThoroughfare=DR;
 streetName=SKYLARK;
 streetSuffix=;
 room=APT G;
 building=;
 floor=;
 descriptiveLocation=;
 city=FT PIERCE;
 state=FL;
 zipCode=;
 unnumberedHouseIndicator=0;
 crossBoundary=;
 postalBox=;
 route=;
 rateZone=;
 driveInstructions=;
 companyIndicator=;
 addressPattern{
 }
}
```

**API Output:**

```
Status{
 msgId=BLPE931ADR;
 msgTxt=NO MATCH ON EXACT STREET NAME AND COMMUNITY/STATE;
}
AlternativeAddressList{
 0{
  alternativeaddress{
   fieldedaddress{

    houseNumber=<null>;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=<null>;
    streetName=<null>;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
```

```
    descriptiveLocation=<null>;
    city=FANTASY;
    state=FL;
    zipCode=90099;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
 1{
  alternativeaddress{
   fieldedaddress{
        houseNumber=<null>;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=<null>;
    streetName=<null>;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FAST FOOD;
    state=FL;
    zipCode=00000;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
```

```
      }
     }
    }
   }
   npanxx=<null>;
   localServiceTermination=<null>;
   telephoneInfo{
   }
   areaTransferInfo{
    areaTransferCutDate=<null>;
    areaTransferNumChgDate=<null>;
    areaTransferNpaNxx=<null>;
    tariffExchangeCode=<null>;
   }
  }
  2{
   alternativeaddress{
    fieldedaddress{
     houseNumber=<null>;
     houseNumberSuffix=<null>;
     streetDirectional=0;
     streetThoroughfare=<null>;
     streetName=<null>;
     streetSuffix=<null>;
     room=<null>;
     building=<null>;
     floor=<null>;
     descriptiveLocation=<null>;
     city=FLORIST;
     state=FL;
     zipCode=38105;
     unnumberedHouseIndicator=0;
     crossBoundary=<null>;
     postalBox=<null>;
     route=<null>;
     rateZone=<null>;
     driveInstructions=<null>;
     companyIndicator=<null>;
     addressPattern{
      0{
       structurePattern=<null>;
       elevationPattern=<null>;
       unitPattern=<null>;
      }
     }
    }
   }
   npanxx=<null>;
   localServiceTermination=<null>;
   telephoneInfo{
   }
   areaTransferInfo{
    areaTransferCutDate=<null>;
    areaTransferNumChgDate=<null>;
    areaTransferNpaNxx=<null>;
    tariffExchangeCode=<null>;
   }
  }
  3{
```

```
 alternativeaddress{
  fieldedaddress{

   houseNumber=<null>;
   houseNumberSuffix=<null>;
   streetDirectional=0;
   streetThoroughfare=<null>;
   streetName=<null>;
   streetSuffix=<null>;
   room=<null>;
   building=<null>;
   floor=<null>;
   descriptiveLocation=<null>;
   city=FLOWERTON;
   state=FL;
   zipCode=00000;
   unnumberedHouseIndicator=0;
   crossBoundary=<null>;
   postalBox=<null>;
   route=<null>;
   rateZone=<null>;
   driveInstructions=<null>;
   companyIndicator=<null>;
   addressPattern{
    0{
     structurePattern=<null>;
     elevationPattern=<null>;
     unitPattern=<null>;
    }
   }
  }
 }
 npanxx=<null>;
 localServiceTermination=<null>;
 telephoneInfo{
 }
 areaTransferInfo{
  areaTransferCutDate=<null>;
  areaTransferNumChgDate=<null>;
  areaTransferNpaNxx=<null>;
  tariffExchangeCode=<null>;
 }
}
4{
 alternativeaddress{
  fieldedaddress{
   houseNumber=<null>;
   houseNumberSuffix=<null>;
   streetDirectional=0;
   streetThoroughfare=<null>;
   streetName=<null>;
   streetSuffix=<null>;
   room=<null>;
   building=<null>;
   floor=<null>;
   descriptiveLocation=<null>;
   city=FORT LAUDERDALE;
   state=FL;
   zipCode=33301;
```

```
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
 5{
  alternativeaddress{
   fieldedaddress{
    houseNumber=<null>;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=<null>;
    streetName=<null>;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
    state=FL;
    zipCode=34945;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
```

```
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
}
AddressInfoHeader{
 ServiceRestrictionList{
 }
 ServiceInstructionText=<null>;
}
```

**xstTestClient Input:**

```
RequestType = avq;
address = {
        houseNumber = 704;
        streetThoroughfare = DR;
        streetName = SKYLARK;
        room = "APT G";
        city = "FT PIERCE";
        state = FL;
        unnumberedHouseIndicator = N;
};
```

**xstTestClient Output:**

```
status={
 msgId=BLPE931ADR;
 msgTxt="NO MATCH ON EXACT STREET NAME AND COMMUNITY/STATE";
};
messageHeader={
 inquiryNumber=397bdafb00000001;
 dateSent=1999090314540916;
};
alternativeaddressInformation=(
 {
  alternativeaddress={
   fieldedaddress={
    streetDirectional=;
    city=FANTASY;
    state=FL;
    zipCode=90099;
    unnumberedHouseIndicator=N;
    addressPattern=(
     {
     };
    );
   };
  };
  telephoneInfo=(
  );
```

```
 areaTransferInfo={
 };
};
{
 alternativeaddress={
  fieldedaddress={
   streetDirectional=;
   city="FAST FOOD";
   state=FL;
   zipCode=00000;
   unnumberedHouseIndicator=N;
   addressPattern=(
    {
    };
   );
  };
 };
 telephoneInfo=(
 );
 areaTransferInfo={
 };
};
{
 alternativeaddress={
  fieldedaddress={
   streetDirectional=;
   city=FLORIST;
   state=FL;
   zipCode=38105;
   unnumberedHouseIndicator=N;
   addressPattern=(
    {
    };
   );
  };
 };
 telephoneInfo=(
 );
 areaTransferInfo={
 };
};
{
 alternativeaddress={
  fieldedaddress={
   streetDirectional=;
   city=FLOWERTON;
   state=FL;
   zipCode=00000;
   unnumberedHouseIndicator=N;
   addressPattern=(
    {
    };
   );
  };
 };
 telephoneInfo=(
 );
 areaTransferInfo={
 };
```

```
 };
 {
  alternativeaddress={
   fieldedaddress={
    streetDirectional=;
    city="FORT LAUDERDALE";
    state=FL;
    zipCode=33301;
    unnumberedHouseIndicator=N;
    addressPattern=(
     {
     };
    );
   };
  };
  telephoneInfo=(
  );
  areaTransferInfo={
  };
 };
 {
  alternativeaddress={
   fieldedaddress={
    streetDirectional=;
    city="FORT PIERCE";
    state=FL;
    zipCode=34945;
    unnumberedHouseIndicator=N;
    addressPattern=(
     {
     };
    );
   };
  };
  telephoneInfo=(
  );
  areaTransferInfo={
  };
 };
);
addressInfoHeader={
 serviceRestrictionInfoList=(
 );
};
```

2.1.8 tag.clec.avq08 Street Names Returned (F)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Request with numbered address input, which returns street names.

**Interface Type:** xstaAddressValidation.verifyWithAddress

**Special Instructions:** None

**Note:** The data shown is not current, data not available.

**API Input:**

```
Address{

 houseNumber=2604;
 houseNumberSuffix=;
 streetDirectional=0;
 streetThoroughfare=DR;
 streetName=SKYLAR;
 streetSuffix=;
 room=;
 building=;
 floor=;
 descriptiveLocation=;
 city=FT PIERCE;
 state=FL;
 zipCode=;
 unnumberedHouseIndicator=0;
 crossBoundary=;
 postalBox=;
 route=;
 addressPattern{
  0{
   structurePattern=;
   elevationPattern=;
   unitPattern=;
  }
  1{
   structurePattern=;
   elevationPattern=;
   unitPattern=;
  }
 }
}
```

**API Output:**

```
Status{
 msgId=BLPE902ADR;
 msgTxt=NO EXACT MATCH ON STREET NAME;
}
AlternativeAddressList{
 0{
  alternativeaddress{
   addresswithrange{

    houseNumber=;
    houseNumberRange{
     fromHouseNumber=1700;
```

```
     toHouseNumber=1799;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=0;
    streetThoroughfare=CT;
    streetName=SCHOOL;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
  }
 }
 1{
  alternativeaddress{
   addresswithrange{
    houseNumber=;
    houseNumberRange{
     fromHouseNumber=700;
     toHouseNumber=999;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=0;
    streetThoroughfare=DR;
    streetName=SKYLARK;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
```

```
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
  }
 }
 2{
  alternativeaddress{
   addresswithrange{
    houseNumber=;
    houseNumberRange{
     fromHouseNumber=2400;
     toHouseNumber=2499;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=0;
    streetThoroughfare=CT;
    streetName=SKYLOCK;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
  }
 }
}
AddressInfoHeader{
 ServiceRestrictionList{
 }
```

```
 ServiceInstructionText=<null>;
}
```

**xstTestClient Input:**

```
ADDRESS:|2604|||DR|SKYLAR||||||FT PIERCE|FL||N|||||||
```

**API Output:**

**xstTestClient Output:**

### 2.1.9 tag.clec.avq09 Descriptive Names Returned (G)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Request with descriptive address input, which returns descriptive names.

**Interface Type:** xstaAddressValidation.verifyWithAddress

**Special Instructions:** None

**API Input:**

```
Address{

 houseNumber=1002;
 houseNumberSuffix=;
 streetDirectional=0;
 streetThoroughfare=DR;
 streetName=SEAWAY;
 streetSuffix=;
 room=RM  Z;
 building=;
 floor=;
 descriptiveLocation=;
 city=FT PIERCE;
 state=FL;
 zipCode=;
 unnumberedHouseIndicator=0;
 crossBoundary=;
 postalBox=;
 route=;
 rateZone=;
 driveInstructions=;
 companyIndicator=;
 addressPattern{
 }
}
```

**API Output:**

```
Status{
 msgId=BLPE903ADR;
 msgTxt=NO MATCH ON DESCRIPTIVE ADDRESS;
```

```
}
AlternativeAddressList{
 0{
  alternativeaddress{
   fieldedaddress{
        houseNumber=<null>;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=<null>;
    streetName=<null>;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=DARVEESH APTS;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
 1{
  alternativeaddress{
   fieldedaddress{
        houseNumber=<null>;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=<null>;
    streetName=<null>;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=DESCRIPTIVE FOR I009;
    city=FORT PIERCE;
```

```
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
 2{
  alternativeaddress{
   fieldedaddress{

    houseNumber=<null>;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=<null>;
    streetName=<null>;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=DESCRIPTIVE 3.2;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
```

```
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
 3{
  alternativeaddress{
   fieldedaddress{

    houseNumber=<null>;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=<null>;
    streetName=<null>;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=DOCK LOTS;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
 4{
```

```
 alternativeaddress{
  fieldedaddress{

   houseNumber=<null>;
   houseNumberSuffix=<null>;
   streetDirectional=0;
   streetThoroughfare=<null>;
   streetName=<null>;
   streetSuffix=<null>;
   room=<null>;
   building=<null>;
   floor=<null>;
   descriptiveLocation=DREAM ACRES;
   city=FORT PIERCE;
   state=FL;
   zipCode=<null>;
   unnumberedHouseIndicator=0;
   crossBoundary=<null>;
   postalBox=<null>;
   route=<null>;
   rateZone=<null>;
   driveInstructions=<null>;
   companyIndicator=<null>;
   addressPattern{
    0{
     structurePattern=<null>;
     elevationPattern=<null>;
     unitPattern=<null>;
    }
   }
  }
 }
 npanxx=<null>;
 localServiceTermination=<null>;
 telephoneInfo{
 }
 areaTransferInfo{
  areaTransferCutDate=<null>;
  areaTransferNumChgDate=<null>;
  areaTransferNpaNxx=<null>;
  tariffExchangeCode=<null>;
 }
}
5{
 alternativeaddress{
  fieldedaddress{

   houseNumber=<null>;
   houseNumberSuffix=<null>;
   streetDirectional=0;
   streetThoroughfare=<null>;
   streetName=<null>;
   streetSuffix=<null>;
   room=<null>;
   building=<null>;
   floor=<null>;
   descriptiveLocation=DREW'S DONUT SHOP;
   city=FORT PIERCE;
   state=FL;
```

```
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
 6{
  alternativeaddress{
   fieldedaddress{
        houseNumber=<null>;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=<null>;
    streetName=<null>;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=DSCRP W NO LINES;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
```

```
 }
 npanxx=<null>;
 localServiceTermination=<null>;
 telephoneInfo{
 }
 areaTransferInfo{
  areaTransferCutDate=<null>;
  areaTransferNumChgDate=<null>;
  areaTransferNpaNxx=<null>;
  tariffExchangeCode=<null>;
 }
}
7{
 alternativeaddress{
  fieldedaddress{

   houseNumber=<null>;
   houseNumberSuffix=<null>;
   streetDirectional=0;
   streetThoroughfare=<null>;
   streetName=<null>;
   streetSuffix=<null>;
   room=<null>;
   building=<null>;
   floor=<null>;
   descriptiveLocation=DSCRP W NO SUPPL;
   city=FORT PIERCE;
   state=FL;
   zipCode=<null>;
   unnumberedHouseIndicator=0;
   crossBoundary=<null>;
   postalBox=<null>;
   route=<null>;
   rateZone=<null>;
   driveInstructions=<null>;
   companyIndicator=<null>;
   addressPattern{
    0{
     structurePattern=<null>;
     elevationPattern=<null>;
     unitPattern=<null>;
    }
   }
  }
 }
 npanxx=<null>;
 localServiceTermination=<null>;
 telephoneInfo{
 }
 areaTransferInfo{
  areaTransferCutDate=<null>;
  areaTransferNumChgDate=<null>;
  areaTransferNpaNxx=<null>;
  tariffExchangeCode=<null>;
 }
}
8{
 alternativeaddress{
  fieldedaddress{
```

```
    houseNumber=<null>;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=<null>;
    streetName=<null>;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=DSCRP W SUPPL & NO LINES;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
 9{
  alternativeaddress{
   fieldedaddress{

    houseNumber=<null>;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=<null>;
    streetName=<null>;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=DUP TEST;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
```

```
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
}
```

**xstTestClient Input:**

```
RequestType = avq;
address = {
        houseNumber = 1002;
        streetThoroughfare = DR;
        streetName = SEAWAY;
        room = "RM  Z";
        city = "FT PIERCE";
        state = FL;
        unnumberedHouseIndicator = N;
};
```

**xstTestClient Output:**

```
status={
 msgId=BLPE903ADR;
 msgTxt="NO MATCH ON DESCRIPTIVE ADDRESS";
};
messageHeader={
 inquiryNumber=337baf0600000001;
 dateSent=1999090315081901;
};
alternativeaddressInformation=(
 {
  alternativeaddress={
   fieldedaddress={
    streetDirectional=;
    descriptiveLocation="DARVEESH APTS";
    city="FORT PIERCE";
```

```
    state=FL;
    unnumberedHouseIndicator=N;
    addressPattern=(
     {
     };
    );
   };
  };
  telephoneInfo=(
  );
  areaTransferInfo={
  };
 };
 {
  alternativeaddress={
   fieldedaddress={
    streetDirectional=;
    descriptiveLocation="DESCRIPTIVE FOR I009";
    city="FORT PIERCE";
    state=FL;
    unnumberedHouseIndicator=N;
    addressPattern=(
     {
     };
    );
   };
  };
  telephoneInfo=(
  );
  areaTransferInfo={
  };
 };
 {
  alternativeaddress={
   fieldedaddress={
    streetDirectional=;
    descriptiveLocation="DESCRIPTIVE 3.2";
    city="FORT PIERCE";
    state=FL;
    unnumberedHouseIndicator=N;
    addressPattern=(
     {
     };
    );
   };
  };
  telephoneInfo=(
  );
  areaTransferInfo={
  };
 };
 {
  alternativeaddress={
   fieldedaddress={
    streetDirectional=;
    descriptiveLocation="DOCK LOTS";
    city="FORT PIERCE";
    state=FL;
    unnumberedHouseIndicator=N;
```

```
    addressPattern=(
     {
     };
    );
   };
  };
  telephoneInfo=(
  );
  areaTransferInfo={
  };
 };
 {
  alternativeaddress={
   fieldedaddress={
    streetDirectional=;
    descriptiveLocation="DREAM ACRES";
    city="FORT PIERCE";
    state=FL;
    unnumberedHouseIndicator=N;
    addressPattern=(
     {
     };
    );
   };
  };
  telephoneInfo=(
  );
  areaTransferInfo={
  };
 };
 {
  alternativeaddress={
   fieldedaddress={
    streetDirectional=;
    descriptiveLocation="DREW'S DONUT SHOP";
    city="FORT PIERCE";
    state=FL;
    unnumberedHouseIndicator=N;
    addressPattern=(
     {
     };
    );
   };
  };
  telephoneInfo=(
  );
  areaTransferInfo={
  };
 };
 {
  alternativeaddress={
   fieldedaddress={
    streetDirectional=;
    descriptiveLocation="DSCRP W NO LINES";
    city="FORT PIERCE";
    state=FL;
    unnumberedHouseIndicator=N;
    addressPattern=(
     {
```

```
      };
     );
    };
   };
   telephoneInfo=(
   );
   areaTransferInfo={
   };
  };
  {
   alternativeaddress={
    fieldedaddress={
     streetDirectional=;
     descriptiveLocation="DSCRP W NO SUPPL";
     city="FORT PIERCE";
     state=FL;
     unnumberedHouseIndicator=N;
     addressPattern=(
      {
      };
     );
    };
   };
   telephoneInfo=(
   );
   areaTransferInfo={
   };
  };
  {
   alternativeaddress={
    fieldedaddress={
     streetDirectional=;
     descriptiveLocation="DSCRP W SUPPL & NO LINES";
     city="FORT PIERCE";
     state=FL;
     unnumberedHouseIndicator=N;
     addressPattern=(
      {
      };
     );
    };
   };
   telephoneInfo=(
   );
   areaTransferInfo={
   };
  };
  {
   alternativeaddress={
    fieldedaddress={
     streetDirectional=;
     descriptiveLocation="DUP TEST";
     city="FORT PIERCE";
     state=FL;
     unnumberedHouseIndicator=N;
     addressPattern=(
      {
      };
     );
```

```
    };
   };
   telephoneInfo=(
   );
   areaTransferInfo={
   };
  };
);
addressInfoHeader={
 serviceRestrictionInfoList=(
 );
};
```

2.1.10 tag.clec.avq10 House Numbers Returned (F)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Request with numbered address input, which returns house numbers.

**Interface Type:** xstaAddressValidation.verifyWithAddress

**Special Instructions:** None

**API Input:**

```
Address{
 houseNumber=704;
 houseNumberSuffix=;
 streetDirectional=0;
 streetThoroughfare=DR;
 streetName=SKYLARK;
 streetSuffix=;
 room=APT G;
 building=;
 floor=;
 descriptiveLocation=;
 city=FT PIERCE;
 state=FL;
 zipCode=;
 unnumberedHouseIndicator=0;
 crossBoundary=;
 postalBox=;
 route=;
 rateZone=;
 driveInstructions=;
 companyIndicator=;
 addressPattern{
 }
}
```

**API Output:**

```
Status{
 msgId=BLPE902ADR;
 msgTxt=NO EXACT MATCH ON STREET NAME;
}
AlternativeAddressList{
 0{
  alternativeaddress{
   addresswithrange{

    houseNumber=;
    houseNumberRange{
     fromHouseNumber=1700;
     toHouseNumber=1799;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=0;
    streetThoroughfare=CT;
    streetName=SCHOOL;
    streetSuffix=<null>;
```

```
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
 1{
  alternativeaddress{
   addresswithrange{

    houseNumber=;
    houseNumberRange{
     fromHouseNumber=700;
     toHouseNumber=999;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=0;
    streetThoroughfare=DR;
    streetName=SKYLARK;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
```

```
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
 2{
  alternativeaddress{
   addresswithrange{
        houseNumber=;
    houseNumberRange{
     fromHouseNumber=2400;
     toHouseNumber=2499;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=0;
    streetThoroughfare=CT;
    streetName=SKYLOCK;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
}
AddressInfoHeader{
 ServiceRestrictionList{
 }
 ServiceInstructionText=<null>;
}
```

**xstTestClient Input:**

```
RequestType = avq;
address = {
        houseNumber = 704;
        streetThoroughfare = DR;
        streetName = SKYLARK;
        room = "APT G";
        city = "FT PIERCE";
        state = FL;
        unnumberedHouseIndicator = N;
};
```

**xstTestClient Output:**

```
status={
 msgId=BLPE902ADR;
 msgTxt="NO EXACT MATCH ON STREET NAME";
};
messageHeader={
 inquiryNumber=5df0e39b00000001;
 dateSent=1999090315032042;
};
alternativeaddressInformation=(
 {
  alternativeaddress={
   addresswithrange={
    houseNumberRange={
     fromHouseNumber=1700;
     toHouseNumber=1799;
    };
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=;
    streetThoroughfare=CT;
    streetName=SCHOOL;
    city="FORT PIERCE";
    state=FL;
   };
  };
  telephoneInfo=(
  );
  areaTransferInfo={
  };
 };
 {
  alternativeaddress={
   addresswithrange={
    houseNumberRange={
     fromHouseNumber=700;
     toHouseNumber=999;
    };
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=;
    streetThoroughfare=DR;
    streetName=SKYLARK;
    city="FORT PIERCE";
    state=FL;
   };
  };
```

```
  telephoneInfo=(
  );
  areaTransferInfo={
  };
 };
 {
  alternativeaddress={
   addresswithrange={
    houseNumberRange={
     fromHouseNumber=2400;
     toHouseNumber=2499;
    };
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=;
    streetThoroughfare=CT;
    streetName=SKYLOCK;
    city="FORT PIERCE";
    state=FL;
   };
  };
  telephoneInfo=(
  );
  areaTransferInfo={
  };
 };
);
addressInfoHeader={
 serviceRestrictionInfoList=(
 );
};
```

2.1.11 tag.clec.avq11 Location Standards Returned (1)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Request with numbered address input, which returns location standards.

**Interface Type:** xstaAddressValidation.verifyWithAddress

**Special Instructions:** None

**API Input:**

```
Address{
 houseNumber=1002;
 houseNumberSuffix=;
 streetDirectional=0;
 streetThoroughfare=DR;
 streetName=SEAWAY;
 streetSuffix=;
 room=RM  Z;
 building=;
 floor=;
 descriptiveLocation=;
 city=FT PIERCE;
 state=FL;
 zipCode=;
 unnumberedHouseIndicator=0;
 crossBoundary=;
 postalBox=;
 route=;
 rateZone=;
 driveInstructions=;
 companyIndicator=;
 addressPattern{
 }
```

**API Output:**

```
Status{
 msgId=BLPE941ADR;
 msgTxt=SIMILAR ADDRESS FOUND - TYPE I DATA RETURNED;
}
AlternativeAddressList{
 0{
  alternativeaddress{
   fieldedaddress{
    houseNumber=601;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=DR;
    streetName=SEAWAY;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
```

```
    descriptiveLocation=CAUSEWAY MHP;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=UNITA;
     }
     1{
      structurePattern=BLDGA;
      elevationPattern=<null>;
      unitPattern=APT A A-N;
     }
     2{
      structurePattern=PIERA;
      elevationPattern=<null>;
      unitPattern=LOT AN;
     }
     3{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=APT A-N;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
}
AddressInfoHeader{
 ServiceRestrictionList{
 }
 ServiceInstructionText=LOC STANDARD: LOT A-N (12-91) OR LOT N-A;
}
```

**xstTestClient Input:**

```
RequestType = avq;
address = {
        houseNumber = 1002;
        streetThoroughfare = DR;
```

```
        streetName = SEAWAY;
        room = "RM  Z";
        city = "FT PIERCE";
        state = FL;
        unnumberedHouseIndicator = N;
};
```

**xstTestClient Output:**

```
status={
 msgId=BLPE941ADR;
 msgTxt="SIMILAR ADDRESS FOUND - TYPE I DATA RETURNED";
};
messageHeader={
 inquiryNumber=26eb093400000001;
 dateSent=1999090315344113;
};
alternativeaddressInformation=(
 {
  alternativeaddress={
   fieldedaddress={
    houseNumber=601;
    streetDirectional=;
    streetThoroughfare=DR;
    streetName=SEAWAY;
    descriptiveLocation="CAUSEWAY MHP";
    city="FORT PIERCE";
    state=FL;
    unnumberedHouseIndicator=N;
    addressPattern=(
     {
      unitPattern=UNITA;
     };
     {
      structurePattern=BLDGA;
      unitPattern="APT A A-N";
     };
     {
      structurePattern=PIERA;
      unitPattern="LOT AN";
     };
     {
      unitPattern="APT A-N";
     };
    );
   };
  };
  telephoneInfo=(
  );
  areaTransferInfo={
  };
 };
);
addressInfoHeader={
 serviceRestrictionInfoList=(
 );
 serviceInstructionText="LOC STANDARD: LOT A-N (12-91) OR LOT N-A";
};
```

2.1.12 tag.clec.avq12 Supplemental Addresses Returned (J)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Request with numbered address input, which returns supplemental addresses.

**Interface Type:** xstaAddressValidation.verifyWithAddress

**Special Instructions:** None

**Note:** The data shown is not current, data not available.

**API Input:**

```
Address{

 houseNumber=1211;
 houseNumberSuffix=;
 streetDirectional=0;
 streetThoroughfare=DR;
 streetName=SEAWAY;
 streetSuffix=;
 room=APT D;
 building=;
 floor=;
 descriptiveLocation=;
 city=FT*P;
 state=FL;
 zipCode=;
 unnumberedHouseIndicator=0;
 crossBoundary=;
 postalBox=;
 route=;
 addressPattern{
  0{
   structurePattern=;
   elevationPattern=;
   unitPattern=;
  }
  1{
   structurePattern=;
   elevationPattern=;
   unitPattern=;
  }
 }
}
```

**API Output:**

```
Status{
 msgId=BLPE911ADR;
 msgTxt=NO EXACT MATCH ON SUPPLEMENTAL ADDRESS.;
}
AlternativeAddressList{
 0{
  alternativeaddress{
   fieldedaddress{
    houseNumber=1211;
```

```
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=DR;
    streetName=SEAWAY;
    streetSuffix=<null>;
    room=LOT 9999;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
   0{
    telephoneNumber=<null>;
    quickServeIndicator=<null>;
    addressStatus=<null>;
    availableFacilitiesIndicator=<null>;
   }
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
  }
 }
 1{
  alternativeaddress{
   fieldedaddress{

    houseNumber=1211;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=DR;
    streetName=SEAWAY;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=FLR 52;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
```

```
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
   0{
    telephoneNumber=4079432333;
    quickServeIndicator=<null>;
    addressStatus=W;
    availableFacilitiesIndicator=<null>;
   }
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
  }
 }
 2{
  alternativeaddress{
   fieldedaddress{

    houseNumber=1211;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=DR;
    streetName=SEAWAY;
    streetSuffix=<null>;
    room=<null>;
    building=BLDG55;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
```

```
    0{
     structurePattern=<null>;
     elevationPattern=<null>;
     unitPattern=<null>;
    }
   }
  }
 }
 npanxx=<null>;
 localServiceTermination=<null>;
 telephoneInfo{
  0{
   telephoneNumber=4072349233;
   quickServeIndicator=<null>;
   addressStatus=W;
   availableFacilitiesIndicator=<null>;
  }
 }
 areaTransferInfo{
  areaTransferCutDate=<null>;
  areaTransferNumChgDate=<null>;
  areaTransferNpaNxx=<null>;
 }
}
3{
 alternativeaddress{

   houseNumber=1211;
   houseNumberSuffix=<null>;
   streetDirectional=0;
   streetThoroughfare=DR;
   streetName=SEAWAY;
   streetSuffix=<null>;
   room=APT A;
   building=<null>;
   floor=<null>;
   descriptiveLocation=<null>;
   city=FORT PIERCE;
   state=FL;
   zipCode=<null>;
   unnumberedHouseIndicator=0;
   crossBoundary=<null>;
   postalBox=<null>;
   route=<null>;
   rateZone=<null>;
   driveInstructions=<null>;
   companyIndicator=<null>;
   addressPattern{
    0{
     structurePattern=<null>;
     elevationPattern=<null>;
     unitPattern=<null>;
    }
   }
  }
 }
 npanxx=<null>;
 localServiceTermination=<null>;
 telephoneInfo{
```

```
  0{
   telephoneNumber=4074645636;
   quickServeIndicator=<null>;
   addressStatus=W;
   availableFacilitiesIndicator=<null>;
  }
 }
 areaTransferInfo{
  areaTransferCutDate=<null>;
  areaTransferNumChgDate=<null>;
  areaTransferNpaNxx=<null>;
 }
}
4{
 alternativeaddress{
  fieldedaddress{
       houseNumber=1211;
   houseNumberSuffix=<null>;
   streetDirectional=0;
   streetThoroughfare=DR;
   streetName=SEAWAY;
   streetSuffix=<null>;
   room=APT B;
   building=<null>;
   floor=<null>;
   descriptiveLocation=<null>;
   city=FORT PIERCE;
   state=FL;
   zipCode=<null>;
   unnumberedHouseIndicator=0;
   crossBoundary=<null>;
   postalBox=<null>;
   route=<null>;
   rateZone=<null>;
   driveInstructions=<null>;
   companyIndicator=<null>;
   addressPattern{
    0{
     structurePattern=<null>;
     elevationPattern=<null>;
     unitPattern=<null>;
    }
   }
  }
 }
 npanxx=<null>;
 localServiceTermination=<null>;
 telephoneInfo{
  0{
   telephoneNumber=4074669479;
   quickServeIndicator=<null>;
   addressStatus=W;
   availableFacilitiesIndicator=<null>;
  }
 }
 areaTransferInfo{
  areaTransferCutDate=<null>;
  areaTransferNumChgDate=<null>;
  areaTransferNpaNxx=<null>;
```

```
  }
 }
 5{
  alternativeaddress{
   fieldedaddress{
        houseNumber=1211;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=DR;
    streetName=SEAWAY;
    streetSuffix=<null>;
    room=APT C;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
   0{
    telephoneNumber=<null>;
    quickServeIndicator=<null>;
    addressStatus=<null>;
    availableFacilitiesIndicator=<null>;
   }
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
  }
 }
}
AddressInfoHeader{
 ServiceRestrictionList{
 }
 ServiceInstructionText=<null>;
}
```

**xstTestClient Input:**

