

# MiCollab Advanced Messaging 9.3 Web Services API Developer Guide Developer Resources Document

For version 9.3 and above

## Notice

The information contained in this document is believed to be accurate in all respects but is not warranted by Mitel Networks™ Corporation (MITEL®). Mitel makes no warranty of any kind with regards to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The information is subject to change without notice and should not be construed in any way as a commitment by Mitel or any of its affiliates or subsidiaries. Mitel and its affiliates and subsidiaries assume no responsibility for any errors or omissions in this document. Revisions of this document or new editions of it may be issued to incorporate such changes.

No part of this document can be reproduced or transmitted in any form or by any means - electronic or mechanical - for any purpose without written permission from Mitel Networks Corporation.

## Trademarks

The trademarks, service marks, logos and graphics (collectively "Trademarks") appearing on Mitel's Internet sites or in its publications are registered and unregistered trademarks of Mitel Networks Corporation (MNC) or its subsidiaries (collectively "Mitel") or others. Use of the Trademarks is prohibited without the express consent from Mitel. Please contact our legal department at [legal@mitel.com](mailto:legal@mitel.com) for additional information. For a list of the worldwide Mitel Networks Corporation registered trademarks, please refer to the website: <http://www.mitel.com/trademarks>.

© Copyright 2022, Mitel Networks Corporation

All rights reserved

# License Agreement

## Mitel Networks Coporation LICENSE AGREEMENT

### READ CAREFULLY BEFORE USING MiCollab Advanced Messaging 9.3

BY USING THE MiCollab AM SOFTWARE, CUSTOMER ACCEPTS THE TERMS OF THIS LICENSE AGREEMENT. IF CUSTOMER DOES NOT AGREE TO THIS LICENSE AGREEMENT, PROMPTLY RETURN THE MiCollab AM SOFTWARE, DOCUMENTATION AND ALL ITEMS SUPPLIED WITH IT TO THE PERSON WHO SUPPLIED THEM TO YOU. YOUR MONEY WILL BE REFUNDED IN FULL.

#### 1. Definitions

**Software.** *Software* means the computer software and accompanying documentation known as MiCollab AM, and all additions, corrections, updates, and enhancements provided under this agreement, together with copies of such items

#### 2. License.

Mitel Networks Coporation (*Mitel*) grants Customer a nonexclusive, nontransferable license to use the Software in accordance with terms of this agreement. Mitel remains the owner of the Software and of any copy of the Software which Customer may make.

#### 3. Proprietary Rights.

The Software is the valuable trade secret property of Mitel and is protected under U.S. copyright laws and international treaties. UNAUTHORIZED COPYING, USE, OR DISCLOSURE OF THE SOFTWARE IS AN INFRINGEMENT OF Mitel's COPYRIGHT, TRADE SECRET, AND OTHER INTELLECTUAL PROPERTY RIGHTS.

#### 4. Use Restrictions.

4.1 Customer may make copies of the Software for its own internal use on the number of workstations licensed to Customer and as a backup; provided that Customer (a) includes in and on such copies all notices of copyright and proprietary rights appearing in and on the Software, and (b) erases or destroys such copies when they are no longer required. Customer will not permit anyone else to copy any portion of the Software. Customer may not copy the written materials accompanying the Software.

4.2 Customer shall not disassemble, reverse engineer, or decode any portion of the Software or attempt to bypass the metering device contained in the Software. Except as expressly set forth in this Agreement, Customer shall not modify any portion of the Software, or merge or embed the Software into another computer program.

#### 5. Distribution.

Customer is licensed to create derivatives of the MiCollab AM Web Services API and to incorporate such derivatives of the MiCollab AM Web Services API into Customer's applications. However, Customer has no right to distribute any components of the MiCollab AM Web Services API except as incorporated into Customer's application as provided in the preceding sentence. Customer may use files included in the

MiCollab AM Web Services API to build Customer's applications. However, Customer has no right to distribute any of such files or any related documentation. Customer may not distribute the Software as a whole or distribute a developer's kit created using the Software.

## **6. Limited Warranty and Disclaimer of Implied Warranties.**

6.1 Mitel warrants that, upon delivery by Mitel or Mitel's Vendor to you, the Software will perform substantially in accordance with the specifications stated in the accompanying written materials. Mitel does not warrant that the Software is free from all bugs, errors, and omissions. This Limited Warranty is void if the failure of the Software has resulted from your improper use or misapplication of it.

6.2 If the Software does not comply with the Limited Warranty, Mitel will use reasonable efforts to correct the noncompliance either by, at Mitel's option: (a) repair or replacement of the Software, or (b) return of a fair portion of the price paid. You must return all copies of the Software to Mitel or Mitel's vendor with a copy of your receipt within 90 days of the date you received the Software to receive this remedy. Any replacement Software will be warranted for the remainder of the original 90 day warranty period or 30 days from the date you received the replacement, whichever is longer.

6.3 THE WARRANTY PROVIDED UNDER SECTION 6.1 OF THIS LIMITED WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED. Mitel MAKES NO OTHER WARRANTIES, EXPRESS OR IMPLIED. Mitel DISCLAIMS ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT.

## **7. Term and Termination**

7.1 Customer may terminate the license at any time.

7.2 If Customer fails to comply with any provision of this agreement, Mitel may, in addition to all other remedies available, terminate the license by notice to Customer. Upon termination of the license, Customer shall cease using the Software. Customer's obligations to hold the Software in confidence and all related obligations shall survive termination of the license.

## **8. Assignment.**

This Software is licensed only to Customer. Customer shall not transfer the Software or this license to anyone. In no event may Customer transfer, assign, rent, lease, sell, or otherwise dispose of the Software on a temporary or permanent basis.

