Cisco BTS 10200 Softswitch SOAP Adapter Interface Specification Programmer's Guide, Release 6.0.1

August 21, 2008

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Text Part Number: OL-15336-02

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

CCDE, CCENT, Cisco Eos, Cisco Lumin, Cisco Nexus, Cisco StadiumVision, Cisco TelePresence, the Cisco logo, DCE, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn and Cisco Store are service marks; and Access Registrar, Aironet, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, EtherFast, EtherSwitch, Event Center, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, iQuick Study, IronPort, the IronPort logo, LightStream, Linksys, MediaTone, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, Network Registrar, PCNow, PIX, PowerPanels, ProConnect, ScriptShare, SenderBase, SMARTnet, Spectrum Expert, StackWise, The Fastest Way to Increase Your Internet Quotient, TransPath, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0807R)

Cisco BTS 10200 Softswitch SOAP Adapter Interface Specification Programmer's Guide, Release 6.0.1
Copyright © 2008 Cisco Systems, Inc. All rights reserved.

CONTENTS

Preface

Revised: August 21, 2008, OL-15336-02

This document is the *Cisco BTS 10200 Softswitch SOAP Adapter Interface Specification Programmer's Guide, Release 6.0.1* for the Cisco BTS 10200 Softswitch, Release 6.0.1.

Audience

This document is designed for intermediate and advanced SOAP programmers. It presumes an intermediate or higher level of SOAP programming familiarity.

Organization

This document is divided into the following chapters and appendixes:

- Chapter 1, "SOAP Architecture and Application Programming Interface"—This chapter describes the Simple Object Access Protocol (SOAP) adapter architecture and application programming interface (API) for the Cisco BTS 10200 Softswitch.
- Chapter 2, "Extensible Markup Language Processing"—This chapter describes the Extensible Markup Language (XML) process in the Simple Object Access Protocol (SOAP) interface.
- Chapter 2, "Troubleshooting"—This chapter describes the resolution of the events and alarms associated with the Simple Object Access Protocol (SOAP) interface.
- Appendix A, "XML Description Documents"—This appendix provides examples of the XML description documents.
- Appendix B, "XML Test Drivers"—This appendix details the XML test drivers.

Conventions

This document uses the following conventions:

Note Means *reader take note.* Notes contain helpful suggestions or references to material not covered in the manual.

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Subscribe to the *What's New in Cisco Product Documentation* as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS version 2.0.

Document Change History

Table 1 provides the revision history for the *Cisco BTS 10200 Softswitch SOAP Adapter Interface Specification Programmer's Guide, Release 6.0.x*.

Table 1 Revision History

Version Number	Issue Date	Status	Reason for Change
OL-15336-01	31 Mar 2008	First issue	—
OL-15336-02	21 Aug 2008	Revised	Added Troubleshooting chapter

CHAPTER 1

SOAP Architecture and Application Programming Interface

Revised: August 21, 2008, OL-15336-02

The *Cisco BTS 10200 Softswitch Release 6.0 SOAP Adapter Interface Specification Programmer Guide, Release 6.0.1* describes the Simple Object Access Protocol (SOAP) adapter. The SOAP adapter provides a machine-to-machine interface (MMI) over SOAP.

The goal of the SOAP interface is to provide a provisioning method for the Cisco BTS 10200 Softswitch product that parallels the Command Line Interface (CLI) adapter in capabilities. SOAP provides an abstraction of the BTS 10200 in a consistent, object-oriented model. Discussion of the actual object model for this interface is not within the scope of this document.

The Cisco BTS 10200 Softswitch Command Line Interface Database is the definitive source for token descriptions (parameters) and their values, as used in the SOAP interface.

Overview

This chapter describes the Simple Object Access Protocol (SOAP) adapter architecture and application programming interface (API) for the BTS 10200. This chapter includes the following sections:

SOAP Adapter Architecture

The SOAP adapter interface leverages the adapter architecture of the Element Management System (EMS) component in the BTS 10200. This architecture allows for a variety of adapters to provide operations, administration, management, and provisioning (OAM&P) by adapting the external interface to a common infrastructure in the EMS.

Application Interface

The SOAP adapter uses the Tomcat AXIS version 1.4 package to develop and deploy the SOAP application. AXIS 1.4 is a third-party freeware provided as part of the BTS 10200 Tomcat package.

XML Interface

The XML interface is abstracted by SOAP itself. The Tomcat package uses either the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP) for connections. Narrowing on the NameService will also produce the BTS10200 objects. Narrowing is covered in great detail in the coding examples in the BTS 10200 SDK package. This is bundled with the BTS 10200 application.

To create separate application-level connections or objects requires some object pooling and numerous logins to obtain valid login keys for every instance of the BTS10200 object. Examples are available in the bundled SDK package.

Batch Data Retrieval (Paging)

Batch data retrieval (paging) is available using the CLI **show** command. Paging is required to view large data sets, and the initial setup of paging impacts BTS 10200 performance. Paged data is cached for quick retrieval upon subsequent requests. Paged data must be contiguous for optimal system performance. For more information, see the *Cisco BTS 10200 Softswitch Operations and Maintenance Guide,* the *Cisco BTS 10200 Softswitch Troubleshooting Guide,* and the Cisco BTS 10200 Softswitch Command Line Interface Database.

Connections and Transactions

SOAP supports up to 100 simultaneous sessions. A session is any valid login to the SOAP interface. A transaction is any specific request; for example, **show** or **change**.

Each login session allocates its own set of resources, much like an individual CLI session. If a command is not executed within 30 minutes, the session is identified as idle. An idle session is removed from the interface, and all resources are closed.

Note Idle time is configured through the bts.properties file of the Tomcat application jar file. The bts.properties file is located on the BTS 10200 in the /opt/BTSsoap/etc directory.

Note Modification of the bts.properties file should not be attempted without Cisco TAC support or supervision.

SOAP Interface Specifications

The SOAP interface adapter conforms to SOAP 1.2 specifications.

Compiler Tools

The SOAP adapter utilizes the J2SE Development Kit (JDK)–1.6.0. The BTS 10200 uses JDK 1.6.0 for compilation and the JDK 1.5.0 or JDK 6 for the Java Runtime Environment (JRE).

Client-side applications might require the following tools:

- Xerces parsers
- ECS Report Builder

WSDL Stub Generation

The SOAP adapter uses the Apache AXIS toolkit to generate skeletons and stubs. The skeletons and stubs abstract the SOAP interface from the BTS 10200 middleware.

The following example of SOAP script code uses the Java package tree as developed in the BTS 10200 product. Such code can vary. Other clients can specify a different package tree to contain the Web Services Description Language (WSDL) interface objects. See the SDK for a detailed breakdown of this script.

```
#!/bin/sh
##############################################################################
# Copyright (c) 2002, 2006 by Cisco Systems, Inc.
#
#
##############################################################################
set -e
set -a
#set -x

#
# List required jar files
#
AXISLIB=/opt/Tomcat/webapps/axis/WEB-INF/lib
CLASSPATH=$AXISLIB/axis.jar:$AXISLIB/commons-logging-1.0.4.jar:$AXISLIB/jaxrpc:$AXISLIB/co
mmons-discovery-0.2.jar:$AXISLIB/saaj:$AXISLIB/wsdl4j-1.5.1.jar

export CLASSPATH

java -classpath $CLASSPATH org.apache.axis.wsdl.WSDL2Java -o . -p
com.sswitch.oam.soap.client  bts10200.wsdl
```

Java files are generated in a local directory tree specified in the package directory. This package path is required in the bind logic to find the object interface implementation.

```
#!/bin/sh
##############################################################################
# Copyright (c) 2002, 2006 by Cisco Systems, Inc.
#
#
##############################################################################
set -e
set -a
#set -x
#
# List required jar files
#
AXISLIB=/opt/Tomcat/webapps/axis/WEB-INF/lib
```

```
CLASSPATH=$AXISLIB/axis.jar:$AXISLIB/commons-logging-1.0.4.jar:$AXISLIB/jaxrpc:$AXISLIB/co
mmons-discovery-0.2.jar:$AXISLIB/saaj:$AXISLIB/wsdl4j-1.5.1.jar

export CLASSPATH

javac -classpath $CLASSPATH -d ./ com/sswitch/oam/soap/client/*.java
```

The preceding example suggests compiling a package of Java files to generate the required class files. These class files must exist in the client classpath.

Note The preceding example shows a common method for building all java files in a single directory with a single command. This is one of the fastest ways to compile bulk java code.