```
RequestType=avq;
Address ={
       houseNumber=1211;
       streetThoroughfare=DR
       streetName=SEAWAY;
       room=  APT D
       city=''FT *P'';
       state=FL;
       unnumberedHouseIndicator=N;
}
```

**xstTestClient Output:**

### 2.1.13 tag.clec.avq13 Cross Boundary Lines Returned (F)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Request with numbered address input, which returns cross boundary lines.

**Interface Type:** xstaAddressValidation.verifyWithAddress

**Special Instructions:** None

**Note**: The data shown is not current, data not available.

**API Input:**

```
Address{

 houseNumber=2154;
 houseNumberSuffix=;
 streetDirectional=0;
 streetThoroughfare=RD;
 streetName=SANDY GROVE;
 streetSuffix=;
 room=;
 building=;
 floor=;
 descriptiveLocation=;
 city=BENNETTSVILLE;
 state=SC;
 zipCode=;
 unnumberedHouseIndicator=0;
 crossBoundary=;
 postalBox=;
 route=;
 addressPattern{
  0{
   structurePattern=;
   elevationPattern=;
   unitPattern=;
  }
  1{
   structurePattern=;
   elevationPattern=;
   unitPattern=;
  }
 }
```

```
}
```

**API Output:**

```
Status{
 msgId=BLPE902ADR;
 msgTxt=NO EXACT MATCH ON STREET NAME;
}
AlternativeAddressList{
 0{
  alternativeaddress{
   addresswithrange{

    houseNumber=;
    houseNumberRange{
     fromHouseNumber=100;
     toHouseNumber=170;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=0;
    streetThoroughfare=CIR;
    streetName=SANDSPUR;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=BENNETTSVILLE;
    state=SC;
    zipCode=<null>;
    crossBoundary=NC;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
  }
 }
 1{
  alternativeaddress{
   addresswithrange{
    houseNumber=;
    houseNumberRange{
     fromHouseNumber=2000;
     toHouseNumber=2374;
    }
    houseNumberSuffix=<null>;
```

```
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=0;
    streetThoroughfare=LN;
    streetName=SANDY;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=BENNETTSVILLE;
    state=SC;
    zipCode=<null>;
    crossBoundary=NC;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
  }
 }
 2{
  alternativeaddress{
   addresswithrange{

    houseNumber=;
    houseNumberRange{
     fromHouseNumber=726;
     toHouseNumber=1733;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=0;
    streetThoroughfare=RD;
    streetName=SANDY GROVE CH;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=BENNETTSVILLE;
    state=SC;
    zipCode=<null>;
    crossBoundary=NC;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
```

```
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
  }
 }
 3{
  alternativeaddress{
   addresswithrange{
        houseNumber=;
    houseNumberRange{
     fromHouseNumber=200;
     toHouseNumber=599;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=0;
    streetThoroughfare=ST;
    streetName=SHANDON;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=BENNETTSVILLE;
    state=SC;
    zipCode=<null>;
    crossBoundary=NC;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
  }
 }
 4{
  alternativeaddress{
   addresswithrange{

    houseNumber=;
    houseNumberRange{
```

```
     fromHouseNumber=1;
     toHouseNumber=199;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=2;
    streetDirectional=0;
    streetThoroughfare=ST;
    streetName=SHENANDOAH;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=BENNETTSVILLE;
    state=SC;
    zipCode=<null>;
    crossBoundary=NC;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
  }
 }
 5{
  alternativeaddress{
   addresswithrange{
        houseNumber=;
    houseNumberRange{
     fromHouseNumber=1;
     toHouseNumber=6;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=2;
    streetDirectional=0;
    streetThoroughfare=RD;
    streetName=SMITH;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=BENNETTSVILLE;
    state=SC;
    zipCode=<null>;
    crossBoundary=NC;
    postalBox=<null>;
```

```
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
  }
 }
 6{
  alternativeaddress{
   addresswithrange{

    houseNumber=;
    houseNumberRange{
     fromHouseNumber=500;
     toHouseNumber=699;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=0;
    streetThoroughfare=RD;
    streetName=SMITH;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=BENNETTSVILLE;
    state=SC;
    zipCode=<null>;
    crossBoundary=NC;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
  }
 }
 7{
  alternativeaddress{
```

```
  addresswithrange{

   houseNumber=;
   houseNumberRange{
    fromHouseNumber=1;
    toHouseNumber=99;
   }
   houseNumberSuffix=<null>;
   oddEvenIndicator=B;
   assignedHouseNumberStatus=1;
   streetDirectional=0;
   streetThoroughfare=ST;
   streetName=SMITH;
   streetSuffix=<null>;
   room=<null>;
   building=<null>;
   floor=<null>;
   descriptiveLocation=<null>;
   city=BENNETTSVILLE;
   state=SC;
   zipCode=<null>;
   crossBoundary=NC;
   postalBox=<null>;
   route=<null>;
   rateZone=<null>;
   driveInstructions=<null>;
   companyIndicator=<null>;
  }
 }
 npanxx=<null>;
 localServiceTermination=<null>;
 telephoneInfo{
 }
 areaTransferInfo{
  areaTransferCutDate=<null>;
  areaTransferNumChgDate=<null>;
 }
}
8{
 alternativeaddress{
  addresswithrange{
       houseNumber=;
   houseNumberRange{
    fromHouseNumber=200;
    toHouseNumber=250;
   }
   houseNumberSuffix=<null>;
   oddEvenIndicator=B;
   assignedHouseNumberStatus=1;
   streetDirectional=0;
   streetThoroughfare=ST;
   streetName=SMYTHE;
   streetSuffix=<null>;
   room=<null>;
   building=<null>;
   floor=<null>;
   descriptiveLocation=<null>;
   city=BENNETTSVILLE;
   state=SC;
```

```
    zipCode=<null>;
    crossBoundary=NC;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
  }
 }
9{
  alternativeaddress{
   addresswithrange{
        houseNumber=;
    houseNumberRange{
     fromHouseNumber=1100;
     toHouseNumber=1299;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=0;
    streetThoroughfare=LN;
    streetName=SUMMIT;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=BENNETTSVILLE;
    state=SC;
    zipCode=<null>;
    crossBoundary=NC;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
  }
 }
```

```
}
AddressInfoHeader{
 ServiceRestrictionList{
 }
 ServiceInstructionText=<null>;
}
```

**xstTestClient Input:**

```
RequestType=avq;
Address ={
       houseNumber=2154;
       streetThoroughfare=DR
       streetName=SANDY GROVE;
       city=BENNETTSVILLE;
       state=SC;
       unnumberedHouseIndicator=N;
}
```

**xstTestClient Output:**

2.1.14 tag.clec.avq14 Resend (B)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Resend Request with existing inquiry number provided.

**Interface Type:** xstaAddressValidation.verifyWithAddress

**Special Instructions:** None

**API Input:**

```
Address{

 houseNumber=101;
 houseNumberSuffix=A;
 streetDirectional=6;
 streetThoroughfare=DR;
 streetName=SKYLARK;
 streetSuffix=E;
 room=APT G;
 building=BLDG1;
 floor=FLR 3;
 descriptiveLocation=SEAGATE;
 city=MIAMI;
 state=FL;
 zipCode=999999;
 unnumberedHouseIndicator=0;
 crossBoundary=GA;
 postalBox=123;
 route=14W;
 rateZone=;
 driveInstructions=;
 companyIndicator=;
 addressPattern{
 }
}
```

**API Output:**

```
Status{
 msgId=BLPW906ADR;
 msgTxt=ADDRESS INFORMATION ONLY/ASSIGN AHN;
}
AlternativeAddressList{
 0{
  alternativeaddress{
   addresswithrange{

    houseNumber=;
    houseNumberRange{
     fromHouseNumber=1;
     toHouseNumber=199;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=2;
    streetDirectional=0;
```

```
    streetThoroughfare=BLVD;
    streetName=CRANE;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=SUGARLOAF KEY;
    state=FL;
    zipCode=33042;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=SGKYFLMA;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
```

**xstTestClient Input:**

```
RequestType = avq;
inquiryNumber = 5a29272100000001;
address = {
        houseNumber = 101;
        houseNumberSuffix = A;
        streetDirectional = NW;
        streetThoroughfare = DR;
        streetName = SKYLARK;
        streetSuffix = E;
        room = "APT G";
        building = BLDG1;
        floor = "FLR 3";
        descriptiveLocation = SEAGATE;
        city = MIAMI;
        state = FL;
        zipCode = 999999;
        unnumberedHouseIndicator = N;
        crossBoundary = GA;
        postalBox = 123;
        route = 14W;
};
```

**xstTestClient Output:**

```
status={
 msgId=BLPW906ADR;
 msgTxt="ADDRESS INFORMATION ONLY/ASSIGN AHN";
```

```
};
messageHeader={
 inquiryNumber=5a29272100000001;
 dateSent=1999090316082153;
};
alternativeaddressInformation=(
 {
  alternativeaddress={
   addresswithrange={
    houseNumberRange={
     fromHouseNumber=1;
     toHouseNumber=199;
    };
    oddEvenIndicator=B;
    assignedHouseNumberStatus=2;
    streetDirectional=;
    streetThoroughfare=BLVD;
    streetName=CRANE;
    city="SUGARLOAF KEY";
    state=FL;
    zipCode=33042;
   };
  };
  localServiceTermination=SGKYFLMA;
  telephoneInfo=(
  );
  areaTransferInfo={
  };
 };
);
addressInfoHeader={
 serviceRestrictionInfoList=(
 );
};
```

2.1.15 tag.clec.avq15 Area Transfer, Restrictions and Instructions (A)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Response with area transfer information, service restrictions and service instructions.

**Interface Type:** xstaAddressValidation.verifyWithAddress

**Special Instructions:**

**Note:** The data shown is not current, data not available.

**API Input:**

```
Address{

 houseNumber=100;
 houseNumberSuffix=;
 streetDirectional=0;
 streetThoroughfare=LN;
 streetName=BELLS CAMP;
 streetSuffix=;
 room=;
 building=;
 floor=;
 descriptiveLocation=;
 city=BROCKPORT;
 state=NY;
 zipCode=;
 unnumberedHouseIndicator=0;
 crossBoundary=;
 postalBox=;
 route=;
 addressPattern{
  0{
   structurePattern=;
   elevationPattern=;
   unitPattern=;
  }
  1{
   structurePattern=;
   elevationPattern=;
   unitPattern=;
  }
 }
}
```

**API Output:**

```
Status{
 msgId=BLP0000ADR;
 msgTxt=COMPLETED SUCCESSFULLY;
}
AlternativeAddressList{
 0{
  alternativeaddress{
   fieldedaddress{
```

```
    houseNumber=100;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=LN;
    streetName=BELLS CAMP;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=BROCKPORT;
    state=NY;
    zipCode=14512;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=IQ;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=716616;
  localServiceTermination=ROCHNY01;
  telephoneInfo{
   0{
    telephoneNumber=4074443333;
    quickServeIndicator=N;
    addressStatus=W;
    availableFacilitiesIndicator=N;
   }
  }
  areaTransferInfo{
   areaTransferCutDate=19990909;
   areaTransferNumChgDate=19990909;
   areaTransferNpaNxx=407745;
  }
 }
}
AddressInfoHeader{
 ServiceRestrictionList{
  0{
   estServiceDate=20000104;
   restrictionCode=NF;
   restrictionText=NO FACILITIES;
  }
 }
ServiceInstructionText=THIS IS A TEST OF THE NUMBER OF SERVICE INSTRUCTIONS THA T CAN
BE SENT TO CONTRA   CTS VIA RSAGADDR CONTRACT... BASED UPON SERVICE INSTRUCTIONS BEING
ADDED AT THE   GLOBAL LEVELS....THIS ONE IS AT THE GLOBAL LEVEL OF NY1 (EXCHANGE).
IT IS FI   VE LINES LONG, SO THAT THIS TEST CAN BE AS COMPLETE AS POSSIBLE.  SERVICE
INSTR   AT THE BASIC ADDRESS LEVEL ARE ONLY 2 LINES LONG, AT 70-SOME CHAR'S PER LINE.
```

```
THIS IS A TEST OF THE NUMBER OF SERVICE INSTRUCTIONS THAT CAN BE SENT TO CONTRA   CTS
VIA RSAGADDR CONTRACT... BASED UPON SERVICE INSTRUCTIONS BEING ADDED AT THE   GLOBAL
LEVELS....THIS ONE IS AT THE GLOBAL LEVEL OF NY1 (EXCHANGE).   IT IS FI   VE LINES
LONG, SO THAT THIS TEST CAN BE AS COMPLETE AS POSSIBLE.  SERVICE INSTR   AT THE BASIC
ADDRESS LEVEL ARE ONLY 2 LINES LONG, AT 70-SOME CHAR'S PER LINE.;
}
```

**xstTestClient Input:**

```
RequestType=avq;
Address ={
       houseNumber=100;
       streetThoroughfare=LN
       streetName= BELLS CAMP;
       city=BROCKPORT;
       state=NY
       unnumberedHouseIndicator=N;
}
```

**xstTestClient Output:**

### 2.1.16 tag.clec.avq16 Range of House Numbers (H)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Response with range of house numbers.

**Interface Type:** xstaAddressValidation.verifyWithAddress

**Special Instructions:** None

**API Input:**

```
Address{
 houseNumber=;
 houseNumberSuffix=;
 streetDirectional=0;
 streetThoroughfare=;
 streetName=;
 streetSuffix=;
 room=;
 building=;
 floor=;
 descriptiveLocation=;
 city=FT PIERCE;
 state=FL;
 zipCode=99999;
 unnumberedHouseIndicator=0;
 crossBoundary=;
 postalBox=;
 route=;
 rateZone=;
 driveInstructions=;
 companyIndicator=;
 addressPattern{
 }
}
```

**API Output:**

```
Status{
 msgId=BLPE905ADR;
 msgTxt=NO MATCH ON HOUSE NUMBER OR AHN;
}
AlternativeAddressList{
 0{
  alternativeaddress{
   addresswithrange{
        houseNumber=;
    houseNumberRange{
     fromHouseNumber=5;
     toHouseNumber=500;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=0;
    streetThoroughfare=<null>;
    streetName=BIMINI;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=MACEDON;
    state=NY;
    zipCode=<null>;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
}
AddressInfoHeader{
 ServiceRestrictionList{
 }
 ServiceInstructionText=<null>;
}
```

**xstTestClient Input:**

```
RequestType = avq;
```

```
address = {
        city = "FT PIERCE";
        state = FL;
        zipCode = 99999;
        unnumberedHouseIndicator = N;
};
```

**xstTestClient Output:**

```
status={
 msgId=BLPE905ADR;
 msgTxt="NO MATCH ON HOUSE NUMBER OR AHN";
};
messageHeader={
 inquiryNumber=50b67b1200000001;
 dateSent=1999090316302122;
};
alternativeaddressInformation=(
 {
  alternativeaddress={
   addresswithrange={
    houseNumberRange={
     fromHouseNumber=5;
     toHouseNumber=500;
    };
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=;
    streetName=BIMINI;
    city=MACEDON;
    state=NY;
   };
  };
  telephoneInfo=(
  );
  areaTransferInfo={
  };
 };
);
addressInfoHeader={
 serviceRestrictionInfoList=(
 );
};
```

## 2.1.17 tag.clec.avq17 Menu of Basic Addresses - Location (K)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Response with a menu of basic addresses for a descriptive location.

**Interface Type:** xstaAddressValidation.verifyWithAddress

**Special Instructions:** None

**API Input:**

```
Address{
 houseNumber=5455;
 houseNumberSuffix=;
 streetDirectional=0;
 streetThoroughfare=DR;
 streetName=SEAWAY;
 streetSuffix=;
 room=;
 building=;
 floor=;
 descriptiveLocation=;
 city=PORT SAINT LUCIE;
 state=FL;
 zipCode=;
 unnumberedHouseIndicator=0;
 crossBoundary=;
 postalBox=;
 route=;
 rateZone=;
 driveInstructions=;
 companyIndicator=;
 addressPattern{
 }
}
```

**API Output:**

```
Status{
 msgId=BLPE919ADR;
 msgTxt=MORE THAN ONE MATCH ON DESCRIPTIVE. SELECT BASIC ADDRESS;
}
AlternativeAddressList{
 0{
  alternativeaddress{
   fieldedaddress{
     houseNumber=3;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=AV;
    streetName=PAYNE;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
```

```
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
   0{
    telephoneNumber=4043931005;
    quickServeIndicator=<null>;
    addressStatus=W;
    availableFacilitiesIndicator=<null>;
   }
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
 1{
  alternativeaddress{
   fieldedaddress{

    houseNumber=17;
    houseNumberSuffix=<null>;
    streetDirectional=0;
    streetThoroughfare=AV;
    streetName=PAYNE;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
```

```
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
   0{
    telephoneNumber=<null>;
    quickServeIndicator=<null>;
    addressStatus=<null>;
    availableFacilitiesIndicator=<null>;
   }
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
}
AddressInfoHeader{
 ServiceRestrictionList{
 }
 ServiceInstructionText=<null>;
}
```

**xstTestClient Input:**

```
RequestType = avq;
address = {
        houseNumber = 5455;
        streetThoroughfare = DR;
        streetName = SEAWAY;
        city = "PORT SAINT LUCIE";
        state = FL;
        unnumberedHouseIndicator = N;
};
```

**xstTestClient Output:**

```
status={
 msgId=BLPE919ADR;
 msgTxt="MORE THAN ONE MATCH ON DESCRIPTIVE. SELECT BASIC ADDRESS";
};
messageHeader={
 inquiryNumber=76b8d83000000001;
 dateSent=1999090316333231;
};
alternativeaddressInformation=(
 {
  alternativeaddress={
   fieldedaddress={
    houseNumber=3;
```

```
    streetDirectional=;
    streetThoroughfare=AV;
    streetName=PAYNE;
    city="FORT PIERCE";
    state=FL;
    unnumberedHouseIndicator=N;
    addressPattern=(
     {
     };
    );
   };
  };
  telephoneInfo=(
   {
    telephoneNumber=4043931005;
    addressStatus=W;
   };
  );
  areaTransferInfo={
  };
 };
 {
  alternativeaddress={
   fieldedaddress={
    houseNumber=17;
    streetDirectional=;
    streetThoroughfare=AV;
    streetName=PAYNE;
    city="FORT PIERCE";
    state=FL;
    unnumberedHouseIndicator=N;
    addressPattern=(
     {
     };
    );
   };
  };
  telephoneInfo=(
   {
   };
  );
  areaTransferInfo={
  };
 };
);
addressInfoHeader={
 serviceRestrictionInfoList=(
 );
};
```

2.1.18 tag.clec.avq18 Summary Information (M)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Response with summary information.

**Interface Type:** xstaAddressValidation.verifyWithAddress

**Special Instructions:** None

**API Input:**

```
Address{

 houseNumber=101;
 houseNumberSuffix=A;
 streetDirectional=6;
 streetThoroughfare=DR;
 streetName=SKYLARK;
 streetSuffix=E;
 room=APT G;
 building=BLDG1;
 floor=FLR 3;
 descriptiveLocation=SEAGATE;
 city=MIAMI;
 state=FL;
 zipCode=999999;
 unnumberedHouseIndicator=0;
 crossBoundary=GA;
 postalBox=123;
 route=14W;
 rateZone=;
 driveInstructions=;
 companyIndicator=;
 addressPattern{
 }
}
```

**API Output:**

```
Status{
 msgId=BLPW906ADR;
 msgTxt=ADDRESS INFORMATION ONLY/ASSIGN AHN;
}
AlternativeAddressList{
 0{
  alternativeaddress{
   addresswithrange{
        houseNumber=;
    houseNumberRange{
     fromHouseNumber=1;
     toHouseNumber=199;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=2;
    streetDirectional=0;
```

```
    streetThoroughfare=BLVD;
    streetName=CRANE;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=SUGARLOAF KEY;
    state=FL;
    zipCode=33042;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=SGKYFLMA;
  telephoneInfo{
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
}
AddressInfoHeader{
 ServiceRestrictionList{
 }
 ServiceInstructionText=<null>;
```

**xstTestClient Input:**

```
RequestType = avq;
inquiryNumber = 5a29272100000001;
address = {
        houseNumber = 101;
        houseNumberSuffix = A;
        streetDirectional = NW;
        streetThoroughfare = DR;
        streetName = SKYLARK;
        streetSuffix = E;
        room = "APT G";
        building = BLDG1;
        floor = "FLR 3";
        descriptiveLocation = SEAGATE;
        city = MIAMI;
        state = FL;
        zipCode = 999999;
        unnumberedHouseIndicator = N;
        crossBoundary = GA;
        postalBox = 123;
        route = 14W;
};
```

**xstTestClient Output:**

```
status={
 msgId=BLPW906ADR;
 msgTxt="ADDRESS INFORMATION ONLY/ASSIGN AHN";
};
messageHeader={
 inquiryNumber=5a29272100000001;
 dateSent=1999090316373740;
};
alternativeaddressInformation=(
 {
  alternativeaddress={
   addresswithrange={
    houseNumberRange={
     fromHouseNumber=1;
     toHouseNumber=199;
    };
    oddEvenIndicator=B;
    assignedHouseNumberStatus=2;
    streetDirectional=;
    streetThoroughfare=BLVD;
    streetName=CRANE;
    city="SUGARLOAF KEY";
    state=FL;
    zipCode=33042;
   };
  };
  localServiceTermination=SGKYFLMA;
  telephoneInfo=(
  );
  areaTransferInfo={
  };
 };
);
addressInfoHeader={
 serviceRestrictionInfoList=(
 );
};
```

2.1.19 tag.clec.avq19 Menu of Address Telephones (N)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Response with a menu of address telephones.

**Interface Type:** xstaAddressValidation.verifyWithTelephone

**Special Instructions:** None

**API Input:**

```
telephone=4074646476
```

**API Output:**

```
Status{
```

```
 msgId=BLPE921ADR;
 msgTxt=MORE THAN ONE MATCH ON TELEPHONE/CIRCUIT NUMBER;
}
AlternativeAddressList{
 0{
  alternativeaddress{
   fieldedaddress{

    houseNumber=39;
    houseNumberSuffix=AHN;
    streetDirectional=0;
    streetThoroughfare=BLVD;
    streetName=CRANE;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=SUGARLOAF KEY;
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
   0{
    telephoneNumber=3057452661;
    quickServeIndicator=<null>;
    addressStatus=W;
    availableFacilitiesIndicator=<null>;
   }
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
 1{
  alternativeaddress{
   fieldedaddress{
        houseNumber=39;
    houseNumberSuffix=AHN;
```

```
    streetDirectional=0;
    streetThoroughfare=BLVD;
    streetName=CRANE;
    streetSuffix=<null>;
    room=UNITPOLE;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=SUGARLOAF KEY;
    state=FL;
    zipCode=<null>;
    unnumberedHouseIndicator=0;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
    addressPattern{
     0{
      structurePattern=<null>;
      elevationPattern=<null>;
      unitPattern=<null>;
     }
    }
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
   0{
    telephoneNumber=3057452661;
    quickServeIndicator=<null>;
    addressStatus=N;
    availableFacilitiesIndicator=<null>;
   }
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
}
AddressInfoHeader{
 ServiceRestrictionList{
 }
 ServiceInstructionText=<null>;
}
```

**xstTestClient Input:**

```
RequestType = avq_tn;
queryTelephone = 4074646476;
```

**xstTestClient Output:**

```
status={
 msgId=BLPE921ADR;
 msgTxt="MORE THAN ONE MATCH ON TELEPHONE/CIRCUIT NUMBER";
};
messageHeader={
 inquiryNumber=15afa8b300000001;
 dateSent=1999090317062397;
};
alternativeaddressInformation=(
 {
  alternativeaddress={
   fieldedaddress={
    houseNumber=39;
    houseNumberSuffix=AHN;
    streetDirectional="";
    streetThoroughfare=BLVD;
    streetName=CRANE;
    city="SUGARLOAF KEY";
    state=FL;
    unnumberedHouseIndicator=N;
    addressPattern=(
     {
     };
    );
   };
  };
  telephoneInfo=(
   {
    telephoneNumber=3057452661;
    addressStatus=W;
   };
  );
  areaTransferInfo={
  };
 };
 {
  alternativeaddress={
   fieldedaddress={
    houseNumber=39;
    houseNumberSuffix=AHN;
    streetDirectional="";
    streetThoroughfare=BLVD;
    streetName=CRANE;
    room=UNITPOLE;
    city="SUGARLOAF KEY";
    state=FL;
    unnumberedHouseIndicator=N;
    addressPattern=(
     {
     };
    );
   };
  };
  telephoneInfo=(
   {
    telephoneNumber=3057452661;
    addressStatus=N;
   };
  );
```

```
  areaTransferInfo={
  };
 };
);
addressInfoHeader={
 serviceRestrictionInfoList=(
 );
};
```

2.1.20 tag.clec.avq20 Menu of Basic Addresses - Street (O)

**Purpose:** This test demonstrates that the CLEC Client Application can process an Address Validation Response with menu of basic addresses for a street.

**Interface Type:** xstaAddressValidation.verifyWithAddress

**Special Instructions:** None

**API Input:**

```
Address{
 houseNumber=7800;
 houseNumberSuffix=;
 streetDirectional=0;
 streetThoroughfare=ST;
 streetName=BANYON;
 streetSuffix=;
 room=;
 building=;
 floor=;
 descriptiveLocation=;
 city=FT PERCE;
 state=FL;
 zipCode=;
 unnumberedHouseIndicator=0;
 crossBoundary=;
 postalBox=;
 route=;
 rateZone=;
 driveInstructions=;
 companyIndicator=;
 addressPattern{
 }
}
```

**API Output:**

```
Status{
 msgId=BLPW908ADR;
 msgTxt=RNS - THIS ADDRESS ALSO HAS LIVING UNITS WITH SUPPLEMENTAL ADDRE;
}
AlternativeAddressList{
 0{
  alternativeaddress{
   addresswithrange{

    houseNumber=1002;
```

```
    houseNumberRange{
     fromHouseNumber=400;
     toHouseNumber=2099;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=0;
    streetThoroughfare=DR;
    streetName=SEAWAY;
    streetSuffix=<null>;
    room=<null>;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
   0{
    telephoneNumber=<null>;
    quickServeIndicator=<null>;
    addressStatus=<null>;
    availableFacilitiesIndicator=<null>;
   }
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
 1{
  alternativeaddress{
   addresswithrange{
        houseNumber=1002;
    houseNumberRange{
     fromHouseNumber=400;
     toHouseNumber=2099;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=0;
    streetThoroughfare=DR;
    streetName=SEAWAY;
    streetSuffix=<null>;
    room=<null>;
```

```
   building=<null>;
   floor=<null>;
   descriptiveLocation=<null>;
   city=FORT PIERCE;
   state=FL;
   zipCode=<null>;
   crossBoundary=<null>;
   postalBox=<null>;
   route=<null>;
   rateZone=<null>;
   driveInstructions=<null>;
   companyIndicator=<null>;
  }
 }
 npanxx=<null>;
 localServiceTermination=<null>;
 telephoneInfo{
  0{
   telephoneNumber=4074663666;
   quickServeIndicator=<null>;
   addressStatus=N;
   availableFacilitiesIndicator=<null>;
  }
 }
 areaTransferInfo{
  areaTransferCutDate=<null>;
  areaTransferNumChgDate=<null>;
  areaTransferNpaNxx=<null>;
  tariffExchangeCode=<null>;
 }
}
2{
 alternativeaddress{
  addresswithrange{

   houseNumber=1002;
   houseNumberRange{
    fromHouseNumber=400;
    toHouseNumber=2099;
   }
   houseNumberSuffix=<null>;
   oddEvenIndicator=B;
   assignedHouseNumberStatus=1;
   streetDirectional=0;
   streetThoroughfare=DR;
   streetName=SEAWAY;
   streetSuffix=<null>;
   room=APT A;
   building=<null>;
   floor=<null>;
   descriptiveLocation=<null>;
   city=FORT PIERCE;
   state=FL;
   zipCode=<null>;
   crossBoundary=<null>;
   postalBox=<null>;
   route=<null>;
   rateZone=<null>;
   driveInstructions=<null>;
```

```
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
   0{
    telephoneNumber=4074688588;
    quickServeIndicator=<null>;
    addressStatus=W;
    availableFacilitiesIndicator=<null>;
   }
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
 3{
  alternativeaddress{
   addresswithrange{

    houseNumber=1002;
    houseNumberRange{
     fromHouseNumber=400;
     toHouseNumber=2099;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=0;
    streetThoroughfare=DR;
    streetName=SEAWAY;
    streetSuffix=<null>;
    room=APT A;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
   0{
    telephoneNumber=4074645356;
    quickServeIndicator=<null>;
    addressStatus=W;
    availableFacilitiesIndicator=<null>;
```

```
  }
 }
 areaTransferInfo{
  areaTransferCutDate=<null>;
  areaTransferNumChgDate=<null>;
  areaTransferNpaNxx=<null>;
  tariffExchangeCode=<null>;
 }
}
4{
 alternativeaddress{
  addresswithrange{

   houseNumber=1002;
   houseNumberRange{
    fromHouseNumber=400;
    toHouseNumber=2099;
   }
   houseNumberSuffix=<null>;
   oddEvenIndicator=B;
   assignedHouseNumberStatus=1;
   streetDirectional=0;
   streetThoroughfare=DR;
   streetName=SEAWAY;
   streetSuffix=<null>;
   room=APT B;
   building=<null>;
   floor=<null>;
   descriptiveLocation=<null>;
   city=FORT PIERCE;
   state=FL;
   zipCode=<null>;
   crossBoundary=<null>;
   postalBox=<null>;
   route=<null>;
   rateZone=<null>;
   driveInstructions=<null>;
   companyIndicator=<null>;
  }
 }
 npanxx=<null>;
 localServiceTermination=<null>;
 telephoneInfo{
  0{
   telephoneNumber=4074684826;
   quickServeIndicator=<null>;
   addressStatus=W;
   availableFacilitiesIndicator=<null>;
  }
 }
 areaTransferInfo{
  areaTransferCutDate=<null>;
  areaTransferNumChgDate=<null>;
  areaTransferNpaNxx=<null>;
  tariffExchangeCode=<null>;
 }
}
5{
 alternativeaddress{
```