## **9. Miscellaneous**

9.1 Mitel's liability (whether in tort, contract, or otherwise and notwithstanding any fault, negligence, product liability, or strict liability of Mitel) under this agreement or with regard to any Software or documentation will in no event exceed the amount paid by Customer to Mitel for the Software. IN NO EVENT WILL MITEL BE LIABLE (WHETHER IN CONTRACT, WARRANTY, TORT, PRODUCT LIABILITY, STRICT LIABILITY OR OTHERWISE) TO CUSTOMER OR ANY OTHER PERSON FOR ANY INDIRECT, INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES.

9.2 This agreement will be governed by the laws of the State of Washington. Customer consents to the nonexclusive jurisdiction of the federal and state courts in Seattle, Washington, in any action arising out of or in connection with this agreement. No agent, employee, or representative of Mitel has any authority to bind Mitel to any affirmation, representation, or warranty, and, unless such is specifically included within

this written agreement, it shall not be enforceable by Customer. If any provision of this agreement is held by a court of competent jurisdiction to be invalid, illegal, or unenforceable, the remainder of this agreement shall remain in full force and effect.

#### U.S. GOVERNMENT RESTRICTED RIGHTS LEGEND

The Software and documentation are provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subdivision (c)(1)(II) of The Rights in Technical Data and Computer Software clause at 48 C.F.R. Section 252.227-7013 or in subparagraphs (c)(1) and (2) of the Commercial Computer Software Restricted Rights clause at C.F.R. Section 52.227-19, as applicable. Contractor/manufacturer is Mitel Networks Coporation.

# Contents

<b>License Agreement</b>	<b>3</b>
<b>Preface</b>	<b>8</b>
References	8
Documentation	8
Documentation Updates	9
Help	9
Document Conventions	9
<b>Getting Started</b>	<b>11</b>
MiCollab AM Web Services API Toolkit	11
Requirements	11
Programming Model	11
Installing and Using the Demonstration Application	12
Server	12
SessionID	12
SessionInstanceID	13
Functions	13
Parameters	13
Reload Functions Button	13
Invoke Button	13
Results Tab	13
Full Response Tab	13
Full Request Tab	13
Starting a New Session	13
Distributing Your Application	14
<b>Using the MiCollab AM Web Services API</b>	<b>15</b>
Calling CXProcessXML	15
Notes on HTTP headers	17
Logging On	18
Session Identification	18
Logon Privilege Levels and Application Type	19

MiCollab AM Objects	19
Mailbox Configuration Access	19
Message Access	19
Line Management	20
Miscellaneous Functions	20
The Update Operation	20
XML Conventions	21
Empty or Missing tags	21
Boolean Values	21
Timestamp Values	21
<b>Exceptions and Error Returns</b>	<b>22</b>
Generic MiCollab AM Application Errors	22
<b>Creating and Using Mailbox Recordings</b>	<b>23</b>
Overview	23
Recording, Saving, and Sending a New Voice Message	24
Playing a Mailbox Recording	24
Creating a New Mailbox Recording	25
<b>MiCollab AM Web Services API WSDL</b>	<b>26</b>

# Preface

The Web Services API Developer's Guide provides the information you need to develop applications that integrate with MiCollab Advanced Messaging (MiCollab AM), using the MiCollab AM Simple Object Access Protocol (SOAP) Application Programming Interface (API). The MiCollab AM Web Services API is an open developer interface to the MiCollab AM voice and call-processing platform.

For a detailed description of each individual function that is part of this API, including the function syntax, its parameters, return values and most common error codes, please refer to the document titled *MiCollab AM Web Services API Functions Reference*.

The MiCollab AM Web Services API provides access to MiCollab AM for the purpose of administering subscriber mailboxes and managing messages and recording objects. Through the MiCollab AM Web Services API, you can integrate MiCollab AM capabilities into your own applications. It provides a SOAP web services interface that provides the following services:

- Log on and off of a MiCollab AM mailbox or administrator account.
- Access to MiCollab AM objects via Add, Search & Get, Update and Delete operations. These include:
  - Access to MiCollab AM Mailbox settings
  - Access to Mailbox related structures such as Devices, Speech Alias, Availability
  - Access to sub-arrays such as Member Array of a Distribution List mailbox
- Message access including ability to send new messages and Forward/Reply to existing messages.
- Ability to place calls and record and play messages and recording objects.
- Other specific operations such as renumbering mailboxes and changing passwords.

## References

A catalog of technical documentation is included on the MiCollab AM Installation Media. If you are installing any advanced applications, such as Networking and Fax Server applications, you should refer to the appropriate technical documentation for application and installation information.

## Documentation

The technical documentation is produced in the PDF format and requires the PDF reader to view it. The documentation set for this MiCollab AM includes the following documents and resources:

- **Developer Resources.** Contains programming guides and API references for developers for integrating the server clients and web applications with MiCollab AM.
- **Integration Technical Notes (ITN).** Contains a set of guides that describe the integration methods and instructions for a variety of phone systems to work with MiCollab AM. The ITNs are generally used by resellers or administrators who are experienced with MiCollab AM and familiar with the integration procedures and terminology.

- **Quick Reference Card (QRC).** Contains shortcuts and quick instructions telling subscribers how to access and use the messaging system.
- **Server Documentation.** Available as a PDF only. Contains administrative guides for administrators about installing, configuring, and administering the messaging system, and user guides for subscribers about accessing the messaging system and checking and sending messages.
- **Spare Parts Documentation.** Contains a set of guides that describe the instructions for installing and configuring hardware parts to work with MiCollab AM. These documents are written for Mitel certified MiCollab AM technicians who are experienced with MiCollab AM and familiar with the procedures and terminology.
- **Software Release Notice (SRN).** This notice introduces the new features, capabilities, and hardware/software requirements for the corresponding MiCollab AM version.