The BTS 10200 offers a Software Developer's Kit (SDK) with a complete range of examples that utilize the SOAP interface. These include many topics such as

- CLI
- Proxy example
- Batch file processing

The following example illustrates the basic extraction of the BTS 10200 objects:

```
package com.sswitch.oam.ccc;

import com.sswitch.oam.soap.client.*;
import java.net.*;

/**
 * Copyright (c) 2002-2004, 2006 by Cisco Systems, Inc.
 */
public class SoapProvAdapter implements XMLAdapter {

        private String key="";

        private String url=null;

        private Bts10200Operations port=null;

        private String host="";

        public SoapProvAdapter(String args[]) {

          //
          // General SSL properties
          //
          java.util.Properties props = System.getProperties();
          props.put( "javax.net.ssl.keyStore", "bts10200_ks" );
          props.put( "javax.net.ssl.keyStorePassword", "Chillan" );
          props.put( "javax.net.ssl.trustStore", "bts10200_ts" );
          props.put( "javax.net.ssl.trustStorePassword", "Chillan" );

          // Plug our context finder class into the SSL extension
                for(int i=0; i<args.length; i++) {

                        if (args[i].equals("-b")) {

                                host=args[i+1];

                                break;
                        }
```

```
                }
            }

        /**
          * Connect to target server specified by destAddr, using default user and password
          * @param destAddr Hostname, ip address or URL of the destination
          */
        public void connect() throws XMLAdapterException {

                connect("btsadmin","btsadmin");

            }

        /**
          * Connect to target server specified by destAddr, with provided user and password
          * @param destAddr Hostname, ip address or URL of the destination
          */
        public void connect(String user, String pass) throws XMLAdapterException {


                try {

                        disconnect();

                } catch(Exception e) {}

                try {

                        if (host.equals("")) {

                                throw new XMLAdapterException(100, "not a valid host
name");
                        }

                        url="https://"+host+"/axis/services/bts10200";

                        port=new Bts10200OperationsServiceLocator().getbts10200(new
URL(url));

                        key=port.login(user,pass);

                } catch(BtsSoapException e) {

                        throw new XMLAdapterException(100, e.getError_string());


                } catch(Exception e) {

                        throw new XMLAdapterException(100, e.toString());

                }
            }

        /**
          * Make a request to execute an normal CLI command
          * @param request The XML request document
          * @param String The XML formatted answer
          */
        public synchronized String request(String request) throws XMLAdapterException {

                try {

                        testConnected();
```

```
                        return port.request(request,key);

        } catch (XMLAdapterException e) {

                        throw e;

                } catch(BtsSoapException e) {

                        throw new XMLAdapterException(100, e.getError_string());

                } catch(Exception e) {

                        throw new XMLAdapterException(100, e.toString());

                }
        }


    /**
      * Make a getCommandDoc request to
      * @param request The XML request document
      * @param String The XML formatted answer
      */
    public synchronized String getCommandDoc(String verb, String noun) throws
XMLAdapterException {

                try {

                        testConnected();

                        return port.getCommandDoc(verb,noun,key);

        } catch (XMLAdapterException e) {

                        throw e;

                } catch(BtsSoapException e) {

                        throw new XMLAdapterException(e.getError_code(),
e.getError_string());

                } catch(Exception e) {

                        throw new XMLAdapterException(1, e.toString());

                }

        }


    public synchronized String getExtCommandDoc(String verb, String noun) throws
XMLAdapterException {

                try {

                        testConnected();

                        return port.getExtCommandDoc(verb,noun,key);

        } catch (XMLAdapterException e) {

                        throw e;
```

```
                } catch(BtsSoapException e) {

                        throw new XMLAdapterException(e.getError_code(),
e.getError_string());

                } catch(Exception e) {

                        throw new XMLAdapterException(1, e.toString());

                }



        }
    public synchronized String executeMacro(String command) throws XMLAdapterException {


                throw new XMLAdapterException(1, "not implemented yet!");

        }


    public void disconnect() throws XMLAdapterException {

                try {

                        if (port != null && key!=null) {

                                port.logout(key);

                        }

                } catch(BtsSoapException e) {

                        throw new XMLAdapterException(e.getError_code(),
e.getError_string());

                } catch(Exception e) {

                        throw new XMLAdapterException(1, e.toString());

                } finally {

                        port=null;
                        key=null;

                }
        }

        private void testConnected() throws XMLAdapterException {


                if (port==null || key==null) {

                        throw new XMLAdapterException(1, "Adapter is not connected to
target box !");
                }
        }

}
```

WSDL Overview

The Web Services Description Language (WSDL) is used to describe services for the interface in the SOAP adapter. This section provides an overview of the WSDL for the Cisco BTS 10200 Softswitch. These WSDL operations define access to the XML descriptions and documents used to provision the Cisco BTS 10200 Softswitch. The XML document is described in a Chapter 2, "Extensible Markup Language Processing." For the most part, SOAP acts as the transport for these documents.

BTS 10200 Softswitch WSDL

The bts10200.wsdl file contains the general system-wide data structures and type definitions. It also contains the error interfaces (exceptions). See the "Cisco BTS 10200 Softswitch WSDL Code" section on page 2-6 for the full text of the bts10200.wsdl file. This file contains all objects that are defined for use in the BTS 10200. The breakdown of each object is listed below.

- **Bts10200_Security**—The primary security object. It is used to create login keys for use in another object. This security object is required to enable access the BTS 10200.
- **Bts10200**—The basic object used to retrieve XML description documents as well as provisioning and control documents.
- **BtsSoapException**—The object used to report all errors in the Cisco BTS 10200 Softswitch SOAP interface.
- **Mgp**—The object used to communicate with media gateways. It uses simple strings to contain MGW-compatible text commands.

Cisco BTS 10200 Softswitch API

This section covers the actual API calls to the SOAP interface. The assumption is that the client application is developed in the Java language. This does not prohibit the use of C++. However, that is not within the scope of this document.

All parameters that are listed are required for each invocation of methods in the associated object.

Cisco BTS 10200 Softswitch Security

The BTS 10200 security object (Bts10200_Security) provides a user several levels of security for the SOAP interface. It allows authorized users to obtain a security key and use this key for all future transactions. This object must be used prior to all other SOAP method invocations in the interface. This key is valid in the SOAP interface for the life of the user's session. The key is no longer valid once the logout method has been invoked. Likewise, the security key expires after 10 minutes if the system has not been accessed during that period of time, and the user is automatically logged out of the SOAP interface. The user name and password values are the same as those allowed in the CLI /MAC adapter interfaces. The same authorization permissions apply.

Both the login and logout methods described in this section are part of the Bts10200_Security interface. The parameters listed are required for each method and must contain data.

Login

The login method provides authentication of a SOAP interface user. It utilizes the same user security as the FTP or CLI adapters. This method returns a string value defined as a key. This key is required for all transactions against the SOAP interface. It is an authentication key to indicate the specific authorization of a particular user. The method signature is defined using the following code:

```
<wsdl:operation name="login">

    <wsdl:input message="impl:loginRequest" name="loginRequest"/>

    <wsdl:output message="impl:loginResponse" name="loginResponse"/>

    <wsdl:fault message="impl:BtsSoapException" name="BtsSoapException"/>

    </wsdl:operation>
```

- **Return value**—Status indicating success or failure of the operation. A failure indication means the facility is unavailable. A successful return means the operation was completed.
- **Exception**—A BtsSoapException means there is an operational error in processing the request. This includes faults with the parameter types, ranges, and database access.

Login sessions expire in 10 minutes with no activity. This means that a command must traverse each session once every 10 minutes to keep a session alive. This is important for any client application that deploys the use of connection pools.

Logout

Logging gout terminates a login session. This destroys the validity of the authentication key. Once this method is complete, the key can no longer be used for other method invocations. The method signature is defined in the following code:

```
<wsdl:operation name="logout">

    <wsdl:input message="impl:logoutRequest" name="logoutRequest"/>

    <wsdl:output message="impl:logoutResponse" name="logoutResponse"/>

    <wsdl:fault message="impl:BtsSoapException" name="BtsSoapException"/>

    </wsdl:operation>
```

Attempts to login can receive two types of status indicators.

- **Return value**—Status indicating success or failure of the operation. A failure indication means the facility is unavailable. A successful return indicates the operation was completed.
- **Exception**—A BtsSoapException means there is an operational error in processing the request. This includes faults within the parameter types, ranges, and in database access.

BTS 10200 Provisioning API

The BTS 10200 object (Bts10200) provides provisioning interface functions to the BTS 10200 CLI engine for authorized users. See Chapter 2, "Extensible Markup Language Processing" for a description of XML input and output. The CLI commands are parsed into an XML document before the document is sent through the SOAP interface. The SOAP adapter web service then executes the CLI provisioning commands and sends back the reply in an XML document. Each method (description of command retrieval) in this section is part of the Bts10200 interface. The parameters listed are required for each method and must contain data.

getCommandDoc

The getCommandDoc method provides command description retrieval. This method obtains the XML document describing the command syntax and options for a specific noun-verb combination. The method is defined using the following code:

```
      <wsdl:operation name="getCommandDoc">

    <wsdl:input message="impl:getCommandDocRequest" name="getCommandDocRequest"/>

    <wsdl:output message="impl:getCommandDocResponse" name="getCommandDocResponse"/>

    <wsdl:fault message="impl:BtsSoapException" name="BtsSoapException"/>

    </wsdl:operation>
```

getExtCommandDoc

The getExtCommandDoc method provides command description retrieval. This method obtains the XML document describing the command syntax and options for a specific noun-verb combination. The method is defined using the following code:

```
      <wsdl:operation name="getExtCommandDoc">

         <wsdl:input message="impl:getExtCommandDocRequest"
name="getExtCommandDocRequest"/>

         <wsdl:output message="impl:getExtCommandDocResponse"
name="getExtCommandDocResponse"/>

         <wsdl:fault message="impl:BtsSoapException" name="BtsSoapException"/>

      </wsdl:operation>
```

request

The request method processes an XML document-based provisioning request through SOAP interface. The SOAP web service executes the provisioning commands and sends back the reply in an XML document. The method signature is defined using the following code:

```
<wsdl:operation name="request">

   <wsdl:input message="impl:requestRequest" name="requestRequest"/>

   <wsdl:output message="impl:requestResponse" name="requestResponse"/>

   <wsdl:fault message="impl:BtsSoapException" name="BtsSoapException"/>

</wsdl:operation>
```