```
  addresswithrange{
       houseNumber=1002;
   houseNumberRange{
    fromHouseNumber=400;
    toHouseNumber=2099;
   }
   houseNumberSuffix=<null>;
   oddEvenIndicator=B;
   assignedHouseNumberStatus=1;
   streetDirectional=0;
   streetThoroughfare=DR;
   streetName=SEAWAY;
   streetSuffix=<null>;
   room=APT C;
   building=<null>;
   floor=<null>;
   descriptiveLocation=<null>;
   city=FORT PIERCE;
   state=FL;
   zipCode=<null>;
   crossBoundary=<null>;
   postalBox=<null>;
   route=<null>;
   rateZone=<null>;
   driveInstructions=<null>;
   companyIndicator=<null>;
  }
 }
 npanxx=<null>;
 localServiceTermination=<null>;
 telephoneInfo{
  0{
   telephoneNumber=4074659007;
   quickServeIndicator=<null>;
   addressStatus=W;
   availableFacilitiesIndicator=<null>;
  }
 }
 areaTransferInfo{
  areaTransferCutDate=<null>;
  areaTransferNumChgDate=<null>;
  areaTransferNpaNxx=<null>;
  tariffExchangeCode=<null>;
 }
}
6{
 alternativeaddress{
  addresswithrange{
       houseNumber=1002;
   houseNumberRange{
    fromHouseNumber=400;
    toHouseNumber=2099;
   }
   houseNumberSuffix=<null>;
   oddEvenIndicator=B;
   assignedHouseNumberStatus=1;
   streetDirectional=0;
   streetThoroughfare=DR;
   streetName=SEAWAY;
```

```
    streetSuffix=<null>;
    room=APT D;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
   0{
    telephoneNumber=4074606777;
    quickServeIndicator=<null>;
    addressStatus=W;
    availableFacilitiesIndicator=<null>;
   }
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
 7{
  alternativeaddress{
   addresswithrange{
        houseNumber=1002;
    houseNumberRange{
     fromHouseNumber=400;
     toHouseNumber=2099;
    }
    houseNumberSuffix=<null>;
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional=0;
    streetThoroughfare=DR;
    streetName=SEAWAY;
    streetSuffix=<null>;
    room=APT E;
    building=<null>;
    floor=<null>;
    descriptiveLocation=<null>;
    city=FORT PIERCE;
    state=FL;
    zipCode=<null>;
    crossBoundary=<null>;
    postalBox=<null>;
    route=<null>;
    rateZone=<null>;
```

```
    driveInstructions=<null>;
    companyIndicator=<null>;
   }
  }
  npanxx=<null>;
  localServiceTermination=<null>;
  telephoneInfo{
   0{
    telephoneNumber=<null>;
    quickServeIndicator=<null>;
    addressStatus=<null>;
    availableFacilitiesIndicator=<null>;
   }
  }
  areaTransferInfo{
   areaTransferCutDate=<null>;
   areaTransferNumChgDate=<null>;
   areaTransferNpaNxx=<null>;
   tariffExchangeCode=<null>;
  }
 }
}
AddressInfoHeader{
 ServiceRestrictionList{
 }
 ServiceInstructionText=<null>;
}
```

**xstTestClient Input:**

```
RequestType = avq;
address = {
        houseNumber = 7800;
        streetThoroughfare = ST;
        streetName = BANYON;
        city = "FT PERCE";
        state = FL;
        unnumberedHouseIndicator = N;
};
```

**xstTestClient Input:**

```
status={
 msgId=BLPW908ADR;
 msgTxt="RNS - THIS ADDRESS ALSO HAS LIVING UNITS WITH SUPPLEMENTAL ADDRE";
};
messageHeader={
 inquiryNumber=01bc482800000001;
 dateSent=1999090317101944;
};
alternativeaddressInformation=(
 {
  alternativeaddress={
   addresswithrange={
    houseNumber=1002;
    houseNumberRange={
     fromHouseNumber=400;
```

```
    toHouseNumber=2099;
   };
   oddEvenIndicator=B;
   assignedHouseNumberStatus=1;
   streetDirectional="";
   streetThoroughfare=DR;
   streetName=SEAWAY;
   city="FORT PIERCE";
   state=FL;
  };
 };
 telephoneInfo=(
  {
  };
 );
 areaTransferInfo={
 };
};
{
 alternativeaddress={
  addresswithrange={
   houseNumber=1002;
   houseNumberRange={
    fromHouseNumber=400;
    toHouseNumber=2099;
   };
   oddEvenIndicator=B;
   assignedHouseNumberStatus=1;
   streetDirectional="";
   streetThoroughfare=DR;
   streetName=SEAWAY;
   city="FORT PIERCE";
   state=FL;
  };
 };
 telephoneInfo=(
  {
   telephoneNumber=4074663666;
   addressStatus=N;
  };
 );
 areaTransferInfo={
 };
};
{
 alternativeaddress={
  addresswithrange={
   houseNumber=1002;
   houseNumberRange={
    fromHouseNumber=400;
    toHouseNumber=2099;
   };
   oddEvenIndicator=B;
   assignedHouseNumberStatus=1;
   streetDirectional="";
   streetThoroughfare=DR;
   streetName=SEAWAY;
   room="APT A";
   city="FORT PIERCE";
```

```
    state=FL;
   };
  };
  telephoneInfo=(
   {
    telephoneNumber=4074688588;
    addressStatus=W;
   };
  );
  areaTransferInfo={
  };
 };
 {
  alternativeaddress={
   addresswithrange={
    houseNumber=1002;
    houseNumberRange={
     fromHouseNumber=400;
     toHouseNumber=2099;
    };
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional="";
    streetThoroughfare=DR;
    streetName=SEAWAY;
    room="APT A";
    city="FORT PIERCE";
    state=FL;
   };
  };
  telephoneInfo=(
   {
    telephoneNumber=4074645356;
    addressStatus=W;
   };
  );
  areaTransferInfo={
  };
 };
 {
  alternativeaddress={
   addresswithrange={
    houseNumber=1002;
    houseNumberRange={
     fromHouseNumber=400;
     toHouseNumber=2099;
    };
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional="";
    streetThoroughfare=DR;
    streetName=SEAWAY;
    room="APT B";
    city="FORT PIERCE";
    state=FL;
   };
  };
  telephoneInfo=(
   {
```

```
    telephoneNumber=4074684826;
    addressStatus=W;
   };
  );
  areaTransferInfo={
  };
 };
 {
  alternativeaddress={
   addresswithrange={
    houseNumber=1002;
    houseNumberRange={
     fromHouseNumber=400;
     toHouseNumber=2099;
    };
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional="";
    streetThoroughfare=DR;
    streetName=SEAWAY;
    room="APT C";
    city="FORT PIERCE";
    state=FL;
   };
  };
  telephoneInfo=(
   {
    telephoneNumber=4074659007;
    addressStatus=W;
   };
  );
  areaTransferInfo={
  };
 };
 {
  alternativeaddress={
   addresswithrange={
    houseNumber=1002;
    houseNumberRange={
     fromHouseNumber=400;
     toHouseNumber=2099;
    };
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional="";
    streetThoroughfare=DR;
    streetName=SEAWAY;
    room="APT D";
    city="FORT PIERCE";
    state=FL;
   };
  };
  telephoneInfo=(
   {
    telephoneNumber=4074606777;
    addressStatus=W;
   };
  );
  areaTransferInfo={
```

```
  };
 };
 {
  alternativeaddress={
   addresswithrange={
    houseNumber=1002;
    houseNumberRange={
     fromHouseNumber=400;
     toHouseNumber=2099;
    };
    oddEvenIndicator=B;
    assignedHouseNumberStatus=1;
    streetDirectional="";
    streetThoroughfare=DR;
    streetName=SEAWAY;
    room="APT E";
    city="FORT PIERCE";
    state=FL;
   };
  };
  telephoneInfo=(
   {
   };
  );
  areaTransferInfo={
  };
 };
);
addressInfoHeader={
 serviceRestrictionInfoList=(
 );
};
```

## 2.2 Appointment Availability

Appointment Availability test cases include two normal responses, a match and a non-match condition. One additional test case is included which returns the maximum size response allowed by the API data structures.

### 2.2.1 tag.clec.aaq01 Request Using Valid NPA-NXX

**Purpose:** This test demonstrates that the CLEC Client Application can process an Appointment Availability Request with valid NPA-NXX input.

**Interface Type:** xstaAppointmentScheduling.verifyAvailAppts

**Special Instructions:** None

**API Input:**

```
npanxx=770992;
```

**API Output:**

```
Status{
 msgId=BLP0000AAV;
 msgTxt=TRANSACTION COMPLETED SUCCESSFULLY;
}
AppointmentData{
 coAvailability{
  0=YYYYYYN;
 }
 pvAvailability{
  0=YYYYYYN;
 }
 interval{
  0{
   pvResidentialReinstall=02;
   reinstall3=03;
   residentialNewinstall1_2=02;
   newinstall3=05;
   newinstall4=06;
   newinstall5=07;
   newinstall6_10=08;
   newinstall11_15=10;
   addLine=03;
   residentialInsideWire=02;
   pvBusinessReinstall=02;
   businessNewinstall1_2=02;
   businessInsideWire=02;
   quickService=00;
  }
 }
 holidays{
  0=19981225;
  1=19991125;
 }
 closeDates{
  0{
   closeDate=19981213;
   reasonCode1=<null>;
   reasonCode2=M;
  }
  1{
   closeDate=19981214;
   reasonCode1=<null>;
   reasonCode2=M;
  }
  2{
   closeDate=19981215;
   reasonCode1=<null>;
   reasonCode2=M;
  }
  3{
   closeDate=19981216;
   reasonCode1=R;
   reasonCode2=M;
  }
```

**xstTestClient Input:**

```
RequestType = aaq;
npanxx = 770992;
```

**xstTestClient Output:**

```
status={
 msgId=BLP0000AAV;
 msgTxt="TRANSACTION COMPLETED SUCCESSFULLY";
};
npanxx=770992;
messageHeader={
 inquiryNumber=60c8813f00000001;
 dateSent=1999090317200811;
};
availableAppointments={
 coAvailability=(
  (
   Y;
   Y;
   Y;
   Y;
   Y;
   Y;
   Y;
  );
 );
 pvAvailability=(
  (
   Y;
   Y;
   Y;
   Y;
   Y;
   Y;
   Y;
  );
 );
 interval=(
  {
   pvResidentialReinstall=02;
   reinstall3=03;
   residentialNewinstall1_2=02;
   newinstall3=05;
   newinstall4=06;
   newinstall5=07;
   newinstall6_10=08;
   newinstall11_15=10;
   addLine=03;
   residentialInsideWire=02;
   pvBusinessReinstall=02;
   businessNewinstall1_2=02;
   businessInsideWire=02;
   quickService=00;
  };
 );
 holidays=(
  19981225;
  19991125;
 );
 closeDates=(
```

```
  {
   closeDate=19981213;
   reasonCode2=M;
  };
  {
   closeDate=19981214;
   reasonCode2=M;
  };
  {
   closeDate=19981215;
   reasonCode2=M;
  };
  {
   closeDate=19981216;
   reasonCode1=R;
   reasonCode2=M;
  };
 );
};
```

### 2.2.2 tag.clec.aaq02 — No Match

**Purpose:** This test demonstrates that the CLEC Client Application can process an Appointment Availability Request which returns no data.

**Interface Type:** xstaAppointmentScheduling.verifyAvailAppts

**Special Instructions:** None

**API Input:**

```
npanxx=777999;
```

**API Output:**

```
FAILURE: invalid data
         inquiry number: 5d51589700000001
         data element: mapDSAPRsp
         msgId: BLP1004COM
         msgTxt: INVALID NPA 777    ;CANNOT PROCESS TRANSACTION
```

**xstTestClient Input:**

```
RequestType = aaq;
npanxx = 777999;
```

**xstTestClient Output:**

```
FAILURE: invalid data
         inquiry number: 5d51589700000001
         data element: mapDSAPRsp
         msgId: BLP1004COM
         msgTxt: INVALID NPA 777    ;CANNOT PROCESS TRANSACTION
```

## 2.3 Customer Service

### 2.3.1 tag.clec.csr01 CR - Match by Account Number

**Purpose:** This test demonstrates that the CLEC Client Application can process a Customer Record Request with a valid BST thirteen character account number input.

**Interface Type:** xstaCustomerRecord.retrieveCustomerRecord

**Special Instructions:** The CLEC will be asked to check for specific components within the customerTextList value returned.

**API Input**

```
customerRequest{
stateCode=;
 customerCode=130;
 exchangeCompanyCircuitId=;
 accountTelephoneNumber=4234819999;
 agencyAuthorizationStatus=Y;
 authorizationName=BARBARASAIDSO;
 authorizationDate=19971120;
 lsiIndicator=N;
}
```

**API Output**

```
Status{
 msgId=BLP0000CSR;
 msgTxt=CSR Transaction Completed Successfully.;
}
customerTextList{
```

**xstTestClient Input:**

```
RequestType = csrq;
customerRequest = {
        agencyAuthorizationStatus = Y;
        authorizationName = BARBARASAIDSO;
        authorizationDate = 19971120;
        stateCode = AT;
        exchangeCompanyCircuitId = 40IBS4519841SB;
        customerCode = 329;
        accountTelephoneNumber = 7709873275;
};
```

**xstTestClient Output:**

```
status={
 msgId=BLP0000CSR;
 msgTxt="CSR Transaction Completed Successfully.";
};
messageHeader={
 inquiryNumber=503e259e00000001;
 dateSent=1999090317423779;
};
customerTextList=(
 9049924110345;
);
```

## 2.3.2 tag.clec.csr02 Match Using Circuit ID

**Purpose:** This test demonstrates that the CLEC Client Application can process a Customer Record Request with a valid BST circuit identifier input.

**Interface Type:** xstaCustomerRecord.retrieveCustomerRecord

**Special Instructions:** None

**API Input:**

```
customerRequest{
 stateCode=AT;
 customerCode=;
 exchangeCompanyCircuitId=40IBS4519841SB;
 accountTelephoneNumber=;
 agencyAuthorizationStatus=Y;
 authorizationName=BARBARASAIDSO;
 authorizationDate=19971120;
 lsiIndicator=N;
}
```

**API Output:**

```
Status{
 msgId=BLP0011CSR;
 msgTxt=ACCOUNT NUMBER FOUND FOR CIRCUIT ID REQUEST.  RESUBMIT QUERY USING ACCOUNT
7709873275329.;
}
customerTextList{
 0=7709873275329;
}
```

**xstTestClient Input:**

```
RequestType = csrq;
customerRequest = {
        agencyAuthorizationStatus = Y;
        authorizationName = BARBARASAIDSO;
        authorizationDate = 19971120;
        stateCode = AT;
        exchangeCompanyCircuitId = 40IBS4519841SB;
        customerCode = "";
        accountTelephoneNumber = "";
};
```

**xstTestClient Output:**

```
status={
 msgId=BLP0011CSR;
 msgTxt="Account number found for circuit ID request.  Resubmit query using account
7709873275329.";
};
messageHeader={
 inquiryNumber=4232d73a00000001;
 dateSent=1999090317451870;
};
customerTextList=(
 7709873275329;
```

```
);
```

### 2.3.3 tag.clec.csr03 — No Match

**Purpose:** This test demonstrates that the CLEC Client Application can process a Customer Record request for an account which does not exist.

**Interface Type:** xstaCustomerRecord.retrieveCustomerRecord

**Special Instructions:** None

**Note:** The data format is not current, data not available.

**API Input:**

```
customerRequest{
 stateCode=;
 customerCode=;
 exchangeCompanyCircuitId=;
 accountTelephoneNumber=4045899999;
 agencyAuthorizationStatus=Y;
 authorizationName=BARBARASAIDSO;
 authorizationDate=19971120;
 lsiIndicator=0;
}
```

**API Output:**

```
InvalidData Exception
data element: mapRSAGRsp
msgId: BLP1003CSR
msgTxt: ACCOUNT 4045899999    NOT FOUND IN BELLSOUTH CSR DATABASE.
```

**xstTestClient Input:**

```
AGENCYAUTH:Y|
AUTHNAME:BARBARASAIDSO|
DATEAGENCYAUTH:19971120|
STATECODE:|
ECCKT:|
CUSTCODE:|
TELEPHONE:4045899999|
LSIIND:N|
```

### 2.3.4 tag.clec.csr04 — Restricted Account

**Purpose:** This test demonstrates that the CLEC Client Application can process a Customer Record request for a BST account which has either CPNI or reseller restrictions.

**Interface Type:** xstaCustomerRecord.retrieveCustomerRecord

**Special Instructions:** None

**Note:** The data format shown is not current, data not available.

**API Input:**

```
customerRequest{
 stateCode=;
 customerCode=;
 exchangeCompanyCircuitId=;
 accountTelephoneNumber=4043519999;
 agencyAuthorizationStatus=Y;
 authorizationName=BARBARASAIDSO;
 authorizationDate=19971120;
 lsiIndicator=0;
}
```

**API Output:**

```
InvalidData Exception
data element: mapRSAGRsp
msgId: BLP1004CSR
msgTxt: BELLSOUTH IS NOT AUTHORIZED TO PROVIDE INFORMATION ON THIS ACCOUNT,
4043519999117
```

**xstTestClient Input:**

```
AGENCYAUTH:Y|
AUTHNAME:BARBARASAIDSO|
DATEAGENCYAUTH:19971120|
STATECODE:|
ECCKT:|
CUSTCODE:|
TELEPHONE:4043519999|
LSIIND:N|
```

### 2.3.5 tag.clec.csr05 LSI - Match by Account Number

**Purpose:** This test demonstrates that the CLEC Client Application can process a Local Service Itemization (LSI) Request with a valid BST thirteen character account number input.

**Interface Type:** xstaCustomerRecord.retrieveCustomerRecord

**Special Instructions:** The CLEC will be asked to check for specific components within the customerTextList value returned.

**API Input**

```
customerRequest{
 stateCode=;
 customerCode=130;
 exchangeCompanyCircuitId=;
 accountTelephoneNumber=4234829999;
 agencyAuthorizationStatus=Y;
 authorizationName=BARBARASAIDSO;
 authorizationDate=19971120;
 lsiIndicator=Y;
}
```

**API Output**

```
Status{
 msgId=BLP0000CSR;
 msgTxt=CSR Transaction Completed Successfully.;
```

```
}
customerTextList{
 0=ACCT 3055589215927                                BN1   AT&T COM OF THEBN2   S
CENTRAL STATES                              BN3   ACCESS MGR-LOCAL  BA4   500 N POINT
PKWY                                PO    ALPH       GA 300051 ESC   Three-Way Calling
1 ESM   Call Forwarding                   1 ESX   Call Waiting                    1 MFD3X
Multiple Feature credit f+      1 NP3   Listing-not in directory +      1 UEPLX
Unbundled Loop Voice Grad+      1 UEPRL Unbundled Exchange Port, +      1 UNECN CLEC
Contact Name and Num+;
}
```

**xstTestClient Input:**

```
RequestType = csrq;
customerRequest = {
        agencyAuthorizationStatus = Y;
        authorizationName = BARBARASAIDSO;
        authorizationDate = 19971120;
        stateCode = AT;
        exchangeCompanyCircuitId = 40IBS4519841SB;
        customerCode = 329;
        accountTelephoneNumber = 7709873275;
        lsiIndicator = Y;
};
```

**xstTestClient Output:**

```
status={
 msgId=BLP0000CSR;
 msgTxt="CSR Transaction Completed Successfully.";
};
messageHeader={
 inquiryNumber=0221ce4300000001;
 dateSent=1999090317495989;
};
customerTextList=(
 "ACCT 3055589215927                                BN1   AT&T COM OF THEBN2   S
CENTRAL STATES                            BN3   ACCESS MGR-LOCALBA4   500 N POINT PKWY
PO    ALPH       GA 30005                              1 ESC   Three-Way Calling 1 ESM
Call Forwarding               1 ESX   Call Waiting                  1 MFD3X Multiple
Feature credit f+      1 NP3   Listing-not in directory +      1 UEPLX Unbundled Loop
Voice Grad+      1 UEPRL Unbundled Exchange Port, +      1 UNECN CLEC Contact Name and
Num+";
);
```

## 2.4 Service Availability

### 2.4.1 tag.clec.saq01 Request Using 11-Character CLLI

**Purpose:** This test demonstrates that the CLEC Client Application can process a Service Availability Request with a valid BST NPA-NXX and eleven-character TAG-TA-LST input.

**Interface Type:** xstaServiceAvailability..verifyService

**Special**

**Instructions:** The CLEC will be asked to check for specific components within the productFeatureInfoList and carrierInfoList values returned.

**API Input:**

```
clli=ACWOGAMA97E;
npanxx=770529;
picSvcOfng=R;
serviceAbbreviationList{
 0=ABC;
 1=D%;
 2=EF%;
}
```

**API Output:**

```
Status{
 msgId=BLP0000SAV;
 msgTxt=Transaction completed successfully;
}
clli=ACWOGAMA97E;
switchInfo{
 switchType=5ES;
 isdnIndicator=Y;
 switchNpaNxxList{
  0=770529;
  1=770917;
  2=770966;
  3=770974;
  4=770975;
 }
 eightHundredServingOffice=ACWOGAMA97E;
 watsServingOffice=ACWOGAMA97E;
 switchClliList{
 }
}
productFeatureInfoList{
 0{
  productId=DID;
  productName=DID;
  featureTitle=DID;
  centralOfficeFeatureAvail=A;
  featureEffectiveDate=19910814;
  accessNumber=<null>;
  tariffNotes=<null>;
  tariffStatus=E;
  tariffEffectiveDate=;
  productFeatureUsocList{
  }
 }
 1{
  productId=DID800;
  productName=800 SVC TO DID LINES;
  featureTitle=800 SVC TO DID LINES;
  centralOfficeFeatureAvail=A;
  featureEffectiveDate=19910814;
  accessNumber=<null>;
  tariffNotes=<null>;
  tariffStatus=E;
  tariffEffectiveDate=;
  productFeatureUsocList{
```

```
 }
}
2{
 productId=DIDFS;
 productName=FASTER SIGNALING ON DID;
 featureTitle=FASTER SIGNALING ON DID;
 centralOfficeFeatureAvail=A;
 featureEffectiveDate=19930716;
 accessNumber=<null>;
 tariffNotes=<null>;
 tariffStatus=E;
 tariffEffectiveDate=19930716;
 productFeatureUsocList{
 }
}
3{
 productId=DIDTQ;
 productName=DID TRUNK QUEUING;
 featureTitle=DID TRUNK QUEUING;
 centralOfficeFeatureAvail=N;
 featureEffectiveDate=;
 accessNumber=<null>;
 tariffNotes=<null>;
 tariffStatus=E;
 tariffEffectiveDate=;
 productFeatureUsocList{
 }
}
4{
 productId=DIDUT;
 productName=DID WITH USER TRANSFER;
 featureTitle=DID WITH USER TRANSFER;
 centralOfficeFeatureAvail=N;
 featureEffectiveDate=;
 accessNumber=<null>;
 tariffNotes=<null>;
 tariffStatus=N;
 tariffEffectiveDate=;
 productFeatureUsocList{
 }
}
5{
 productId=DTMF;
 productName=DTMF SIGNALING ON DID;
 featureTitle=DTMF SIGNALING ON DID;
 centralOfficeFeatureAvail=A;
 featureEffectiveDate=19930716;
 accessNumber=<null>;
 tariffNotes=<null>;
 tariffStatus=E;
 tariffEffectiveDate=19930716;
 productFeatureUsocList{
 }
}
6{
 productId=EFRS;
 productName=ESSX;
 featureTitle=FULLY RESTRICTED SVC  (G);
 centralOfficeFeatureAvail=N;
```

```
  featureEffectiveDate=;
  accessNumber=<null>;
  tariffNotes=<null>;
  tariffStatus=N;
  tariffEffectiveDate=;
  productFeatureUsocList{
  }
 }
}
carrierInfoList{
 0{
  cic=0686;
  serviceOfferingType=RDDD;
  accessCarrierName=CONNECT-N-SAVE/AT&T;
  accessCarrierNameAbbreviation=UCN;
  accessCarrierTelephoneNumber=8004374121;
 }
 1{
  cic=0686;
  serviceOfferingType=RDDDI;
  accessCarrierName=CONNECT-N-SAVE/AT&T;
  accessCarrierNameAbbreviation=UCN;
  accessCarrierTelephoneNumber=8004374121;
 }
}
```

**xstTestClient Input:**

```
RequestType = saq;
npanxx = 770529;
clli = ACWOGAMA97E;
picServiceOffering = R;
serviceAbbreviationList = (
        ABC,
        "D%",
        "EF%",
);
```

**xstTestClient Output:**

```
status={
 msgId=BLP0000SAV;
 msgTxt="Transaction completed successfully";
};
clli=ACWOGAMA97E;
messageHeader={
 inquiryNumber=2c07e45a00000001;
 dateSent=1999090317571459;
};
switchInfo={
 switchType=5ES;
 isdnIndicator=Y;
 switchNpaNxxList=(
  770529;
  770917;
  770966;
  770974;
  770975;
 );
 eightHundredServingOffice=ACWOGAMA97E;
```

```
 watsServingOffice=ACWOGAMA97E;
 switchCLLiList=(
 );
};
productFeatureInfoList=(
 {
  productId=DID;
  productName=DID;
  featureTitle=DID;
  centralOfficeFeatureAvail=A;
  featureEffectiveDate=19910814;
  tariffStatus=E;
  productFeatureUsocList=(
  );
 };
 {
  productId=DID800;
  productName="800 SVC TO DID LINES";
  featureTitle="800 SVC TO DID LINES";
  centralOfficeFeatureAvail=A;
  featureEffectiveDate=19910814;
  tariffStatus=E;
  productFeatureUsocList=(
  );
 };
 {
  productId=DIDFS;
  productName="FASTER SIGNALING ON DID";
  featureTitle="FASTER SIGNALING ON DID";
  centralOfficeFeatureAvail=A;
  featureEffectiveDate=19930716;
  tariffStatus=E;
  tariffEffectiveDate=19930716;
  productFeatureUsocList=(
  );
 };
 {
  productId=DIDTQ;
  productName="DID TRUNK QUEUING";
  featureTitle="DID TRUNK QUEUING";
  centralOfficeFeatureAvail=N;
  tariffStatus=E;
  productFeatureUsocList=(
  );
 };
 {
  productId=DIDUT;
  productName="DID WITH USER TRANSFER";
  featureTitle="DID WITH USER TRANSFER";
  centralOfficeFeatureAvail=N;
  tariffStatus=N;
  productFeatureUsocList=(
  );
 };
 {
  productId=DTMF;
  productName="DTMF SIGNALING ON DID";
  featureTitle="DTMF SIGNALING ON DID";
  centralOfficeFeatureAvail=A;
```

```
   featureEffectiveDate=19930716;
   tariffStatus=E;
   tariffEffectiveDate=19930716;
   productFeatureUsocList=(
   );
  };
  {
   productId=EFRS;
   productName=ESSX;
   featureTitle="FULLY RESTRICTED SVC  (G)";
   centralOfficeFeatureAvail=N;
   tariffStatus=N;
   productFeatureUsocList=(
   );
  };
);
carrierInfoList=(
  {
   cic=0686;
   serviceOfferingType=RDDD;
   accessCarrierName="CONNECT-N-SAVE/AT&T";
   accessCarrierNameAbbreviation=UCN;
   accessCarrierTelephoneNumber=8004374121;
  };
  {
   cic=0686;
   serviceOfferingType=RDDDI;
   accessCarrierName="CONNECT-N-SAVE/AT&T";
   accessCarrierNameAbbreviation=UCN;
   accessCarrierTelephoneNumber=8004374121;
  };
);
```

## 2.5 General Pool Telephone Number

2.5.1 tag.clec.tng01 Request 25 GP Telephone Numbers

**Purpose:** This test demonstrates that the CLEC Client Application can process a Telephone Number Reservation Request for the maximum number of numbers allowed.

**Interface Type:** xstaTelephoneNumberAvailability.getAvailableTelephoneNumbers

**Special Instructions:** None

**Note:** The data format shown is not current, data not available.

**API Input:**

```
npanxx=404252;
quantityRequested=25
exceptionChar=;
options=0;
requestedNumberLow=;
typeOfService=;
clli=ATLNGASS;
```

**API Output:**

```
Status{
 msgId=BLP1000TNX;
 msgTxt=TRANSACTION SUCCESSFUL;
}
clli=ATLNGASSDS1;
confirmationNumber=DC46H14;
telephoneNumbers{
 0=4042521617;
 1=4042522303;
 2=4042521476;
 3=4042521614;
 4=4042522331;
 5=4042521927;
 6=4042522603;
 7=4042522865;
 8=4042524385;
 9=4042521264;
 10=4042521768;
 11=4042524371;
 12=4042522966;
 13=4042522487;
 14=4042521710;
 15=4042523031;
 16=4042524193;
 17=4042521761;
 18=4042522684;
 19=4042521458;
 20=4042522282;
 21=4042523821;
 22=4042523731;
 23=4042521545;
 24=4042523824;
}
```

**xstTestClient Input:**

```
NPANXX:404252|
QUANTITY:25|
EXCEPTCHAR:|
TNOPTIONS:None|
NUMBERLOW:|
TYPEOFSERVICE:|
CLLI:ATLNGASS|
```

### 2.5.2 tag.clec.tng02 Extend GP Telephone Number Using Confirmation Number

**Purpose:** This test demonstrates that the CLEC Client Application can process a Telephone Number Reservation Extension request by confirmation number.

**Interface Type:** xstaTelephoneNumberAvailability.selectAvailableTelephoneNumbers

**Special Instructions:** None

**API Input:**

```
npanxx=404252;
telephoneNumberList{
}
confirmationNumber=3EHFGJ8;
reserveUntilDate=19991221;
clli=ATLNGASS;
```

**API Output:**

```
Status{
 msgId=BLP3002TNX;
 msgTxt=DATES WERE SUCCESSFULLY CHANGED;
}
clli=ATLNGASSDS1;
confirmationNumber=EJ38JEB;
telephoneNumbers{
 0=4042526841;
}
```

**xstTestClient Input:**

```
RequestType = tnsq;
npanxx = 404252;
option = TN;
confirmationNumber = 3EHFGJ8;
reserveUntilDate = 19991221;
clli = ATLNGASS;
```

**xstTestClient Output:**

```
status={
 msgId=BLP3002TNX;
 msgTxt="DATES WERE SUCCESSFULLY CHANGED";
};
```