## Documentation Updates

Documentation updates may be available from the following sources:

- Mitel certified technicians can view or download documents and program files from our partner web site: [connect.mitel.com/connect](http://connect.mitel.com/connect)

## Help

The primary source of information about MiCollab AM is the online help available within any of its administrative utilities. You can access **Help** as follows:

- Click the **Help** button in the dialog box or window in which you are working
- Press the **F1** key at any time.

## Document Conventions

The following conventions are used in this document:

- **Key Names.** Names of keys on the keyboard are shown in a box.

Example: **Enter**

When two keys must be pressed simultaneously, they are joined by a + sign.

Example: **Alt** + **Tab**

- **Reference to Document.** *Italics* fonts can also signify the titles of other documents.

Example: See the *System Installation and Configuration Guide*.

- **UI Element Names.** Names of UI elements such as dialog windows, screens, menu items, tabs, buttons, icons, etc. are shown in bold.

Example: On the **Startup** screen, click the **Start** icon.

- **User Input.** Information required to be typed is shown in italics.

Example: Type the password *voicemail*.

- **Warning, Caution, Important, and Notes.** Text for the contents that require attention are shown as follows:

**WARNING** A warning paragraph advises you of circumstances that can result in the loss of data, harm to the system server platform, or personal harm.

**CAUTION** Failure to follow these recommendations can result in unauthorized access to the system and consequent loss of data.

**IMPORTANT** An important paragraph gives decision-making information or informs you of the order in which tasks need to be completed.

**NOTE** A note gives additional information, provides an explanation, or indicates an exception to the information in the preceding text.

# Getting Started

This chapter describes the system requirements for using the MiCollab AM Web Services API, and explains how to install and use the API test application.

## MiCollab AM Web Services API Toolkit

- The MiCollab AM Web Services API toolkit contains the following:
- Web Services API Developer's Guide (this document)
- Web Services API Functions Reference
- CXIF.XML - WSDL file defining the MiCollab AM Web Services API function
- Demonstration program which provides a GUI interface through which all Web Services API functions may be tested (includes CXSoapDemo.jar, saaj-api.jar, and saaj-impl.jar)
- CXSoapRequests.xml – contains sample XML for all Web Services API functions

## Requirements

MiCollab AM Web Services API allows you to write a variety of applications. The following hardware and software are required to use the MiCollab AM Web Services API.

- A MiCollab AM server software license
- Applications developed using the MiCollab AM Web Services API may be run locally on the MiCollab AM server or from another system which has access to the MiCollab AM server across a TCP/IP network

Developers may write applications using the MiCollab AM Web Services API using any application development language that supports SOAP development.

This manual assumes that the developer is proficient in the use of SOAP and XML within their language of choice.

## Programming Model

The base programming model for the MiCollab AM Web Services API is, naturally, SOAP. As such, you will usually generate a proxy method in your language using the CXIF.XML WSDL. The Web Services API WSDL defines a single SOAP method – **CXProcessXML**. All the MiCollab AM Web Services API functions are invoked via this single method. This is done by passing parameters within an XML request body. Results are returned through an XML response body.

# Installing and Using the Demonstration Application

The SOAP Demonstration program CXSOAPDemo is written in Java and requires that the Java virtual machine version 6 update 1 or higher be installed in order to run.

Copy the CXSOAPDemo files onto your system and execute the CXSOAPDemo.jar file. CXSOAPDemo presents the GUI shown in Figure 1.

