

Sametime
Version 9.0

Sametime 9.0
Software Development Kit
Video and Voice Integration Guide



Edition Notice

Note: Before using this information and the product it supports, read the information in "Notices."

This edition applies to version 9.0 of IBM Sametime (program number **5725-M36**) and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright IBM Corporation 2006, 2013.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 1. Introduction.....	5
Opportunity for users, opportunity for IBM partners.....	5
Today's Sametime telephony and video features.....	5
History of Sametime telephony and video features.....	6
Prerequisite skills and recommended reading.....	6
Skills.....	6
Background References.....	6
Terminology.....	7
Commonly used technologies.....	7
Conferencing terms.....	7
Chapter 2. Overview of the Sametime Media Manager.....	9
Introduction.....	9
Topology.....	9
Main Components.....	10
Partners extend Sametime via TCSPI adapter.....	10
Chapter 3. Implementing the Telephony Service Provider, Part I.....	11
Summary of key classes and interfaces.....	13
Conference Service Interface.....	13
Conference and User Data Model.....	13
ConferenceServiceRequests interface.....	14
Conference Lifecycle.....	14
Actions.....	14
User management.....	14
Property setting and notification.....	14
Interaction of key classes in sample of user scenarios.....	15
Sametime Connect as audio/video endpoint.....	15
Third-party provided non-SIP endpoint (Legacy).....	16
Conference management and creation.....	16
Conference creation.....	16
Conference management process.....	17
Your implementation.....	17
Conference creation.....	17
More on features.....	18
Conference data retrieval.....	18
Persistence.....	18
Chapter 4. Sametime Connect interaction with a third-party MCU.....	19
Sametime Connect and SIP-enabled MCU.....	19
Sequence of Events.....	19
Client endpoint registration.....	20
Call Initiation and Conference Creation.....	20

Conference Start and Endpoint Registration.....	21
Dial Request and Media Negotiation.....	21
Data Flow Established via RTP.....	22
Summary.....	22
Interaction diagram: Start conference with an external MCU.....	23
Chapter 5. Introduction to the MyAV sample service provider.....	24
My A/V sample.....	24
Service Adapter Classes.....	25
MCU Classes.....	25
Chapter 6. Implementing the Telephony Service Provider, Part II.....	26
Extending the Default Conference and User.....	26
Defining your supported required and optional features.....	26
Conference features are specified as XML.....	26
Your telephony hardware.....	27
Guidelines for choosing or using features.....	27
Handling failure scenarios.....	29
System failure scenarios.....	29
Asynchronous response handling.....	31
The TCSPI does not collect events from telephony hardware.....	31
The process.....	31
Process and deliver methods are paired.....	31
Process and deliver sequences.....	32
Example: Dial methods.....	32
Understanding callbacks.....	32
Deliver methods.....	32
processDial() Comparing success to failure.....	32
Handling request timeouts.....	34
Service provider responsibilities.....	34
Timeout process.....	34
Canceling pending requests.....	34
Tips on notifying users.....	34
Appendix A: Setting up your development environment.....	36
Appendix B: Installing the MyAV sample.....	38
Creating a new WebSphere Server Profile.....	38
Installing the MyAV service provider.....	40
Installing the MyAV MCU enterprise application.....	42
How to enable TLS for SIP transport.....	45
Installation verification.....	45
Appendix C: Installing and testing your service provider.....	47

Chapter 1. Introduction

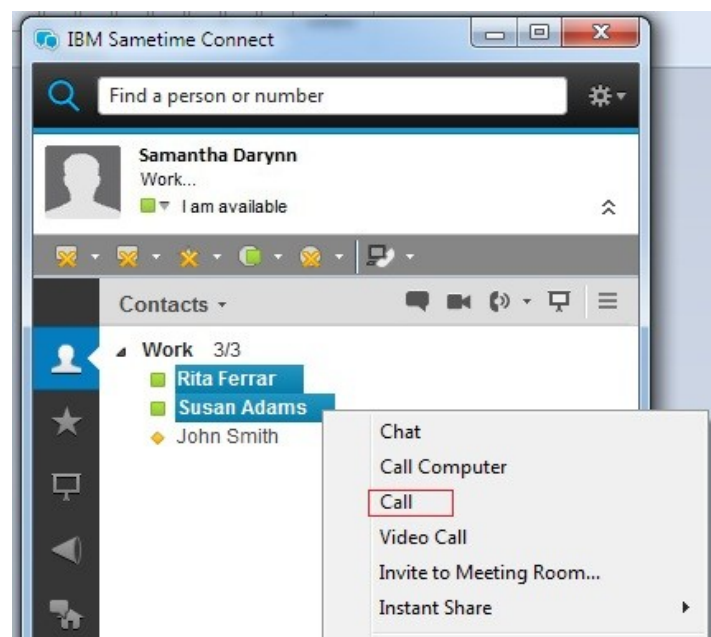
This guide introduces you to the IBM Sametime Telephony Conferencing Service Provider Interface (TCSPI) application program interface (API). The TCSPI is a set of APIs that you will use to create a service provider interface (SPI). Given a third-party provider's implementation of this SPI and the accompanying configuration files, the Sametime administrator can install a TCSPI adapter that will be accessed when community users request audio/video features from the Sametime Connect client or Meetings client. Similarly, the adapter will be notified of user requests like mute/unmute/start video and can notify the Sametime Media Server of user events like speaking started/stopped. Programmers can use this guide and supporting material to create and implement a teleconferencing service provider interface, commonly referred as an adapter.