```
clli=ATLNGASSDS1;
messageHeader={
 inquiryNumber=60c6034700000001;
 dateSent=1999090318480696;
};
```

### 2.5.3 tag.clec.tng03 Cancel GP Telephone Number Using Confirmation Number

**Purpose:** This test demonstrates that the CLEC Client Application can process a Telephone Number Reservation Cancellation request by confirmation number.

**Interface Type:** xstaTelephoneNumberAvailability.cancelTelephoneNumbers

**Special Instructions:** None

**API Input:**

```
npanxx=404252;
confirmationNumber=38GHFK8;
telephoneNumberList{
}
clli=ATLNGASS;
```

**API Output:**

```
Status{
 msgId=BLP3003TNX;
 msgTxt=CONFIRMATION 38GHFK8 HAS BEEN CANCELLED;
}
clli=HLWDFLPEXXX;
```

**xstTestClient Input:**

```
RequestType = tncan_tn;
npanxx = 404252;
confirmationNumber = 38GHFK8;
clli = ATLNGASS;
```

**xstTestClient Output:**

```
status={
 msgId=BLP3003TNX;
 msgTxt="CONFIRMATION 38GHFK8 HAS BEEN CANCELLED";
};
clli=HLWDFLPEXXX;
messageHeader={
 inquiryNumber=35fe43ca00000001;
 dateSent=1999090319233073;
};
```

### 2.5.4 tag.clec.tng04 Request A Specific GP Telephone Number

**Purpose:** This test demonstrates that the CLEC Client Application can process a Telephone Number Reservation Request for a specific telephone number.

**Interface Type:** xstaTelephoneNumberAvailability.getAvailableTelephoneNumbers

**Special Instructions:** None

**API Input:**

```
npanxx=404252;
quantityRequested=1;
exceptionChar=;
options=0;
requestedNumberLow=4042521458;
typeOfService=;
clli=ATLNGASS;
```

**API Output:**

```
Status{
 msgId=BLP1000TNX;
 msgTxt=TRANSACTION SUCCESSFUL;
}
clli=ATLNGASS;
confirmationNumber= DG2B97H;
telephoneNumbers{
 0=4042521458;
}
```

**xstTestClient Input:**

```
RequestType = tnaq;
npanxx = 404252;
quantityRequested = 1;
options = None;
requestedNumberLow = "4042521458";
clli = ATLNGASS;
```

**xstTestClient Output:**

```
status={
 msgId=BLP1000TNX;
 msgTxt="TRANSACTION SUCCESSFUL";
};
clli=ATLNGASSDS1;
messageHeader={
 inquiryNumber=37a50a5000000001;
 dateSent=1999090319302323;
};
confirmationNumber=DG2B97H;
telephoneNumbers=(
 4042521458;
);
```

### 2.5.5 tag.clec.tng05 Extend GP Telephone Number Reservation Using Telephone Number

**Purpose:** This test demonstrates that the CLEC Client Application can process a Telephone Number Reservation Extension request by telephone number.

**Interface Type:** xstaTelephoneNumberAvailability.selectAvailableTelephoneNumbers

**Special Instructions:** None

**API Input:**

```
npanxx=404252;
telephoneNumberList{
 0=4042520701;
 1=4042520702;
 2=4042520703;
 3=4042520704;
 4=4042520705;
 5=4042520706;
 6=4042520707;
 7=4042520708;
 8=4042520709;
 9=4042520710;
 10=4042520711;
 11=4042520712;
 12=4042520713;
 13=4042520714;
 14=4042520715;
 15=4042520716;
 16=4042520717;
 17=4042520718;
 18=4042520719;
 19=4042520720;
 20=4042520721;
 21=4042520722;
 22=4042520723;
 23=4042520724;
 24=4042520725;
}
confirmationNumber=38GJFK8;
reserveUntilDate=19991221;
clli=ATLNGASS;
```

**API Output:**

```
Status{
 msgId=BLP3002TNX;
 msgTxt=DATES WERE SUCCESSFULLY CHANGED;
}
clli=HLWDFLPEXXX;
confirmationNumber=<null>;
telephoneNumbers{
 0=9544321005;
 1=9544321006;
 2=9544321007;
 3=9544321008;
 4=9544321009;
 5=9544321010;
```

```
 6=9544321011;
 7=9544321012;
 8=9544321013;
 9=9544321014;
 10=9544321015;
 11=9544321016;
 12=9544321017;
 13=9544321018;
 14=9544321019;
 15=9544321020;
 16=9544321021;
 17=9544321022;
 18=9544321023;
 19=9544321024;
}
```

**xstTestClient Input:**

```
npanxx = 404252;
option = TN;
confirmationNumber = 38GJFK8;
reserveUntilDate = 19991221;
clli = ATLNGASS;
telephoneNumberList = (
        4042520701,
        4042520702,
        4042520703,
        4042520704,
        4042520705,
        4042520706,
        4042520707,
        4042520708,
        4042520709,
        4042520710,
        4042520711,
        4042520712,
        4042520713,
        4042520714,
        4042520715,
        4042520716,
        4042520717,
        4042520718,
        4042520719,
        4042520720,
        4042520721,
        4042520722,
        4042520723,
        4042520724,
        4042520725,
);
```

**xstTestClient Output:**

```
status={
 msgId=BLP3002TNX;
 msgTxt="DATES WERE SUCCESSFULLY CHANGED";
};
clli=HLWDFLPEXXX;
messageHeader={
 inquiryNumber=61dbeb5000000001;
```

```
 dateSent=1999090319493249;
};
```

## 2.5.6 tag.clec.tng06 Cancel A GP Telephone Number Using Telephone Number

**Purpose:** This test demonstrates that the CLEC Client Application can process a Telephone Number Reservation Cancellation request by telephone number.

**Interface Type:** xstaTelephoneNumberAvailability.cancelTelephoneNumbers

**Special Instructions:** None

**API Input:**

```
npanxx=404252;
confirmationNumber=;
telephoneNumberList{
 0=4042520701;
}
clli=ATLNGASS;
```

**API Output:**

```
Status{
 msgId=BLP1000TNX;
 msgTxt=TRANSACTION SUCCESSFUL;
}
clli=HLWDFLPE;
```

**xstTestClient Input:**

```
RequestType = tncan_tn;
npanxx = 404252;
confirmationNumber = "";
clli = ATLNGASS;
telephoneNumberList = (
        4042520701
);
```

**xstTestClient Output:**

```
status={
 msgId=BLP1000TNX;
 msgTxt="TRANSACTION SUCCESSFUL";
};
clli=HLWDFLPE;
messageHeader={
 inquiryNumber=17d1dc3700000001;
 dateSent=1999090320032048;
};
```

## 2.5.7 tag.clec.tng07 Request An Unavailable Number

**Purpose:** This test demonstrates that the CLEC Client Application can process a Telephone Number Reservation Request for a specific number which is not available.

**Interface Type:** xstaTelephoneNumberAvailability.getAvailableTelephoneNumbers

**Special Instructions:** None

**Note**: The data format shown is not current, data not available.

**API Input:**

```
npanxx=404714;
quantityRequested=1;
exceptionChar=;
options=0;
requestedNumberLow=4045299999;
typeOfService=;
clli=ATLNGAYP;
```

**API Output:**

```
Status{
 msgId=BLP7064TNX;
 msgTxt=NO NUMBERS FOUND TO MATCH YOUR REQUEST;
}
clli=ATLNGAYP;
confirmationNumber=<null>;
telephoneNumbers{
}
```

**xstTestClient Input:**

```
NPANXX:404714|
QUANTITY:1|
EXCEPTCHAR:|
TNOPTIONS:None|
NUMBERLOW:4045299999|
TYPEOFSERVICE:|
CLLI:ATLNGAYP|
```

# 2.6 Direct-In-Dial (DID) Telephone Number

## 2.6.1 tag.clec.tnd01 Request Random DID Telephone Numbers

**Purpose:** This test demonstrates that the CLEC Client Application can process a DID Telephone Number Reservation Request for random numbers.

**Interface Type:** xstaTelephoneNumberAvailability.getAvailableDIDTelephoneNumbers

**Special Instructions:** None

**API Input:**

```
npanxx=910343;
telephoneNumber=;
exceptionChar=8;
quantityRequested=25;
routeIndex=123;
clli=ATLNGASS;
```

**API Output:**

```
Status{
 msgId=BLP3009DID;
 msgTxt=TNS HAVE BEEN SUCCESSFULLY RESERVED.;
}
clli=ATLNGASS;
confirmationNumber=1GC1RDJ;
quantityProvided=10;
telephoneNumberRangeList{
 0{
  lowValue=9103430677;
  highValue=9103430686;
 }
}
```

**xstTestClient Input:**

```
RequestType = tnaq_did;
npanxx = 910343;
exceptionChar = 8;
quantityRequested = 25;
routeIndex = 123;
clli = ATLNGASS;
```

**xstTestClient Output:**

```
status={
 msgId=BLP3009DID;
 msgTxt="TNS HAVE BEEN SUCCESSFULLY RESERVED.";
};
clli=WLMGNCFO76G;
messageHeader={
 inquiryNumber=597d402600000001;
 dateSent=1999090710443513;
};
```

```
confirmationNumber=1GC1RDJ;
quantityProvided=10;
telephoneNumberRangeList=(
 {
  lowValue=9103430677;
  highValue=9103430686;
 };
);
```

### 2.6.2 tag.clec.tnd02 Extend DID Telephone Numbers Using Confirmation Number

**Purpose:** This test demonstrates that the CLEC Client Application can process a DID Telephone Number Reservation Extension request by confirmation number.

**Interface Type:** xstaTelephoneNumberAvailability.selectAvailableTelephoneNumbers

**Special Instructions:** None

**Note**: This example has not been updated to show the new file format.

**API Input:**

```
npanxx=404252;
telephoneNumberList{
 0=;
}
confirmationNumber=7E7J2F6;
reserveUntilDate=19991231;
clli=ATLNGASSDS1;
```

**API Output:**

```
Status{
 msgId=BLP3002DID;
 msgTxt=DATES WERE SUCCESSFULLY CHANGED;
}
clli=ATLNGASSDS1;
quantityProvided=0;
telephoneNumberRangeList{
}
```

**xstTestClient Input:**

```
NPANXX:404252|
TELEPHONE:|
TELEPHONEOPTION:DID|
CONFIRMATION:7E7J2F6|
EXPIRYDATE:19991231|
CLLI:ATLNGASSDS1|
```

### 2.6.3 tag.clec.tnd03 Cancel DID Telephone Numbers Using Confirmation Number

**Purpose:** This test demonstrates that the CLEC Client Application can process a DIDTelephone Number Reservation Cancellation request by confirmation number.

**Interface Type:** xstaTelephoneNumberAvailability.cancelDIDTelephoneNumbers

**Special Instructions:** None

**Note:** This example has not been updated to show the new file format.

**API Input:**

```
npanxx=404252;
confirmationNumber=7E7J2F6;
didRangeList{
}
clli=ATLNGASSDS1;
```

**API Output:**

```
Status{
 msgId=BLP1000DID;
 msgTxt=TRANSACTION SUCCESSFUL;
}
clli=ATLNGASSDS1;
```

**xstTestClient Input:**

```
NPANXX:404252|
TELEPHONE:|
CONFIRMATION:7E7J2F6|
CLLI:ATLNGASSDS1|
```

### 2.6.4 tag.clec.tnd04 Request A Specific DID Telephone Number

**Purpose:** This test demonstrates that the CLEC Client Application can process a DIDTelephone Number Reservation Request for a specific telephone number.

**Interface Type:** xstaTelephoneNumberAvailability.getAvailableDIDTelephoneNumbers

**Special Instructions:** None

**API Input:**

```
npanxx=910343;
telephoneNumber=9103430677;
exceptionChar=;
quantityRequested=10;
routeIndex=123;
clli=ATLNGASS;
```

**API Output:**

```
Status{
 msgId=BLP3009DID;
 msgTxt=TNS HAVE BEEN SUCCESSFULLY RESERVED.;
}
clli=WLMGNCFO76G;
confirmationNumber=1GC1RDJ;
quantityProvided=10;
telephoneNumberRangeList{
 0{
  lowValue=9103430677;
  highValue=9103430686;
 }
}
```

**xstTestClient Input:**

```
RequestType = tnaq_did;
npanxx = 910343;
telephoneNumber = 9103430677;
exceptionChar = "";
quantityRequested = 10;
routeIndex = 123;
clli = ATLNGASS;
```

**xstTestClient Output:**

```
status={
 msgId=BLP3009DID;
 msgTxt="TNS HAVE BEEN SUCCESSFULLY RESERVED.";
};
clli=WLMGNCFO76G;
messageHeader={
 inquiryNumber=597d402600000001;
 dateSent=1999090710443513;
};
confirmationNumber=1GC1RDJ;
```

```
quantityProvided=10;
telephoneNumberRangeList=(
 {
  lowValue=9103430677;
  highValue=9103430686;
 };
);
```

### 2.6.5 tag.clec.tnd05 Extend DID Reservation Using Telephone Number

**Purpose:** This test demonstrates that the CLEC Client Application can process a DID Telephone Number Reservation Extension request by Telephone Number.

**Interface Type:** xstaTelephoneNumberAvailability.selectAvailableTelephoneNumbers

**Special Instructions:** None

**Note:** This example has not been updated to show the new file format.

**API Input:**

```
npanxx=404529;
telephoneNumberList{
 0=4042551002;
}
confirmationNumber=;
reserveUntilDate=19991231;
clli=ATLNGASSDS1;
```

**API Output:**

```
Status{
 msgId=BLP3002DID;
 msgTxt=DATES WERE SUCCESSFULLY CHANGED;
}
clli=ATLNGASSDS1;
quantityProvided=0;
telephoneNumberRangeList{
}
```

**xstTestClient Input:**

```
NPANXX:404529|
TELEPHONE:4042551002|
TELEPHONEOPTION:DID|
CONFIRMATION:|
EXPIRYDATE:19991231|
CLLI:ATLNGASSDS1|
```

## 2.6.6 tag.clec.tnd06 Cancel A Single DID Range

**Purpose:** This test demonstrates that the CLEC Client Application can process a DIDTelephone Number Reservation Cancellation request by DID Range.

**Interface Type:** xstaTelephoneNumberAvailability.cancelDIDTelephoneNumbers

**Special Instructions:** None

**API Input:**

```
npanxx=910343;
confirmationNumber=;
didRangeList{
 0{
  lowValue=9103430677;
  highValue=9103430686;
 }
}
```

**API Output:**

```
status={
 msgId=BLP1000DID;
 msgTxt=TRANSACTION SUCCESSFULL;
}
clli=ATLNGASS;
```

**xstTestClient Input:**

```
RequestType = tncan_did;
npanxx = 910343;
confirmationNumber = "";
clli = ATLNGASS;
didRangeList = (
        {
                lowValue = 9103430677;
                highValue = 9103430686;
        },
);
```

**xstTestClient Output:**

```
status={
 msgId=BLP1000DID;
 msgTxt="TRANSACTION SUCCESSFULL";
};
clli=ATLNGASS;
messageHeader={
 inquiryNumber=1076a32200000001;
 dateSent=1999090711512742;
};
```

## 2.7 Multi-Line Hunt (MLH) Telephone Number

2.7.1 tag.clec.tnm01 Reserve MLH Number With Incoming Terminals

**Purpose:** This test demonstrates that the CLEC Client Application can process a Telephone Number Reservation Request for an MLH number with incoming terminals.

**Interface Type:** xstaTelephoneNumberAvailability.getAvailableMLHTelephoneNumbers

**Special Instructions:** None

**Note:** This example has not been updated to show the new file format.

**API Input:**

```
npanxx=407236;
leadTelephoneNumber=4072369331;
quantityInTerminals=10;
quantityOutTerminals=0;
existingMLHNumberList{
 0=;
}
lastInTerminal=;
lastOutTerminal=;
clli=ORLDFLMA42E;
```

**API Output:**

```
Status{
 msgId=BLP1000MLH;
 msgTxt=TRANSACTION SUCCESSFUL;
}
clli=ORLDFLMA42E;
existingMLHNumberList{
 0=;
}
huntGroupNumberList{
 0{
  huntGroupNumber= 348;
  leadTelephoneNumber=4072369331;
inTerminalRange{
 lowValue=   1;
 highValue=  10;
}
outTerminalRange{
 lowValue=<null>;
 highValue=<null>;
}
 }
}
```

**xstTestClient Input:**

```
NPANXX:407236|
LEADPHONE:4072369331|
INTERM:10|
OUTTERM:|
```

```
EXISTINGMLH:|
LASTINTERM:|
LASTOUTTERM:|
CLLI:ORLDFLMA42E|
```

### 2.7.2 tag.clec.tnm02 Add Internal Terminals to an MLH Number

**Purpose:** This test demonstrates that the CLEC Client Application can process a Telephone Number Reservation Request for adding incoming terminals to an existing MLH number.

**Interface Type:** xstaTelephoneNumberAvailability.getAvailableMLHTelephoneNumbers

**Special Instructions:** None

**Note:** This example has not been updated to show the new file format.

**API Input:**

```
npanxx=407236;
leadTelephoneNumber=4072369331;
quantityInTerminals=5;
quantityOutTerminals=0;
existingMLHNumberList{
 0=348;
}
lastInTerminal=10;
lastOutTerminal=;
clli=ORLDFLMA;
```

**API Output:**

```
Status{
 msgId=BLP3004MLH;
 msgTxt=EXISTING MLH WILL STILL WORK.  ADDITIONAL TERS ARE SHOWN            BELO
W THE MLH NUMBER;
}
clli=ORLDFLMA42E;
existingMLHNumberList{
 0=348;
}
huntGroupNumberList{
 0{
  huntGroupNumber= 348;
  leadTelephoneNumber=4072369331;
inTerminalRange{
 lowValue=  11;
 highValue=  15;
}
outTerminalRange{
 lowValue=<null>;
 highValue=<null>;
}
 }
}
```

**xstTestClient Input:**

```
NPANXX:407236|
LEADPHONE:4072369331|
INTERM:5|
OUTTERM:|
EXISTINGMLH:348|
LASTINTERM:10|
LASTOUTTERM:|
CLLI:ORLDFLMA|
```

### 2.7.3 tag.clec.tnm03 Cancel a Reservation Using MLH Number

**Purpose:** This test demonstrates that the CLEC Client Application can process a Telephone Number Reservation Cancellation Request using MLH number.

**Interface Type:** xstaTelephoneNumberAvailability.cancelMLHTelephoneNumbers

**Special Instructions:** None

**Note:** This example has not been updated to show the new file format.

**API Input:**

```
npanxx=407236;
return1=348;
return2=;
clli=ORLDFLMA42E;
```

**API Output:**

```
Status{
 msgId=BLP3002MLH;
 msgTxt=MLH NUMBER(S) WERE RETURNED TO ATLAS. DO NOT PUT RETURNED          NUMB
ER(S) ON A SERVICE ORDER;
}
clli=ORLDFLMA42E;
```

**xstTestClient Input:**

```
NPANXX:407236|
LEADPHONE:4072369331|
CLLI:ORLDFLMA42E|
TELEPHONE:348|
TELEPHONE:|
```

## 2.8 Telephone Number Assignment (Miscellaneous)

### 2.8.1 Query by City and State Miscellaneous Telephone Number

**API Input**

```
---------
npanxx=;
city=ATLANTA;
state=GA;
quantityRequested=1;
```

**API Output**

```
----------
```

```
Status{
 msgId=BLP0000TNR;
 msgTxt=TRANSACTION SUCCESSFUL. MISCELLANEOUS ACCOUNT NUMBERS ARE VALID FOR 60 DAYS
ONLY.;
}
quantityProvided=0;
npanxxList{
 0=212049;
 1=212050;
 2=212149;
 3=212150;
 4=212151;
 5=212152;
 6=212153;
 7=212154;
 8=212249;
 9=212250;
 10=212251;
 11=212252;
 12=212253;
 13=212254;
 14=212349;
 15=212350;
 16=212351;
 17=212352;
 18=212449;
 19=212450;
 20=212451;
 21=212452;
 22=212453;
 23=212454;
 24=212549;
 25=404M21;
}
telephoneNumberList{
}
```

**xstTestClient Input**

```
-------------------
RequestType = tnaq_misc;
city = ATLANTA;
state = GA;
quantityRequested = 1;
```

**xstTestClient Output**

```
--------------------
status={
 msgId=BLP0000TNR;
 msgTxt="TRANSACTION SUCCESSFUL. MISCELLANEOUS ACCOUNT NUMBERS ARE VALID FOR 60 DAYS
ONLY.";
};
messageHeader={
 inquiryNumber=7ff3a91700000001;
 dateSent=1999120815224388;
};
quantityProvided=0;
```

```
npanxxList=(
 212049;
 212050;
 212149;
 212150;
 212151;
 212152;
 212153;
 212154;
 212249;
 212250;
 212251;
 212252;
 212253;
 212254;
 212349;
 212350;
 212351;
 212352;
 212449;
 212450;
 212451;
 212452;
 212453;
 212454;
 212549;
 404M21;
);
telephoneNumberList=(
);
```

### 2.8.2 Query by NPANXX — Miscellaneous Telephone Number

**API Input**

```
npanxx=404M21;
city=;
state=;
quantityRequested=1;
```

**API Output**

```
Status{
 msgId=BLP0000TNR;
 msgTxt=TRANSACTION SUCCESSFUL. MISCELLANEOUS ACCOUNT NUMBERS ARE VALID FOR 60 DAYS
ONLY.;
}
quantityProvided=3;
npanxxList{
}
telephoneNumberList{
 0=9086991111;
 1=9086991112;
 2=9086991113;
}
```

**xstTestClient Input**

```
RequestType = tnaq_misc;
npanxx = 404M21;
quantityRequested = 1;
```

**xstTestClient Output**

```
status={
 msgId=BLP0000TNR;
 msgTxt="TRANSACTION SUCCESSFUL. MISCELLANEOUS ACCOUNT NUMBERS ARE VALID FOR 60 DAYS
ONLY.";
};
messageHeader={
 inquiryNumber=3f78cd2200000001;
 dateSent=1999120815331581;
};
quantityProvided=3;
npanxxList=(
);
telephoneNumberList=(
 9086991111;
 9086991112;
 9086991113;
);
```

## 2.9 Pre-Order Calculated Due Date

**API Input:**

See the method, getDueDate() in the xstaOrder class.

**API Output:**

**xstTestClient Input:**

```
RequestType=cdd;    #Pre-order   REQTYP=E   ACT=C
# Input
testProdIndicator=T;
lsr={
       ccna=CNA; pon=CRES-022P; version=00;
       lsrNumber=CRES-022P; locqty=123;
       atn=1234567890; serviceCenter=LCSC;
       dateSent =19991122;
       ddd =19991122;
       apptimeDdd=0900-1100;
       dddo =19991122; # Desired Due Date Out
       dfdt=1200;    # Desired Frame Due Time
       project=Project8Identi; chc=Y;
       reqtyp=EB;
       act=c;
       sup=""; exp=Y; cc=CDD1;
       albr=Y; sca=Y; agauth=Y;
       dated =19991122; # Date of Agency Authorization
       authName="Authoriztn Name";
       actl=ACTL1234567; ai=Y; apot=APOT4567890; lst=LST87654321;
       lso=123456;          # Local Service Office
```

```
        tos=2BG;            #see table 2-4 and 2-5
        spec=SPEC467;
#nc=LY--;
#nci=02QC3.RVO;             #see table 2-11 and 2-12
#secnci=02RV2.T;     #see table 2-11 and 2-12
        rpon=RPON1234567890; rord=rord09876543; lspAuth=lspa;
        lspAuthDate =19991122;
        lspAuthName=lsp_auth_name; cic=1234;
        cust="Customer Name Size 25AN";
        bil=A; ban1=qweasd1234098; acna=ABC;
        vta=VTAvariable; init=initiator; initTelNo=7326994801;
        initFaxNo=7326994802; impCon=implementation;
        impConTelNo=7326994803; impConPager=7326994804;
        remarks="this is a remark that is not very long";
        uncommon=N;
};    #END OF LSR

eu={
header={ administrative={ ibt=9; };      # Administrative Section
        locationAndAccess={
                aact=A; locnum=000; euName="jack and jill";
                sano=F61; sasf=abcd;
                sasd=E;
                sasn="york street"; sath=denver;
                sass=suff; sadlo="acorss the bagel shop";
                euFloor=2ndfloor; euRoom= "apt 205";
                euBldg="wing Suk"; euCity=Denver;
                euState=CO; euZipCode=80231;
                lconName="Denver Univ"; lconTelNo=3033061995;
                acc="access using front door";
        };      # Location and Access Header Section
        insideWire={ iwOptions=W; iwconName=Jack;
                iwConTelNo=3036891114; };  # Inside Wire Section
        bill={
                eatn=3033061994; fbi=Y;
                fbBillName="Messrs jack and jill"; fbSbillName=joe;
                fbStreet="1904 s york st"; fbFloor=2ndfloor;
                fbRoom=205; fbCity=denver; fbState=CO; fbZipCode=80231;
                fbBillcon=ESI; fbConTelNo=3036891114;
        };      # Bill Section
};      # END of Header Section
};     #END OF EU form


dsdl={
deliveryAddress={ dact=N; nameDel=NAme; ddasn=12; ddaloc=012345;
        ddast=12; ddazc=12; };
};     #END OF DSDL form

resale={
        ord=012345678901234567;
        rsqty=001;
        serviceDetailList=(
                {
                baList=(      { ba=A;       block=A;      }, );
                cfa=012345678901234567890123456789012345678901;
                ckr=01234567890123456789012345678901234567890;
                cnam=012345678901234;
                ecckt=800.123.4567;
```

```
                fpi=A; ispid=12345678901234;
                insideWireList=( { iwjk=01234; iwjq=12; }; );
                jkCode=01234; jkNum=01; jkPos=12;
                jr=Y; lean=""; leatn="";
                lna=c;
                lneclssvc=01234; lnex=""; lnum=01234;
                locnum=000; lpic=0123; matn=1;
                nidr=Y; npi=C; pic=0123;
                pulse=0123; san=012345678901234567890123456789;
                sdi=A; sgnl=01; ssig=01;
                tcFr=1234567890; ters=0123456789;
                tli=1234567890; tns=0123456789-1234;
                tsp=012345678-01;
                      featureList = (
                            { fa=n; feature=bbc; featureDetail=""; },
                            { fa=n; feature=bcr; featureDetail=""; },
                            { fa=n; feature=bndbs; featureDetail=""; },
                            { fa=n; feature=brd; featureDetail=""; },
                       );      # END of Feature Section
                },   #END OF SERVICE DETAIL Section
        );   #END OF SERVICE DETAIL Section
};    #END OF RESALE Form
```

**xstTestClient Output:**

```
status={
 msgId=00;
 msgTxt="See Due Date Response structure for details.";
};
messageHeader={
 dateSent=1999112214253756;
};
dueDate={
 dueDate=19991203;
 msgId=TAGT0000CDD;
 msgTxt="CALCULATED DUE DATE PROVIDED";
};
```

# 2.10 ESDQ (Pre-Order Estimated Service Date Query)

**API Input:**

See the method, estimatedDueDate() in the xstaDueDate class.

**API Output:**

**xstTestClient Input:**

```
RequestType = esdq;
typeRecord = {
        reqtyp = "AB";
        act = V;
        tos = 1AM;
        nc = LY--;
        secnci = 02IS5;
};
orderInfo = {
        uncommon = "";
        ddd = 20001201;
        lqty = 012;
        npqty = "";
        rsqty = "";
};
serviceAddressList = (
        {
                houseNumber = 111;
                houseNumberSuffix = "";
                streetDirectional = "E";
                streetName = "TENNY";
                streetThoroughfare = "AV";
                streetSuffix = "";
                room = 123412;
                building = TALL;
                floor = "1234";
                city = LOUISVILLE;
                state = KY;
                zipCode = 40214;
                hunting = "N";
                lineInfoList = (
                        {
                                npt = P;
                                lna = P;
                                lneclssvc = "";
                                featureList = (
                                        {
                                                fa = "N";
                                                feature = "HTG";
                                                featureDetail = "Detail";
                                        },
                                        {
                                                fa = "N";
```

```
                                        feature = "HTG";
                                        featureDetail = "Detail 2";
                                    },
                                );
                            },
                    );
            },
);
```

**xstTestClient Input:**

```
status={
 msgId=00;
 msgTxt="THIS IS AN ESTIMATED DUE DATE AND IS SUBJECT TO CHANGE WHEN FIRM ORDER IS
SUBMITTED";
};
messageHeader={
 inquiryNumber=1345d64700000001;
 dateSent=2000072114123767;
};
dueDate={
 dueDate=20001201;
 msgId=TAGT0001CDD;
 msgTxt="THIS IS AN ESTIMATED DUE DATE AND IS SUBJECT TO CHANGE WHEN FIRM ORDER IS
SUBMITTED";
 pvIndicator=Y;
};
```

# 3. Appendix J - Firm Order Test DATA

This is a sample of the Firm Order test client input files for each type of interface types. The firm-order interface types are:

**NOTE**: The Test Data provided here is for example only.

- DIRLIST
- NP
- LOOPNP
- PORT
- LOOPPORT
- LOOP
- RESALE
- CDD

## 3.1 DIRLIST

**API Input:**

```
standardHeader{
 subTransaction=1;
 companyCode=BSTH;
 msgId=;
 msgTxt=;
 leoMessage=;
 applId=Chandra;
 userId=eeb;
 trxId=TagLeoServer_3_1_mut67412a1d;
}
orderHeader{
 base{
  apiVersion=3.1;
  appId=Chandra;
  userid=eeb;
  companyCode=XYZ;
  inquiryNumber=BSTH-PON-0000-11;
  dateSent=1999090812421352;
 }
 cc=BSTH;
 testProdIndicator=0;
}
lsr{
 ccna=CNA;
 pon=PON-0000;
 version=11;
 lsrNumber=LSR_NUMBER;
```

```
locqty=123;
htqty=;
an=;
atn=1234567890;
serviceCenter=LCSC;
dateSent=19991208;
ddd=19991225;
apptimeDdd=0900-1100;
dddo=19991226;
dfdt=1200;
project=TAGMUT;
chc=Y;
reqtyp=JB;
act=N;
sup=01;
exp=Y;
cc=BSTH;
albr=Y;
sca=Y;
agauth=Y;
dated=19991225;
authName=auth name;
porttyp=;
actl=ROOM1234567;
ai=;
apot=;
lst=B0987654321;
lso=123456;
tos=1--;
spec=SVCPR;
nc=LY--;
nci=02QC3.RVO;
secnci=02RV2.T;
rpon=RPON1234567890;
rord=rord0987654321;
lspAuth=lspa;
lspAuthDate=19991231;
lspAuthName=lsp_auth_name;
cic=1234;
cust=customer name;
bi1=A;
ban1=qweasd1234098;
bi2=;
ban2=;
acna=ABC;
ebd=;
billName=;
sbillName=;
billNameStreet=;
billNameFloor=;
billNameRoom=;
billNameCity=;
billNameState=;
billNameZipCode=;
billCon=;
billConTelNo=;
vta=variable;
init=initiator;
initTelNo=7326994801;
```