BtsSoapException

The following basic BtsSoapExceptions can be returned. The numbers given in the sample code refer to the text in the explanation. The text is returned if the sample code is used.

```
public static final int CLIENT_LOGIN_FAIL = 401;
public static final int CLIENT_LOGOUT_FAIL = 402;
public static final int CLIENT_ACCESS_DENIED = 404;
public static final int CLIENT_SESSION_INVALID = 406;
public static final int CLIENT_SESSION_IN_USE = 407;

public static final int CLIENT_BAD_REQUEST = 410;
public static final int CLIENT_INVALID_VALUE = 411;
public static final int CLIENT_SIDE_UNKNOWN = 499;

/** server side exception block **/
public static final int ADAPTER_BLOCKED = 501;
public static final int ADAPTER_INITING = 502;
public static final int ADAPTER_STOPED = 503;
public static final int ADAPTER_FAULT = 504;
public static final int ADAPTER_UNKNOWN = 505;


public static final int BTS_STANDBY = 511;
public static final int BTS_FAULT = 512;
public static final int BTS_UNKNOWN = 513;

public static final int SESSION_OVERLOAD = 520;
public static final int RESOURCE_ALO_FAIL = 530;
public static final int CMD_NOT_SUPPORT = 540;

public static final int SERVER_SIDE_UNKNOWN = 599;
```

Understanding the SOAP Session Manageability Feature

The SSM feature enhances the manageability of user sessions. It impacts four areas:

1. SOAP Software Developer's Kit (SDK)
2. SOAP Interface Servant (SOAP)
3. Session and System Manager (SMG for EMS)
4. User Security Manager (USM for EMS)

The SDK changes better demonstrate the use of API changes and reflect the effects as seen on the client side of a deployment. Session information is processed as part of the existing Session Manager capabilities. SOAP session policy is a timer-controlled process to remove the policy violated sessions.

Software Developer's Kit

There are two new login APIs in the SDK client code. The first API is *loginWithStatus* that returns password aging status. The other API is *loginResetPassword* that resets a new password when an old password is aged. The *SDK Programmer's Guide* contains the details of session management features and the commands to display and clear client sessions in SOAP. It also explains policy behavior and how this behavior impacts login attempts.

The SOAP SDK has the following new components:

- ResetPassword.java—A driver program to utilize the login, reset password, and logout functions.
- SoapProvAdapter.java—Adds access to the password reset API for SOAP, and provides external indications and API for driver logic.

SOAP Interface Servant

This feature impacts the SOAP Interface Servant application. User security is controlled by UserSessionManager and UserAuth objects. User security is the location of password validation and tracking user attributes such as idle login and security keys.

The user security information must have its data externalized through an API. Through this API, queries are available to take snapshots of the present condition of sessions. This information is through database statistics tables in the MySQL database.

Additional message handler functions add an on-demand reporting of session information and acceptance of command requests to terminate the sessions. This interface utilizes the user security API.

The Session Control Policy is handled in a minute-based looping process to screen and remove sessions that match a record in the policy. Policy management is supplied in a new CLI command.

The bts.properties file contains the Maximum User Limit (item name: max.users) and Idle Timeout (item name: idle.timeout) that can be modified manually. The SOAP adapter relies on these numbers to decide if: (1) the maximum user limit is reached, or (2) the user session is idle timeout. The SOAP adapter dynamically reads the file upon user login and during session audit. If the maximum number of users is set to a value higher than 50, the hard limit of 50 maximum users applies.

Session and System Manager

The SOAP Manageability feature adds the following new capabilities:

User Session Display—Display current secure and nonsecure SOAP sessions using the new "show client-session" and "report client-session" commands. The returned data also includes any current CLI sessions.

Manual Session Removal—Remove a SOAP session or a CLI session using the "stop client-session" command. The present "stop session" command applies to CLI users only. Additional information clearly indicates individual sessions.

Policy-driven Smart Session Management—Includes the smart removal of idle sessions allowing new sessions to login, while allowing administrative access at all times. This does not effect idle time. The maximum duration of a session is set whether a session is idle or not. Default idle time is 10 minutes.

Password Aging Notification—Aging notification of the password for a given user when *loginWithStatus* API is used.

SOAP Password Reset—Users can login and reset aging password using the *loginResetPassword* API. If a password expires, access is denied until the password is reset.

Disable Password Aging—Set passwords to never expire when adding a new user or using the "change user" command. Set the status token to PERSIST.

Alarms and Events for Critical Session Handling—Issue warning and major alarms and events when the session threshold of usage is reached. Issue an alarm when the maximum login sessions is reached. Issue an event when a user session is terminated because of a policy violation. This behavior is managed through the Policy table.

User Security Manager

The User Security Manager (USM) has a new status. This status disables the password aging function. The status has the following attributes:

- DISABLED—The user account is locked out and the user cannot access the system.
- ENABLED—The user is active and current for all attributes including password aging.
- PERSIST—The user account has no password aging.

CHAPTER 2

Extensible Markup Language Processing

Revised: August 21, 2008, OL-15336-02

This chapter describes the Extensible Markup Language (XML) process in the Simple Object Access Protocol (SOAP) interface.

Basic XML and SOAP Components

Along with XML, the primary component of the SOAP interface is the Tomcat web application. In addition, there are dependencies on related components in the managed object (MO) Java package. The required packages are as follows:

- Apache XML
- Xerces (parser)
- ECS (XML Document Building Tool Kit)

These packages interoperate with the central to the functional components in the Cisco BTS 10200 Softswitch. There is no larger object model applied to the system. The SOAP interface supports a variety of packet telephony applications.

XML in the SOAP Interface

This section describes how to use XML in the SOAP interface. Terms used in this section follow those used in XML specifications. This is to avoid confusion in the use of terms such as *element*, *subelement*, and *attribute*.

CIS Functions

This section provides the schemas that perform client-side verification of the XML document structure. These schemas cover the following items:

- ManagedObject
- Request
- Reply

The schema for a *ManagedObject* follows the format listed below.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

        <xs:element name="ManagedObject">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="MOAttribute" maxOccurs="unbounded"/>
                        </xs:sequence>
                        <xs:attribute name="Verb" type="xs:string" use="required"/>
                        <xs:attribute name="id" type="xs:string" use="required"/>
                </xs:complexType>
        </xs:element>

        <xs:element name="MOAttribute">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="Required"/>
                                <xs:element ref="Type"/>
                                <xs:element ref="Default"/>
                                <xs:element ref="Width"/>
                                <xs:element ref="HelpText"/>
                                <xs:element ref="Label"/>
                                <xs:element ref="Parser" minOccurs="0"/>
                                <xs:element ref="Permitted" minOccurs="0"/>
                                <xs:element ref="Fk" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attribute name="id" type="xs:string" use="required"/>
                </xs:complexType>
        </xs:element>

        <xs:element name="Required" type="xs:boolean"/>

        <xs:element name="Type">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:enumeration value="single"/>
                                <xs:enumeration value="text"/>
                                <xs:enumeration value="multi"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>

        <xs:element name="Default" type="xs:string"/>

        <xs:element name="HelpText" type="xs:string"/>

        <xs:element name="Label" type="xs:string"/>

        <xs:element name="Noun" type="xs:string"/>

        <xs:element name="Param" type="xs:string"/>

        <xs:element name="Parser">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="JavaScript"/>
                                <xs:element ref="RegExp"/>
                        </xs:sequence>
                        <xs:attribute name="id" use="required" type="xs:string"/>
                </xs:complexType>
        </xs:element>
        <xs:element name="JavaScript" type="xs:string"/>
```

```
        <xs:element name="RegExp" type="xs:string"/>

        <xs:element name="Permitted" type="xs:string"/>

        <xs:element name="Width" type="xs:int" />

        <xs:element name="Fk">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="Noun"/>
                                <xs:element ref="Param"/>
                                <xs:element ref="Fk" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attribute name="id" type="xs:string" use="required"/>
                </xs:complexType>
        </xs:element>

</xs:schema>
```

The schema for a *Request* follows the format listed below.

```
<?xml version="1.0" encoding="UTF-8"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
        <xs:element name="Request">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="Entry" maxOccurs="unbounded"/>
                        </xs:sequence>
                        <xs:attribute name="Verb" type="xs:string" use="required"/>
                        <xs:attribute name="Noun" type="xs:string" use="required"/>
                </xs:complexType>
        </xs:element>
        <xs:element name="Entry">
                <xs:complexType>
                        <xs:attribute name="Key" type="xs:string" use="required"/>
                        <xs:attribute name="Value" type="xs:string" use="required"/>
                </xs:complexType>
        </xs:element>
</xs:schema>
```

The schema for a *Reply* follows the format listed below.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

        <xs:element name="Reply">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="Status"/>
                                <xs:element ref="Reason"/>
                                <xs:element ref="Size"/>
                                <xs:element ref="AbsoluteSize"/>
                                <xs:element ref="StartRow"/>
                                <xs:element ref="DataTable"/>
                        </xs:sequence>
                        <xs:attribute name="id" type="xs:string" use="required"/>
                </xs:complexType>
        </xs:element>

        <xs:element name="Status" type="xs:boolean"/>

        <xs:element name="Reason" type="xs:string"/>

        <xs:element name="Size" type="xs:integer"/>
```

```
        <xs:element name="AbsoluteSize" type="xs:integer"/>

        <xs:element name="StartRow" type="xs:integer"/>

        <xs:element name="DataTable">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="Row" maxOccurs="unbounded"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>

        <xs:element name="Row">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="Column" maxOccurs="unbounded"/>
                        </xs:sequence>
                        <xs:attribute name="id" use="required" type="xs:integer"/>
                </xs:complexType>
        </xs:element>

        <xs:element name="Column" >
                <xs:complexType>
                        <xs:simpleContent>
                                <xs:extension base="xs:string">
                                        <xs:attribute name="id" use="required"/>
                                </xs:extension>
                        </xs:simpleContent>
                </xs:complexType>
        </xs:element>

</xs:schema>
```

The interface definition language (IDL) allows access to XML description documents for each noun and verb combination. For example, the **add subscriber** command generates a matching XML document that defines the element and attributes of this command. The IDL allows command processing based on a well-formed but unverified XML document.