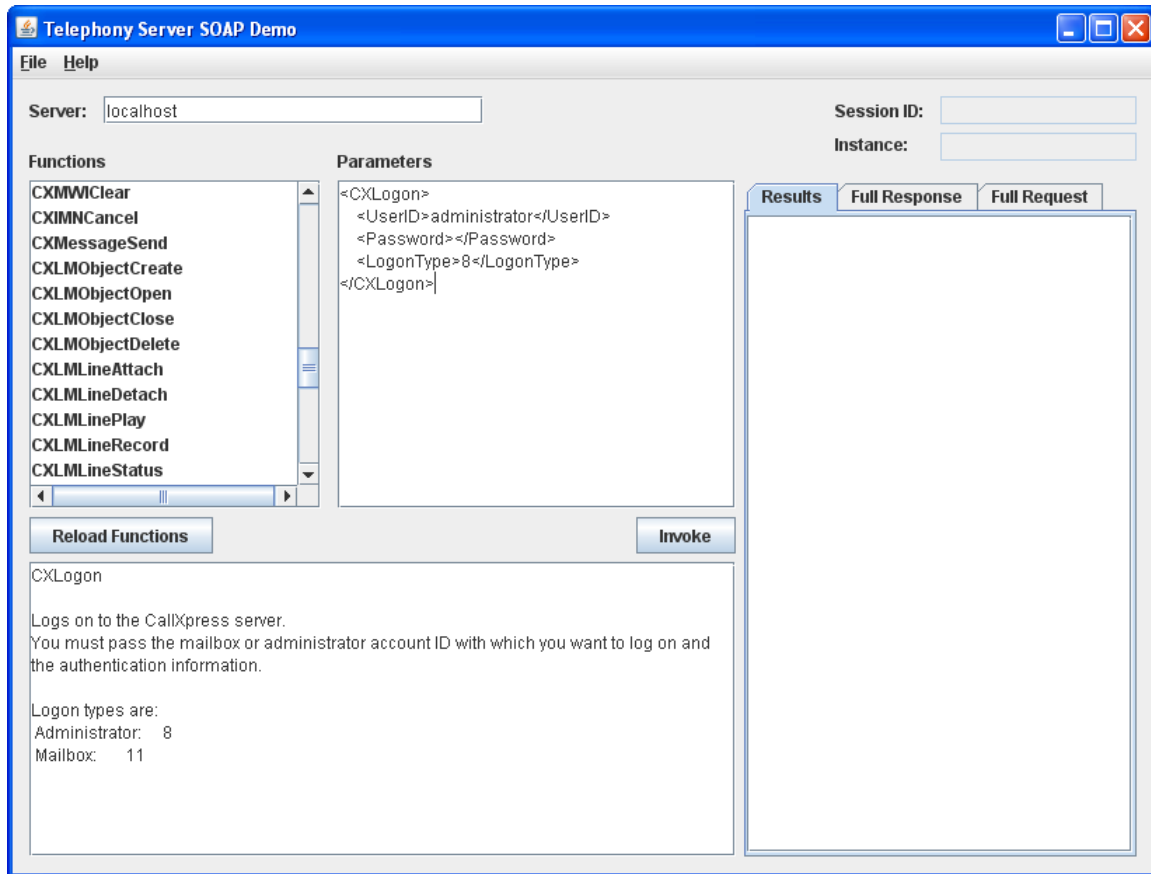


Figure 1. MiCollab AM SOAP Demo GUI

CXSOAPDemo allows you to test all functions provided by the API. The controls of the demo application are described below:

## Server

The URL of the MiCollab AM server. The URL should be a server name or IP address. The URL can begin with the http:// prefix. However, if the prefix is not specified, http:// is implied.

## SessionID

Displays the first part of the unique identifier for the current session. This value is set after a call to the **CXLogon** function and all subsequent calls must include this identifier.

## SessionInstanceID

Displays the second part of the unique identifier for the current session. This value is set after a call to the **CXLogon** function and all subsequent calls must include this identifier. All sessions created on the same server have the same **SessionInstanceID**.

## Functions

Lists all functions available in the demo application. Selecting a function causes the application to display an XML template of the function in the Parameters control and also display a description of the function in the bottom text pane.

## Parameters

Contains a template of the XML for the selected function. Prior to invoking the function, edit the parameter values within the XML as appropriate to the request being made.

## Reload Functions Button

Causes the contents of the CXSoapRequests.xml file to be reloaded in the program. The CXSoapRequests.xml file may be edited to provide any desirable preset parameter values within the XML function definitions.

## Invoke Button

Causes the XML within the Parameters pane to be packaged within a call to the CXProcessXML SOAP method. Results of the function are displayed in the Results pane.

## Results Tab

Displays the XML response returned by the MiCollab AM server.

## Full Response Tab

Displays the complete SOAP response, including all of the SOAP wrappers.

## Full Request Tab

Displays the complete SOAP request, including all of the SOAP wrappers.

## Starting a New Session

A Web Services session must always begin with a call to the **CXLogon** function. The logon call establishes a unique session with the MiCollab AM server and authenticates the application user for access to the MiCollab AM server. State is maintained independently for each unique session with the MiCollab AM

server. The access rights of the authenticated user are established and enforced throughout the session. Additionally, the session maintains state information across certain functions. For example, the **CXLMLinePlay** function uses the state information that was established during a previous call to **CXLMLineAttach**.

The two values **SessionID** and **SessionInstanceID** are the mechanism used by the Server to identify the session with which a request is associated. These are returned in the response to a successful call to the **CXLogon** function and must be provided in all other Web Services API function calls with the exception of **CXGetVersionInfo**.

In general, *CXSOAPDemo* simply packages the request XML that has been entered in the Parameters pane into a SOAP call to the **CXProcessXML** method. However, *CXSOAPDemo* does parse the **SessionID** and **SessionInstanceID** out of the XML return from a **CXLogon** request for display within the Session ID and Instance controls respectively. Additionally, when functions other than **CXLogon** are selected, the values of the Session ID and Instance controls are inserted into the sample XML request displayed in the Parameters pane. This simplifies use of the *CXSOAPDemo*, since the user does not have to place a valid **SessionID** and **SessionInstanceID** in the XML request for each function that is tested.

## Distributing Your Application

You are granted an unlimited license to distribute applications that you create using the MiCollab AM Web Services API.

You are not licensed to distribute any of the components of the MiCollab AM Web Services API package; however, you may distribute your own works which contain derivatives of the API package, such as a proxy method derived from the WSDL.

You may use the files included in the MiCollab AM Web Services API to build your applications. However, you may not distribute any of these files or any accompanying documentation.

# Using the MiCollab AM Web Services API

Using the MiCollab AM Web Services API is similar to other SOAP based client/server application development. You will implement applications which use the MiCollab AM Web Services API using the normal SOAP/XML programming conventions for your language of choice. Typically, you will generate a SOAP proxy method for the **CXProcessXML** SOAP method and use an XML writer and parser that are available in your language.

The MiCollab AM SOAP server listens for unencrypted SOAP calls on port 18276 and encrypted (SSL) SOAP calls on port 18277. Be sure your application does not attempt to connect to the MiCollab AM SOAP on the standard HTTP port 80.

Please note that SSL is used only for encryption of the transport. The SSL certificate used by the SOAP server is self-signed by default and so your application would have to either trust this certificate or turn off certificate validation so that the client-side SSL handshake completes successfully. You can also import your own certificate that is to be used for this purpose.

The SOAP SSL certificate is now stored in the windows certificate store. The process of importing the certificate hasn't changed. However, it is important to understand that the SOAP Server is referencing the certificate to be used from the certificate store. You can run the following netsh command to see the properties of the certificate that is associated with the SOAP SSL port (which is 18277).

**netsh http show sslcert ipport=0.0.0.0:18277**

If traversing a firewall, make sure that the ports 18276 and 18277 are open from the client to the MiCollab AM server. You can determine whether IP packets can be routed from your client machine to the MiCollab AM server by entering the command `netstat -a` in a command window. If the MiCollab AM server is running on your computer, you should see something like the following, where computer is the name of your computer and computerfull is the fully-qualified network name of your computer:

Active Connections

Proto	Local Address	Foreign Address	State
...			
TCP	computer:18276	computerfull:0	LISTENING
TCP	computer:18277	computerfull:0	LISTENING
...			