```
 initFaxNo=7326994802;
 initStreet=;
 initFloor=;
 initRoomMailStop=;
 initCity=;
 initState=;
 initZipCode=;
 impCon=implementation;
 impConTelNo=7326994803;
 impConPager=7326994804;
 altImpCon=;
 altImpConTelNo=;
 altImpConPager=;
 dsgCon=;
 drc=;
 dsgConTelNo=;
 dsgConFaxNo=;
 dsgConStreet=;
 dsgConFloor=;
 dsgConRoomMailStop=;
 dsgConCity=;
 dsgConState=;
 dsgConZipCode=;
 remarks=thisis a remark that is not very long;
 pbt=A;
 bcs=;
 huntGroupInformationList{
 }
}
eu{
 header{
  administrative{
   dqty=;
   ibt=9;
  }
  locationAndAccess{
   aact=;
   locnum=000;
   euName=jack and jill;
   sano=1904;
   sasf=abcd;
   sasd=E;
   sasn=york street;
   sath=denver;
   sass=suff;
   sadlo=acorss the bagel shop;
   euFloor=2ndfloor;
   euRoom=205;
   euBldg=S York;
   euCity=Denver;
   euState=CO;
   euZipCode=80231;
   lconName=Denver Univ;
   lconTelNo=3033061995;
   eumi=Y;
   acc=access using front door;
   wsop=V;
   erl=;
  }
```

```
 insideWire{
  iwOptions=;
  iwConName=;
  iwConTelNo=;
 }
 bill{
  ean=;
  eatn=3033061994;
  fbi=Y;
  fbBillName=Messrs jack and jill;
  fbSbillName=joe;
  fbStreet=1904 s york st;
  fbFloor=2ndfloor;
  fbRoom=205;
  fbCity=denver;
  fbState=CO;
  fbZipCode=80231;
  fbBillCon=ESI;
  fbConTelNo=3036891114;
 }
}
detail{
 locationAndAccessList{
  0{
   locnum=123;
   locact=N;
   euName=jack and jill;
   sano=555;
   sasf=54321;
   sasd=E;
   sasn=South Gate;
   sath=25 South;
   sass=sass;
   sadlo=across the Rockies;
   euFloor=3rdfloor;
   euRoom=305;
   euBldg=south ga;
   euCity=Englewood;
   euState=CO;
   euZipCode=80231;
   lconName=ESI;
   lconTelNo=3036891114;
   acc=ask the receptionist;
  }
 }
 disconnectInformationList{
  0{
   dnum=qwert;
   discNbr=3036891114;
   ter=;
   tcOpt=;
   tcPer=;
   tcToPrimary=;
   tcToSecondary=;
   primary{
    tcid=;
    tcName=;
   }
   secondary{
```

```
      tcid=;
      tcName=;
     }
    }
   }
  }
 }
dsdl{
 listingList{
  0{
   listingControl{
    dlnum=1234;
    lact=N;
    ali=ERT;
    rty=LAC;
    lty=1;
    tt=8;
    styc=SH;
    toa=BP;
   }
   listingInstruction{
    wpp=;
    doi=0;
    ltn=1112223333;
    nstn=6666;
    lnln=louie;
    lnfn=emily;
    lnpl=Y;
    pla=;
    des=;
    tl=IV;
    title1=PhD;
    title2=PE;
    nick=foobar;
    lapr=rty;
    lano=7;
    lasf=ipo;
    lasd=NE;
    lasn=hoes lane;
    lath=zyx;
    lass=NW;
    laloc=sdfg;
    last=AB;
    textList{
     0{
      ltext=blah blah blah some more;
      ltxty=AC;
      ltxnum=7777;
     }
    }
    yph=;
    sic=1234;
   }
   listingIndicators{
    adi=;
    dirname=qwerty;
    dirsub=somewhere else;
    adv=;
    dml=Y;
```

```
    dlnm=Y;
    bro=;
   }
   dscr{
    alirSequencing{
     so=A;
     seqtext=xx;
     seqaddr=xx;
     seqtn=611;
     lvl=0;
     hs=N;
     htn=3334445555;
    }
    levelDetailList{
     0{
      lvl=1;
      ins1=E;
      so1=A;
      seqtext1=xxx;
      seqaddr1=xxx;
      seqtn1=911;
      intn=2223334444;
      innstn=2223334444;
      intext=yyyy;
      inaddr=zzzz;
     }
    }
   }
  }
 }
 deliveryAddress{
  dact=N;
  nameDel=delivery name 1;
  ddapr=;
  ddano=;
  ddasf=;
  ddasd=;
  ddasn=123 hoes lane;
  ddath=;
  ddass=;
  ddalo=;
  ddadlo=;
  ddaloc=012345 metlars lane;
  ddast=FL;
  ddazc=07122;
  directoryTypeList{
   0{
    dirtyp=B;
    dirqtya=1234;
    dirqtync=6789;
   }
  }
 }
}
```

**API Output:**

```
status{
```

```
 msgId=00;
 msgTxt=LSR submitted to BellSouth;
}
dueDate{
 dueDate=19991223;
 msgId=TAGT0000CDD;
 msgText="CALCULATED DUE DATE PROVIDED";
}
```

**xstTestClient Input:**

```
RequestType = dirlist;
testProdIndicator = T;
lsr = {
        ccna = "CNA";
        pon = "PON-0000";
        version = "11";
        lsrNumber = "LSR_NUMBER";
        locqty = "123";
        atn = "1234567890";
        serviceCenter = "LCSC";
        dateSent = "19991208";
        ddd = "19991225";
        apptimeDdd = "0900-1100";
        dddo = "19991226";
        dfdt = "1200";
        project = "TAGMUT";
        chc = "Y";
        reqtyp = "JB";
        act = "N";
        sup = "01";
        exp = "Y";
        cc = "BSTH";
        albr = "Y";
        sca = "Y";
        agauth = "Y";
        dated = "19991225";
        authName = "auth name";
        porttyp = "";
        actl = "ROOM1234567";
        ai = "";
        apot = "";
        lst = "B0987654321";
        lso = "123456";
        tos = "1--";
        spec = "SVCPR";
        nc = "LY--";
        nci = "02QC3.RVO";
        secnci = "02RV2.T";
        rpon = "RPON1234567890";
        rord = "rord0987654321";
        lspAuth = "lspa";
        lspAuthDate = "19991231";
        lspAuthName = "lsp_auth_name";
        cic = "1234";
        cust = "customer name";
        bil = "A";
        banl = "qweasd1234098";
```

```
        acna = "ABC";
        vta = "variable";
        init = "initiator";
        initTelNo = "7326994801";
        initFaxNo = "7326994802";
        impCon = "implementation";
        impConTelNo = "7326994803";
        impConPager = "7326994804";
        remarks = "thisis a remark that is not very long";
        pbt = "A";
};


eu = {
        header = {
                administrative = {
                        ibt = "9";
                };      # Administrative Section
                locationAndAccess = {
                        aact = "";
                        locnum = "000";
                        euName = "jack and jill";
                        sano = "1904";
                        sasf = "abcd";
                        sasd = "E";
                        sasn = "york street";
                        sath = "denver";
                        sass = "suff";
                        sadlo = "acorss the bagel shop";
                        euFloor = "2ndfloor";
                        euRoom = "205";
                        euBldg = "S York";
                        euCity = "Denver";
                        euState = "CO";
                        euZipCode = "80231";
                        lconName = "Denver Univ";
                        lconTelNo = "3033061995";
                        eumi = "Y";
                        acc = "access using front door";
                        wsop = "V";
                        erl = "";
                };      # Location and Access Header Section
                insideWire = {
                        iwOptions = "";
                        iwconName = "";
                        iwConTelNo = "";
                };      # Inside Wire Section
                bill = {
                        eatn = "3033061994";
                        fbi = "Y";
                        fbBillName = "Messrs jack and jill";
                        fbSbillName = "joe";
                        fbStreet = "1904 s york st";
                        fbFloor = "2ndfloor";
                        fbRoom = "205";
                        fbCity = "denver";
                        fbState = "CO";
                        fbZipCode = "80231";
                        fbBillcon = "ESI";
```

```
                fbConTelNo = "3036891114";
        };      # Bill Section
};      # Header Section
detail = {
        locationAndAccessList = (
                {
                        locnum = "123";
                        locact = "N";
                        euName = "jack and jill";
                        sano = "555";
                        sasf = "54321";
                        sasd = "E";
                        sasn = "South Gate";
                        sath = "25 South";
                        sass = "sass";
                        sadlo = "across the Rockies";
                        euFloor = "3rdfloor";
                        euRoom = "305";
                        euBldg = "south ga";
                        euCity = "Englewood";
                        euState = "CO";
                        euZipCode = "80231";
                        lconName = "ESI";
                        lconTelNo = "3036891114";
                        acc = "ask the receptionist";
                },
        );      # Location And Access List
        disconnectInformationList = (
                {
                        dnum = "qwert";
                        discNbr = "3036891114";
                },
        );      # Disconnect Information List
};      # Detail Section
};


dsdl = {
        listingList = (
        {       # element 0
                listingControl = {
                        dlnum = "1234";
                        lact = "N";
                        ali = "ERT";
                        rty = "LAC";
                        lty = "1";
                        tt = "8";
                        styc = "SH";
                        toa = "BP";
                };
                listingInstruction = {
                        wpp = "";
                        doi = "0";
                        ltn = "1112223333";
                        nstn = "6666";
                        lnln = "louie";
                        lnfn = "emily";
                        lnpl = "Y";
                        pla = "";
```

```
                des = "";
                tl = "IV";
                title1 = "PhD";
                title2 = "PE";
                nick = "foobar";
                lapr = "rty";
                lano = "7";
                lasf = "ipo";
                lasd = "NE";
                lasn = "hoes lane";
                lath = "zyx";
                lass = "NW";
                laloc = "sdfg";
                last = "AB";
                textList=(
                {
                        ltext = "blah blah blah some more";
                        ltxty = "AC";
                        ltxnum = "7777";
                },
                );
                yph = "";
                sic = "1234";
        };
        listingIndicators = {
                adi = "";
                dirname = "qwerty";
                dirsub = "somewhere else";
                adv = "";
                dml = "Y";
                dlnm = "Y";
                bro = "";
        };
        dscr = {
                alirSequencing = {
                        so = "A";
                        seqtext = "xx";
                        seqaddr = "xx";
                        seqtn = "611";
                        lvl = "0";
                        hs = "N";
                        htn = "3334445555";
                };
                levelDetailList = (
                {
                        lvl = "1";
                        ins1 = "E";
                        so1 = "A";
                        seqtext1 = "xxx";
                        seqaddr1 = "xxx";
                        seqtn1 = "911";
                        intn = "2223334444";
                        innstn = "2223334444";
                        intext = "yyyy";
                        inaddr = "zzzz";
                },
                );
        };
};
```

```
        );
        deliveryAddress = {
                dact = "N";
                nameDel = "delivery name 1";
                #ddapr = "";
                #ddano = "";
                #ddasf = "";
                #ddasd = "";
                ddasn = "123 hoes lane";
                #ddath = "";
                #ddass = "";
                #ddalo = "";
                #ddadlo = "";
                ddaloc = "012345 metlars lane";
                ddast = "FL";
                ddazc = "07122";
                directoryTypeList=(
                {
                        dirtyp = "B";
                        dirqtya = "1234";
                        dirqtync = "6789";
                },
                );
        };
};
```

**xstTestClient Output:**

```
status={
 msgId=00;
 msgTxt="LSR submitted to BellSouth";
};
messageHeader={
 dateSent=1999090712423732;
};
dueDate={
 dueDate=19991223;
 msgId=TAGT0000CDD;
 msgText="CALCULATED DUE DATE PROVIDED";
};
```

## 3.2 NP

**API Input:**

```
standardHeader{
 subTransaction=1;
 companyCode=BSTH;
 msgId=;
 msgTxt=;
 leoMessage=;
 applId=Chandra;
 userId=eeb;
 trxId=TagLeoServer_3_1_mut36343256;
}
orderHeader{
 base{
  apiVersion=3.1;
  appId=Chandra;
  userid=eeb;
  companyCode=XYZ;
  inquiryNumber=BSTH-NP1-11;
  dateSent=1999090713345103;
 }
 cc=BSTH;
 testProdIndicator=0;
}
lsr{
 ccna=CNA;
 pon=NP1;
 version=11;
 lsrNumber=LSR_NUMBER;
 locqty=123;
 htqty=;
 an=;
 atn=1234567890;
 serviceCenter=LCSC;
 dateSent=19991208;
 ddd=19991225;
 apptimeDdd=0900-1100;
 dddo=19991226;
 dfdt=1200;
 project=TAGMUT;
 chc=Y;
 reqtyp=CB;
 act=C;
 sup=01;
 exp=Y;
 cc=BSTH;
 albr=Y;
 sca=Y;
 agauth=Y;
 dated=19991225;
 authName=auth name;
 porttyp=;
 actl=ROOM1234567;
 ai=Y;
 apot=A1234567890;
```

```
lst=B0987654321;
lso=123456;
tos=1A-;
spec=SVCPR;
nc=LY--;
nci=02QC3.RVO;
secnci=02RV2.T;
rpon=RPON1234567890;
rord=rord0987654321;
lspAuth=lspa;
lspAuthDate=19991231;
lspAuthName=lsp_auth_name;
cic=1234;
cust=customer name;
bi1=A;
ban1=qweasd1234098;
bi2=;
ban2=;
acna=ABC;
ebd=;
billName=;
sbillName=;
billNameStreet=;
billNameFloor=;
billNameRoom=;
billNameCity=;
billNameState=;
billNameZipCode=;
billCon=;
billConTelNo=;
vta=variable;
init=initiator;
initTelNo=7326994801;
initFaxNo=7326994802;
initStreet=;
initFloor=;
initRoomMailStop=;
initCity=;
initState=;
initZipCode=;
impCon=implementation;
impConTelNo=7326994803;
impConPager=7326994804;
altImpCon=;
altImpConTelNo=;
altImpConPager=;
dsgCon=;
drc=;
dsgConTelNo=;
dsgConFaxNo=;
dsgConStreet=;
dsgConFloor=;
dsgConRoomMailStop=;
dsgConCity=;
dsgConState=;
dsgConZipCode=;
remarks=thisis a remark that is not very long;
pbt=A;
bcs=;
```

```
 huntGroupInformationList{
 }
}
eu{
 header{
  administrative{
   dqty=;
   ibt=9;
  }
  locationAndAccess{
   aact=A;
   locnum=;
   euName=jack and jill;
   sano=12375;
   sasf=abcd;
   sasd=E;
   sasn=testing;
   sath=denver;
   sass=suff;
   sadlo=acorss the bagel shop;
   euFloor=twenty one;
   euRoom=123456789;
   euBldg=building1;
   euCity=Denver;
   euState=CO;
   euZipCode=80231;
   lconName=local contact;
   lconTelNo=1234567890;
   eumi=Y;
   acc=access using front door;
   wsop=V;
   erl=;
  }
  insideWire{
   iwOptions=;
   iwConName=;
   iwConTelNo=;
  }
  bill{
   ean=1234567890;
   eatn=;
   fbi=;
   fbBillName=;
   fbSbillName=joe;
   fbStreet=1904 s york st;
   fbFloor=;
   fbRoom=;
   fbCity=denver;
   fbState=CO;
   fbZipCode=80231;
   fbBillCon=ESI;
   fbConTelNo=3036891114;
  }
 }
 detail{
  locationAndAccessList{
   0{
    locnum=123;
    locact=N;
```

```
    euName=jack and jill;
    sano=555;
    sasf=54321;
    sasd=E;
    sasn=South Gate;
    sath=25 South;
    sass=sass;
    sadlo=across the Rockies;
    euFloor=;
    euRoom=305;
    euBldg=south ga;
    euCity=Englewood;
    euState=CO;
    euZipCode=80231;
    lconName=ESI;
    lconTelNo=3036891114;
    acc=ask the receptionist;
   }
  }
  disconnectInformationList{
   0{
    dnum=qwert;
    discNbr=3036891114;
    ter=;
    tcOpt=;
    tcPer=;
    tcToPrimary=;
    tcToSecondary=;
    primary{
     tcid=;
     tcName=;
    }
    secondary{
     tcid=;
     tcName=;
    }
   }
  }
 }
}
dsdl{
 listingList{
  0{
   listingControl{
    dlnum=0027;
    lact=D;
    ali=ABC;
    rty=;
    lty=;
    tt=;
    styc=CI;
    toa=;
   }
   listingInstruction{
    wpp=;
    doi=;
    ltn=3037770000;
    nstn=;
    lnln=NAMES R US;
```

```
    lnfn=;
    lnpl=;
    pla=;
    des=;
    tl=;
    title1=;
    title2=;
    nick=;
    lapr=;
    lano=;
    lasf=;
    lasd=;
    lasn=123 MAIN ST;
    lath=;
    lass=;
    laloc=;
    last=;
    textList{
    }
    yph=YPH1234;
    sic=2345;
   }
   listingIndicators{
    adi=;
    dirname=;
    dirsub=;
    adv=;
    dml=;
    dlnm=;
    bro=;
   }
   dscr{
    alirSequencing{
     so=;
     seqtext=;
     seqaddr=;
     seqtn=;
     lvl=;
     hs=;
     htn=;
    }
    levelDetailList{
     0{
      lvl=;
      ins1=;
      so1=;
      seqtext1=;
      seqaddr1=;
      seqtn1=;
      intn=;
      innstn=;
      intext=;
      inaddr=;
     }
    }
   }
  }
 }
 deliveryAddress{
```

```
  dact=N;
  nameDel=RIGHT HERE;
  ddapr=;
  ddano=;
  ddasf=;
  ddasd=;
  ddasn=123 MAIN ST;
  ddath=;
  ddass=;
  ddalo=;
  ddadlo=;
  ddaloc=ANYTOWN;
  ddast=AL;
  ddazc=13579;
  directoryTypeList{
  }
 }
}
np{
 npqty=00001;
 serviceDetailList{
  0{
   baList{
    0{
     ba=A;
     block=B;
    }
   }
   cftn=;
   ckr=abcdefghijklmnopqrstuvwxyz123456789012345;
   ecckt=101.T1.NCYMNY50.NYCMNY54W01;
   fpi=A;
   lean=;
   leatn=;
   lna=P;
   lnum=12345;
   locnum=123;
   lpic=LPC1;
   npi=A;
   npt=A;
   nptg=1234ABCD;
   portedNbr=3033061995;
   rti=123abc;
   tcFr=1234567890;
   tcOpt=ST;
   tcPer=20000112;
   tcToPrimary=1234567890;
   tcToSecondary=1234567890;
   primary{
    tcid=01;
    tcName=abcdefghijklmnopqrstuvwxyz 12345678;
   }
   secondary{
    tcid=02;
    tcName=12345678 abcdefghijklmnopqrstuvwxyz;
   }
   tnp=123;
  }
 }
```

```
}
```

**API Output:**

```
status{
 msgId=00;
 msgTxt=LSR submitted to BellSouth;
}
dueDate{
 dueDate=19991223;
 msgId=TAGT0000CDD;
 msgText="CALCULATED DUE DATE PROVIDED";
}
```

**xstTestClient Input:**

```
RequestType = np;
testProdIndicator = T;
lsr = {
        ccna = "CNA";
        pon = "NP1";
        version = "11";
        lsrNumber = "LSR_NUMBER";
        locqty = "123";
        atn = "1234567890";
        serviceCenter = "LCSC";
        dateSent = "19991208";
        ddd = "19991225";
        apptimeDdd = "0900-1100";
        dddo = "19991226";
        dfdt = "1200";
        project = "TAGMUT";
        chc = "Y";
        reqtyp = "CB";
        act = "C";
        sup = "01";
        exp = "Y";
        cc = "BSTH";
        albr = "Y";
        sca = "Y";
        agauth = "Y";
        dated = "19991225";
        authName = "auth name";
        actl = "ROOM1234567";
        ai = "Y";
        apot = "A1234567890";
        lst = "B0987654321";
        lso = "123456";
        tos = "1A-";
        spec = "SVCPR";
        nc = "LY--";
        nci = "02QC3.RVO";
        secnci = "02RV2.T";
        rpon = "RPON1234567890";
        rord = "rord0987654321";
        lspAuth = "lspa";
        lspAuthDate = "19991231";
        lspAuthName = "lsp_auth_name";
```

```
        cic = "1234";
        cust = "customer name";
        bil = "A";
        banl = "qweasdl234098";
        acna = "ABC";
        vta = "variable";
        init = "initiator";
        initTelNo = "7326994801";
        initFaxNo = "7326994802";
        impCon = "implementation";
        impConTelNo = "7326994803";
        impConPager = "7326994804";
        remarks = "thisis a remark that is not very long";
        pbt = "A";
};


eu = {
        header = {
                administrative = {
                        ibt = "9";
                };      # Administrative Section
                locationAndAccess = {
                        aact = "A";
                        locnum = "";
                        euName = "jack and jill";
                        sano = "12375";
                        sasf = "abcd";
                        sasd = "E";
                        sasn = "testing";
                        sath = "denver";
                        sass = "suff";
                        sadlo = "acorss the bagel shop";
                        euFloor = "twenty one";
                        euRoom = "123456789";
                        euBldg = "building1";
                        euCity = "Denver";
                        euState = "CO";
                        euZipCode = "80231";
                        lconName = "local contact";
                        lconTelNo = "1234567890";
                        eumi = "Y";
                        acc = "access using front door";
                        wsop = "V";
                };      # Location and Access Header Section
                bill = {
                        ean = "1234567890";
                        eatn = "";
                        fbi = "";
                        fbBillName = "";
                        fbSbillName = "joe";
                        fbStreet = "1904 s york st";
                        fbFloor = "";
                        fbRoom = "";
                        fbCity = "denver";
                        fbState = "CO";
                        fbZipCode = "80231";
                        fbBillcon = "ESI";
                        fbConTelNo = "3036891114";
```

```
                };      # Bill Section
        };      # Header Section
        detail = {
                locationAndAccessList = (
                        {
                                locnum = "123";
                                locact = "N";
                                euName = "jack and jill";
                                sano = "555";
                                sasf = "54321";
                                sasd = "E";
                                sasn = "South Gate";
                                sath = "25 South";
                                sass = "sass";
                                sadlo = "across the Rockies";
                                euFloor = "";
                                euRoom = "305";
                                euBldg = "south ga";
                                euCity = "Englewood";
                                euState = "CO";
                                euZipCode = "80231";
                                lconName = "ESI";
                                lconTelNo = "3036891114";
                                acc = "ask the receptionist";
                        },
                );      # Location And Access List
                disconnectInformationList = (
                        {
                                dnum = "qwert";
                                discNbr = "3036891114";
                        },
                );      # Disconnect Information List
        };      # Detail Section
};


np = {
        npqty = "00001";
        serviceDetailList = (
                {
                        baList = (
                                {
                                        ba = "A";
                                        block = "B";
                                };
                        );      # Blocking Activity List
                        cftn = "";
                        ckr = "abcdefghijklmnopqrstuvwxyz123456789012345";
                        ecckt = "101.T1.NCYMNY50.NYCMNY54W01";
                        fpi = "A";
                        lean = "";
                        leatn = "";
                        lna = "P";
                        lnum = "12345";
                        locnum = "123";
                        lpic = "LPC1";
                        npi = "A";
                        npt = "A";
                        nptg = "1234ABCD";
```

```
                        portedNbr = "3033061995";
                        rti = "123abc";
                        tcFr = "1234567890";
                        tcOpt = "ST";
                        tcPer = "20000112";
                        tcToPrimary = "1234567890";
                        tcToSecondary = "1234567890";
                        primary = {
                                tcid = "01";
                                tcName = "abcdefghijklmnopqrstuvwxyz 12345678";
                        };      # Transfer of Calls Primary
                        secondary = {
                                tcid = "02";
                                tcName = "12345678 abcdefghijklmnopqrstuvwxyz";
                        };      # Transfer of Calls Secondary
                        tnp = "123";
                },
        );      # Service Detail List
};

dsdl = {
        listingList = (
                {
                        listingControl = {
                                dlnum = 0027;
                                lact = "D";
                                ali = ABC;
                                rty = "";
                                lty = "";
                                tt = "";
                                styc = CI;
                                toa = "";
                        };
                        listingInstruction = {
                                wpp = "";
                                doi = "";
                                ltn = 3037770000;
                                nstn = "";
                                lnln = "NAMES R US";
                                lnfn = "";
                                lnpl = "";
                                pla = "";
                                des = "";
                                tl = "";
                                title1 = "";
                                title2 = "";
                                nick = "";
                                lapr = "";
                                lano = "";
                                lasf = "";
                                lasd = "";
                                lasn = "123 MAIN ST";
                                lath = "";
                                lass = "";
                                laloc = "";
                                last = "";
                                yph = YPH1234;
                                sic = 2345;
                        };
```

```
                        listingIndicators = {
                                adi = "";
                                dirname = "";
                                dirsub = "";
                                adv = "";
                                dml = "";
                                dlnm = "";
                                bro = "";
                        };
                        dscr = {
                                alirSequencing = {
                                        so = "";
                                        seqtext = "";
                                        seqaddr = "";
                                        seqtn = "";
                                        lvl = "";
                                        hs = "";
                                        htn = "";
                                };      # ALIR Sequencing Section
                                levelDetailList = (
                                        {
                                                lvl = "";
                                                ins1 = "";
                                                so1 = "";
                                                seqtext1 = "";
                                                seqaddr1 = "";
                                                seqtn1 = "";
                                                intn = "";
                                                innstn = "";
                                                intext = "";
                                                inaddr = "";
                                        },
                                );      # Level Detail List
                        };      # Caption Request
                },
        );
        deliveryAddress = {
                dact = "N";
                nameDel = "RIGHT HERE";
                ddapr = "";
                ddano = "";
                ddasf = "";
                ddasd = "";
                ddasn = "123 MAIN ST";
                ddath = "";
                ddass = "";
                ddalo = "";
                ddadlo = "";
                ddaloc = ANYTOWN;
                ddast = AL;
                ddazc = 13579;
        };
};
```

**xstTestClient Output:**

```
status={
```

```
 msgId=00;
 msgTxt="LSR submitted to BellSouth";
};
messageHeader={
 dateSent=1999090712423732;
};
dueDate={
 dueDate=19991223;
 msgId=TAGT0000CDD;
 msgText="CALCULATED DUE DATE PROVIDED";
};
```

# 3.3 LOOPNP

**API Input:**

```
standardHeader{
 subTransaction=1;
 companyCode=BSTH;
 msgId=;
 msgTxt=;
 leoMessage=;
 applId=Chandra;
 userId=eeb;
 trxId=TagLeoServer_3_1_mut39cc2269;
}
orderHeader{
 base{
  apiVersion=3.1;
  appId=Chandra;
  userid=eeb;
  companyCode=XYZ;
  inquiryNumber=BSTH-PON-0000-11;
  dateSent=1999090812494941;
 }
 cc=BSTH;
 testProdIndicator=0;
}
lsr{
 ccna=CNA;
 pon=PON-0000;
 version=11;
 lsrNumber=LSR_NUMBER;
 locqty=123;
 htqty=;
 an=;
 atn=1234567890;
 serviceCenter=LCSC;
 dateSent=19991208;
 ddd=19991225;
 apptimeDdd=0900-1100;
 dddo=19991226;
 dfdt=1200;
 project=TAGMUT;
 chc=Y;
 reqtyp=BB;
 act=V;
 sup=01;
 exp=Y;
 cc=BSTH;
 albr=Y;
 sca=Y;
 agauth=Y;
 dated=19991225;
 authName=auth name;
 porttyp=;
 actl=ROOM1234567;
 ai=Y;
```

```
apot=A1234567890;
lst=B0987654321;
lso=123456;
tos=4C-;
spec=SVCPR;
nc=TY--;
nci=;
secnci=;
rpon=RPON1234567890;
rord=rord0987654321;
lspAuth=lspa;
lspAuthDate=19991231;
lspAuthName=lsp_auth_name;
cic=1234;
cust=customer name;
bi1=A;
ban1=qweasd1234098;
bi2=A;
ban2=zxcdsa0987123;
acna=ABC;
ebd=;
billName=;
sbillName=;
billNameStreet=;
billNameFloor=;
billNameRoom=;
billNameCity=;
billNameState=;
billNameZipCode=;
billCon=;
billConTelNo=;
vta=variable;
init=initiator;
initTelNo=7326994801;
initFaxNo=7326994802;
initStreet=;
initFloor=;
initRoomMailStop=;
initCity=;
initState=;
initZipCode=;
impCon=implementation;
impConTelNo=7326994803;
impConPager=7326994804;
altImpCon=;
altImpConTelNo=;
altImpConPager=;
dsgCon=;
drc=;
dsgConTelNo=;
dsgConFaxNo=;
dsgConStreet=;
dsgConFloor=;
dsgConRoomMailStop=;
dsgConCity=;
dsgConState=;
dsgConZipCode=;
remarks=thisis a remark that is not very long;
pbt=;
```

```
 bcs=;
 huntGroupInformationList{
 }
}
eu{
 header{
  administrative{
   dqty=;
   ibt=9;
  }
  locationAndAccess{
   aact=A;
   locnum=000;
   euName=jack and jill;
   sano=1904;
   sasf=abcd;
   sasd=E;
   sasn=york street;
   sath=denver;
   sass=suff;
   sadlo=acorss the bagel shop;
   euFloor=2ndfloor;
   euRoom=205;
   euBldg=S York;
   euCity=Denver;
   euState=CO;
   euZipCode=80231;
   lconName=Denver Univ;
   lconTelNo=;
   eumi=;
   acc=access using front door;
   wsop=V;
   erl=Y;
  }
  insideWire{
   iwOptions=;
   iwConName=;
   iwConTelNo=3036891114;
  }
  bill{
   ean=;
   eatn=3033061994;
   fbi=N;
   fbBillName=Messrs jack and jill;
   fbSbillName=joe;
   fbStreet=1904 s york st;
   fbFloor=2ndfloor;
   fbRoom=205;
   fbCity=denver;
   fbState=CO;
   fbZipCode=80231;
   fbBillCon=ESI;
   fbConTelNo=;
  }
 }
 detail{
  locationAndAccessList{
   0{
    locnum=123;
```