The IDL allows command access to supported media gateway (MGW) devices. The IDL commands do not follow the XML access format defined in the schema. The CIS supports MGW commands that are native to the MGW internal command structure.

XML documents that originate in the BTS 10200 are dynamically generated and include all command-description documents.

ManagedObject

A *ManagedObject* has one element, the *MOAttribute*. A *ManagedObject* also has two attributes: the *id* of the ManagedObject and the *verb*. The *id* represents the object on which some action is to be taken. The *verb* indicates the action to be taken. For example, subscriber, or termination is a valid ID. This *id* a required attribute.

The following list describes the various parts of the schema and he valid values for each part:

- **Verb**—This attribute defines the action to take on a given ManagedObject. This is a required attribute and is composed of character data.
- **MOAttribute**—The ManagedObject can contain none, one, or more of these elements. It has one attribute named *id*. This character data acts as a label for the element. The order of these elements does not imply any specific behavior. They are arbitrarily listed.
- **Required**—This subelement has two values defined as true or false.
- **Type**—This subelement defines whether the MOAttribute has a single value, multiple values, or is a text. The multi or single option implies that the permitted element offers a list of choices.
- **Default**—This subelement indicates the default value for the MOAttribute.
- **Width**—This subelement indicates the total field width of the data. For example, if the MOAttribute is a description, this indicates the length of the description.
- **HelpText**—This subelement offers a brief text to indicate the nature of the MOAttribute.
- **Permitted**—This subelement specifies the possible values or ranges for the MOAttribute.
- **Parser**—This subelement indicates what type of validation is required. There is a single attribute to this subelement. The attribute is specified as character data in an ID field. The subelements are listed below.
  - **JavaScript**—This subelement represents a JavaScript, which performs validation or regular expression matching.
  - **RegExp**—This subelement defines the regular expression in character data format.

Request Schema

The Request schema consists of a single element that contains no subelements or one or more Entry subelements. An Entry subelement can have as many as two attributes.

The following list presents the parts of the Request schema and the values that are valid for those parts:

- **Noun**—This attribute defines the item on which some operation is requested. This is expressed as character data.
- **Verb**—This attribute defines the action to perform on the “Noun” attribute. This is expressed as character data.

The *Entry* element is allowed to be empty. It can also contain two attributes. These attributes are defined as follows:

- **Key**—This is the *id* value derived from the MOAttribute in the ManagedObject. It is expressed as character data.
- **Value**—This is the client-derived value assigned to the Key attribute. The Value is expressed as character data. It should also conform to the subelements in MOAttribute from which this Key/Value pair was derived.

Reply Schema

The Reply schema defines the structure of returned data generated in response to a Request. The Reply contains three elements and no attributes. These elements are defined as follows:

- **Status**—The Status element has two possible values. Either true or false is applied to this element.

- **Reason**—The Reason element contains character data. This element explains the cause for an error in processing a command or returns a success indication.
- **DataTable**—This element has one attribute and one subelement, which are defined below. The DataTable is used as the container for data that results from the execution of a request. Each Reply can contain a *DataTable* element.
  - **Row**—This subelement defines a single complete item of data. A *DataTable* can contain one or more Row subelements. A Row has one attribute. This character data defines the row ID. The ID is always a sequential value based on the number of returned rows. The *id* attribute is required.
  - **Col**—Each Row contains a subelement known as a Col, which is expressed as character data. The subelement Col has one attribute, *id*, which is expressed as a character value. This is the same ID value that is used in the MOAttribute. The *id* attribute is required.

SOAP Interface Adapter Implementation

The SOAP interface adapter is a web service application within Tomcat that specifies an external interface. This section provides more detail about the structure of the document interchange between the SOAP adapter and a client-side program.

One global issue for the external interface is that all documents covered here are defined as *well-formed* but not *verified*. This means that the schema is not an embedded part of the XML document. If you embed the schema in a document, parser packages are used to validate the structure of the document. However, this impedes the transition to XML schemas, should schemas be desired by other customers. The client side can still use the schema, included in this document, to perform validation.

Cisco BTS 10200 Softswitch WSDL Code

This section describes the system Web Services Description Language (WSDL) file for the SOAP interface in the BTS 10200. This WSDL code applies to BTS 10200, Release 6.0.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://www.cisco.com/BTS10200/i01" xmlns:apac
hesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://www.cisco.com/BTS1020
0/i01" xmlns:intf="http://www.cisco.com/BTS10200/i01" xmlns:wsdl="http://schemas
.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns
:xsd="http://www.w3.org/2001/XMLSchema">
<!--
Copyright (c) 2002-2006 by Cisco Systems, Inc.
-->
<!--WSDL created by Apache Axis version: 1.4
Built on Jul 23, 2006 (06:38:00 CST)-->
 <wsdl:types>
  <schema elementFormDefault="qualified" targetNamespace="http://www.cisco.com/B
TS10200/i01" xmlns="http://www.w3.org/2001/XMLSchema">
   <element name="request">
    <complexType>
     <sequence>
      <element name="xmlstring" type="xsd:string"/>
      <element name="key" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="requestResponse">
    <complexType>
```

```
  <sequence>
   <element name="requestReturn" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<complexType name="BtsSoapException">
 <sequence>
  <element name="error_code" type="xsd:int"/>
  <element name="error_string" nillable="true" type="xsd:string"/>
 </sequence>
</complexType>
<element name="fault" type="impl:BtsSoapException"/>
<element name="login">
 <complexType>
  <sequence>
   <element name="user" type="xsd:string"/>
   <element name="password" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="loginResponse">
 <complexType>
  <sequence>
   <element name="loginReturn" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="logout">
 <complexType>
  <sequence>
   <element name="key" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="logoutResponse">
 <complexType/>
</element>
<element name="getCommandDoc">
 <complexType>
  <sequence>
   <element name="noun" type="xsd:string"/>
   <element name="verb" type="xsd:string"/>
   <element name="key" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="getCommandDocResponse">
 <complexType>
  <sequence>
   <element name="getCommandDocReturn" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="getExtCommandDoc">
 <complexType>
  <sequence>
   <element name="noun" type="xsd:string"/>
   <element name="verb" type="xsd:string"/>
   <element name="key" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="getExtCommandDocResponse">
 <complexType>
```

```
    <sequence>
     <element name="getExtCommandDocReturn" type="xsd:string"/>
    </sequence>
   </complexType>
  </element>
 </schema>
</wsdl:types>

  <wsdl:message name="getExtCommandDocRequest">

     <wsdl:part element="impl:getExtCommandDoc" name="parameters"/>

  </wsdl:message>

  <wsdl:message name="BtsSoapException">

     <wsdl:part element="impl:fault" name="fault"/>

  </wsdl:message>

  <wsdl:message name="requestRequest">

     <wsdl:part element="impl:request" name="parameters"/>

  </wsdl:message>

  <wsdl:message name="loginRequest">

     <wsdl:part element="impl:login" name="parameters"/>

  </wsdl:message>

  <wsdl:message name="getCommandDocRequest">

     <wsdl:part element="impl:getCommandDoc" name="parameters"/>

  </wsdl:message>

  <wsdl:message name="logoutResponse">

     <wsdl:part element="impl:logoutResponse" name="parameters"/>

  </wsdl:message>

  <wsdl:message name="logoutRequest">

     <wsdl:part element="impl:logout" name="parameters"/>

  </wsdl:message>

  <wsdl:message name="getExtCommandDocResponse">

     <wsdl:part element="impl:getExtCommandDocResponse" name="parameters"/>

  </wsdl:message>

  <wsdl:message name="getCommandDocResponse">

     <wsdl:part element="impl:getCommandDocResponse" name="parameters"/>

  </wsdl:message>

  <wsdl:message name="requestResponse">
```

```
      <wsdl:part element="impl:requestResponse" name="parameters"/>
   </wsdl:message>
   <wsdl:message name="loginResponse">
      <wsdl:part element="impl:loginResponse" name="parameters"/>
   </wsdl:message>
   <wsdl:portType name="Bts10200Operations">
      <wsdl:operation name="request">
         <wsdl:input message="impl:requestRequest" name="requestRequest"/>
         <wsdl:output message="impl:requestResponse" name="requestResponse"/>
         <wsdl:fault message="impl:BtsSoapException" name="BtsSoapException"/>
      </wsdl:operation>
      <wsdl:operation name="login">
         <wsdl:input message="impl:loginRequest" name="loginRequest"/>
         <wsdl:output message="impl:loginResponse" name="loginResponse"/>
         <wsdl:fault message="impl:BtsSoapException" name="BtsSoapException"/>
      </wsdl:operation>
      <wsdl:operation name="logout">
         <wsdl:input message="impl:logoutRequest" name="logoutRequest"/>
         <wsdl:output message="impl:logoutResponse" name="logoutResponse"/>
         <wsdl:fault message="impl:BtsSoapException" name="BtsSoapException"/>
      </wsdl:operation>
      <wsdl:operation name="getCommandDoc">
         <wsdl:input message="impl:getCommandDocRequest" name="getCommandDocRequ
est"/>
         <wsdl:output message="impl:getCommandDocResponse" name="getCommandDocRe
sponse"/>
         <wsdl:fault message="impl:BtsSoapException" name="BtsSoapException"/>
      </wsdl:operation>
      <wsdl:operation name="getExtCommandDoc">
         <wsdl:input message="impl:getExtCommandDocRequest" name="getExtCommandD
ocRequest"/>
         <wsdl:output message="impl:getExtCommandDocResponse" name="getExtComman
dDocResponse"/>
         <wsdl:fault message="impl:BtsSoapException" name="BtsSoapException"/>
```

