



Using the Data Services API (DSAPI) with the S6100 1.5 Audio Conferencing Server

04-601899
Issue 1
March 2007

Notice

While reasonable efforts were made to ensure that the information in this document was complete and accurate at the time of printing, Avaya Inc. can assume no liability for any errors. Changes and corrections to the information in this document may be incorporated in future releases.

For full legal page information, please see the complete document, Avaya Legal Page for Hardware Documentation, Document number 03-600759.

To locate this document on our website, simply go to <http://www.avaya.com/support> and search for the document number in the search box.

Documentation disclaimer

Avaya Inc. is not responsible for any modifications, additions, or deletions to the original published version of this documentation unless such modifications, additions, or deletions were performed by Avaya. Customer and/or End User agree to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation to the extent made by the Customer or End User.

Link disclaimer

Avaya Inc. is not responsible for the contents or reliability of any linked Web sites referenced elsewhere within this documentation, and Avaya does not necessarily endorse the products, services, or information described or offered within them. We cannot guarantee that these links will work all of the time and we have no control over the availability of the linked pages.

Warranty

Avaya Inc. provides a limited warranty on this product. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya's standard warranty language, as well as information regarding support for this product, while under warranty, is available through the following Web site:

<http://www.avaya.com/support>

Copyright

Except where expressly stated otherwise, the Product is protected by copyright and other laws respecting proprietary rights. Unauthorized reproduction, transfer, and/or use can be a criminal, as well as a civil, offense under the applicable law. All non-Avaya trademarks are the property of their respective owners.

Avaya support

Avaya provides a telephone number for you to use to report problems or to ask questions about your product. In the United States and Canada, the support telephone numbers are: 1-800-242-2121 or 1-800-866-5469. In locations that can not call "800" phone numbers, dial: +1-720-444-0734.

E-mail: MXCustomerSupp@avaya.com

Fax: 1-978-677-5134

Outside the United States and Canada, use these numbers: 353-1-207-5667 or 353-1-207-5668.

E-mail: MXSupport@avaya.com

For additional support telephone numbers, see the Avaya Web site:

<http://www.avaya.com/support>

Advisory

Customers who accept responsibility to implement (install and configure) this solution with their own support team will be responsible for all associated implementation and technical assistance on a time and material basis if required from Avaya Global Services."

Contents

Preface	5
Introduction	5
Intended audience	5
Related documents	6
Conventions	6
Contacting Avaya	7
Glossary of terms	8
Chapter 1: Getting Started	11
In this chapter	11
Getting Started	11
Chapter 2: DSAPI Overview	13
Background	13
Class Dependencies	13
Transport Layer Security (TLS)	14
JAAS	15
Exception Handling	15
Logging	16
Service Locator	16
Object Creation	17
Object Updates	17
Object Deletion	17
Object Searching	17
Paging	18
Chapter 3: Account Management	19
In this chapter	19
Accounts	19
Account Creation	19
Account Updates	20
Account Deletion	20
Account Retrieval	20
Account Searching	20

Chapter 4: Reservation Management 21

 In this chapter 21

 Reservations 21

 Reservation Instances. 21

 Port Pools 22

 Conference Profiles 22

 Reservation Creation 23

 Reservation Updates 24

 Reservation Cancellation 24

 Reservation Searching 24

Index 25

Preface

The preface provides information on the following topics:

- [Introduction](#)
- [Intended audience](#)
- [Related documents](#)
- [Conventions](#)
- [Contacting Avaya](#)
- [Glossary of terms](#)

Introduction

The Meeting Exchange® Express Edition Data Services API (DSAPI) is an object-oriented, programming interface written in Java that describes the business services layer for the Meeting Exchange® S6100 server software. Using DSAPI, application programmers can develop third-party or customized applications for integration with the Meeting Exchange® S6100 database. DSAPI can be used by any application that can make a Java Remote Method Invocation (RMI) call for the purpose of storing or retrieving information in the S6100 database. This document will give an overview of DSAPI and then will focus on how to manage user accounts and reservations using DSAPI.

Intended audience

The intended audiences for this guide are application developers and technical support representatives for Avaya Tier IV conferencing.

Related documents

This guide is designed for use with the following Meeting Exchange documents:

- *DSAPI Javadoc*
- *Avaya Meeting Exchange® Express Edition Release 1.5 Administration and Maintenance Guide*
- *Avaya Meeting Exchange® Express Edition Release 1.5 Installation and Configuration Guide*
- *S6100 Web Interface Installer Online Help*
- *S6100 Web Interface System Administrator Online Help*



Tip:

Use Internet Explorer 6.0 with SP1 or later for optimum online help performance.