```
    locact=N;
    euName=jack and jill;
    sano=555;
    sasf=54321;
    sasd=E;
    sasn=South Gate;
    sath=25 South;
    sass=sass;
    sadlo=across the Rockies;
    euFloor=3rdfloor;
    euRoom=305;
    euBldg=sougate;
    euCity=Englewood;
    euState=CO;
    euZipCode=80231;
    lconName=ESI;
    lconTelNo=;
    acc=ask the receptionist;
   }
  }
  disconnectInformationList{
   0{
    dnum=qwert;
    discNbr=3036891114;
    ter=;
    tcOpt=;
    tcPer=;
    tcToPrimary=;
    tcToSecondary=;
    primary{
     tcid=;
     tcName=;
    }
    secondary{
     tcid=;
     tcName=;
    }
   }
  }
 }
}
dsdl{
 listingList{
 }
 deliveryAddress{
  dact=;
  nameDel=;
  ddapr=;
  ddano=;
  ddasf=;
  ddasd=;
  ddasn=;
  ddath=;
  ddass=;
  ddalo=;
  ddadlo=;
  ddaloc=;
  ddast=;
  ddazc=;
```

```
  directoryTypeList{
  }
 }
}
lsnp{
 lqty=123;
 npqty=12345;
 serviceDetailList{
  0{
   baList{
    0{
     ba=;
     block=;
    }
   }
   cableId=P1234;
   cfa=;
   cftn=;
   chanPair=CHANP;
   chanPair2=CHANP2;
   ckr=;
   ecckt=;
   fpi=;
   insideWireList{
    0{
     iwjk=;
     iwjq=;
    }
   }
   jkCode=;
   jkNum=;
   jkPos=;
   jr=;
   lean=;
   leatn=;
   lna=V;
   lnum=00001;
   locnum=;
   lpic=;
   nidr=;
   npi=;
   npt=A;
   nptg=12345678;
   portedNbr=1234567890;
   relayRack=;
   rti=A12345;
   san=;
   shelf=;
   slot=;
   sltn=;
   systemId=;
   tcFr=;
   tcOpt=;
   tcPer=;
   tcToPrimary=;
   tcToSecondary=;
   primary{
    tcid=;
    tcName=;
```

```
    }
    secondary{
     tcid=;
     tcName=;
    }
    tnp=;
    tsp=;
   }
  }
}
```

**API Output:**

```
status{
 msgId=00;
 msgTxt=LSR submitted to BellSouth;
}
dueDate{
 dueDate=19991223;
 msgId=TAGT0000CDD;
 msgText="CALCULATED DUE DATE PROVIDED";
}
```

**xstTestClient Input:**

```
RequestType = loopnp;
testProdIndicator = T;
lsr = {
        ccna = "CNA";
        pon = "PON-0000";
        version = "11";
        lsrNumber = "LSR_NUMBER";
        locqty = "123";
        atn = "1234567890";
        serviceCenter = "LCSC";
        dateSent = "19991208";
        ddd = "19991225";
        apptimeDdd = "0900-1100";
        dddo = "19991226";
        dfdt = "1200";
        project = "TAGMUT";
        chc = "Y";
        reqtyp = "BB";
        act = "V";
        sup = "01";
        exp = "Y";
        cc = "BSTH";
        albr = "Y";
        sca = "Y";
        agauth = "Y";
        dated = "19991225";
        authName = "auth name";
        actl = "ROOM1234567";
        ai = "Y";
        apot = "A1234567890";
        lst = "B0987654321";
        lso = "123456";
        tos = "4C-";
        spec = "SVCPR";
```

```
        nc = "TY--";
        rpon = "RPON1234567890";
        rord = "rord0987654321";
        lspAuth = "lspa";
        lspAuthDate = "19991231";
        lspAuthName = "lsp_auth_name";
        cic = "1234";
        cust = "customer name";
        bi1 = "A";
        ban1 = "qweasd1234098";
        bi2 = "A";
        ban2 = "zxcdsa0987123";
        acna = "ABC";
        vta = "variable";
        init = "initiator";
        initTelNo = "7326994801";
        initFaxNo = "7326994802";
        impCon = "implementation";
        impConTelNo = "7326994803";
        impConPager = "7326994804";
        remarks = "thisis a remark that is not very long";
};

eu = {
        header = {
                administrative = {
                        ibt = "9";
                };      # Administrative Section
                locationAndAccess = {
                        aact = "A";
                        locnum = "000";
                        euName = "jack and jill";
                        sano = "1904";
                        sasf = "abcd";
                        sasd = "E";
                        sasn = "york street";
                        sath = "denver";
                        sass = "suff";
                        sadlo = "acorss the bagel shop";
                        euFloor = "2ndfloor";
                        euRoom = "205";
                        euBldg = "S York";
                        euCity = "Denver";
                        euState = "CO";
                        euZipCode = "80231";
                        lconName = "Denver Univ";
                        acc = "access using front door";
                        wsop = "V";
                        erl = "Y";
                };      # Location and Access Header Section
                insideWire = {
                        iwconTelNo = "3036891114";
                };      # Inside Wire Section
                bill = {
                        eatn = "3033061994";
                        fbi = "N";
                        fbBillName = "Messrs jack and jill";
                        fbSbillName = "joe";
                        fbStreet = "1904 s york st";
```

```
                        fbFloor = "2ndfloor";
                        fbRoom = "205";
                        fbCity = "denver";
                        fbState = "CO";
                        fbZipCode = "80231";
                        fbBillcon = "ESI";
                };      # Bill Section
        };      # Header Section
        detail = {
                locationAndAccessList = (
                        {
                                locnum = "123";
                                locact = "N";
                                euName = "jack and jill";
                                sano = "555";
                                sasf = "54321";
                                sasd = "E";
                                sasn = "South Gate";
                                sath = "25 South";
                                sass = "sass";
                                sadlo = "across the Rockies";
                                euFloor = "3rdfloor";
                                euRoom = "305";
                                euBldg = "sougate";
                                euCity = "Englewood";
                                euState = "CO";
                                euZipCode = "80231";
                                lconName = "ESI";
                                acc = "ask the receptionist";
                        },
                );      # Location And Access List
                disconnectInformationList = (
                        {
                                dnum = "qwert";
                                discNbr = "3036891114";
                        },
                );      # Disconnect Information List
        };      # Detail Section
};

lsnp = {
        lqty = "123";
        npqty = "12345";
        serviceDetailList = (
                {
                        baList = (
                                {
                                        ba = "";
                                        block = "";
                                },
                        );      # Blocking Activity List
                        cableId = "P1234";
                        cfa = "";
                        cftn = "";
                        chanPair = "CHANP";
                        chanPair2 = "CHANP2";
                        ckr = "";
                        ecckt = "";
                        fpi = "";
```

```
                        insideWireList = (
                                {
                                        iwjk = "";
                                        iwjq = "";
                                },
                        );      # Inside Wire List
                        jkCode = "";
                        jkNum = "";
                        jkPos = "";
                        jr = "";
                        lean = "";
                        leatn = "";
                        lna = "V";
                        lnum = "00001";
                        locnum = "";
                        lpic = "";
                        nidr = "";
                        npi = "";
                        npt = "A";
                        nptg = "12345678";
                        portedNbr = "1234567890";
                        relayRack = "";
                        rti = "A12345";
                        san = "";
                        shelf = "";
                        slot = "";
                        systemId = "";
                        tcOpt = "";
                        tcPer = "";
                        tcToPrimary = "";
                        tcToSecondary = "";
                        primary = {
                                tcid = "";
                                tcName = "";
                        };      # Transfer of Calls Primary
                        secondary = {
                                tcid = "";
                                tcName = "";
                        };      # Transfer of Calls Secondary
                        tnp = "";
                        tsp = "";
                },
        );      # Service Detail List
};
```

**xstTestClient Output:**

```
status={
 msgId=00;
 msgTxt="LSR submitted to BellSouth";
};
messageHeader={
 dateSent=1999090712423732;
};
dueDate={
 dueDate=19991223;
 msgId=TAGT0000CDD;
```

```
 msgText="CALCULATED DUE DATE PROVIDED";
};
```

## 3.4 PORT

**API Input:**

```
standardHeader{
 subTransaction=1;
 companyCode=BSTH;
 msgId=;
 msgTxt=;
 leoMessage=;
 applId=Chandra;
 userId=eeb;
 trxId=TagLeoServer_3_1_mut47ccfd2a;
}
orderHeader{
 base{
  apiVersion=3.1;
  appId=Chandra;
  userid=eeb;
  companyCode=XYZ;
  inquiryNumber=BSTH-PON-0000-11;
  dateSent=1999090813104093;
 }
 cc=BSTH;
 testProdIndicator=0;
}
lsr{
 ccna=CNA;
 pon=PON-0000;
 version=11;
 lsrNumber=LSR_NUMBER;
 locqty=123;
 htqty=;
 an=;
 atn=1234567890;
 serviceCenter=LCSC;
 dateSent=19991208;
 ddd=19991225;
 apptimeDdd=0159-0823;
 dddo=19991226;
 dfdt=1200;
 project=Project1Identifi;
 chc=Y;
 reqtyp=FB;
 act=R;
 sup=01;
 exp=N;
 cc=BSTH;
 albr=N;
 sca=N;
 agauth=Y;
 dated=19991225;
 authName=Authorizatn Nam;
 porttyp=L;
 actl=ROOM1234567;
 ai=Y;
```

```
apot=A1234567890;
lst=Local1Servi;
lso=;
tos=1AM;
spec=SPEC567;
nc=LY--;
nci=02QC3.RVO;
secnci=02RV2.T;
rpon=RELAT4.PURCH;
rord=rord0987654321;
lspAuth=LSPA;
lspAuthDate=19991231;
lspAuthName=lsp_auth_name;
cic=1234;
cust=customer name;
bil=A;
ban1=qweasd1234098;
bi2=;
ban2=;
acna=;
ebd=;
billName=;
sbillName=;
billNameStreet=;
billNameFloor=;
billNameRoom=;
billNameCity=;
billNameState=;
billNameZipCode=;
billCon=;
billConTelNo=;
vta=Variable1Term1Agr;
init=initiator;
initTelNo=7326994801;
initFaxNo=7326994802;
initStreet=;
initFloor=;
initRoomMailStop=;
initCity=;
initState=;
initZipCode=;
impCon=implementation;
impConTelNo=7326994803;
impConPager=Pager1Numebr1IMP1CONTACTs;
altImpCon=;
altImpConTelNo=;
altImpConPager=;
dsgCon=DSG_CON;
drc=DRC;
dsgConTelNo=7326994806;
dsgConFaxNo=1234567890;
dsgConStreet=street address;
dsgConFloor=;
dsgConRoomMailStop=;
dsgConCity=;
dsgConState=NJ;
dsgConZipCode=08905;

remarks=12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678
```

```
901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234
567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890;
 pbt=A;
 bcs=;
 huntGroupInformationList{
 }
}
eu{
 header{
  administrative{
   dqty=;
   ibt=9;
  }
  locationAndAccess{
   aact=A;
   locnum=000;
   euName=jack and jill;
   sano=1904;
   sasf=abcd;
   sasd=E;
   sasn=york street;
   sath=denver;
   sass=suff;
   sadlo=acorss the bagel shop;
   euFloor=2ndfloor;
   euRoom=205;
   euBldg=S York;
   euCity=Denver;
   euState=CO;
   euZipCode=80231;
   lconName=Denver Univ;
   lconTelNo=3033061995;
   eumi=;
   acc=access using front door;
   wsop=;
   erl=;
  }
  insideWire{
   iwOptions=;
   iwConName=;
   iwConTelNo=;
  }
  bill{
   ean=;
   eatn=3033061994;
   fbi=Y;
   fbBillName=Messrs jack and jill;
   fbSbillName=joe;
   fbStreet=1904 s york st;
   fbFloor=2ndfloor;
   fbRoom=205;
   fbCity=denver;
   fbState=CO;
   fbZipCode=80231;
   fbBillCon=ESI;
   fbConTelNo=3036891114;
  }
 }
 detail{
```

```
  locationAndAccessList{
   0{
    locnum=123;
    locact=N;
    euName=jack and jill;
    sano=555;
    sasf=54321;
    sasd=E;
    sasn=South Gate;
    sath=25 South;
    sass=sass;
    sadlo=across the Rockies;
    euFloor=3rdfloor;
    euRoom=305;
    euBldg=south te;
    euCity=Englewood;
    euState=CO;
    euZipCode=80231;
    lconName=3036891114;
    lconTelNo=;
    acc=ask the receptionist;
   }
  }
  disconnectInformationList{
   0{
    dnum=qwert;
    discNbr=3036891114;
    ter=;
    tcOpt=;
    tcPer=;
    tcToPrimary=;
    tcToSecondary=;
    primary{
     tcid=;
     tcName=;
    }
    secondary{
     tcid=;
     tcName=;
    }
   }
  }
 }
}
dsdl{
 listingList{
 }
 deliveryAddress{
  dact=N;
  nameDel=asdfg;
  ddapr=;
  ddano=;
  ddasf=;
  ddasd=;
  ddasn=qwert;
  ddath=;
  ddass=;
  ddalo=;
  ddadlo=;
```

```
   ddaloc=asdfg;
   ddast=fg;
   ddazc=12345;
   directoryTypeList{
   }
  }
 }
 port{
  ord=;
  pqty=001;
  serviceDetailList{
   0{
    baList{
    }
    cableId=;
    cfa=;
    chanPair=BBBB1;
    ckr=;
    ecckt=800.123.4567;
    fpi=;
    lean=;
    leatn=;
    lna=R;
    lneclssvc=;
    lnex=;
    lnum=00001;
    locnum=;
    lpic=;
    matn=;
    npi=;
    otn=;
    pic=;
    pulse=;
    relayRack=;
    san=;
    sdi=;
    sgnl=;
    shelf=QWAAAE;
    slot=;    sltn=;
    ssig=;
    systemId=;
    tcFr=;
    tcOpt=;
    tcPer=;
    tcToPrimary=;
    tcToSecondary=;
    primary{
     tcid=;
     tcName=;
    }
    secondary{
     tcid=;
     tcName=;
    }
    ters=;
    tli=;
    tns=1234567890-1230;
    tsp=;
    featureList{
```

```
    }
   }
  }
 }
}
```

**API Output:**

```
status{
 msgId=00;
 msgTxt=LSR submitted to BellSouth;
}
dueDate{
 dueDate=19991223;
 msgId=TAGT0000CDD;
 msgText="CALCULATED DUE DATE PROVIDED";
}
```

**xstTestClient Input:**

```
RequestType = port;
testProdIndicator = T;
lsr = {
        ccna = "CNA";
        pon = "PON-0000";
        version = "11";
        lsrNumber = "LSR_NUMBER";
        locqty = "123";
        atn = "1234567890";
        serviceCenter = "LCSC";
        dateSent = "19991208";
        ddd = "19991225";
        apptimeDdd = "0159-0823";
        dddo = "19991226";
        dfdt = "1200";
        project = "Project1Identifi";
        chc = "Y";
        reqtyp = "FB";
        act = "R";
        sup = "01";
        exp = "N";
        cc = "BSTH";
        albr = "N";
        sca = "N";
        agauth = "Y";
        dated = "19991225";
        authName = "Authorizatn Nam";
        porttyp = "L";
        actl = "ROOM1234567";
        ai = "Y";
        apot = "A1234567890";
        lst = "Local1Servi";
        tos = "1AM";
        spec = "SPEC567";
        nc = "LY--";
        nci = "02QC3.RVO";
        secnci = "02RV2.T";
        rpon = "RELAT4.PURCH";
        rord = "rord0987654321";
```

```
        lspAuth = "LSPA";
        lspAuthDate = "19991231";
        lspAuthName = "lsp_auth_name";
        cic = "1234";
        cust = "customer name";
        bil = "A";
        banl = "qweasd1234098";
        vta = "VariablelTermlAgr";
        init = "initiator";
        initTelNo = "7326994801";
        initFaxNo = "7326994802";
        impCon = "implementation";
        impConTelNo = "7326994803";
        impConPager = "Pagerl Numebrl IMPlCONTACTs";
        dsgCon = "DSG_CON";
        drc = "DRC";
        dsgConTelNo = "7326994806";
        dsgConFaxNo = "1234567890";
        dsgConStreet = "street address";
        dsgConState = "NJ";
        dsgConZipCode = "08905";
        remarks =
"12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345
67890123456789012345678901234567890123456789012345678901234567890123456789012345678901
23456789012345678901234567890123456789012345678901234567890123456789012345678901234567890";
        pbt = "A";
};

eu = {
        header = {
                administrative = {
                        #dqty = "123";
                        ibt = "9";
                };      # Administrative Section
                locationAndAccess = {
                        aact = "A";
                        locnum = "000";
                        euName = "jack and jill";
                        sano = "1904";
                        sasf = "abcd";
                        sasd = "E";
                        sasn = "york street";
                        sath = "denver";
                        sass = "suff";
                        sadlo = "acorss the bagel shop";
                        euFloor = "2ndfloor";
                        euRoom = "205";
                        euBldg = "S York";
                        euCity = "Denver";
                        euState = "CO";
                        euZipCode = "80231";
                        lconName = "Denver Univ";
                        lconTelNo = "3033061995";
                        acc = "access using front door";
                };      # Location and Access Header Section
                bill = {
                        eatn = "3033061994";
                        fbi = "Y";
                        fbBillName = "Messrs jack and jill";
```

```
                        fbSbillName = "joe";
                        fbStreet = "1904 s york st";
                        fbFloor = "2ndfloor";
                        fbRoom = "205";
                        fbCity = "denver";
                        fbState = "CO";
                        fbZipCode = "80231";
                        fbBillcon = "ESI";
                        fbConTelNo = "3036891114";
                };      # Bill Section
        };      # Header Section
        detail = {
                locationAndAccessList = (
                        {
                                locnum = "123";
                                locact = "N";
                                euName = "jack and jill";
                                sano = "555";
                                sasf = "54321";
                                sasd = "E";
                                sasn = "South Gate";
                                sath = "25 South";
                                sass = "sass";
                                sadlo = "across the Rockies";
                                euFloor = "3rdfloor";
                                euRoom = "305";
                                euBldg = "south te";
                                euCity = "Englewood";
                                euState = "CO";
                                euZipCode = "80231";
                                lconName = "ESI";
                                lconName = "3036891114";
                                acc = "ask the receptionist";
                        },
                );      # Location And Access List
                disconnectInformationList = (
                        {
                                dnum = "qwert";
                                discNbr = "3036891114";
                        },
                );      # Disconnect Information List
        };      # Detail Section
};
dsdl = {
        deliveryAddress = {
                dact=N;
                ddasn=qwert;
                nameDel=asdfg;
                ddaloc=asdfg;
                ddast=fg;
                ddazc=12345;
        };

};
port = {
        pqty = "001";
        serviceDetailList = (
                {
                        chanPair = "BBBB1";
```

```
                        ecckt = "800.123.4567";
                        lna = "R";
                        lnum = "00001";
                        shelf = "QWAAAE";
                        tns = "1234567890-1230";

                },
        );      # Service Detail List
};
```

**xstTestClient Output:**

```
status={
 msgId=00;
 msgTxt="LSR submitted to BellSouth";
};
messageHeader={
 dateSent=1999090712423732;
};
dueDate={
 dueDate=19991223;
 msgId=TAGT0000CDD;
 msgText="CALCULATED DUE DATE PROVIDED";
};
```

## 3.5 LOOPPORT

**API Input:**

```
standardHeader{
 subTransaction=1;
 companyCode=BSTH;
 msgId=;
 msgTxt=;
 leoMessage=;
 applId=Chandra;
 userId=eeb;
 trxId=TagLeoServer_3_1_mut62121cbb;
}
orderHeader{
 base{
  apiVersion=3.1;
  appId=Chandra;
  userid=eeb;
  companyCode=XYZ;
  inquiryNumber=BSTH-PON-0000-11;
  dateSent=1999090813175294;
 }
 cc=BSTH;
 testProdIndicator=0;
}
lsr{
 ccna=CNA;
 pon=PON-0000;
 version=11;
 lsrNumber=LSR_NUMBER;
 locqty=123;
```

```
htqty=;
an=;
atn=1234567890;
serviceCenter=LCSC;
dateSent=19991208;
ddd=19991225;
apptimeDdd=0900-1100;
dddo=19991226;
dfdt=1200;
project=TAGMUT;
chc=Y;
reqtyp=MB;
act=C;
sup=01;
exp=Y;
cc=BSTH;
albr=Y;
sca=Y;
agauth=Y;
dated=19991225;
authName=auth name;
porttyp=L;
actl=ROOM1234567;
ai=Y;
apot=A1234567890;
lst=B0987654321;
lso=123456;
tos=1AM;
spec=SVCPR;
nc=LY--;
nci=02QC3.RVO;
secnci=02RV2.T;
rpon=RPON1234567890;
rord=rord0987654321;
lspAuth=lspa;
lspAuthDate=19991231;
lspAuthName=lsp_auth_name;
cic=1234;
cust=customer name;
bi1=A;
ban1=qweasd1234098;
bi2=;
ban2=;
acna=ABC;
ebd=;
billName=;
sbillName=;
billNameStreet=;
billNameFloor=;
billNameRoom=;
billNameCity=;
billNameState=;
billNameZipCode=;
billCon=;
billConTelNo=;
vta=variable;
init=initiator;
initTelNo=7326994801;
initFaxNo=7326994802;
```

```
 initStreet=;
 initFloor=;
 initRoomMailStop=;
 initCity=;
 initState=;
 initZipCode=;
 impCon=implementation;
 impConTelNo=7326994803;
 impConPager=7326994804;
 altImpCon=;
 altImpConTelNo=;
 altImpConPager=;
 dsgCon=;
 drc=;
 dsgConTelNo=;
 dsgConFaxNo=;
 dsgConStreet=;
 dsgConFloor=;
 dsgConRoomMailStop=;
 dsgConCity=;
 dsgConState=;
 dsgConZipCode=;
 remarks=thisis a remark that is not very long;
 pbt=;
 bcs=;
 huntGroupInformationList{
 }
}
eu{
 header{
  administrative{
   dqty=;
   ibt=9;
  }
  locationAndAccess{
   aact=A;
   locnum=000;
   euName=jack and jill;
   sano=1904;
   sasf=abcd;
   sasd=E;
   sasn=york street;
   sath=denver;
   sass=suff;
   sadlo=acorss the bagel shop;
   euFloor=2ndfloor;
   euRoom=205;
   euBldg=S York;
   euCity=Denver;
   euState=CO;
   euZipCode=80231;
   lconName=Denver Univ;
   lconTelNo=3033061995;
   eumi=Y;
   acc=access using front door;
   wsop=V;
   erl=;
  }
  insideWire{
```

```
  iwOptions=W;
  iwConName=Jack;
  iwConTelNo=3036891114;
 }
 bill{
  ean=;
  eatn=3033061994;
  fbi=Y;
  fbBillName=Messrs jack and jill;
  fbSbillName=joe;
  fbStreet=1904 s york st;
  fbFloor=2ndfloor;
  fbRoom=205;
  fbCity=denver;
  fbState=CO;
  fbZipCode=80231;
  fbBillCon=ESI;
  fbConTelNo=3036891114;
 }
}
detail{
 locationAndAccessList{
  0{
   locnum=123;
   locact=N;
   euName=jack and jill;
   sano=555;
   sasf=54321;
   sasd=E;
   sasn=South Gate;
   sath=25 South;
   sass=sass;
   sadlo=across the Rockies;
   euFloor=3rdfloor;
   euRoom=305;
   euBldg=south ga;
   euCity=Englewood;
   euState=CO;
   euZipCode=80231;
   lconName=ESI;
   lconTelNo=3036891114;
   acc=ask the receptionist;
  }
 }
 disconnectInformationList{
  0{
   dnum=qwert;
   discNbr=3036891114;
   ter=;
   tcOpt=;
   tcPer=;
   tcToPrimary=;
   tcToSecondary=;
   primary{
    tcid=;
    tcName=;
   }
   secondary{
    tcid=;
```

```
      tcName=;
     }
    }
   }
  }
 }
dsdl{
 listingList{
 }
 deliveryAddress{
  dact=N;
  nameDel=NAme;
  ddapr=;
  ddano=;
  ddasf=;
  ddasd=;
  ddasn=12;
  ddath=;
  ddass=;
  ddalo=;
  ddadlo=;
  ddaloc=012345;
  ddast=12;
  ddazc=12;
  directoryTypeList{
  }
 }
}
rs{
 ord=01234567890123456789;
 rsqty=123;
 serviceDetailList{
  0{
   baList{
    0{
     ba=A;
     block=A;
    }
   }
   cfa=012345678901234567890123456789012345678901;
   ckr=01234567890123456789012345678901234567890;
   cnam=012345678901234;
   ecckt=800.123.4567;
   fpi=A;
   ispid=12345678901234;
   insideWireList{
    0{
     iwjk=01234;
     iwjq=12;
    }
   }
   jkCode=01234;
   jkNum=01;
   jkPos=12;
   jr=Y;
   lean=;
   leatn=;
   lna=N;
   lneclssvc=01234;
```

```
    lnex=;
    lnum=01234;
    locnum=123;
    lpic=0123;
    matn=1;
    nidr=Y;
    npi=C;
    otn=;
    pic=0123;
    pulse=0123;
    san=01234567890123456789012345678 9;
    sdi=A;
    sgnl=01;
    ssig=01;
    tcFr=1234567890;
    tcOpt=ST;
    tcPer=19991212;
    tcToPrimary=1234557890;
    tcToSecondary=1234567810;
    primary{
     tcid=01;
     tcName=01234567890123456789012345678901234;
    }
    secondary{
     tcid=02;
     tcName=01234567890123456789012345678901234;
    }
    ters=0123456789;
    tli=1234567890;
    tns=0123456789-1234;
    tsp=012345678-01;
    featureList{
    }
   }
  }
 }
 port{
  ord=;
  pqty=123;
  serviceDetailList{
   0{
    baList{
    }
    cableId=;
    cfa=;
    chanPair=;
    ckr=;
    ecckt=;
    fpi=;
    lean=;
    leatn=;
    lna=C;
    lneclssvc=;
    lnex=;
    lnum=12345;
    locnum=;
    lpic=;
    matn=;
    npi=;
```

```
   otn=;
   pic=;
   pulse=;
   relayRack=;
   san=;
   sdi=;
   sgnl=;
   shelf=;
   slot=;
   sltn=;
   ssig=;
   systemId=;
   tcFr=;
   tcOpt=;
   tcPer=;
   tcToPrimary=;
   tcToSecondary=;
   primary{
    tcid=;
    tcName=;
   }
   secondary{
    tcid=;
    tcName=;
   }
   ters=;
   tli=;
   tns=0123456789-1234;
   tsp=;
   featureList{
   }
  }
 }
}
```

**API Output:**

```
status{
 msgId=00;
 msgTxt=LSR submitted to BellSouth;
}
dueDate{
 dueDate=19991223;
 msgId=TAGT0000CDD;
 msgText="CALCULATED DUE DATE PROVIDED";
}
```

**xstTestClient Input:**

```
RequestType = loopport;
testProdIndicator = T;
lsr = {
        ccna = "CNA";
        pon = "PON-0000";
        version = "11";
        lsrNumber = "LSR_NUMBER";
        locqty = "123";
        atn = "1234567890";
        serviceCenter = "LCSC";
```

```
        dateSent = "19991208";
        ddd = "19991225";
        apptimeDdd = "0900-1100";
        dddo = "19991226";
        dfdt = "1200";
        project = "TAGMUT";
        chc = "Y";
        reqtyp = "MB";
        act = "C";
        sup = "01";
        exp = "Y";
        cc = "BSTH";
        albr = "Y";
        sca = "Y";
        agauth = "Y";
        dated = "19991225";
        authName = "auth name";
        porttyp = "L";
        actl = "ROOM1234567";
        ai = "Y";
        apot = "A1234567890";
        lst = "B0987654321";
        lso = "123456";
        tos = "1AM";
        spec = "SVCPR";
        nc = "LY--";
        nci = "02QC3.RVO";
        secnci = "02RV2.T";
        rpon = "RPON1234567890";
        rord = "rord0987654321";
        lspAuth = "lspa";
        lspAuthDate = "19991231";
        lspAuthName = "lsp_auth_name";
        cic = "1234";
        cust = "customer name";
        bil = "A";
        banl = "qweasd1234098";
        acna = "ABC";
        vta = "variable";
        init = "initiator";
        initTelNo = "7326994801";
        initFaxNo = "7326994802";
        impCon = "implementation";
        impConTelNo = "7326994803";
        impConPager = "7326994804";
        remarks = "thisis a remark that is not very long";
};

eu = {
        header = {
                administrative = {
                        ibt = "9";
                };      # Administrative Section
                locationAndAccess = {
                        aact = "A";
                        locnum = "000";
                        euName = "jack and jill";
                        sano = "1904";
                        sasf = "abcd";
```

```
                sasd = "E";
                sasn = "york street";
                sath = "denver";
                sass = "suff";
                sadlo = "acorss the bagel shop";
                euFloor = "2ndfloor";
                euRoom = "205";
                euBldg = "S York";
                euCity = "Denver";
                euState = "CO";
                euZipCode = "80231";
                lconName = "Denver Univ";
                lconTelNo = "3033061995";
                eumi = "Y";
                acc = "access using front door";
                wsop = "V";
                erl = "";
        };      # Location and Access Header Section
        insideWire = {
                iwOptions = "W";
                iwconName = "Jack";
                iwConTelNo = "3036891114";
        };      # Inside Wire Section
        bill = {
                eatn = "3033061994";
                fbi = "Y";
                fbBillName = "Messrs jack and jill";
                fbSbillName = "joe";
                fbStreet = "1904 s york st";
                fbFloor = "2ndfloor";
                fbRoom = "205";
                fbCity = "denver";
                fbState = "CO";
                fbZipCode = "80231";
                fbBillcon = "ESI";
                fbConTelNo = "3036891114";
        };      # Bill Section
};      # Header Section
detail = {
        locationAndAccessList = (
                {
                        locnum = "123";
                        locact = "N";
                        euName = "jack and jill";
                        sano = "555";
                        sasf = "54321";
                        sasd = "E";
                        sasn = "South Gate";
                        sath = "25 South";
                        sass = "sass";
                        sadlo = "across the Rockies";
                        euFloor = "3rdfloor";
                        euRoom = "305";
                        euBldg = "south ga";
                        euCity = "Englewood";
                        euState = "CO";
                        euZipCode = "80231";
                        lconName = "ESI";
                        lconTelNo = "3036891114";
```