```
      </wsdl:operation>

   </wsdl:portType>

   <wsdl:binding name="bts10200SoapBinding" type="impl:Bts10200Operations">

      <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/s
oap/http"/>

      <wsdl:operation name="request">

         <wsdlsoap:operation soapAction=""/>

         <wsdl:input name="requestRequest">

            <wsdlsoap:body use="literal"/>

         </wsdl:input>

         <wsdl:output name="requestResponse">

            <wsdlsoap:body use="literal"/>

         </wsdl:output>

         <wsdl:fault name="BtsSoapException">

            <wsdlsoap:fault name="BtsSoapException" use="literal"/>

         </wsdl:fault>

      </wsdl:operation>

      <wsdl:operation name="login">

         <wsdlsoap:operation soapAction=""/>

         <wsdl:input name="loginRequest">

            <wsdlsoap:body use="literal"/>

         </wsdl:input>

         <wsdl:output name="loginResponse">

            <wsdlsoap:body use="literal"/>

         </wsdl:output>

         <wsdl:fault name="BtsSoapException">

            <wsdlsoap:fault name="BtsSoapException" use="literal"/>

         </wsdl:fault>

      </wsdl:operation>

      <wsdl:operation name="logout">

         <wsdlsoap:operation soapAction=""/>

         <wsdl:input name="logoutRequest">

            <wsdlsoap:body use="literal"/>
```

```
      </wsdl:input>
      <wsdl:output name="logoutResponse">
         <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="BtsSoapException">
         <wsdlsoap:fault name="BtsSoapException" use="literal"/>
      </wsdl:fault>
   </wsdl:operation>
   <wsdl:operation name="getCommandDoc">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getCommandDocRequest">
         <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getCommandDocResponse">
         <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="BtsSoapException">
         <wsdlsoap:fault name="BtsSoapException" use="literal"/>
      </wsdl:fault>
   </wsdl:operation>
   <wsdl:operation name="getExtCommandDoc">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getExtCommandDocRequest">
         <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getExtCommandDocResponse">
         <wsdlsoap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="BtsSoapException">
         <wsdlsoap:fault name="BtsSoapException" use="literal"/>
      </wsdl:fault>
   </wsdl:operation>
```

```
    </wsdl:binding>
    <wsdl:service name="Bts10200OperationsService">
       <wsdl:port binding="impl:bts10200SoapBinding" name="bts10200">
          <wsdlsoap:address location="https://localhost/axis/services/bts10200"/>
       </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

CHAPTER 3

Troubleshooting

Revised: August 21, 2008, OL-15336-02

This chapter describes the resolution of the events and alarms associated with the Simple Object Access Protocol (SOAP) interface.

SOAP Operations Support System Events

This section provides the information that you need to monitor and correct Operations Support System events. Table 3-1 lists all Operations Support System events in numerical order and provides a cross reference to each subsection in this section.

Table 3-1 SOAP Session Manageability OSS Events

Event Type	Event Name	Event Severity
OSS 17	Session Has Been Removed (OSS 17)	INFO
OSS 18	Invalid Session Request (OSS 18)	INFO
OSS 19	Interface Is Active and Operational (OSS 19)	INFO
OSS 20	Interface Is Not Started or Not Operational (OSS 20)	MINOR

Session Has Been Removed (OSS 17)

The Session Has Been Removed event serves as an information alert that the session has been removed. The primary cause of the informational alert is that the session was removed because it was idle over the timeout limit.

Invalid Session Request (OSS 18)

The Invalid Session Request event serves as an information alert that the noun, verb, or parameters of the **request** command are not valid. To correct the primary cause of the Invalid Session Request event, check and correct the **request** command.

Interface Is Active and Operational (OSS 19)

The Interface Is Active and Operational event serves as an informational alert that the application interface is active and operational. The event is informational only and no further action is required.

Interface Is Not Started or Not Operational (OSS 20)

The Interface Is Not Started or Is Not Operational alarm (minor) indicates that application interface has failed to start or is not operational. To troubleshoot and correct the cause of the alarm, restart the application interface.

Alarms

The following alarms might be returned by this feature.

OSS (17)

To monitor and correct the cause of the event, refer to the "Session Has Been Removed (OSS 17)" section.

DESCRIPTION	Session Has Been Removed
SEVERITY	INFO
THRESHOLD	100
THROTTLE	0
DATAWORDS	Session Type - STRING [16] User ID - STRING [16] Session Key - STRING [20]
PRIMARY CAUSE	The session was removed since it was idle over the timeout limit.
PRIMARY ACTION	N/A.

OSS (18)

To monitor and correct the cause of the event, refer to the "Invalid Session Request (OSS 18)" section.

DESCRIPTION	Invalid Session Request
SEVERITY	INFO
THRESHOLD	100
THROTTLE	0
DATAWORDS	User ID - STRING [16] Session Key - STRING [20] Request - STRING [256]
PRIMARY CAUSE	The noun, verb, or parameters of the **request** command are not valid.
PRIMARY ACTION	Check and correct the **request** command.

OSS (19)

To monitor and correct the cause of the event, refer to the "Interface Is Active and Operational (OSS 19)" section.

DESCRIPTION	Interface is Active and Operational
SEVERITY	INFO
THRESHOLD	100
THROTTLE	0
DATAWORDS	Session Type - STRING [16]
PRIMARY CAUSE	The application interface is active and operational.
PRIMARY ACTION	N/A.

OSS (20)

To monitor and correct the cause of the event, refer to the "Interface Is Not Started or Not Operational (OSS 20)" section.

DESCRIPTION	Interface Is Not Started or Is Not Operational
SEVERITY	MINOR
THRESHOLD	100
THROTTLE	0
DATAWORDS	Session Type - STRING [16]
PRIMARY CAUSE	The application interface has failed to start or is not operational.
PRIMARY ACTION	Restart the application interface.

APPENDIX A

XML Description Documents

Revised: August 21, 2008, OL-15336-02

This appendix provides examples of the XML description documents.