Conventions

[Table 1](#) describes conventions commonly used in this book.

Table 1: Writing conventions

Convention	Indicates
Choose	A command from the menu bar. For example, choose File > Exit .
Click	An item selection in GUI-based dialogs or Web pages. For example, click OK or Print .
Enter	Text input from a keyboard. For example, enter: <code>pmadm -r -p</code>
Keyname	Keyboard keys, for example Esc or Enter . In certain cases, keynames refer a combination of keys, such as Ctrl+Enter .
ASCII text	Specific Ascii-based keys or key sequences, macros, commands, fields or items, and lines or strings in a file or system display. For example, Open the file in the text editor of your choice and locate the following term: <code>TemplateDir</code>
1 of 2	

Table 1: Writing conventions (continued)

Convention	Indicates
Bold	GUI-based items, such as menu commands, buttons, box labels, and fields, as well as numbers, letters, symbols on a keyboard or keypad. For example, when you click Configure , the Administration Configuration dialog displays.
Italic	Titles of books and CD-ROMs or verbatim (exactly as shown) text passages. Provides emphasis on terms where needed. For example: - See the <i>Installation and Configuration Guide for Groupware Edition for Lotus Notes</i>. - Make sure that the switch is in the <i>off</i> position before removing the high-voltage transformer.
Link	URLs, cross-references and other hypertext fields. For example, see Installing XYZ on page 123.
2 of 2	

Contacting Avaya

The following table lists telephone numbers and E-mail addresses.

For information about Meeting Exchange servers and features	
US and Canada Technical Support	+1-800-242-2121 +1-800-866-5469 +1-720-444-0734 E-mail: MXCustomerSupp@avaya.com Fax: +1-978-677-5134
International Technical Support	+353-1-207-5666 (CS7000) +353-1-207-5667 E-mail: MXSupport@avaya.com
Avaya Support Web site	www.avaya.com/support For the latest information about this product and accompanying documentation, go to the product link located on this site.
Avaya Web site	www.avaya.com

Glossary of terms

[Table 2](#) describes terms commonly used in this document.

Table 2: Terminology

Term	Definition
Conference	The audio mixing of conferees dialed into a bridge to create a single physical session or instance, with the mixed output sent to everyone in the conference.
DSAPI	Data Services Application Programming Interface. The Data Services API is an object-oriented, programming interface that is written in Java. Programmers use DSAPI to write applications that communicate with the S6100 conference server application.
DTMF	Dual Tone Multi-Frequency
EJB	Enterprise JavaBean™. An architecture for setting up program components that are written in Java. JavaBeans run on the server side of a computer network that uses the client/server model.
instance	An individual conference session within a reservation.
JAAS	Java Authentication and Authorization Service.
JBoss	An application server written in Java that can host business components developed in Java. JBoss is an open-source implementation of J2EE that relies on the EJB specification for functionality.
JDK	Java Development Kit
JNDI	Java Naming and Directory Interface
JVM	Java Virtual Machine. A platform-independent programming language that converts the byte code of a Java application into computer language and executes the code
Provisioning	A process used by administrators to assign system resources and privileges and enforce security policies for a defined set of roles.
Reservation	A group of scheduled conference sessions or instances created as a single booking.
1 of 2	

Table 2: Terminology (continued)

Term	Definition
RMI	Remote Method Invocation. A set of protocols developed by Sun Microsystems™ that enables Java objects to communicate remotely with other Java objects.
SCP	Secure Copy
TLS	Transport Layer Security, essentially its SSL
2 of 2	

Chapter 1: Getting Started

In this chapter

This chapter provides information on connecting to a demo DSAPI client application.

Getting Started

To start using DSAPI, a demo DSAPI client application is available on your Meeting Exchange® S6100 conferencing server. To access the demo application, follow the steps below:

1. Go to: `/usr/crystal/dsapi_client/crystal_dsapi_client.zip`
2. Using SCP, copy this zip file to your local machine.
3. Unzip the file.
4. Locate the `readme.txt` file exists in the root directory.

The `readme.txt` file provides instructions on how to get started. The demo application illustrates account and reservation management along with miscellaneous DSAPI functionality. Once the demo application is working, the DSAPI programmer can use it as a working platform for further development or for a reference point to a working DSAPI client, in conjunction with this document and the DSAPI javadocs located in `<demo-location>\mxee\dsapi\javadocs`.

This document assumes the reader has a reasonable knowledge of the Java programming language and using RMI.