## Calling CXProcessXML

The single generic **CXProcessXML** SOAP method is to be called for invoking any and all kinds of service functions. The XML that is part of this request specifies the actual function that needs to be invoked. Thus, in the request below, **CXLogon** is specified as the function to be invoked.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <CXProcessXML xmlns="http://www.avstgroup.com/CXIf">
```

```

        <XMLRequestData>
          <CXLogon>
            <UserID>administrator</UserID>
            <Password></Password>
            <LogonType>8</LogonType>
          </CXLogon>
        </XMLRequestData>
      </CXProcessXML>
    </s:Body>
  </s:Envelope>

```

Within the request XML, there is always an **XMLRequestData** element. This **XMLRequestData** element contains a single child element whose name is the name of the function to be invoked and whose direct child-elements are named-parameters to the function. This is the Parameter List structure. This structure is used throughout the MiCollab AM SOAP interface for all functions.

In the above example, **UserID**, **Password** and **LogonType** are input parameters to the **CXLogon** function. These are all simple parameters with the parameter-value specified as the content of the Parameter XML element. A parameter can also be a complex XML structure such as **MB**, **StoredMessage** etc. In this case, the Parameter may be viewed as an xml tree-structure with the name of the Parameter being the top-level element's tag name.

Assuming that the above **CXLogon** function succeeds, the server will send a generic response XML that looks like this:

```

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <CXProcessXMLResponse xmlns="http://www.avstgroup.com/CXIf">
      <XMLResponse>
        <CXResponse>
          <Completed>1</Completed>
          <SessionID>2060976129</SessionID>
          <SessionInstanceID>216f14ac-84fd-4ae8-91e1-
e43757361bd6</SessionInstanceID>
        </CXResponse>
      </XMLResponse>
    </CXProcessXMLResponse>
  </s:Body>
</s:Envelope>

```

Within the response XML, the sub-structure starting from the **CXResponse** element is a Parameter List structure. The **Completed**, **SessionID** and **SessionInstanceID** are thus the named output parameters that are part of this response. All the output parameters of the **CXLogon** function thus happen to be simple parameters.

If however the **CXLogon** function call fails, the server will return an error response using an xml that looks like this:

```

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <s:Fault>
      <faultcode>s:Server</faultcode>
      <faultstring xml:lang="en-US">Invalid credentials</faultstring>
    </s:Fault>
  </s:Body>
</s:Envelope>

```

```

        <detail>
            <CXFault xmlns="http://www.avstgroup.com/CXIf">
                <Error>2147749809</Error>
                <Description>Invalid credentials</Description>
            </CXFault>
        </detail>
    </s:Fault>
</s:Body>
</s:Envelope>

```

Starting from the **Fault** element is the standard SOAP fault structure. The **faultstring** SOAP element contains the error string. The **detail** SOAP element holds the **CXFault** structure. **CXFault** is the custom structure that the Server sends as part of the error response. Within **CXFault**, the **Error** element contains the Error code and the **Description** element contains the error string.

Please note that all elements starting from the **CXProcessXML** element in the request XML, all elements starting from the **CXProcessXMLResponse** element in the response XML and all elements starting from the **CXFault** element in the error response are namespace qualified with the **http://www.avstgroup.com/CXIf** namespace. Thus, these elements and all their child elements and any nested child elements, all belong to this namespace. Note, however, that when generating an XML, there are alternate ways of specifying this same namespace rule. So for example, the request XML shown previously could also be specified like the one below and this would also be equally acceptable since it essentially means the same thing:

```

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
    <s:Body>
        <CX:CXProcessXML xmlns:CX="http://www.avstgroup.com/CXIf">
            <CX:XMLRequestData>
                <CX:CXLogon>
                    <CX:UserID>administrator</CX:UserID>
                    <CX:Password></CX:Password>
                    <CX:LogonType>8</CX:LogonType>
                </CX:CXLogon>
            </CX:XMLRequestData>
        </CX:CXProcessXML>
    </s:Body>
</s:Envelope>

```

## Notes on HTTP headers

We have changed the MiCollab AM SOAP implementation from gSOAP-based to WWS API-based. While making this change, we have kept the interface the same. This means that the WSDL for the MiCollab AM SOAP Server hasn't changed at all. So, the expectation is that clients should be able to talk to the new SOAP Server just like they were talking to the previous gSOAP-based SOAP Server. Even though it has been our goal to keep the interface the same, the reality is that the two implementations do differ in some respects. So, while the SOAP payload itself hasn't changed, we have found that some apps may require minor changes just due to some differences in HTTP headers that need to be set carefully. There are two headers that need particular attention:

1. **SOAPAction header is now mandatory**

Our prior gSOAP-based implementation would work even when this HTTP header is not specified in the HTTP request that is sent to the server. However, in order for the requests to be processed properly, the new implementation does require this header. Here is the right way to set this header:

SOAPAction: ""

2. **User-Agent header must not contain the word "Mozilla"**

This is most definitely a defect on the WWS side. If you are seeing the token "Mozilla" being sent to the SOAPServer and you are not getting a proper response from the SOAPServer for your request then you are most likely running into this defect. The easy thing to do is to set the User-Agent header to a string that is appropriate to your app. And, yes, do not include the token "Mozilla" in it.

In addition to the above headers, it is also required to make sure that the namespaces are specified correctly. The WWS-based implementation is very strict and consistent when it comes to the validation and enforcement of namespaces.

## Logging On

MiCollab AM requires a logon prior to using any API function. A logon does the following:

- Authenticates the ID and password provided
- Establishes a unique session with the MiCollab AM server. This allows the server to store any state information associated with this session.
- Establishes the privileges based on the identification provided and the rights that have been granted to the ID

An application uses the **CXLogon** and **CXLogoff** functions to log on and off of the MiCollab AM system for both subscriber mailbox and administrator accounts.

MiCollab AM allows multiple simultaneous logons to the same mailbox and also allows multiple administrators to post updates to a specific mailbox. In a multiple update situation, the last update posted supersedes any prior update. In most cases, this is not an issue since the Web Services API allows the updating of specific fields rather than a full replacement of an entire record.

For administrator accounts, MiCollab AM allows for multiple, simultaneous logons up to the limit set for the administrator account, which may be unlimited.

Any session which remains logged on for more than 30 minutes without any activity will be automatically logged off. This allows the SOAP server to clean up abandoned sessions.

## Session Identification

An API session is identified by two properties; **SessionID** and **SessionInstanceID**. The values for the two session identifiers are returned in the results of **CXLogon**. These two session identifiers must be used with

all other API functions. Please note that **CXGetVersionInfo** is the only function that can be invoked prior to establishing the session.

## Logon Privilege Levels and Application Type

Two classes of applications, administrative and subscriber, may be written using the API. The type of logon used determines the class of application. Administrative applications that are designed to administer multiple mailboxes must use an administrator account logon. Subscriber mailbox logons should be used by applications that are designed for individual users to edit their mailbox settings, send and receive messages and make calls.

## MiCollab AM Objects

The MiCollab AM Web Services API functions can be thought of as a set of tools to make use of objects residing on the MiCollab AM server. Please refer to the MiCollab AM Web Services API Functions Reference document for a detailed description of the objects and the operations that can be performed on them.

## Mailbox Configuration Access

A MiCollab AM mailbox can be viewed as an entity within the MiCollab AM directory. Many MiCollab AM API functions require the session to be logged on using an administrator account with specific access rights before some of these settings can be changed.

**CXMBXXX** functions allow mailbox configuration settings to be retrieved and changed.

**CXPromptRecordingXXX** functions allow mailbox recording object IDs to be retrieved and changed.

**CXObjectXXX** functions allow mailbox recording objects to be retrieved and changed.

**CXAvailabilityXXX** functions relate to management of a mailbox user's Availability settings.

**CXDevRecXXX** functions relate to management of a mailbox user's telephone devices.

**CXMBGroupXXX** functions relate to the association of subscriber mailboxes with organizational groupings such as departments and locations.

**CXMBSpeechAliasXXX** functions allow management of speech aliases which are used to identify different ways that a subscriber's name may be spoken, such as a nick name.

## Message Access

Functions in the category **CXMessageXXX** allow messages to be retrieved and modified and new messages to be sent. Most Messaging functions are only available when the session is logged on using a Subscriber mailbox.

## Line Management

Functions in the category **CXLMXXX** support placing calls from the MiCollab AM server to a designated telephone for the purpose of playing and recording messages, greetings and spoken names. Once connected, a call may also be transferred to another telephone.

## Miscellaneous Functions

Several MiCollab AM Web Services API functions provide access to different configuration information or expose special operations that are not covered elsewhere.

## The Update Operation

For the Update operations, in addition to supplying the object's XML with the modified values, you can also supply the *Old Object*. This lets the server know the old state of the Object prior to the edit. This information is used by the server to minimize the database changes required and to minimize the possibility of inadvertently overwriting a field that was just updated by some other user. It is recommended that this parameter be specified wherever possible.

Also note that for the **ListUpdate** or **ArrayUpdate** type of functions, this is the only way to indicate member additions and deletions to the server. To perform deletion of a member, you would put the member in the old list but not put it in the new list. To add a new member, you would do the opposite, that is put it in the new list but keep it missing in the old list. Note that this rule is automatically followed if the application first reads the data from the server, then caches the read data and then makes the changes on a separate copy of the object. The old object in this case is simply the original cached object.

The old object is specified using the **OldObject** parameter and it always has the same structure as the object being updated. An example xml follows:

```
<CXMBSpeechAliasUpdate>
  <SessionID>333775392</SessionID>
  <SessionInstanceID>821d35ef-ee1a-414e-b414-9690cc240750</SessionInstanceID>
  <MBSpeechAlias>
    <ID>26</ID>
    <MBID>59003</MBID>
    <Phrase>Test Two</Phrase>
    <IsPrimaryAlias>1</IsPrimaryAlias>
  </MBSpeechAlias>
  <OldObject>
    <MBSpeechAlias>
      <ID>26</ID>
      <MBID>59003</MBID>
      <Phrase>Test One</Phrase>
      <IsPrimaryAlias>1</IsPrimaryAlias>
    </MBSpeechAlias>
  </OldObject>
</CXMBSpeechAliasUpdate>
```

# XML Conventions

## Empty or Missing tags

Generally speaking, no distinction is made between these two cases and the function implementation will always try to pick a default value. However, if the value is mandatory, an error will be returned.

## Boolean Values

Boolean values are represented as a 0 = False and 1 = True.

## Timestamp Values

Timestamps conform to the **dateTime** data type defined by XML schema. The format of **dateTime** is as shown below:

YYYY-MM-DDTHH:MM:SSZ

Please note that all timestamp values, unless indicated specifically, are time values in UTC. This is indicated by the Z (for Zulu) at the end of the format. Having times in UTC, allows the application to adjust the time as necessary based on the time-zone that it is currently using. This also allows the server to maintain a value that is kind of the *absolute* value. Here is an example timestamp:

2011-01-24T19:23:28Z

# Exceptions and Error Returns

There are two classes of errors that may occur in your use of the MiCollab AM Web Services API. SOAP errors or exceptions that occur outside of the actual execution of an XML request on the MiCollab AM server result in exception handling native to your programming language. An example of this type of exception is an attempt to access a MiCollab AM server using an incorrect URL or when MiCollab AM is not running. The other class of errors has errors specific to the MiCollab AM application and data structures. Examples of this class of error include conditions such as: invalid session, object does not exist, invalid request and invalid mailbox. These errors are returned with an error code and description in the XML format shown previously in the section titled [Calling CXProcessXML](#).

Each MiCollab AM Web Service API topic contains a list of the most common errors as follows:

- Error value            Description text

Followed by a brief explanation of how the error occurred.

## Generic MiCollab AM Application Errors

The following errors can be encountered in almost every MiCollab AM API call:

- 2147749804            Session is Invalid

The SessionID provided is invalid, the SessionInstanceID provided is invalid, the session has been previously logged off, or the MiCollab AM server has timed out of the session.

- 2147749305            Invalid Argument

One or more of the function parameters are not correct or have not been supplied.

- 2147749306            Invalid Request Name

A function with this name does not exist.

- 2147749319            Method not supported

This operation is not supported.

- 2147750405            ECM Error: Node not reachable

The remote node (System Server or Call Server) is not reachable. This is only returned from a function that depends on the connectivity to the remote node to perform its operation.

- 2147750611            Data-layer error: Insufficient privileges

The specified operation could not be completed because the logged-on user has insufficient privileges.

# Creating and Using Mailbox Recordings

This section describes how to use the line management functions to create and play mailbox recordings and messages.

## Overview

Use the line management functions to enable an external client to place a telephone call to an arbitrary telephone to connect with a user. Use the recording object functions to create and play various types of MiCollab AM recording objects such as messages, greetings and spoken names.

Call the **CXMPromptRecordingListGet** function to get the identifiers for spoken names and greetings associated with MiCollab AM mailboxes. Call the **CXMessageSearch** function to get the identifiers for the messages. Once a message identifier is retrieved, the message can be immediately played by referencing the message identifier in a call to **CXLMLinePlay**.

Call the **CXLMLObjectOpen** function to open an existing recording object so that it can be played. Call **CXLMLObjectCreate** to create a new recording object for purposes of creating a new recording object.

Call the **CXLMLLinePlay** or **CXLMLLineRecord** function, respectively, to play or record the recording object. These functions contain two positioning elements, **StartPos** and **EndPos** which are in milliseconds. The **StartPos** element specifies the position where recording or playback is to begin. The **EndPos** element specifies the position where the operation should end. During playback, if the value of **EndPos** is unspecified, playback continues to the end of the recording. During recording, if the value of **EndPos** is unspecified, recording continues until a terminate command is received or the line is disconnected.

Call the **CXLMLLineWait** function to terminate recording or playback. **CXLMLLineWait** returns the number of milliseconds recorded or played. To start recording or playback, call the **CXLMLLineRecord** or **CXLMLLinePlay** function, respectively, and set the value of the **StartPos** element to the desired position. When recording or playback is complete, call **CXLMLLineWait** to retrieve the length, in milliseconds, that was played or recorded and add the length to the originating position value to derive a new position. The new position within the recording object is maintained by the client and may also be updated as a result of user controls such as a position slider or backup or fast-forward buttons. Play and Record operations can be called multiple times allowing the application to respond to user actions like rewind, forward, stop and continue.

Call the **CXLMLLineStatus** function to determine whether the line is still connected to a telephone. You should call this function periodically while recording or playing to determine if the connection has been broken by a hang up.

Call **CXLMLObjectClose** to close the recording object that was opened or created via calls to **CXLMLObjectOpen** or **CXLMLObjectCreate** functions. If the Recording object needs to be saved then a **RecordingTypeID** parameter must be passed to this function so that the server can determine the storage location for this object.

## Recording, Saving, and Sending a New Voice Message

To record, save, and send a new voice message:

- 1 Log on to a subscriber mailbox using the **CXLogon** function.
- 2 Connect to a telephone line using the **CXLMLineAttach** function.
- 3 Create a recording object using the **CXLMLObjectCreate** function. Save the returned value of the **ObjectID** parameter.
- 4 Start recording using the **CXLMLLineRecord** function, passing in the saved **ObjectID** value and setting the values of the **StartPos** parameter to 0. Note that the SOAP call returns immediately and recording continues asynchronously.
- 5 When the message is recorded, call the **CXLMLLineWait** function, setting the value of the Terminate parameter to 1. Note that the **CXLMLLineWait** function returns the length, in milliseconds, of the recording.
- 6 (Optional) Review the recording using the **CXLMLLinePlay** function, passing in the saved **ObjectID** value and setting the value of the **StartPos** parameter to 0.