Subscriber Noun and Add Verb

The following sample XML description document provides an example of the subscriber noun and add verb.

```
================================================
<ManagedObject Verb="add" id="subscriber"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ManagedObject.xsd">
  <MOAttribute id="dn1">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>14</Width>
    <HelpText>Enter at least 1, but not more than 14 characters from the following set:
{0123456789-}.</HelpText>
    <Label>Dn1</Label>
    <Parser id="GenericDNParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="tg">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>32</Width>
    <HelpText>Enter at least 1 character, but not more than 32 characters.</HelpText>
    <Label>Tg</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="policy_id">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>16</Width>
    <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
```

```
    <Label>POLICY_ID</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="category">
    <Required>false</Required>
    <Type>single</Type>
    <Default>[INDIVIDUAL]</Default>
    <Width>15</Width>
    <HelpText>Enter one of the following values: [INDIVIDUAL, MLHG, MLHG_INDIVIDUAL,
MLHG_PREF_INDIV, CTXG, CTXG_INDIVIDUAL, PBX, CTXG_TG, CTXG_MLHG, RACF, IVR]</HelpText>
    <Label>Category</Label>
   <Permitted>[INDIVIDUAL, MLHG, MLHG_INDIVIDUAL, MLHG_PREF_INDIV, CTXG, CTXG_INDIVIDUAL,
PBX, CTXG_TG, CTXG_MLHG, RACF, IVR]</Permitted>
  </MOAttribute>
  <MOAttribute id="ss_number">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>11</Width>
   <HelpText>Enter a Social Security Number in the form ###-##-#### where # is digit from
0-9.</HelpText>
    <Label>Ss Number</Label>
    <Parser id="SocSecParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="ctxg_id">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>16</Width>
    <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
    <Label>Ctxg Id</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="name">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>32</Width>
    <HelpText>Enter at least 1 character, but not more than 32 characters.</HelpText>
    <Label>Name</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="mlhg_pref_list_id">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>16</Width>
    <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
    <Label>Mlhg Pref List Id</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
```

```
    </Parser>
  </MOAttribute>
  <MOAttribute id="address2">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>32</Width>
    <HelpText>Enter at least 1 character, but not more than 32 characters.</HelpText>
    <Label>Address2</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="address1">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>32</Width>
    <HelpText>Enter at least 1 character, but not more than 32 characters.</HelpText>
    <Label>Address1</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="city">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>16</Width>
    <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
    <Label>City</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="terminating_immediate_rel">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[N]</Default>
    <Width>3</Width>
    <HelpText>Enter a boolean value of Y for yes or N for no.</HelpText>
    <Label>Terminating Immediate Release</Label>
    <Parser id="BooleanParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="billing_dn">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>32</Width>
    <HelpText>Enter at least 1, but not more than 32 characters from the following set:
{0123456789-}.</HelpText>
    <Label>Billing Dn</Label>
    <Parser id="GenericDNParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="language">
```

```
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>16</Width>
    <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
    <Label>Language</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="email">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>64</Width>
    <HelpText>Enter an email address in the form text@text where text is a set of
characters with no spaces.</HelpText>
    <Label>Email</Label>
    <Parser id="EmailParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="mlhg_id">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>16</Width>
    <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
    <Label>MLHG ID</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="tgn_id">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>8</Width>
    <HelpText>Enter a number from 0 to 99999999.</HelpText>
    <Label>Trunk Group Number ID</Label>
    <Parser id="DecimalParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="mgw_id">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>32</Width>
    <HelpText>Enter at least 0 characters, but not more than 32 characters.</HelpText>
    <Label>Media Gateway ID</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="status">
    <Required>false</Required>
    <Type>single</Type>
    <Default>[ACTIVE]</Default>
```

```
    <Width>17</Width>
    <HelpText>Enter one of the following values: [ACTIVE, TEMP_OOS, TEMP_DISCONNECTED,
TEMP_UNAVAILABLE]</HelpText>
    <Label>Status</Label>
    <Permitted>[ACTIVE, TEMP_OOS, TEMP_DISCONNECTED, TEMP_UNAVAILABLE]</Permitted>
  </MOAttribute>
  <MOAttribute id="term_id">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>32</Width>
    <HelpText>Enter at least 1 character, but not more than 32 characters.</HelpText>
    <Label>Termination ID</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="usage_sens">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[Y]</Default>
    <Width>3</Width>
    <HelpText>Enter a boolean value of Y for yes or N for no.</HelpText>
    <Label>Usage Sens</Label>
    <Parser id="BooleanParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="id">
    <Required>true</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>30</Width>
    <HelpText>Enter at least 1 character, but not more than 30 characters.</HelpText>
    <Label>ID</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="grp">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[N]</Default>
    <Width>3</Width>
    <HelpText>Enter a boolean value of Y for yes or N for no.</HelpText>
    <Label>Grp</Label>
    <Parser id="BooleanParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="sub_profile_id">
    <Required>true</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>16</Width>
    <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
    <Label>Sub Profile Id</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
```

```
  </Parser>
</MOAttribute>
<MOAttribute id="country">
  <Required>false</Required>
  <Type>text</Type>
  <Default>[USA]</Default>
  <Width>16</Width>
  <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
  <Label>Country</Label>
  <Parser id="TextParser">
    <JavaScript>TBD</JavaScript>
    <RegExp>TBD</RegExp>
  </Parser>
</MOAttribute>
<MOAttribute id="cos_restrict_id">
  <Required>false</Required>
  <Type>text</Type>
  <Default>[null]</Default>
  <Width>16</Width>
  <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
  <Label>COS Restrict ID</Label>
  <Parser id="TextParser">
    <JavaScript>TBD</JavaScript>
    <RegExp>TBD</RegExp>
  </Parser>
</MOAttribute>
<MOAttribute id="qos_id">
  <Required>false</Required>
  <Type>text</Type>
  <Default>[null]</Default>
  <Width>16</Width>
  <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
  <Label>QOS ID</Label>
  <Parser id="TextParser">
    <JavaScript>TBD</JavaScript>
    <RegExp>TBD</RegExp>
  </Parser>
</MOAttribute>
<MOAttribute id="term_type">
  <Required>false</Required>
  <Type>single</Type>
  <Default>[TERM]</Default>
  <Width>5</Width>
  <HelpText>Enter one of the following values: [TERM, TG, ROUTE, RG]</HelpText>
  <Label>TERM TYPE</Label>
  <Permitted>[TERM, TG, ROUTE, RG]</Permitted>
</MOAttribute>
<MOAttribute id="ring_type_dn1">
  <Required>false</Required>
  <Type>text</Type>
  <Default>[1]</Default>
  <Width>1</Width>
  <HelpText>Enter a number from 1 to 3.</HelpText>
  <Label>Ring Type Dn1</Label>
  <Parser id="DecimalParser">
    <JavaScript>TBD</JavaScript>
    <RegExp>TBD</RegExp>
  </Parser>
</MOAttribute>
<MOAttribute id="immediate_release">
  <Required>false</Required>
  <Type>text</Type>
  <Default>[N]</Default>
  <Width>3</Width>
```

```
  <HelpText>Enter a boolean value of Y for yes or N for no.</HelpText>
  <Label>Immediate Release</Label>
  <Parser id="BooleanParser">
    <JavaScript>TBD</JavaScript>
    <RegExp>TBD</RegExp>
  </Parser>
</MOAttribute>
<MOAttribute id="sip_url">
  <Required>false</Required>
  <Type>text</Type>
  <Default>[null]</Default>
  <Width>32</Width>
  <HelpText>Enter at least 1 character, but not more than 32 characters.</HelpText>
  <Label>Sip Url</Label>
  <Parser id="TextParser">
    <JavaScript>TBD</JavaScript>
    <RegExp>TBD</RegExp>
  </Parser>
</MOAttribute>
<MOAttribute id="zipcode">
  <Required>false</Required>
  <Type>text</Type>
  <Default>[null]</Default>
  <Width>10</Width>
  <HelpText>Enter at least 1 character, but not more than 10 characters.</HelpText>
  <Label>Zipcode</Label>
  <Parser id="TextParser">
    <JavaScript>TBD</JavaScript>
    <RegExp>TBD</RegExp>
  </Parser>
</MOAttribute>
<MOAttribute id="pic3">
  <Required>false</Required>
  <Type>text</Type>
  <Default>[null]</Default>
  <Width>4</Width>
  <HelpText>Enter a PIC value as four numeric characters, NPIC, or NONE.</HelpText>
  <Label>Pic3</Label>
  <Parser id="PicParser">
    <JavaScript>TBD</JavaScript>
    <RegExp>TBD</RegExp>
  </Parser>
</MOAttribute>
<MOAttribute id="pic2">
  <Required>false</Required>
  <Type>text</Type>
  <Default>[null]</Default>
  <Width>4</Width>
  <HelpText>Enter a PIC value as four numeric characters, NPIC, or NONE.</HelpText>
  <Label>Pic2</Label>
  <Parser id="PicParser">
    <JavaScript>TBD</JavaScript>
    <RegExp>TBD</RegExp>
  </Parser>
</MOAttribute>
<MOAttribute id="privacy">
  <Required>false</Required>
  <Type>single</Type>
  <Default>[NONE]</Default>
  <Width>4</Width>
  <HelpText>Enter one of the following values: [FULL, NAME, NONE]</HelpText>
  <Label>Privacy</Label>
  <Permitted>[FULL, NAME, NONE]</Permitted>
</MOAttribute>
```

```
  <MOAttribute id="pic1">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>4</Width>
    <HelpText>Enter a PIC value as four numeric characters, NPIC, or NONE.</HelpText>
    <Label>Pic1</Label>
    <Parser id="PicParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="state">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>16</Width>
    <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
    <Label>State</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
</ManagedObject>
```