Chapter 2: DSAPI Overview

This chapter provides information about the following topics:

- [Class Dependencies](#)
- [Transport Layer Security \(TLS\)](#)
- [JAAS](#)
- [Exception Handling](#)
- [Logging](#)
- [Service Locator](#)
- [Object Creation](#)
- [Object Updates](#)
- [Object Deletion](#)
- [Object Searching](#)
- [Paging](#)

Background

The DSAPI server consists of stateless beans deployed on a JBOSS application server (Release 4.0.3, SP1). The stateless session beans provide read/write access to the database. For write operations, a series of business rules are applied to the data, before it is persisted. The stateless session beans provide Java interfaces that clients can use to invoke these operations using RMI. These interfaces are retrieved using the JNDI service that is also deployed on JBOSS AS.

Class Dependencies

DSAPI clients require the following files in its classpath:

- **jbossall-client.jar**
 - JBOSS client side library
- **Crystal_common.jar**
 - DSAPI client side library

- **Log4j.jar**
 - Logging module
- **Config dir.** This contains:
 - **Crystal.cfg**
 - Configure location of DSAPI server here
 - **Crystal.truststore**
 - Contains X.509 cert, detailed later in this section
 - **CrystalAuth.cfg**
 - Required for JAAS, detailed later in this section
 - **CrystalErrors.txt**
 - Contains error messages for DSAPI error codes, internally used by DSAPI client library
 - **CrystalTags.txt**
 - Used in conjunction with CrystalErrors.txt by DSAPI client library
 - **CrystalLog.xml**
 - Contains log4J configuration DSAPI client library. Currently two logs files are generated in the current directory, dsapi.log & dsapi_errors.log

Note:

The demo application ships with all the above files.

Transport Layer Security (TLS)

All DSAPI communication to the S6100 server is over TLS. Clients using DSAPI require an X.509 security certificate to communicate. The required certificate can be found in:

```
<demo-location>\mxee\dsapi\config\crystal.truststore
```

This file can be examined using the JDK keytool program.

Password: crystal

Cert alias: crystal

This self-signed certificate is generated during the installation of the Meeting Exchange® S6100. To allow the certificate to be used, ensure that the DSAPI client JVM has the following parameter set:

```
Djavax.net.ssl.trustStore=.<truststore location>
```

JAAS

All DSAPI clients must be authenticated and authorized using JAAS. (Java Authentication and Authorization Service). JAAS is used to reliably and securely determine who is currently executing Java code and whether they have sufficient permissions to execute the code. The demo application uses the default Administrator user name and password, created at install time, to gain access. Alternatively a previously configured user's email and password can be used.

Exception Handling

All DSAPI methods can throw a number of exceptions. Once an exception is thrown by the DSAPI server, if any changes were made, they are undone, and an exception is thrown back to the client. The following exception types exist:

- **RemoteException**

All DSAPI methods can throw this exception. This exception is part of the RMI protocol.

- **SystemException**

All DSAPI methods can throw this exception. This exception means a system error occurred on the DSAPI server. For example, the server lost connection to the database, the server failed to allocate memory, etc. The system administrator should be consulted when this exception is thrown.

- **BusinessException**

DSAPI methods which cause modification to the database can throw this exception by violating a business rule, for example, a DSAPI dataobject for insertion contained invalid characters, etc. A BusinessException contains one or more errors that occurred during method invocation.

 Tip:

Refer to the demo code on how to interrogate DSAPI exceptions.

Logging

The DSAPI client library, `crystal_common.jar`, uses log4j for its internal logging. The standard priority levels are applicable:

- DEBUG
- INFO
- WARNING
- ERROR
- FATAL

Note:

The demo code comes with logging level set to INFO and outputs to current working directory.

Service Locator

The `ServiceLocator` class is a wrapper around JNDI to provide an easy method of locating the exposed session interfaces. The `getEJBObject()` method takes the name of the desired home interface as a parameter and invokes its `create()` method to return the resulting remote interface.



Important:

These interfaces exist in the `com.avaya.crystal.common.interfaces.*` package

The `ServiceLocator` class maintains a cache of home interfaces to reduce network round trips on subsequent lookups. Example usage can be seen in the demo code.

The `ServiceLocator` class picks up the location of the DSAPI server in the `Crystal.cfg` file which must be in the classpath.