```
                                acc = "ask the receptionist";
                        },
                );      # Location And Access List
                disconnectInformationList = (
                        {
                                dnum = "qwert";
                                discNbr = "3036891114";
                        },
                );      # Disconnect Information List
        };      # Detail Section
};


dsdl = {
        deliveryAddress = {
                dact = "N";
                nameDel = "NAme";
                ddasn = "12";
                ddaloc = "012345";
                ddast = "12";
                ddazc = "12";
        };
};


port = {
        pqty = 123;
        serviceDetailList = (
                {
                        lna = C;
                        lnum = 12345;
                        tns = "0123456789-1234";
                },
        );
};


resale = {
        ord = "01234567890123456789";
        rsqty = 123;
        serviceDetailList = (
                {
                        baList = (
                                {
                                        ba = A;
                                        block = A;
                                },
                        );
                        cfa = 012345678901234567890123456789012345678901;
                        ckr = 01234567890123456789012345678901234567890;
                        cnam = 012345678901234;
                        ecckt = "800.123.4567";
                        fpi = A;
                        ispid = 12345678901234;
                        insideWireList = (
                                {
                                        iwjk = 01234;
                                        iwjq = 12;
```

```
                        };
                    );
                    jkCode = 01234;
                    jkNum = 01;
                    jkPos = 12;
                    jr = Y;
                    lean = "";
                    leatn = "";
                    lna = N;
                    lneclssvc = 01234;
                    lnex = "";
                    lnum = 01234;
                    locnum = 123;
                    lpic = 0123;
                    matn = 1;
                    nidr = Y;
                    npi = C;
                    #otn = 1234567890;
                    pic = 0123;
                    pulse = 0123;
                    san = 012345678901234567890123456789;
                    sdi = A;
                    sgnl = 01;
                    ssig = 01;
                    tcFr = 1234567890;
                    tcOpt = ST;
                    tcPer = 19991212;
                    tcToPrimary = 1234557890;
                    tcToSecondary = 1234567810;
                    primary = {
                            tcid = 01;
                            tcName = 01234567890123456789012345678901234;
                    };
                    secondary = {
                            tcid = 02;
                            tcName = 01234567890123456789012345678901234;
                    };
                    ters = 0123456789;
                    tli = 1234567890;
                    tns = 0123456789-1234;
                    tsp = 012345678-01;
                },
        );
};
```

**xstTestClient Output:**

```
status={
 msgId=00;
 msgTxt="LSR submitted to BellSouth";
};
messageHeader={
 dateSent=1999090712423732;
};
dueDate={
 dueDate=19991223;
 msgId=TAGT0000CDD;
 msgText="CALCULATED DUE DATE PROVIDED";
```

```
};
```

## 3.6 LOOP

**API Input:**

```
standardHeader{
 subTransaction=1;
 companyCode=BSTH;
 msgId=;
 msgTxt=;
 leoMessage=;
 applId=Chandra;
 userId=eeb;
 trxId=TagLeoServer_3_1_mut56c4748d;
}
orderHeader{
 base{
  apiVersion=3.1;
  appId=Chandra;
  userid=eeb;
  companyCode=XYZ;
  inquiryNumber=BSTH-LOOP-0000-11;
  dateSent=1999090712451325;
 }
 cc=BSTH;
 testProdIndicator=0;
}
lsr{
 ccna=CNA;
 pon=LOOP-0000;
 version=11;
 lsrNumber=LSR_NUMBER;
 locqty=123;
 htqty=;
 an=;
 atn=1234567890;
 serviceCenter=LCSC;
 dateSent=19991208;
 ddd=19991225;
 apptimeDdd=0900-1100;
 dddo=19991226;
 dfdt=1200;
 project=TAGMUT;
 chc=Y;
 reqtyp=AB;
 act=R;
 sup=01;
 exp=Y;
 cc=BSTH;
 albr=Y;
 sca=Y;
 agauth=Y;
 dated=19991225;
 authName=auth name;
 porttyp=;
 actl=ROOM1234567;
```

```
ai=Y;
apot=A1234567890;
lst=B0987654321;
lso=123456;
tos=1AM;
spec=SVCPR;
nc=LY--;
nci=02QC3.RVO;
secnci=02RV2.T;
rpon=RPON1234567890;
rord=rord0987654321;
lspAuth=lspa;
lspAuthDate=19991231;
lspAuthName=lsp_auth_name;
cic=1234;
cust=customer name;
bi1=A;
ban1=qweasd1234098;
bi2=;
ban2=;
acna=ABC;
ebd=;
billName=;
sbillName=;
billNameStreet=;
billNameFloor=;
billNameRoom=;
billNameCity=;
billNameState=;
billNameZipCode=;
billCon=;
billConTelNo=;
vta=variable;
init=initiator;
initTelNo=7326994801;
initFaxNo=7326994802;
initStreet=;
initFloor=;
initRoomMailStop=;
initCity=;
initState=;
initZipCode=;
impCon=implementation;
impConTelNo=7326994803;
impConPager=7326994804;
altImpCon=;
altImpConTelNo=;
altImpConPager=;
dsgCon=;
drc=;
dsgConTelNo=;
dsgConFaxNo=;
dsgConStreet=;
dsgConFloor=;
dsgConRoomMailStop=;
dsgConCity=;
dsgConState=;
dsgConZipCode=;
remarks=thisis a remark that is not very long;
```

```
 pbt=;
 bcs=;
 huntGroupInformationList{
 }
}
eu{
 header{
  administrative{
   dqty=;
   ibt=9;
  }
  locationAndAccess{
   aact=A;
   locnum=000;
   euName=jack & jill;
   sano=1904;
   sasf=abcd;
   sasd=E;
   sasn=york street;
   sath=denver;
   sass=suff;
   sadlo=acorss the bagel shop;
   euFloor=2ndfloor;
   euRoom=205;
   euBldg=S York;
   euCity=Denver;
   euState=CO;
   euZipCode=80231;
   lconName=Denver Univ;
   lconTelNo=3033061995;
   eumi=;
   acc=access using front door;
   wsop=;
   erl=;
  }
  insideWire{
   iwOptions=W;
   iwConName=Jack;
   iwConTelNo=3036891114;
  }
  bill{
   ean=;
   eatn=3033061994;
   fbi=Y;
   fbBillName=Messrs jack and jill;
   fbSbillName=joe;
   fbStreet=1904 s york st;
   fbFloor=2ndfloor;
   fbRoom=205;
   fbCity=denver;
   fbState=CO;
   fbZipCode=80231;
   fbBillCon=ESI;
   fbConTelNo=3036891114;
  }
 }
 detail{
  locationAndAccessList{
   0{
```

```
    locnum=001;
    locact=N;
    euName=jack & jill;
    sano=555;
    sasf=54321;
    sasd=E;
    sasn=South Gate;
    sath=25 South;
    sass=sass;
    sadlo=across the Rockies;
    euFloor=3rdfloor;
    euRoom=305;
    euBldg=south ga;
    euCity=Englewood;
    euState=CO;
    euZipCode=80231;
    lconName=ESI;
    lconTelNo=3036891114;
    acc=ask the receptionist;
   }
  }
  disconnectInformationList{
   0{
    dnum=qwert;
    discNbr=3036891114;
    ter=;
    tcOpt=;
    tcPer=;
    tcToPrimary=;
    tcToSecondary=;
    primary{
     tcid=;
     tcName=;
    }
    secondary{
     tcid=;
     tcName=;
    }
   }
  }
 }
}
dsdl{
 listingList{
 }
 deliveryAddress{
  dact=;
  nameDel=;
  ddapr=;
  ddano=;
  ddasf=;
  ddasd=;
  ddasn=;
  ddath=;
  ddass=;
  ddalo=;
  ddadlo=;
  ddaloc=;
  ddast=;
```

```
  ddazc=;
  directoryTypeList{
  }
 }
}
loop{
 lqty=001;
 serviceDetailList{
  0{
   cableId=PAAA1;
   cfa=abcdefghijklmnopqrstuvwxyz1234567890123456;
   chanPair=BBBB1;   chanPair2=BBBB2;
   ckr=abcdefghijklmnopqrstuvwxyz-12345678901234;
   discNbr=1234567890;
   ecckt=800.123.4567;
   insideWireList{
    0{
     iwjk=123ab;
     iwjq=12;
    }
   }
   jkCode=123ab;
   jkNum=1A;
   jkPos=12;
   jr=Y;
   lean=1234567890123;
   leatn=1234567890;
   lna=R;
   lnum=12352;
   locnum=000;
   nidr=N;
   relayRack=1234567890;
   san=abcdefghijklmnopqrstuvwxyz1234;
   shelf=123abc;
   slot=abc123;
   systemId=abc12;
   tcFr=1234567890;
   tcOpt=ST;
   tcPer=20001031;
   tcToPrimary=1234567890;
   tcToSecondary=1234567890;
   primary{
    tcid=01;
    tcName=abcdefghijklmnopqrstuvwxyz 12345678;
   }
   secondary{
    tcid=02;
    tcName=12345678 abcdefghijklmnopqrstuvwxyz;
   }
   ters=123;
   tsp=123456abcdef;
  }
 }
}
```

**API Output:**

```
status{
```

```
 msgId=00;
 msgTxt=LSR submitted to BellSouth;
}
dueDate{
 dueDate=19991223;
 msgId=TAGT0000CDD;
 msgText="CALCULATED DUE DATE PROVIDED";
}
```

**xstTestClient Input:**

```
RequestType = loop;
testProdIndicator = T;
lsr = {
        ccna = "CNA";
        pon = "LOOP-0000";
        version = "11";
        lsrNumber = "LSR_NUMBER";
        locqty = "123";
        atn = "1234567890";
        serviceCenter = "LCSC";
        dateSent = "19991208";
        ddd = "19991225";
        apptimeDdd = "0900-1100";
        dddo = "19991226";
        dfdt = "1200";
        project = "TAGMUT";
        chc = "Y";
        reqtyp = "AB";
        act = "R";
        sup = "01";
        exp = "Y";
        cc = "BSTH";
        albr = "Y";
        sca = "Y";
        agauth = "Y";
        dated = "19991225";
        authName = "auth name";
        actl = "ROOM1234567";
        ai = "Y";
        apot = "A1234567890";
        lst = "B0987654321";
        lso = "123456";
        tos = "1AM";
        spec = "SVCPR";
        nc = "LY--";
        nci = "02QC3.RVO";
        secnci = "02RV2.T";
        rpon = "RPON1234567890";
        rord = "rord0987654321";
        lspAuth = "lspa";
        lspAuthDate = "19991231";
        lspAuthName = "lsp_auth_name";
        cic = "1234";
        cust = "customer name";
        bil = "A";
        banl = "qweasd1234098";
        acna = "ABC";
        vta = "variable";
```

```
        init = "initiator";
        initTelNo = "7326994801";
        initFaxNo = "7326994802";
        impCon = "implementation";
        impConTelNo = "7326994803";
        impConPager = "7326994804";
        remarks = "thisis a remark that is not very long";
};

eu = {
        header = {
                administrative = {
                        ibt = "9";
                };      # Administrative Section
                locationAndAccess = {
                        aact = "A";
                        locnum = "000";
                        euName = "jack & jill";
                        sano = "1904";
                        sasf = "abcd";
                        sasd = "E";
                        sasn = "york street";
                        sath = "denver";
                        sass = "suff";
                        sadlo = "acorss the bagel shop";
                        euFloor = "2ndfloor";
                        euRoom = "205";
                        euBldg = "S York";
                        euCity = "Denver";
                        euState = "CO";
                        euZipCode = "80231";
                        lconName = "Denver Univ";
                        lconTelNo = "3033061995";
                        eumi = "";
                        acc = "access using front door";
                        wsop = "";
                        erl = "";
                };      # Location and Access Header Section
                insideWire = {
                        iwOptions = "W";
                        iwConName = "Jack";
                        iwConTelNo = "3036891114";
                };      # Inside Wire Section
                bill = {
                        eatn = "3033061994";
                        fbi = "Y";
                        fbBillName = "Messrs jack and jill";
                        fbSbillName = "joe";
                        fbStreet = "1904 s york st";
                        fbFloor = "2ndfloor";
                        fbRoom = "205";
                        fbCity = "denver";
                        fbState = "CO";
                        fbZipCode = "80231";
                        fbBillcon = "ESI";
                        fbConTelNo = "3036891114";
                };      # Bill Section
        };      # Header Section
        detail = {
```

```
                locationAndAccessList = (
                        {
                                locnum = "001"; #pp
                                locact = "N";
                                euName = "jack & jill";
                                sano = "555";
                                sasf = "54321";
                                sasd = "E";
                                sasn = "South Gate";
                                sath = "25 South";
                                sass = "sass";
                                sadlo = "across the Rockies";
                                euFloor = "3rdfloor";
                                euRoom = "305";
                                euBldg = "south ga";
                                euCity = "Englewood";
                                euState = "CO";
                                euZipCode = "80231";
                                lconName = "ESI";
                                lconTelNo = "3036891114";
                                acc = "ask the receptionist";
                        },
                );      # Location And Access List
                disconnectInformationList = (
                        {
                                dnum = "qwert";
                                discNbr = "3036891114";
                        },
                );      # Disconnect Information List
        };      # Detail Section
};
loop = {
        lqty = "001";
        serviceDetailList = (
                {
                        cableId = "PAAA1";
                        cfa = "abcdefghijklmnopqrstuvwxyz1234567890123456";
                        chanPair = "BBBB1";
                        chanPair2 = "BBBB2
                        ckr = "abcdefghijklmnopqrstuvwxyz-12345678901234";
                        discNbr = "1234567890";
                        ecckt = "800.123.4567";
                        insideWireList = (
                                {
                                        iwjk = "123ab";
                                        iwjq = "12";
                                },
                        );      # Inside Wire List
                        jkCode = "123ab";
                        jkNum = "1A";
                        jkPos = "12";
                        jr = "Y";
                        lean = "1234567890123";
                        leatn = "1234567890";
                        lna = "R";
                        lnum = "12352";
                        locnum = "000";
                        nidr = "N";
                        relayRack = "1234567890";
```

```
                        san = "abcdefghijklmnopqrstuvwxyz1234";
                        shelf = "123abc";
                        slot = "abc123";
                        systemId = "abc12";
                        tcFr = "1234567890";
                        tcOpt = "ST";
                        tcPer = "20001031";
                        tcToPrimary = "1234567890";
                        tcToSecondary = "1234567890";
                        primary = {
                                tcid = "01";
                                tcName = "abcdefghijklmnopqrstuvwxyz 12345678";
                        };      # Transfer of Calls Primary
                        secondary = {
                                tcid = "02";
                                tcName = "12345678 abcdefghijklmnopqrstuvwxyz";
                        };      # Transfer of Calls Secondary
                        ters = "123";
                        tsp = "123456abcdef";
                },
        );      # Service Detail List
};
```

**xstTestClient Output:**

```
status={
 msgId=00;
 msgTxt="LSR submitted to BellSouth";
};
messageHeader={
 dateSent=1999090712423732;
};
dueDate={
 dueDate=19991223;
 msgId=TAGT0000CDD;
 msgText="CALCULATED DUE DATE PROVIDED";
};
```

# 3.7 RESALE

**API Input:**

```
standardHeader{
 subTransaction=1;
 companyCode=BSTH;
 msgId=;
 msgTxt=;
 leoMessage=;
 applId=Chandra;
 userId=eeb;
 trxId=TagLeoServer_3_1_mut3b798847;
}
orderHeader{
 base{
  apiVersion=3.1;
  appId=Chandra;
  userid=eeb;
  companyCode=XYZ;
  inquiryNumber=BSTH-PON-0000-11;
  dateSent=1999090811041377;
 }
 cc=BSTH;
 testProdIndicator=0;
}
lsr{
 ccna=CNA;
 pon=PON-0000;
 version=11;
 lsrNumber=LSR_NUMBER;
 locqty=123;
 htqty=;
 an=;
 atn=1234567890;
 serviceCenter=LCSC;
 dateSent=19991208;
 ddd=19991225;
 apptimeDdd=0900-1100;
 dddo=19991226;
 dfdt=1200;
 project=TAGMUT;
 chc=Y;
 reqtyp=EB;
 act=N;
 sup=01;
 exp=Y;
 cc=BSTH;
 albr=Y;
 sca=Y;
 agauth=Y;
 dated=19991225;
 authName=auth name;
 porttyp=;
 actl=ROOM1234567;
 ai=Y;
 apot=A1234567890;
 lst=B0987654321;
 lso=123456;
```

```
tos=1AM;
spec=SVCPR;
nc=LY--;
nci=02QC3.RVO;
secnci=02RV2.T;
rpon=RPON1234567890;
rord=rord0987654321;
lspAuth=lspa;
lspAuthDate=19991231;
lspAuthName=lsp_auth_name;
cic=1234;
cust=customer name;
bi1=A;
ban1=qweasd1234098;
bi2=;
ban2=;
acna=ABC;
ebd=;
billName=;
sbillName=;
billNameStreet=;
billNameFloor=;
billNameRoom=;
billNameCity=;
billNameState=;
billNameZipCode=;
billCon=;
billConTelNo=;
vta=variable;
init=initiator;
initTelNo=7326994801;
initFaxNo=7326994802;
initStreet=;
initFloor=;
initRoomMailStop=;
initCity=;
initState=;
initZipCode=;
impCon=implementation;
impConTelNo=7326994803;
impConPager=7326994804;
altImpCon=;
altImpConTelNo=;
altImpConPager=;
dsgCon=;
drc=;
dsgConTelNo=;
dsgConFaxNo=;
dsgConStreet=;
dsgConFloor=;
dsgConRoomMailStop=;
dsgConCity=;
dsgConState=;
dsgConZipCode=;
remarks=thisis a remark that is not very long;
pbt=;
bcs=;
huntGroupInformationList{
}
```

```
}
eu{
 header{
  administrative{
   dqty=;
   ibt=9;
  }
  locationAndAccess{
   aact=A;
   locnum=000;
   euName=jack and jill;
   sano=1904;
   sasf=abcd;
   sasd=E;
   sasn=york street;
   sath=denver;
   sass=suff;
   sadlo=acorss the bagel shop;
   euFloor=2ndfloor;
   euRoom=205;
   euBldg=S York;
   euCity=Denver;
   euState=CO;
   euZipCode=80231;
   lconName=Denver Univ;
   lconTelNo=3033061995;
   eumi=Y;
   acc=access using front door;
   wsop=V;
   erl=;
  }
  insideWire{
   iwOptions=W;
   iwConName=Jack;
   iwConTelNo=3036891114;
  }
  bill{
   ean=;
   eatn=3033061994;
   fbi=Y;
   fbBillName=Messrs jack and jill;
   fbSbillName=joe;
   fbStreet=1904 s york st;
   fbFloor=2ndfloor;
   fbRoom=205;
   fbCity=denver;
   fbState=CO;
   fbZipCode=80231;
   fbBillCon=ESI;
   fbConTelNo=3036891114;
  }
 }
 detail{
  locationAndAccessList{
   0{
    locnum=123;
    locact=N;
    euName=jack and jill;
    sano=555;
```

```
    sasf=54321;
    sasd=E;
    sasn=South Gate;
    sath=25 South;
    sass=sass;
    sadlo=across the Rockies;
    euFloor=3rdfloor;
    euRoom=305;
    euBldg=south ga;
    euCity=Englewood;
    euState=CO;
    euZipCode=80231;
    lconName=ESI;
    lconTelNo=3036891114;
    acc=ask the receptionist;
   }
  }
  disconnectInformationList{
   0{
    dnum=qwert;
    discNbr=3036891114;
    ter=;
    tcOpt=;
    tcPer=;
    tcToPrimary=;
    tcToSecondary=;
    primary{
     tcid=;
     tcName=;
    }
    secondary{
     tcid=;
     tcName=;
    }
   }
  }
 }
}
dsdl{
 listingList{
 }
 deliveryAddress{
  dact=N;
  nameDel=NAme;
  ddapr=;
  ddano=;
  ddasf=;
  ddasd=;
  ddasn=12;
  ddath=;
  ddass=;
  ddalo=;
  ddadlo=;
  ddaloc=012345;
  ddast=12;
  ddazc=12;
  directoryTypeList{
  }
 }
```

```
}
rs{
 ord=01234567890123456789;
 rsqty=123;
 serviceDetailList{
  0{
   baList{
    0{
     ba=A;
     block=A;
    }
   }
   cfa=012345678901234567890123456789012345678901;
   ckr=01234567890123456789012345678901234567890;
   cnam=012345678901234;
   ecckt=800.123.4567;
   fpi=A;
   ispid=12345678901234;
   insideWireList{
    0{
     iwjk=01234;
     iwjq=12;
    }
   }
   jkCode=01234;
   jkNum=01;
   jkPos=12;
   jr=Y;
   lean=;
   leatn=;
   lna=N;
   lneclssvc=01234;
   lnex=;
   lnum=01234;
   locnum=123;
   lpic=0123;
   matn=1;
   nidr=Y;
   npi=C;
   otn=;
   pic=0123;
   pulse=0123;
   san=01234567890123456789012345678 9;
   sdi=A;
   sgnl=01;
   ssig=01;
   tcFr=1234567890;
   tcOpt=ST;
   tcPer=19991212;
   tcToPrimary=1234557890;
   tcToSecondary=1234567810;
   primary{
    tcid=01;
    tcName=012345678901234567890123456789012 34;
   }
   secondary{
    tcid=02;
    tcName=012345678901234567890123456789012 34;
   }
```

```
    ters=0123456789;
    tli=1234567890;
    tns=0123456789-1234;
    tsp=012345678-01;
    featureList{
    }
   }
  }
}
```

**API Output:**

```
status{
 msgId=00;
 msgTxt=LSR submitted to BellSouth;
}
dueDate{
 dueDate=19991223;
 msgId=TAGT0000CDD;
 msgText="CALCULATED DUE DATE PROVIDED";
}
```

**xstTestClient Input:**

```
RequestType = Resale;
testProdIndicator = T;
lsr = {
        ccna = "CNA";
        pon = "PON-0000";
        version = "11";
        lsrNumber = "LSR_NUMBER";
        locqty = "123";
        atn = "1234567890";
        serviceCenter = "LCSC";
        dateSent = "19991208";
        ddd = "19991225";
        apptimeDdd = "0900-1100";
        dddo = "19991226";
        dfdt = "1200";
        project = "TAGMUT";
        chc = "Y";
        reqtyp = "EB";
        act = "N";
        sup = "01";
        exp = "Y";
        cc = "BSTH";
        albr = "Y";
        sca = "Y";
        agauth = "Y";
        dated = "19991225";
        authName = "auth name";
        #porttyp = "L";
        actl = "ROOM1234567";
        ai = "Y";
        apot = "A1234567890";
        lst = "B0987654321";
        lso = "123456";
        tos = "1AM";
```

```
        spec = "SVCPR";
        nc = "LY--";
        nci = "02QC3.RV0";
        secnci = "02RV2.T";
        rpon = "RPON1234567890";
        rord = "rord0987654321";
        lspAuth = "lspa";
        lspAuthDate = "19991231";
        lspAuthName = "lsp_auth_name";
        cic = "1234";
        cust = "customer name";
        bil = "A";
        banl = "qweasd1234098";
        acna = "ABC";
        vta = "variable";
        init = "initiator";
        initTelNo = "7326994801";
        initFaxNo = "7326994802";
        impCon = "implementation";
        impConTelNo = "7326994803";
        impConPager = "7326994804";
        remarks = "thisis a remark that is not very long";
};

eu = {
        header = {
                administrative = {
                        ibt = "9";
                };      # Administrative Section
                locationAndAccess = {
                        aact = "A";
                        locnum = "000";
                        euName = "jack and jill";
                        sano = "1904";
                        sasf = "abcd";
                        sasd = "E";
                        sasn = "york street";
                        sath = "denver";
                        sass = "suff";
                        sadlo = "acorss the bagel shop";
                        euFloor = "2ndfloor";
                        euRoom = "205";
                        euBldg = "S York";
                        euCity = "Denver";
                        euState = "CO";
                        euZipCode = "80231";
                        lconName = "Denver Univ";
                        lconTelNo = "3033061995";
                        eumi = "Y";
                        acc = "access using front door";
                        wsop = "V";
                        erl = "";
                };      # Location and Access Header Section
                insideWire = {
                        iwOptions = "W";
                        iwconName = "Jack";
                        iwConTelNo = "3036891114";
                };      # Inside Wire Section
                bill = {
```

```
                        eatn = "3033061994";
                        fbi = "Y";
                        fbBillName = "Messrs jack and jill";
                        fbSbillName = "joe";
                        fbStreet = "1904 s york st";
                        fbFloor = "2ndfloor";
                        fbRoom = "205";
                        fbCity = "denver";
                        fbState = "CO";
                        fbZipCode = "80231";
                        fbBillcon = "ESI";
                        fbConTelNo = "3036891114";
                };      # Bill Section
        };      # Header Section
        detail = {
                locationAndAccessList = (
                        {
                                locnum = "123";
                                locact = "N";
                                euName = "jack and jill";
                                sano = "555";
                                sasf = "54321";
                                sasd = "E";
                                sasn = "South Gate";
                                sath = "25 South";
                                sass = "sass";
                                sadlo = "across the Rockies";
                                euFloor = "3rdfloor";
                                euRoom = "305";
                                euBldg = "south ga";
                                euCity = "Englewood";
                                euState = "CO";
                                euZipCode = "80231";
                                lconName = "ESI";
                                lconTelNo = "3036891114";
                                acc = "ask the receptionist";
                        },
                );      # Location And Access List
                disconnectInformationList = (
                        {
                                dnum = "qwert";
                                discNbr = "3036891114";
                        },
                );      # Disconnect Information List
        };      # Detail Section
};


dsdl = {
       deliveryAddress = {
              dact = "N";
              nameDel = "NAme";
              ddasn = "12";
              ddaloc = "012345";
              ddast = "12";
              ddazc = "12";
        };
};
```

```
resale = {
        ord = "01234567890123456789";
        rsqty = 123;
        serviceDetailList = (
                {
                        baList = (
                                {
                                        ba = A;
                                        block = A;
                                },
                        );
                        cfa = 012345678901234567890123456789012345678901;
                        ckr = 01234567890123456789012345678901234567890;
                        cnam = 012345678901234;
                        ecckt = "800.123.4567";
                        fpi = A;
                        ispid = 12345678901234;
                        insideWireList = (
                                {
                                        iwjk = 01234;
                                        iwjq = 12;
                                };
                        );
                        jkCode = 01234;
                        jkNum = 01;
                        jkPos = 12;
                        jr = Y;
                        lean = "";
                        leatn = "";
                        lna = N;
                        lneclssvc = 01234;
                        lnex = "";
                        lnum = 01234;
                        locnum = 123;
                        lpic = 0123;
                        matn = 1;
                        nidr = Y;
                        npi = C;
                        #otn = 1234567890;
                        pic = 0123;
                        pulse = 0123;
                        san = 012345678901234567890123456789;
                        sdi = A;
                        sgnl = 01;
                        ssig = 01;
                        tcFr = 1234567890;
                        tcOpt = ST;
                        tcPer = 19991212;
                        tcToPrimary = 1234557890;
                        tcToSecondary = 1234567810;
                        primary = {
                                tcid = 01;
                                tcName = 01234567890123456789012345678901234;
                        };
                        secondary = {
                                tcid = 02;
                                tcName = 01234567890123456789012345678901234;
                        };
                        ters = 0123456789;
```

```
                        tli = 1234567890;
                        tns = 0123456789-1234;
                        tsp = 012345678-01;
                    },
            );
};
```

**xstTestClient Output:**

```
status={
 msgId=00;
 msgTxt="LSR submitted to BellSouth";
};
messageHeader={
 dateSent=1999090712423732;
};
dueDate={
 dueDate=19991223;
 msgId=TAGT0000CDD;
 msgText="CALCULATED DUE DATE PROVIDED";
};
```

# 3.8 LOOP

## 3.8.1 LOOP_MAKEUP_WORKING

An example of a LOOP_MAKEUP_WORKING query is illustrated below:

**Input File:**

```
testProdIndicator=T;
```

RequestType = loop_makeup_working;

```
ClecId=CLECIDENTIFIER;
cc=BS12;
cktId="38.SBGS.404.477.3999.T22.123";
address={
                houseNumber=1001;
                houseNumberSuffix=A;
                streetName=Lincoln;
                streetDirectional=N;
                streetThoroughfare=AVE;
                streetSuffix=W;
                building=123;
                floor=2;
                room=212;
                city=FREEHOLD;
                state=NJ;
                zipCode=07728;
                unnumberedHouseIndicator=N;
        };
```

**Output File:**

```
status={
 msgId=00;
 msgTxt=SUCCESS;
};
messageHeader={
 inquiryNumber=2C6F4DC80000037D;
 dateSent=2000061317574011;
};
loopList=(
 {
  lpstat=spare12;
  ssc=A;
  rtf=4;
  fnList=(
   {
    ca=1able1289;
    pr=pair;
    abp=ab12;
    tea=alphanumerics1234567890;
    trmed=abcde1234;
    lmuList=(
     {
      lmstat=1ortyalphanumerics1234567890;
      luint=1a;
      nld=a1;
      coil=12ab;
      es=12345abcd;
      ldsp=1234567890abcde;
      boList=(
       {
        boCap=123ab;
        boRes=abcd1;
        boOff=12345abcd;
       };
      );
      splList=(
       {
        ga=12345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
      );
     };
    );
   };
   {
    ca=2able1289;
    pr=pair;
    abp=ab12;
    tea=alphanumerics1234567890;
    trmed=abcde1234;
    lmuList=(
     {
      lmstat=2ortyalphanumerics1234567890;
      luint=1a;
```

```
nld=a1;
coil=12ab;
es=12345abcd;
ldsp=1234567890abcde;
boList=(
 {
  boCap=223ab;
  boRes=abcd1;
  boOff=12345abcd;
 };
 {
  boCap=323ab;
  boRes=abcd1;
  boOff=12345abcd;
 };
);
splList=(
 {
  ga=22345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=32345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=42345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=52345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=62345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=72345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
```