Foreign Key Relationships

The following sample XML extended format description contains the foreign key relationships for a given command.

```
================================================
<ManagedObject Verb="add" id="subscriber"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ManagedObject.xsd">
  <MOAttribute id="dn1">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>14</Width>
    <HelpText>Enter at least 1, but not more than 14 characters from the following set:
{0123456789-}.</HelpText>
    <Label>Dn1</Label>
    <Parser id="GenericDNParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="tg">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>32</Width>
    <HelpText>Enter at least 1 character, but not more than 32 characters.</HelpText>
    <Label>Tg</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="policy_id">
```

```
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>16</Width>
    <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
    <Label>POLICY_ID</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="category">
    <Required>false</Required>
    <Type>single</Type>
    <Default>[INDIVIDUAL]</Default>
    <Width>15</Width>
    <HelpText>Enter one of the following values: [INDIVIDUAL, MLHG, MLHG_INDIVIDUAL,
MLHG_PREF_INDIV, CTXG, CTXG_INDIVIDUAL, PBX, CTXG_TG, CTXG_MLHG, RACF, IVR]</HelpText>
    <Label>Category</Label>
    <Permitted>[INDIVIDUAL, MLHG, MLHG_INDIVIDUAL, MLHG_PREF_INDIV, CTXG, CTXG_INDIVIDUAL,
PBX, CTXG_TG, CTXG_MLHG, RACF, IVR]</Permitted>
  </MOAttribute>
  <MOAttribute id="ss_number">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>11</Width>
    <HelpText>Enter a Social Security Number in the form ###-##-#### where # is digit from
0-9.</HelpText>
    <Label>Ss Number</Label>
    <Parser id="SocSecParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="ctxg_id">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>16</Width>
    <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
    <Label>Ctxg Id</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
    <Fk id="CENTREX_GRP_PK">
      <Noun>centrex_grp</Noun>
      <Param>id</Param>
    </Fk>
  </MOAttribute>
  <MOAttribute id="name">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>32</Width>
    <HelpText>Enter at least 1 character, but not more than 32 characters.</HelpText>
    <Label>Name</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="mlhg_pref_list_id">
```

```
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>16</Width>
    <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
    <Label>Mlhg Pref List Id</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
    <Fk id="MLH_PREF_LIST_PK">
      <Noun>mlhg_pref_list</Noun>
      <Param>id</Param>
    </Fk>
  </MOAttribute>
  <MOAttribute id="address2">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>32</Width>
    <HelpText>Enter at least 1 character, but not more than 32 characters.</HelpText>
    <Label>Address2</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="address1">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>32</Width>
    <HelpText>Enter at least 1 character, but not more than 32 characters.</HelpText>
    <Label>Address1</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="city">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>16</Width>
    <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
    <Label>City</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="terminating_immediate_rel">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[N]</Default>
    <Width>3</Width>
    <HelpText>Enter a boolean value of Y for yes or N for no.</HelpText>
    <Label>Terminating Immediate Release</Label>
    <Parser id="BooleanParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="billing_dn">
```

```
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>32</Width>
    <HelpText>Enter at least 1, but not more than 32 characters from the following set:
{0123456789-}.</HelpText>
    <Label>Billing Dn</Label>
    <Parser id="GenericDNParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="language">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>16</Width>
    <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
    <Label>Language</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="email">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>64</Width>
    <HelpText>Enter an email address in the form text@text where text is a set of
characters with no spaces.</HelpText>
    <Label>Email</Label>
    <Parser id="EmailParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="mlhg_id">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>16</Width>
    <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
    <Label>MLHG ID</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
    <Fk id="MLH_PREF_LIST_PK">
      <Noun>mlhg_pref_list</Noun>
      <Param>mlhg_id</Param>
      <Fk id="MLHG_PK">
        <Noun>mlhg</Noun>
        <Param>id</Param>
      </Fk>
    </Fk>
  </MOAttribute>
  <MOAttribute id="tgn_id">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>8</Width>
    <HelpText>Enter a number from 0 to 99999999.</HelpText>
    <Label>Trunk Group Number ID</Label>
```

```
      <Parser id="DecimalParser">
        <JavaScript>TBD</JavaScript>
        <RegExp>TBD</RegExp>
      </Parser>
      <Fk id="TRUNK_GRP_PK">
        <Noun>trunk_grp</Noun>
        <Param>id</Param>
      </Fk>
    </MOAttribute>
    <MOAttribute id="mgw_id">
      <Required>false</Required>
      <Type>text</Type>
      <Default>[null]</Default>
      <Width>32</Width>
      <HelpText>Enter at least 0 characters, but not more than 32 characters.</HelpText>
      <Label>Media Gateway ID</Label>
      <Parser id="TextParser">
        <JavaScript>TBD</JavaScript>
        <RegExp>TBD</RegExp>
      </Parser>
      <Fk id="TERMINATION_PK">
        <Noun>termination</Noun>
        <Param>mgw_id</Param>
        <Fk id="MGW_PK">
          <Noun>mgw</Noun>
          <Param>id</Param>
        </Fk>
      </Fk>
    </MOAttribute>
    <MOAttribute id="status">
      <Required>false</Required>
      <Type>single</Type>
      <Default>[ACTIVE]</Default>
      <Width>17</Width>
      <HelpText>Enter one of the following values: [ACTIVE, TEMP_OOS, TEMP_DISCONNECTED,
TEMP_UNAVAILABLE]</HelpText>
      <Label>Status</Label>
      <Permitted>[ACTIVE, TEMP_OOS, TEMP_DISCONNECTED, TEMP_UNAVAILABLE]</Permitted>
    </MOAttribute>
    <MOAttribute id="term_id">
      <Required>false</Required>
      <Type>text</Type>
      <Default>[null]</Default>
      <Width>32</Width>
      <HelpText>Enter at least 1 character, but not more than 32 characters.</HelpText>
      <Label>Termination ID</Label>
      <Parser id="TextParser">
        <JavaScript>TBD</JavaScript>
        <RegExp>TBD</RegExp>
      </Parser>
      <Fk id="TERMINATION_PK">
        <Noun>termination</Noun>
        <Param>id</Param>
      </Fk>
    </MOAttribute>
    <MOAttribute id="usage_sens">
      <Required>false</Required>
      <Type>text</Type>
      <Default>[Y]</Default>
      <Width>3</Width>
      <HelpText>Enter a boolean value of Y for yes or N for no.</HelpText>
      <Label>Usage Sens</Label>
      <Parser id="BooleanParser">
        <JavaScript>TBD</JavaScript>
```

```
    <RegExp>TBD</RegExp>
  </Parser>
</MOAttribute>
<MOAttribute id="id">
  <Required>true</Required>
  <Type>text</Type>
  <Default>[null]</Default>
  <Width>30</Width>
  <HelpText>Enter at least 1 character, but not more than 30 characters.</HelpText>
  <Label>ID</Label>
  <Parser id="TextParser">
    <JavaScript>TBD</JavaScript>
    <RegExp>TBD</RegExp>
  </Parser>
</MOAttribute>
<MOAttribute id="grp">
  <Required>false</Required>
  <Type>text</Type>
  <Default>[N]</Default>
  <Width>3</Width>
  <HelpText>Enter a boolean value of Y for yes or N for no.</HelpText>
  <Label>Grp</Label>
  <Parser id="BooleanParser">
    <JavaScript>TBD</JavaScript>
    <RegExp>TBD</RegExp>
  </Parser>
</MOAttribute>
<MOAttribute id="sub_profile_id">
  <Required>true</Required>
  <Type>text</Type>
  <Default>[null]</Default>
  <Width>16</Width>
  <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
  <Label>Sub Profile Id</Label>
  <Parser id="TextParser">
    <JavaScript>TBD</JavaScript>
    <RegExp>TBD</RegExp>
  </Parser>
  <Fk id="SUBSCRIBER_PROFILE_PK">
    <Noun>subscriber_profile</Noun>
    <Param>id</Param>
  </Fk>
</MOAttribute>
<MOAttribute id="country">
  <Required>false</Required>
  <Type>text</Type>
  <Default>[USA]</Default>
  <Width>16</Width>
  <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
  <Label>Country</Label>
  <Parser id="TextParser">
    <JavaScript>TBD</JavaScript>
    <RegExp>TBD</RegExp>
  </Parser>
</MOAttribute>
<MOAttribute id="cos_restrict_id">
  <Required>false</Required>
  <Type>text</Type>
  <Default>[null]</Default>
  <Width>16</Width>
  <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
  <Label>COS Restrict ID</Label>
  <Parser id="TextParser">
    <JavaScript>TBD</JavaScript>
```

```
    <RegExp>TBD</RegExp>
  </Parser>
  <Fk id="COST_RESTRICT_PK">
    <Noun>cos_restrict</Noun>
    <Param>id</Param>
  </Fk>
</MOAttribute>
<MOAttribute id="qos_id">
  <Required>false</Required>
  <Type>text</Type>
  <Default>[null]</Default>
  <Width>16</Width>
  <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
  <Label>QOS ID</Label>
  <Parser id="TextParser">
    <JavaScript>TBD</JavaScript>
    <RegExp>TBD</RegExp>
  </Parser>
  <Fk id="QOS_PK">
    <Noun>qos</Noun>
    <Param>id</Param>
  </Fk>
</MOAttribute>
<MOAttribute id="term_type">
  <Required>false</Required>
  <Type>single</Type>
  <Default>[TERM]</Default>
  <Width>5</Width>
  <HelpText>Enter one of the following values: [TERM, TG, ROUTE, RG]</HelpText>
  <Label>TERM TYPE</Label>
  <Permitted>[TERM, TG, ROUTE, RG]</Permitted>
</MOAttribute>
<MOAttribute id="ring_type_dn1">
  <Required>false</Required>
  <Type>text</Type>
  <Default>[1]</Default>
  <Width>1</Width>
  <HelpText>Enter a number from 1 to 3.</HelpText>
  <Label>Ring Type Dn1</Label>
  <Parser id="DecimalParser">
    <JavaScript>TBD</JavaScript>
    <RegExp>TBD</RegExp>
  </Parser>
</MOAttribute>
<MOAttribute id="immediate_release">
  <Required>false</Required>
  <Type>text</Type>
  <Default>[N]</Default>
  <Width>3</Width>
  <HelpText>Enter a boolean value of Y for yes or N for no.</HelpText>
  <Label>Immediate Release</Label>
  <Parser id="BooleanParser">
    <JavaScript>TBD</JavaScript>
    <RegExp>TBD</RegExp>
  </Parser>
</MOAttribute>
<MOAttribute id="sip_url">
  <Required>false</Required>
  <Type>text</Type>
  <Default>[null]</Default>
  <Width>32</Width>
  <HelpText>Enter at least 1 character, but not more than 32 characters.</HelpText>
  <Label>Sip Url</Label>
  <Parser id="TextParser">
```

```
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="zipcode">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>10</Width>
    <HelpText>Enter at least 1 character, but not more than 10 characters.</HelpText>
    <Label>Zipcode</Label>
    <Parser id="TextParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="pic3">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>4</Width>
    <HelpText>Enter a PIC value as four numeric characters, NPIC, or NONE.</HelpText>
    <Label>Pic3</Label>
    <Parser id="PicParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="pic2">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>4</Width>
    <HelpText>Enter a PIC value as four numeric characters, NPIC, or NONE.</HelpText>
    <Label>Pic2</Label>
    <Parser id="PicParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="privacy">
    <Required>false</Required>
    <Type>single</Type>
    <Default>[NONE]</Default>
    <Width>4</Width>
    <HelpText>Enter one of the following values: [FULL, NAME, NONE]</HelpText>
    <Label>Privacy</Label>
    <Permitted>[FULL, NAME, NONE]</Permitted>
  </MOAttribute>
  <MOAttribute id="pic1">
    <Required>false</Required>
    <Type>text</Type>
    <Default>[null]</Default>
    <Width>4</Width>
    <HelpText>Enter a PIC value as four numeric characters, NPIC, or NONE.</HelpText>
    <Label>Pic1</Label>
    <Parser id="PicParser">
      <JavaScript>TBD</JavaScript>
      <RegExp>TBD</RegExp>
    </Parser>
  </MOAttribute>
  <MOAttribute id="state">
    <Required>false</Required>
    <Type>text</Type>
```

```
      <Default>[null]</Default>
      <Width>16</Width>
      <HelpText>Enter at least 1 character, but not more than 16 characters.</HelpText>
      <Label>State</Label>
      <Parser id="TextParser">
        <JavaScript>TBD</JavaScript>
        <RegExp>TBD</RegExp>
      </Parser>
    </MOAttribute>
  </ManagedObject>
```

APPENDIX **B**

XML Test Drivers

Revised: August 21, 2008, OL-15336-02

This appendix details the XML test drivers.