Object Creation

Objects can be persisted in the database by constructing the desired data object from the package `com.avaya.crystal.common.dataobjects.*` and then calling the `add()` method on the relevant session interface that handles that type of dataobject. Most dataobjects have a default constructor which details mandatory fields that must be populated by the client before calling `add`. Business logic is applied to the dataobject in the server and if successful, the dataobject is persisted to the database, and the newly created object is returned to the client. The returned object will have an ID assigned to it, which is persisted in the database also and is accessible via `getId()`.

Note:

Clients cannot specify the ID at creation time.

Object Updates

Previously persisted data objects can be updated by calling the `update()` method on the relevant session interface that handles that type of dataobject. Business logic will also be applicable for updates.

Object Deletion

Previously persisted data objects can be deleted from the database by calling the `delete()` method on the relevant session interface.

Object Searching

Data objects can be searched in the database by calling the appropriate `find()` method on the relevant session interface that handles that type of dataobject. The `find()` method takes a dataobject search object as a parameter and returns a list of dataobjects of that type that have values matching those set in the dataobject search parameter.

- Null fields in the search object are excluded from the search.
- For string fields in the search objects, a wild card can be used to widen the search. For example, to search for all reservations with a name ending in 'daily', `setName("%daily")` should be called on the reservation search object.

If the id of a dataobject is known, it can be retrieved by calling the `getById()` method on the relevant session interface.

Paging

Each DSAPI interface has `find` methods that take advantage of a paging mechanism to allow clients to review data one page at a time. The DSAPI server provides this paging logic.

Paging is the preferred mechanism where large amounts of data are expected and response time is critical. For example, a reservation viewer would use the paging interface to allow end users to view reservations one page at a time.



Tip:

Refer to the demo code and javadocs to understand how the paging mechanism works.

Chapter 3: Account Management

In this chapter

This chapter provides information about the following topics:

- [Accounts](#)
- [Account Creation](#)
- [Account Updates](#)
- [Account Deletion](#)
- [Account Retrieval](#)
- [Account Searching](#)

Note:

This chapter should be read in conjunction with the demo code and javadocs to understand account management.

Accounts

Accounts, also known as End Users, are required for reservation bookings . An account is associated with each reservation. You do not need an account to participate in a conference.

Accounts are managed via the **LogonSession** interface. Usage of this interface is illustrated in the demo application.

Account Creation

Create accounts using the `LogonSession.add()` method which takes a newly created Account data object. Accounts must be associated with a previously created Company data object on creation and each account must have a unique email address. The demo code shows how to create a company or organization and then associate this company's ID with an account. If desired, accounts may be sorted based on different companies. The `LogonSession.add()` returns the newly created account object which will now have its identifier field populated, which can be retrieved using `Account.getID()`.

Account Updates

Update accounts using the `LogonSession.update()` method, which takes the modified Account object as a parameter.

Account Deletion

Delete accounts using the `LogonSession.delete()` method, which takes the `Account.getId()` field object as a parameter.

Account Retrieval

If the ID of an account is known, retrieve it by using the `LogonSession.getbyId()` method.

Account Searching

Search for accounts using the `LogonSession.find()` methods.

Chapter 4: Reservation Management

In this chapter

This chapter provides information about the following topics:

- [Reservations](#)
- [Reservation Creation](#)
- [Reservation Updates](#)
- [Reservation Cancellation](#)
- [Reservation Searching](#)

Note:

This chapter should be read in conjunction with the demo code and javadocs to understand reservation management.

Reservations

Reservations are conference bookings. Reservations are managed via the `ReservationSession` interface. It is important to understand the following conferencing terminology when using DSAPI:

- **Demand Reservation**

Conference can be opened at any time for use. Only one demand reservation can be associated with an account.

- **Scheduled Reservation**

Conference starts and ends at specified dates and times. Multiple scheduled reservations can be associated with an account.

Reservation Instances

A reservation instance represents an actual conference, indicating its start and end time.

- **Scheduled Reservations:** instances are pre-populated according to a recurrence pattern.
- **Demand Reservations:** Instances are created as demand conferences that can open and close at any time.

Port Pools

Each conference participant requires a port to access a conference. The system is partitioned into two port pools:

- **Reserved:** If a reservation is configured to use reserved ports, then at conference time, participants are guaranteed to get into their conference.
- **Unreserved:** If a reservation uses unreserved ports, callers enter the conference on a first come, first served basis.

Tip:

Typically demand reservations use unreserved ports, and scheduled reservations use reserved ports. The system is partitioned between reserved and unreserved. By default, the reserved port pool size is zero and the unreserved port pool size is the number of licensed ports. Information on changing the default configuration is found in *Meeting Exchange® Express Edition Release 1.5 Administration and Maintenance Guide*.