**NOTE** You can use the **StartPos** parameter to position the point at which to record or play. For playback, this provides a mechanism to implement start, stop, backup, and fast-forward functions. For recording, positioning allows a recording to have additional audio appended to it or an existing recording may be partially or completely recorded over.

- 7 If you reviewed the recording in the optional previous step, terminate the playback using the **CXLMLLineWait** function.
- 8 Save the recording using the **CXLMLObjectClose** function, setting the value of the **RecordingTypeID** parameter to *Message* to save the recording as a message attachment.
- 9 Send the message using the **CXMessageSend** function. *Specify the following fields (these are the minimum fields to be specified) within the Message parameter:*
  - Set the *Type* element to *Voice*.
  - Set the **VoiceMsgSubType** element to *Normal*.
  - Specify one or more *Recipient* structures with their **MBID** fields set.
  - Specify an *Attachment* structure with the value of the **FileType** field set to 4 (WAV file) and the **FileName** field set to the recording **ObjectID** that was saved earlier.

## Playing a Mailbox Recording

To play an existing mailbox recording:

- 1 Log on to a subscriber mailbox using the **CXLogon** function.

- 2 Call the **CXPromptRecordingListGet** function to get the identifiers for spoken name/greeting associated with this mailbox.
- 3 Get the ID for this Recording returned in the **MediaFileID** field for the appropriate Prompt language.
- 4 Open the recording using the **CXLMOjectOpen** function, setting the **ObjectID** parameter to the recording name retrieved in the prior step and the **RecordingTypeID** parameter to the type of recording. For example, **RecordingTypeID** should be set to *Greeting* for the Standard subscriber greeting.
- 5 Connect to a telephone line using the **CXLMLineAttach** function.
- 6 Play the recording using the **CXLMLinePlay** function, specifying the **ObjectID** of the recording.

## Creating a New Mailbox Recording

To create a new mailbox recording:

- 1 Log on to a subscriber mailbox using the **CXLogon** function.
- 2 Connect to a telephone line using the **CXLMLineAttach** function.
- 3 Create a recording object using the **CXLMOjectCreate** function. Save the returned value of the **ObjectID** parameter.
- 4 Start recording using the **CXLMLineRecord** function, passing in the saved **ObjectID** value and specifying the **StartPos** parameter value as 0. Note that the SOAP call returns immediately and recording continues asynchronously.
- 5 When the recording is finished, call the **CXLMLineWait** function, setting the value of the **Terminate** element to 1. Note that the **CXLMLineWait** function returns the length, in milliseconds, of the recording.
- 6 (Optional) Review the recording using the **CXLMLinePlay** function, passing in the saved **ObjectID** value and setting the **StartPos** parameter to 0.

**NOTE** You can use the **StartPos** parameter to position the point at which to record or play. For playback, this provides a mechanism to implement start, stop, backup, and fast-forward functions. For recording, positioning allows a recording to have additional audio appended to it or an existing recording may be partially or completely recorded over.

- 7 If you reviewed the recording in the optional previous step, terminate the playback using the **CXLMLineWait** function.
- 8 Save the recording using the **CXLMOjectClose** function, setting the value of the **RecordingTypeID** parameter to the type of recording. For example, **RecordingTypeID** should be set to *Greeting* for the Standard subscriber greeting.
- 9 Update the mailbox configuration using the **CXPromptRecordingUpdate** function, setting the value of the **MediaFileID** field to the new recording object and the **PromptLangID** field to the prompt language to which this recording applies.

# MiCollab AM Web Services API WSDL

The listing below shows the MiCollab AM Web Services API WSDL:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="CXIf"
  targetNamespace="http://www.avstgroup.com/CXIf"
  xmlns:tns="http://www.avstgroup.com/CXIf"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:nsm="http://www.avstgroup.com/CXIf"
  xmlns:SOAP="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:MIME="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:DIME="http://schemas.xmlsoap.org/ws/2002/04/dime/wsdl/"
  xmlns:WSDL="http://schemas.xmlsoap.org/wsdl/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"

<types>

<schema targetNamespace="http://www.avstgroup.com/CXIf"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:nsm="http://www.avstgroup.com/CXIf"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="qualified">
  <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
  <!-- fault element -->
  <element name="CXFault">
    <complexType>
      <sequence>
        <element name="Error" type="xsd:unsignedLong" minOccurs="1" maxOccurs="1"/>
        <element name="Description" type="xsd:string" minOccurs="0" maxOccurs="1"
nillable="true"/>
      </sequence>
    </complexType>
  </element>
  <complexType name="CXProcessXML">
    <sequence>
      <element name="XMLRequestData" minOccurs="0" maxOccurs="1"
nillable="true"><complexType><sequence><any
maxOccurs="1"/></sequence></complexType></element>
    </sequence>
  </complexType>
  <!-- operation request element -->
```

```

    <element name="CXProcessXML">
      <complexType>
        <sequence>
          <element name="XMLRequestData" minOccurs="0" maxOccurs="1"
nillable="true"><complexType><sequence><any
maxoccurs="1"/></sequence></complexType></element>
        </sequence>
      </complexType>
    </element>
    <!-- operation response element -->
    <element name="CXProcessXMLResponse">
      <complexType>
        <sequence>
          <element name="XMLResponse" minOccurs="0" maxOccurs="1"
nillable="true"><complexType><sequence><any
maxoccurs="1"/></sequence></complexType></element>
        </sequence>
      </complexType>
    </element>
  </schema>

</types>

<message name="CXProcessXML">
  <part name="parameters" element="nsm:CXProcessXML"/>
</message>

<message name="CXProcessXMLResponse">
  <part name="parameters" element="nsm:CXProcessXMLResponse"/>
</message>

<message name="CXFaultFault">
  <part name="fault" element="nsm:CFault"/>
</message>

<portType name="CXIfPortType">
  <operation name="CXProcessXML">
    <documentation>Service definition of function nsm__CXProcessXML</documentation>
    <input message="tns:CXProcessXML"/>
    <output message="tns:CXProcessXMLResponse"/>
    <fault name="CFault" message="tns:CFaultFault"/>
  </operation>
</portType>

<binding name="CXIf" type="tns:CXIfPortType">
  <SOAP:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="CXProcessXML">
    <SOAP:operation soapAction=""/>
    <input>
      <SOAP:body parts="parameters" use="literal"/>
    </input>
    <output>
      <SOAP:body parts="parameters" use="literal"/>
    </output>
  </operation>
</binding>

```

```
</output>
<fault name="CXFault">
  <SOAP:fault name="CXFault" use="literal"/>
</fault>
</operation>
</binding>

<service name="CXIf">
  <documentation>gSOAP 2.7.8c generated service definition</documentation>
  <port name="CXIf" binding="tns:CXIf">
    <SOAP:address location="http://localhost:18276"/>
  </port>
</service>

</definitions>
```