CLI to SOAP/CORBA XML Transaction

The following sample test driver executes a normal CLI command, but processes it as a SOAP or CORBA XML transaction.

```
package com.sswitch.oam.drv;

import java.lang.*;
import java.io.*;
import java.util.*;
import java.text.*;
// XML Stuff
import org.apache.ecs.xml.*;
import org.apache.ecs.*;
import org.w3c.dom.*;
import org.xml.sax.*;
import org.apache.xml.serialize.*;
// BTS Utility Code objects...
import com.sswitch.oam.cad.*;
import com.sswitch.oam.xml.*;
import com.sswitch.oam.util.*;
import com.sswitch.oam.ccc.*;

/**
 * XmlCli.java
 * Copyright (c) 2002-2003, 2006 by Cisco Systems, Inc.
 * --Test driver for the XML/CORBA interface...
 *    This test driver executes a normal CLI command and processes the request
 *    as a CORBA XML transaction. The reply is then displayed. I am no as
 *    concerned with complex data show(s) as with the ability to issue
 *    provisioning commands. Note that this example can be built with the
 *    provided tool "oo-cc" this simple script creates the correct CLASSPATH
 *    and invokes the compiler with the correct options. Also, the "oo-idl"
 *    tool can be used to generate the correct IDL output.
 *
 *   @author   A. J. Blanchard
 *   @version  3.1
 *   @since    900-04.01.00
 *   @since    900-03.05.02
 *
```

```
 */

public class XmlCli {

    /*
     * Class private data
     */
    private String []                                   objArgs;
    private XMLAdapter                                  objBts;
    private String                                      objLoginName;
    protected String                                    objPassword;
    protected String                                    objSwitch;
    protected String                                    adapterType;

    /**
     * Generic Constructor for the test driver.
     */
    protected        XmlCli(String[] args)
    {
      // Initialize the BTS ORB interface object.
      try {Log log = Log.getInstance("XmlCli");} catch(Exception e){}
      objArgs = args;
      objLoginName="optiuser";        // Name to login the BTS
      objPassword="optiuser";         // Password to login the BTS
      objSwitch="BTS";         // Default switch name
      parseArgs(objArgs);
      try{
        objBts = XMLAdapterFactory.getInstance().createAdapter(adapterType,args)
;
      }catch(XMLAdapterException e){
        System.out.println("Create Adapter Failure = " + e.err_code +"\n"
                           + e.err_desc);
        System.exit(1);
      }
      return;
    }


    /**
     * This is the main method for the application.
     */
    public static void main(String[] args)
    {
      XmlCli me = new XmlCli(args);
      System.out.println("start me.go()...");
      me.go();
      System.out.println("me.go() done.");
      return;
    }



    /**
     * This is the primary execution method for the object. It performs the
     * actual request and calls for the print of the reply.
     */
    protected void       go()
    {
      //
      // Log into the target machine with generic optiuser
      //
      try {
        System.out.println("Attempt to connect with:"+objLoginName+"/"+objPasswo
rd);
```

```
    objBts.connect(objLoginName, objPassword);
    System.out.println("BTS10200 Login successful...\n");
    System.out.println("Type 'bye' or 'exit' to terminate the program.\n");
    }
  catch (XMLAdapterException e) {
    System.out.println("XMLAdapterException in login = " + e.err_code +"\n"
                       + e.err_desc);
    System.exit(1);
    }
  catch (Exception e) {
    System.out.println("Exception at login = " + e.toString());
    System.exit(2);
    }
  //
  // Read in the file and send request...
  //
  while(true)
      {
      try {
        openCLI();                      // Put out the prompt...
        CommandParser parser = new CommandParser();
        String cmd = readCLI().trim(); // Fetch the request file...
        String reply = "";
        if((cmd.equals("")))
           continue;
        if((cmd.equals("exit"))||(cmd.equals("bye"))||(cmd.equals("quit")))
            break;
        else
          {
          // Issue request to BTS 10200
          reply = objBts.request(parser.toXML(cmd));
          System.out.println("RETURN VALUE: ");
          //parser.prettyPrint(reply);
          System.out.println(printResult(reply));
          }
        }
    catch (XMLAdapterException ce) {
        System.out.println("CIS Command Exception: CODE="+ce.err_code +
                           "\n"+ce.err_desc);
        }
    catch (Exception e) {
        Log.fatal("General Command Exception:"+
                  Util.stackTraceToString(e));
        System.out.println("General Command Exception:");
        e.printStackTrace();
        break;
        }
    } // end while(1)
  closeCLI();
  // Clean up and logout
  try {objBts.disconnect();} catch (XMLAdapterException cad) {}

  return;
} // end go()



/**
 * Open the input file for reading. This allows the read method to suck
 * a line at a time of the CLI style input.
 */
protected void        openCLI()
{
  System.out.print(objLoginName+"@"+objSwitch+"> ");
```

```
  return;
}

/**
 * This is the method that closes and clean up after a file has been
 * processed.
 */
protected void        closeCLI()
{
  System.out.println("\n Bye...");
  return;
}

/**
 * Read in the file provided as the request. Just exit on errors. Don't
 * worry about throwing an error exception.
 */
protected String      readCLI()  throws java.io.IOException
{
  int          temp=0;
  int          idx=0;
  byte []      buf= new byte[256];

  while(true)
      {
      temp = System.in.read();
      if(temp==10)           // <ENTER Key>
        break;
      buf[idx++]=(byte) temp;
      }
  return new String(buf);
}

//=====================================================================
// Tools and utilities...
//=====================================================================

/**
 * This method reads in the startup arguments for the HUB process
 * The only presently supported argument is the HUB slave or master
 * mode assignment. Master is the default. It does not reach out to find
 * the slave HUBs.
 *
 * @param  args  The list of startup arguments
 */
protected void   parseArgs(String[]  args)
{
int  lp=0;

  if(args==null)
     return;

  try {
    for(lp=0;lp < args.length;lp++)
      {
      if(args[lp].startsWith("-n"))
         objLoginName=args[(lp+1)];

      if(args[lp].startsWith("-p"))
         objPassword=args[(lp+1)];

      if(args[lp].startsWith("-b"))
         objSwitch=args[(lp+1)];
```

```
      if(args[lp].startsWith("-t"))
         adapterType = args[++lp].toUpperCase();
      }
    }
  catch (java.lang.NumberFormatException nfe) {
    Log.fatal("Error in args: "+args [lp]+args [lp+1]+"\n"+
              Util.stackTraceToString(nfe));
    System.exit(1);
    }
  catch (java.lang.ArrayIndexOutOfBoundsException e) {
    return;
    }
  return;
}

/**
 * This is the reply data table parser
 */
public String   printResult(String reply)
{

String  output="";
int     size = 0;
HashMap table = null;

try {
  XMLReply parseResult = new XMLReply(reply);
  if(parseResult.getStatus().equals("true"))
     {
     output+="SUCCESS:";
     size = parseResult.getSize();
     }
  else
     output+="FAIL:";
  String temp = parseResult.getReason();
  temp+="\n";
  if((table=parseResult.getReplyData())!=null)
     temp += buildText(parseResult.getReplyData(),0);
  temp += "================================================\n";
  output+=temp;
  for(int lp=1; lp < size; lp++)
    {
    output += buildText(parseResult.getReplyData(),lp);
    output +="================================================\n";
    }

  }
catch (Exception e) {
  System.out.println("Reply Exception = " +e.toString());
  Log.error("Error parsing reply \n"+reply+"\n"+
            Util.stackTraceToString(e));
  }
return output;
} // end printResult

/**
 * This is the text builder for row level data in a reply table.
 */
public String   buildText(HashMap table, int rowNum)
{
  String  data="";
  HashMap rowSet=(HashMap) table.get(Integer.toString(rowNum));

  if(rowSet==null)
```

```
            return "No Data available at row "+rowNum;

          Iterator list = rowSet.keySet().iterator();
          for(;list.hasNext();)
            {
            String key = (String) list.next();
            String value = (String) rowSet.get( key);
            data+=key+"="+value+"\n";
            }
          return data;
        }

} // end XmlCli
```