Conference Profiles

A reservation must be configured to use one of the pre-configured conference profiles.

Please refer *Meeting Exchange® Express Edition Release 1.5 Administration and Maintenance Guide* which details the characteristics of these conference profiles. Reservations inherit behavior from conference profiles. If a conference profile indicates that 'recording enabled' is off, then no recordings will happen for any reservation using that profile.

A subset of the behavior inherited from conference profiles may also be modifiable on a per reservation basis. Again this depends on the selected profile. If 'recording enabled' can be overridden for a particular profile, any reservation associated with that profile can determine whether recording should be enabled or disabled.

Tip:

This feature is documented in the demo code.

- **Message Profiles**

A reservation must be configured to use one of the pre-configured message profiles. A message profile details what prompts to play at conference setup and during the conference. As only one message profile currently exists, reservations must always use the default profile.

- **Conferee**

A participant of a conference.

- **Moderator**

A conference can be moderator controlled. A moderator is a conferee who has special conference privileges such as dialing out to other participants. Only one moderator can exist per conference. It is an optional setting. This behavior is also inherited from the conference profile associated with the reservation.

- **Passcodes**

These are DTMF digit sequences that allow a participant to enter a conference. Each reservation must have a conferee passcode, and if it is moderator controlled, it must also have a moderator passcode. Passcodes can be auto-generated by the system, but this functionality depends on what conference profile is associated with the reservation.

- **Conferee count**

Number of conferees allowed to enter the conference.

- **Recurrence details**

This feature refers solely to scheduled reservations. It specifies what recurrent pattern and dates and times to use in the reservation.



Tip:

Refer to Javadocs for a comprehensive description of the supported patterns.

Reservation Creation

Reservations can be created using the `ReservationSession.add()` method, which takes a newly constructed `DemandReservation` or `ScheduledReservation` data object. Reservations must be associated with a previously created Account data object on creation. The `ReservationSession.add()` returns the newly created reservation object which will now have its identifier field populated that can be retrieved using `Reservation.getID()`.

For scheduled reservations, the `Reservation.getInstances()` field is populated with a set of Instance data objects. Each object instance represents an expected conference in the future according the specified recurrence pattern.

For demand reservations, the `Reservation.getInstances()` field is empty at creation time. However entries are added to the set each time a demand conference is opened.



Tip:

To create a reservation using reserved ports, the Reserved Ports pool size must first be set using the Meeting Exchange® Express Edition web interface. Otherwise a `BusinessException` will be thrown indicating there are no free reserved ports.

Reservation Updates

Reservations can be updated using the `ReservationSession.update()` method, which takes the modified Reservation data object as a parameter. For scheduled reservations, if the update involves a change to `Reservation.getEarlyStartMinutes()` or `Reservation.getRecurrenceDetails()`, all the previously associated current and future instances are cancelled. A new set of instances are created to reflect updated values. This type of update will throw a `BusinessException` if one of the existing instances represents a conference that has started or is due to start.

Reservation Cancellation

Reservations can be cancelled using the `ReservationSession.cancel()` method, which takes the `Reservation.getId()` field object as a parameter. Reservation instances can also be cancelled. Refer to javadoc for specific details.

Once a reservation or reservation instance is cancelled, one can no longer join that conference with the passcode(s) specified in the reservation, and this operation cannot be undone.

Reservation Searching

If the ID of an reservation is known, it can be retrieved using the `ReservationSession.getbyId()` method.

Reservations can be searched using the `ReservationSession.find()` methods. There are a number different search methods depending on the needs of the application.



Refer to Javadoc for further details.

Index

A

Account Creation	19
Account Deletion.	20
Account Retrieval	20
Account Searching.	20
Accounts	19

C

Class Dependencies	13
Conference	
definition	8
Connecting to DSAPI Javadocs.	11
Conventions	
of book.	6

D

Documentation	
listed of related	6
DSAPI	
definition	8
DSAPI Overview.	13

E

EJB	
definition	8
Exception Handling	15

I

Instance	
definition	8

J

JAAS	
definition	8
JBOSS	
definition	8
JMS	
definition	8

M

Material	
related documentation	6

O

Object Creation.	17
Object Deletion	17
Object Searching	17
Object Updates.	17

P

Paging	18
Provisioning	
definition	8

R

Reservation Cancellation	24
Reservation Creation	23
Reservation Searching	24
Reservation Updates	24
Reservations	21
RMI	
definition	9

S

SCP	
definition	9
Service Locator.	16

T

Technical support	
contact information	7
TLS	14
Transport Layer Security (TLS)	14