```
      {
       ga=82345ab;
       lgth=12345abcd;
       uba=a;
       capac=123ab;
       btoff=09876abcd;
      };
      {
       ga=92345ab;
       lgth=12345abcd;
       uba=a;
       capac=123ab;
       btoff=09876abcd;
      };
      {
       ga=a2345ab;
       lgth=12345abcd;
       uba=a;
       capac=123ab;
       btoff=09876abcd;
      };
      {
       ga=b2345ab;
       lgth=12345abcd;
       uba=a;
       capac=123ab;
       btoff=09876abcd;
      };
     );
    };
   );
  };
  {
   ca=3able1289;
   pr=pair;
   abp=ab12;
   tea=alphanumerics1234567890;
   trmed=abcde1234;
   lmuList=(
    {
     lmstat=3ortyalphanumerics1234567890;
     luint=1a;
     nld=a1;
     coil=12ab;
     es=12345abcd;
     ldsp=1234567890abcde;
     boList=(
      {
       boCap=223ab;
       boRes=abcd1;
       boOff=12345abcd;
      };
      {
       boCap=323ab;
       boRes=abcd1;
       boOff=12345abcd;
      };
     );
     splList=(
```

```
{
 ga=22345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=32345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=42345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=52345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=62345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=72345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=82345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=92345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=a2345ab;
 lgth=12345abcd;
```

```
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=b2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
      );
     };
    );
   };
   {
    ca=4able1289;
    pr=pair;
    abp=ab12;
    tea=alphanumerics1234567890;
    trmed=abcde1234;
    lmuList=(
     {
      lmstat=4ortyalphanumerics1234567890;
      luint=1a;
      nld=a1;
      coil=12ab;
      es=12345abcd;
      ldsp=1234567890abcde;
      boList=(
       {
        boCap=223ab;
        boRes=abcd1;
        boOff=12345abcd;
       };
       {
        boCap=323ab;
        boRes=abcd1;
        boOff=12345abcd;
       };
      );
      splList=(
       {
        ga=22345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=32345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=42345ab;
        lgth=12345abcd;
```

```
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=52345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=62345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=72345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=82345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=92345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=a2345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=b2345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
);
};
);
};
{
 ca=5able1289;
```

```
pr=pair;
abp=ab12;
tea=alphanumerics1234567890;
trmed=abcde1234;
lmuList=(
 {
  lmstat=5ortyalphanumerics1234567890;
  luint=1a;
  nld=a1;
  coil=12ab;
  es=12345abcd;
  ldsp=1234567890abcde;
  boList=(
   {
    boCap=223ab;
    boRes=abcd1;
    boOff=12345abcd;
   };
   {
    boCap=323ab;
    boRes=abcd1;
    boOff=12345abcd;
   };
  );
  splList=(
   {
    ga=22345ab;
    lgth=12345abcd;
    uba=a;
    capac=123ab;
    btoff=09876abcd;
   };
   {
    ga=32345ab;
    lgth=12345abcd;
    uba=a;
    capac=123ab;
    btoff=09876abcd;
   };
   {
    ga=42345ab;
    lgth=12345abcd;
    uba=a;
    capac=123ab;
    btoff=09876abcd;
   };
   {
    ga=52345ab;
    lgth=12345abcd;
    uba=a;
    capac=123ab;
    btoff=09876abcd;
   };
   {
    ga=62345ab;
    lgth=12345abcd;
    uba=a;
    capac=123ab;
    btoff=09876abcd;
```

```
        };
        {
         ga=72345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
        {
         ga=82345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
        {
         ga=92345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
        {
         ga=a2345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
        {
         ga=b2345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
       );
      };
     );
    };
    {
     ca=6able1289;
     pr=pair;
     abp=ab12;
     tea=alphanumerics1234567890;
     trmed=abcde1234;
     lmuList=(
      {
       lmstat=6ortyalphanumerics1234567890;
       luint=1a;
       nld=a1;
       coil=12ab;
       es=12345abcd;
       ldsp=1234567890abcde;
       boList=(
        {
         boCap=223ab;
         boRes=abcd1;
         boOff=12345abcd;
```

```
 };
 {
  boCap=323ab;
  boRes=abcd1;
  boOff=12345abcd;
 };
);
splList=(
 {
  ga=22345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=32345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=42345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=52345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=62345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=72345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=82345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=92345ab;
```

```
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=a2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=b2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
      );
     };
    );
   };
   {
    ca=7able1289;
    pr=pair;
    abp=ab12;
    tea=alphanumerics1234567890;
    trmed=abcde1234;
    lmuList=(
     {
      lmstat=7ortyalphanumerics1234567890;
      luint=1a;
      nld=a1;
      coil=12ab;
      es=12345abcd;
      ldsp=1234567890abcde;
      boList=(
       {
        boCap=223ab;
        boRes=abcd1;
        boOff=12345abcd;
       };
       {
        boCap=323ab;
        boRes=abcd1;
        boOff=12345abcd;
       };
      );
      splList=(
       {
        ga=22345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=32345ab;
```

```
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=42345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=52345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=62345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=72345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=82345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=92345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=a2345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=b2345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
```

```
        btoff=09876abcd;
       };
      );
     };
    );
   };
   {
    ca=8able1289;
    pr=pair;
    abp=ab12;
    tea=alphanumerics1234567890;
    trmed=abcde1234;
    lmuList=(
     {
      lmstat=8ortyalphanumerics1234567890;
      luint=1a;
      nld=a1;
      coil=12ab;
      es=12345abcd;
      ldsp=1234567890abcde;
      boList=(
       {
        boCap=223ab;
        boRes=abcd1;
        boOff=12345abcd;
       };
       {
        boCap=323ab;
        boRes=abcd1;
        boOff=12345abcd;
       };
      );
      splList=(
       {
        ga=22345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=32345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=42345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=52345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
```

```
 btoff=09876abcd;
};
{
 ga=62345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=72345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=82345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=92345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=a2345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=b2345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
);
};
);
};
{
ca=9able1289;
pr=pair;
abp=ab12;
tea=alphanumerics1234567890;
trmed=abcde1234;
lmuList=(
 {
  lmstat=9ortyalphanumerics1234567890;
  luint=1a;
  nld=a1;
```

```
coil=12ab;
es=12345abcd;
ldsp=1234567890abcde;
boList=(
 {
  boCap=223ab;
  boRes=abcd1;
  boOff=12345abcd;
 };
 {
  boCap=323ab;
  boRes=abcd1;
  boOff=12345abcd;
 };
);
splList=(
 {
  ga=22345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=32345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=42345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=52345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=62345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=72345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
```

```
        ga=82345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=92345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=a2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=b2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
      );
     };
    );
   };
  );
 };
);
```

;

### 3.8.2 LOOP MAKEUP SPARE

An example of a LOOP MAKEUP SPARE query is illustrated below:

**Input File:**

```
testProdIndicator=T;
RequestType = loop_makeup_spare;
ClecId=CLECIDENTIFIER;
cc=12GH;
numberRequested=12;
nc=LXRA;
nci="02QB9.00A";
secnci="02DU9.00A";
address={
       houseNumber=1234567890123;
       houseNumberSuffix=ZXCVB;
       streetName="&XYZ";
       streetDirectional=SE;
       streetThoroughfare=STREET1234;
```

```
        streetSuffix=NOTH;
        building=Abuilding123456;
        floor=12345678901234;
        room=presidentS14123;
        city=ABCDE12345678901234567890123456 7;
        state=AB;
        zipCode=08854;
        unnumberedHouseIndicator=Y;
};
```

## Output File:

```
status={
 msgId=00;
 msgTxt=SUCCESS;
};
messageHeader={
 inquiryNumber=D909C8720000041C;
 dateSent=2000061317585104;
};
loopList=(
 {
  lpstat=spare12;
  ssc=A;
  rtf=4;
  fnList=(
   {
    ca=lable1289;
    pr=pair;
    abp=ab12;
    tea=alphanumerics1234567890;
    trmed=abcde1234;
    lmuList=(
     {
      lmstat=lortyalphanumerics1234567890;
      luint=1a;
      nld=a1;
      coil=12ab;
      es=12345abcd;
      ldsp=1234567890abcde;
      boList=(
       {
        boCap=123ab;
        boRes=abcd1;
        boOff=12345abcd;
       };
      );
      splList=(
       {
        ga=12345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
      );
     };
```

```
 );
};
{
 ca=2able1289;
 pr=pair;
 abp=ab12;
 tea=alphanumerics1234567890;
 trmed=abcde1234;
 lmuList=(
  {
   lmstat=2ortyalphanumerics1234567890;
   luint=1a;
   nld=a1;
   coil=12ab;
   es=12345abcd;
   ldsp=1234567890abcde;
   boList=(
    {
     boCap=223ab;
     boRes=abcd1;
     boOff=12345abcd;
    };
    {
     boCap=323ab;
     boRes=abcd1;
     boOff=12345abcd;
    };
   );
   splList=(
    {
     ga=22345ab;
     lgth=12345abcd;
     uba=a;
     capac=123ab;
     btoff=09876abcd;
    };
    {
     ga=32345ab;
     lgth=12345abcd;
     uba=a;
     capac=123ab;
     btoff=09876abcd;
    };
    {
     ga=42345ab;
     lgth=12345abcd;
     uba=a;
     capac=123ab;
     btoff=09876abcd;
    };
    {
     ga=52345ab;
     lgth=12345abcd;
     uba=a;
     capac=123ab;
     btoff=09876abcd;
    };
    {
     ga=62345ab;
```

```
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=72345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=82345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=92345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=a2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=b2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
      );
     };
    );
   };
   {
    ca=3able1289;
    pr=pair;
    abp=ab12;
    tea=alphanumerics1234567890;
    trmed=abcde1234;
    lmuList=(
     {
      lmstat=3ortyalphanumerics1234567890;
      luint=1a;
      nld=a1;
      coil=12ab;
      es=12345abcd;
      ldsp=1234567890abcde;
      boList=(
```

```
  {
   boCap=223ab;
   boRes=abcd1;
   boOff=12345abcd;
  };
  {
   boCap=323ab;
   boRes=abcd1;
   boOff=12345abcd;
  };
 );
 splList=(
  {
   ga=22345ab;
   lgth=12345abcd;
   uba=a;
   capac=123ab;
   btoff=09876abcd;
  };
  {
   ga=32345ab;
   lgth=12345abcd;
   uba=a;
   capac=123ab;
   btoff=09876abcd;
  };
  {
   ga=42345ab;
   lgth=12345abcd;
   uba=a;
   capac=123ab;
   btoff=09876abcd;
  };
  {
   ga=52345ab;
   lgth=12345abcd;
   uba=a;
   capac=123ab;
   btoff=09876abcd;
  };
  {
   ga=62345ab;
   lgth=12345abcd;
   uba=a;
   capac=123ab;
   btoff=09876abcd;
  };
  {
   ga=72345ab;
   lgth=12345abcd;
   uba=a;
   capac=123ab;
   btoff=09876abcd;
  };
  {
   ga=82345ab;
   lgth=12345abcd;
   uba=a;
   capac=123ab;
```

```
        btoff=09876abcd;
       };
       {
        ga=92345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=a2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=b2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
      );
     };
    );
   };
   {
    ca=4able1289;
    pr=pair;
    abp=ab12;
    tea=alphanumerics1234567890;
    trmed=abcde1234;
    lmuList=(
     {
      lmstat=4ortyalphanumerics1234567890;
      luint=1a;
      nld=a1;
      coil=12ab;
      es=12345abcd;
      ldsp=1234567890abcde;
      boList=(
       {
        boCap=223ab;
        boRes=abcd1;
        boOff=12345abcd;
       };
       {
        boCap=323ab;
        boRes=abcd1;
        boOff=12345abcd;
       };
      );
      splList=(
       {
        ga=22345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
```

```
 btoff=09876abcd;
};
{
 ga=32345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=42345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=52345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=62345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=72345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=82345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=92345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=a2345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
```

```
        ga=b2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
      );
     };
    );
   };
   {
    ca=5able1289;
    pr=pair;
    abp=ab12;
    tea=alphanumerics1234567890;
    trmed=abcde1234;
    lmuList=(
     {
      lmstat=5ortyalphanumerics1234567890;
      luint=1a;
      nld=a1;
      coil=12ab;
      es=12345abcd;
      ldsp=1234567890abcde;
      boList=(
       {
        boCap=223ab;
        boRes=abcd1;
        boOff=12345abcd;
       };
       {
        boCap=323ab;
        boRes=abcd1;
        boOff=12345abcd;
       };
      );
      splList=(
       {
        ga=22345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=32345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=42345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
```

```
        ga=52345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=62345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=72345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=82345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=92345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=a2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=b2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
      );
     };
    );
   };
   {
    ca=6able1289;
    pr=pair;
    abp=ab12;
    tea=alphanumerics1234567890;
    trmed=abcde1234;
    lmuList=(
```

```
{
 lmstat=6ortyalphanumerics1234567890;
 luint=1a;
 nld=a1;
 coil=12ab;
 es=12345abcd;
 ldsp=1234567890abcde;
 boList=(
  {
   boCap=223ab;
   boRes=abcd1;
   boOff=12345abcd;
  };
  {
   boCap=323ab;
   boRes=abcd1;
   boOff=12345abcd;
  };
 );
 splList=(
  {
   ga=22345ab;
   lgth=12345abcd;
   uba=a;
   capac=123ab;
   btoff=09876abcd;
  };
  {
   ga=32345ab;
   lgth=12345abcd;
   uba=a;
   capac=123ab;
   btoff=09876abcd;
  };
  {
   ga=42345ab;
   lgth=12345abcd;
   uba=a;
   capac=123ab;
   btoff=09876abcd;
  };
  {
   ga=52345ab;
   lgth=12345abcd;
   uba=a;
   capac=123ab;
   btoff=09876abcd;
  };
  {
   ga=62345ab;
   lgth=12345abcd;
   uba=a;
   capac=123ab;
   btoff=09876abcd;
  };
  {
   ga=72345ab;
   lgth=12345abcd;
   uba=a;
```

```
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=82345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=92345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=a2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=b2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
      );
     };
    );
   };
   {
    ca=7able1289;
    pr=pair;
    abp=ab12;
    tea=alphanumerics1234567890;
    trmed=abcde1234;
    lmuList=(
     {
      lmstat=7ortyalphanumerics1234567890;
      luint=1a;
      nld=a1;
      coil=12ab;
      es=12345abcd;
      ldsp=1234567890abcde;
      boList=(
       {
        boCap=223ab;
        boRes=abcd1;
        boOff=12345abcd;
       };
       {
        boCap=323ab;
        boRes=abcd1;
        boOff=12345abcd;
```

```
 };
);
splList=(
 {
  ga=22345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=32345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=42345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=52345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=62345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=72345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=82345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=92345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
```

```
        {
         ga=a2345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
        {
         ga=b2345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
       );
      };
     );
    };
    {
     ca=8able1289;
     pr=pair;
     abp=ab12;
     tea=alphanumerics1234567890;
     trmed=abcde1234;
     lmuList=(
      {
       lmstat=8ortyalphanumerics1234567890;
       luint=1a;
       nld=a1;
       coil=12ab;
       es=12345abcd;
       ldsp=1234567890abcde;
       boList=(
        {
         boCap=223ab;
         boRes=abcd1;
         boOff=12345abcd;
        };
        {
         boCap=323ab;
         boRes=abcd1;
         boOff=12345abcd;
        };
       );
       splList=(
        {
         ga=22345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
        {
         ga=32345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
```

```
{
 ga=42345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=52345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=62345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=72345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=82345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=92345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=a2345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=b2345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
);
};
);
```

```
};
{
 ca=9able1289;
 pr=pair;
 abp=ab12;
 tea=alphanumerics1234567890;
 trmed=abcde1234;
 lmuList=(
  {
   lmstat=9ortyalphanumerics1234567890;
   luint=1a;
   nld=a1;
   coil=12ab;
   es=12345abcd;
   ldsp=1234567890abcde;
   boList=(
    {
     boCap=223ab;
     boRes=abcd1;
     boOff=12345abcd;
    };
    {
     boCap=323ab;
     boRes=abcd1;
     boOff=12345abcd;
    };
   );
   splList=(
    {
     ga=22345ab;
     lgth=12345abcd;
     uba=a;
     capac=123ab;
     btoff=09876abcd;
    };
    {
     ga=32345ab;
     lgth=12345abcd;
     uba=a;
     capac=123ab;
     btoff=09876abcd;
    };
    {
     ga=42345ab;
     lgth=12345abcd;
     uba=a;
     capac=123ab;
     btoff=09876abcd;
    };
    {
     ga=52345ab;
     lgth=12345abcd;
     uba=a;
     capac=123ab;
     btoff=09876abcd;
    };
    {
     ga=62345ab;
     lgth=12345abcd;
```

```
                uba=a;
                capac=123ab;
                btoff=09876abcd;
               };
               {
                ga=72345ab;
                lgth=12345abcd;
                uba=a;
                capac=123ab;
                btoff=09876abcd;
               };
               {
                ga=82345ab;
                lgth=12345abcd;
                uba=a;
                capac=123ab;
                btoff=09876abcd;
               };
               {
                ga=92345ab;
                lgth=12345abcd;
                uba=a;
                capac=123ab;
                btoff=09876abcd;
               };
               {
                ga=a2345ab;
                lgth=12345abcd;
                uba=a;
                capac=123ab;
                btoff=09876abcd;
               };
               {
                ga=b2345ab;
                lgth=12345abcd;
                uba=a;
                capac=123ab;
                btoff=09876abcd;
               };
              );
             };
            );
           };
  );
 };
);
```

### 3.8.3 LOOP_RESERVATION_SPARE

An example of a LOOP_RESERVATION_SPARE query is illustrated below:

**Input File:**

```
RequestType=LOOP_RESERVATION_SPARE;
testProdIndicator=T;
ClecId=CLECIDENTIFIER;
cc=TQ11;
```

```
numberRequested=12;
nc="LX--";
nci="02QC3.OOF";
secnci=02NO2;
address={
       houseNumber=11111222223ab;
       houseNumberSuffix=ABC12;
       streetName=123456789012345678901111aaabbccc22;
       streetDirectional=NE;
       streetThoroughfare=thoroughfa;
       streetSuffix=ABC1;
       building=ABCD12345612345;
       floor=1212121212ABCD;
       room=14141141414ABCD;
       city=CDE12345ABCDE12345ABCDE12345ABCD;
       state=AB;
       zipCode=ABC12;
       unnumberedHouseIndicator=A;
};
```

**Output File:**

```
status={
 msgId=00;
 msgTxt=SUCCESS;
};
messageHeader={
 inquiryNumber=43C96D5200000459;
 dateSent=2000061317592153;
};
numberReserved=20;
resid=LMUworking;
loopList=(
 {
  lpstat=spare12;
  ssc=A;
  rtf=4;
  fnList=(
   {
    ca=1able1289;
    pr=pair;
    abp=ab12;
    tea=alphanumerics1234567890;
    trmed=abcde1234;
    lmuList=(
     {
      lmstat=1ortyalphanumerics1234567890;
      luint=1a;
      nld=a1;
      coil=12ab;
      es=12345abcd;
      ldsp=1234567890abcde;
      boList=(
       {
        boCap=123ab;
        boRes=abcd1;
        boOff=12345abcd;
       };
      );
      splList=(
```

```
      {
       ga=12345ab;
       lgth=12345abcd;
       uba=a;
       capac=123ab;
       btoff=09876abcd;
      };
     );
    };
   );
  };
  {
   ca=2able1289;
   pr=pair;
   abp=ab12;
   tea=alphanumerics1234567890;
   trmed=abcde1234;
   lmuList=(
    {
     lmstat=2ortyalphanumerics1234567890;
     luint=1a;
     nld=a1;
     coil=12ab;
     es=12345abcd;
     ldsp=1234567890abcde;
     boList=(
      {
       boCap=223ab;
       boRes=abcd1;
       boOff=12345abcd;
      };
      {
       boCap=323ab;
       boRes=abcd1;
       boOff=12345abcd;
      };
     );
     splList=(
      {
       ga=22345ab;
       lgth=12345abcd;
       uba=a;
       capac=123ab;
       btoff=09876abcd;
      };
      {
       ga=32345ab;
       lgth=12345abcd;
       uba=a;
       capac=123ab;
       btoff=09876abcd;
      };
      {
       ga=42345ab;
       lgth=12345abcd;
       uba=a;
       capac=123ab;
       btoff=09876abcd;
      };
```

```
        {
         ga=52345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
        {
         ga=62345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
        {
         ga=72345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
        {
         ga=82345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
        {
         ga=92345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
        {
         ga=a2345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
        {
         ga=b2345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
       );
      };
     );
    };
    {
     ca=3able1289;
     pr=pair;
     abp=ab12;
     tea=alphanumerics1234567890;
     trmed=abcde1234;
```

```
lmuList=(
 {
  lmstat=3ortyalphanumerics1234567890;
  luint=1a;
  nld=a1;
  coil=12ab;
  es=12345abcd;
  ldsp=1234567890abcde;
  boList=(
   {
    boCap=223ab;
    boRes=abcd1;
    boOff=12345abcd;
   };
   {
    boCap=323ab;
    boRes=abcd1;
    boOff=12345abcd;
   };
  );
  splList=(
   {
    ga=22345ab;
    lgth=12345abcd;
    uba=a;
    capac=123ab;
    btoff=09876abcd;
   };
   {
    ga=32345ab;
    lgth=12345abcd;
    uba=a;
    capac=123ab;
    btoff=09876abcd;
   };
   {
    ga=42345ab;
    lgth=12345abcd;
    uba=a;
    capac=123ab;
    btoff=09876abcd;
   };
   {
    ga=52345ab;
    lgth=12345abcd;
    uba=a;
    capac=123ab;
    btoff=09876abcd;
   };
   {
    ga=62345ab;
    lgth=12345abcd;
    uba=a;
    capac=123ab;
    btoff=09876abcd;
   };
   {
    ga=72345ab;
    lgth=12345abcd;
```

```
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=82345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=92345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=a2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=b2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
      );
     };
    );
   };
   {
    ca=4able1289;
    pr=pair;
    abp=ab12;
    tea=alphanumerics1234567890;
    trmed=abcde1234;
    lmuList=(
     {
      lmstat=4ortyalphanumerics1234567890;
      luint=1a;
      nld=a1;
      coil=12ab;
      es=12345abcd;
      ldsp=1234567890abcde;
      boList=(
       {
        boCap=223ab;
        boRes=abcd1;
        boOff=12345abcd;
       };
       {
        boCap=323ab;
        boRes=abcd1;
```

```
  boOff=12345abcd;
 };
);
splList=(
 {
  ga=22345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=32345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=42345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=52345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=62345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=72345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=82345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=92345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
```

```
        };
        {
         ga=a2345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
        {
         ga=b2345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
       );
      };
     );
    };
    {
     ca=5able1289;
     pr=pair;
     abp=ab12;
     tea=alphanumerics1234567890;
     trmed=abcde1234;
     lmuList=(
      {
       lmstat=5ortyalphanumerics1234567890;
       luint=1a;
       nld=a1;
       coil=12ab;
       es=12345abcd;
       ldsp=1234567890abcde;
       boList=(
        {
         boCap=223ab;
         boRes=abcd1;
         boOff=12345abcd;
        };
        {
         boCap=323ab;
         boRes=abcd1;
         boOff=12345abcd;
        };
       );
       splList=(
        {
         ga=22345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
        };
        {
         ga=32345ab;
         lgth=12345abcd;
         uba=a;
         capac=123ab;
         btoff=09876abcd;
```

```
};
{
 ga=42345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=52345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=62345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=72345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=82345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=92345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=a2345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=b2345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
);
};
```

```
 );
};
{
 ca=6able1289;
 pr=pair;
 abp=ab12;
 tea=alphanumerics1234567890;
 trmed=abcde1234;
 lmuList=(
  {
   lmstat=6ortyalphanumerics1234567890;
   luint=1a;
   nld=a1;
   coil=12ab;
   es=12345abcd;
   ldsp=1234567890abcde;
   boList=(
    {
     boCap=223ab;
     boRes=abcd1;
     boOff=12345abcd;
    };
    {
     boCap=323ab;
     boRes=abcd1;
     boOff=12345abcd;
    };
   );
   splList=(
    {
     ga=22345ab;
     lgth=12345abcd;
     uba=a;
     capac=123ab;
     btoff=09876abcd;
    };
    {
     ga=32345ab;
     lgth=12345abcd;
     uba=a;
     capac=123ab;
     btoff=09876abcd;
    };
    {
     ga=42345ab;
     lgth=12345abcd;
     uba=a;
     capac=123ab;
     btoff=09876abcd;
    };
    {
     ga=52345ab;
     lgth=12345abcd;
     uba=a;
     capac=123ab;
     btoff=09876abcd;
    };
    {
     ga=62345ab;
```

```
      lgth=12345abcd;
      uba=a;
      capac=123ab;
      btoff=09876abcd;
     };
     {
      ga=72345ab;
      lgth=12345abcd;
      uba=a;
      capac=123ab;
      btoff=09876abcd;
     };
     {
      ga=82345ab;
      lgth=12345abcd;
      uba=a;
      capac=123ab;
      btoff=09876abcd;
     };
     {
      ga=92345ab;
      lgth=12345abcd;
      uba=a;
      capac=123ab;
      btoff=09876abcd;
     };
     {
      ga=a2345ab;
      lgth=12345abcd;
      uba=a;
      capac=123ab;
      btoff=09876abcd;
     };
     {
      ga=b2345ab;
      lgth=12345abcd;
      uba=a;
      capac=123ab;
      btoff=09876abcd;
     };
    );
   };
  );
 };
 {
  ca=7able1289;
  pr=pair;
  abp=ab12;
  tea=alphanumerics1234567890;
  trmed=abcde1234;
  lmuList=(
   {
    lmstat=7ortyalphanumerics1234567890;
    luint=1a;
    nld=a1;
    coil=12ab;
    es=12345abcd;
    ldsp=1234567890abcde;
    boList=(
```

```
 {
  boCap=223ab;
  boRes=abcd1;
  boOff=12345abcd;
 };
 {
  boCap=323ab;
  boRes=abcd1;
  boOff=12345abcd;
 };
);
splList=(
 {
  ga=22345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=32345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=42345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=52345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=62345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=72345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
  btoff=09876abcd;
 };
 {
  ga=82345ab;
  lgth=12345abcd;
  uba=a;
  capac=123ab;
```

```
        btoff=09876abcd;
       };
       {
        ga=92345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=a2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
       {
        ga=b2345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
        btoff=09876abcd;
       };
      );
     };
    );
   };
   {
    ca=8able1289;
    pr=pair;
    abp=ab12;
    tea=alphanumerics1234567890;
    trmed=abcde1234;
    lmuList=(
     {
      lmstat=8ortyalphanumerics1234567890;
      luint=1a;
      nld=a1;
      coil=12ab;
      es=12345abcd;
      ldsp=1234567890abcde;
      boList=(
       {
        boCap=223ab;
        boRes=abcd1;
        boOff=12345abcd;
       };
       {
        boCap=323ab;
        boRes=abcd1;
        boOff=12345abcd;
       };
      );
      splList=(
       {
        ga=22345ab;
        lgth=12345abcd;
        uba=a;
        capac=123ab;
```

```
 btoff=09876abcd;
};
{
 ga=32345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=42345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=52345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=62345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=72345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=82345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=92345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
 ga=a2345ab;
 lgth=12345abcd;
 uba=a;
 capac=123ab;
 btoff=09876abcd;
};
{
```

```
      ga=b2345ab;
      lgth=12345abcd;
      uba=a;
      capac=123ab;
      btoff=09876abcd;
     };
    );
   };
  );
 };
{
 ca=9able1289;
 pr=pair;
 abp=ab12;
 tea=alphanumerics1234567890;
 trmed=abcde1234;
 lmuList=(
  {
   lmstat=9ortyalphanumerics1234567890;
   luint=1a;
   nld=a1;
   coil=12ab;
   es=12345abcd;
   ldsp=1234567890abcde;
   boList=(
    {
     boCap=223ab;
     boRes=abcd1;
     boOff=12345abcd;
    };
    {
     boCap=323ab;
     boRes=abcd1;
     boOff=12345abcd;
    };
   );
   splList=(
    {
     ga=22345ab;
     lgth=12345abcd;
     uba=a;
     capac=123ab;
     btoff=09876abcd;
    };
    {
     ga=32345ab;
     lgth=12345abcd;
     uba=a;
     capac=123ab;
     btoff=09876abcd;
    };
    {
     ga=42345ab;
     lgth=12345abcd;
     uba=a;
     capac=123ab;
     btoff=09876abcd;
    };
    {
```

```
                ga=52345ab;
                lgth=12345abcd;
                uba=a;
                capac=123ab;
                btoff=09876abcd;
               };
               {
                ga=62345ab;
                lgth=12345abcd;
                uba=a;
                capac=123ab;
                btoff=09876abcd;
               };
               {
                ga=72345ab;
                lgth=12345abcd;
                uba=a;
                capac=123ab;
                btoff=09876abcd;
               };
               {
                ga=82345ab;
                lgth=12345abcd;
                uba=a;
                capac=123ab;
                btoff=09876abcd;
               };
               {
                ga=92345ab;
                lgth=12345abcd;
                uba=a;
                capac=123ab;
                btoff=09876abcd;
               };
               {
                ga=a2345ab;
                lgth=12345abcd;
                uba=a;
                capac=123ab;
                btoff=09876abcd;
               };
               {
                ga=b2345ab;
                lgth=12345abcd;
                uba=a;
                capac=123ab;
                btoff=09876abcd;
               };
              );
             };
            );
           };
          );
         };
        );
```

### 3.8.4 LOOP RESERVATION CANCEL

An example of a LOOP RESERVATION CANCEL query is illustrated below:

**Input File:**

```
RequestType=LOOP_RESERVATION_CANCEL;
testProdIndicator=T;
ClecId=CLECIDENTIFIER;
cc=ABC1;
resId=ABC12;
address={
        houseNumber=a1222223ab;
        houseNumberSuffix=12;
        streetName=111aaabb;
        streetDirectional=SW;
        streetThoroughfare=RD12;
        streetSuffix=C1;
        building=12ABC15;
        floor=121ABCD1;
        room=14ABCD1;
        city=A2345;
        state=B1;
        zipCode=1BC12;
        unnumberedHouseIndicator=A;
};
```

**Output File:**

```
status={
msgId=00;
msgTxt=SUCCESS;
};
messageHeader={
inquiryNumber=D909C8720000041D;
dateSent=2000061317585109;
};
```