Opportunity for users, opportunity for IBM partners

Instant messaging is an integral part of today's work environment, enabling teams to collaborate across wide geographical areas efficiently and cost effectively. As part of the IBM® Sametime® instant messaging offering, users can have the convenience of click-to-dial access from their instant messaging contact list. Sametime version 8.5 further enhances the user's experience by offering point-to-point and n-way audio and video conferencing. Third-party telephony and conferencing solution providers can seamlessly integrate their product offering into the Sametime user interface by implementing the service provider interface (SPI) defined by the Sametime Media Server. By writing an adapter, third party telephony and conferencing solution providers can leverage the existing Sametime user interface for call management and conference control at no additional development cost.

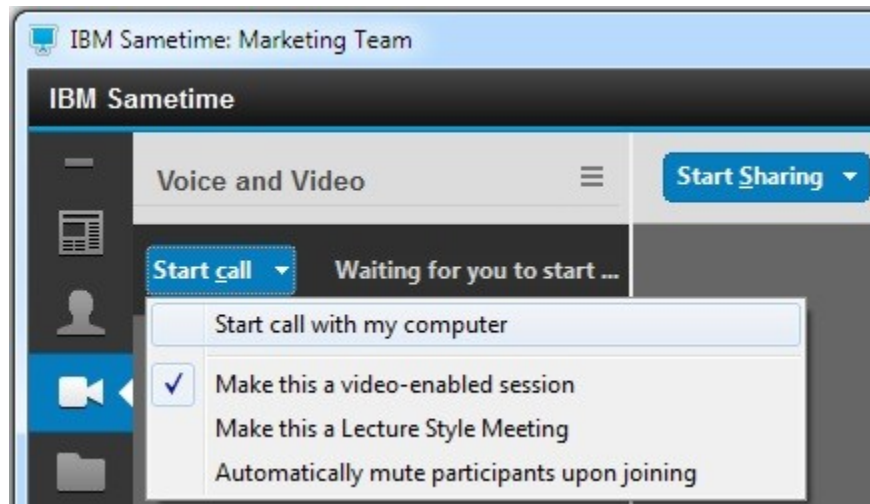
Today's Sametime telephony and video features

Click to Call: Users can click to call others, known as the click-to-call feature. A user simply chooses Call from the context menu of one of their Sametime Connect contacts to start a point-to-point computer-based call or a third-party supplied endpoint. They are able to add other participants to the call with the standard *Invite Others* dialog, much like they can add additional participants to a chat session.



Sametime Connect's right-click menu showing the Click to Call icon.

With click-to-call, users can invite one or more others to a telephone call that is managed by a third party telephony service provider. Click-to-call feature interacts with the provider's TCSPI adapter on the Sametime server.



The Click to Call Conferencing dialog box

Meetings Integration: Users can easily include audio and video features when they create a meeting room. The participant list includes information on who is speaking, along with the moderator's ability to control the conference by providing the ability to mute/unmute, lock the conference and invite other participants.

In order to connect to your existing teleconferencing system, you will use the APIs and SPIs that are part of the TCSPI toolkit to create a Sametime server-side Java adapter. This adapter provides integration with the IBM Sametime Conference Manager by enabling your implementation to be aware of Sametime users' audio/video-related events and requested actions.

History of Sametime telephony and video features

The SPIs introduced by previous editions of this guide were specific to voice-only integration. Starting with Sametime version 8.5, these SPIs include interfaces for enabling video features. Below is a summary of the voice and video capabilities of each release:

Sametime 7.5 – TCSPI for click-to-dial audio calls for external devices (phones, conferencing systems).

Sametime 8.0 – Features of the previous release plus point-to-point computer-based audio/video calls and client-side user interface extensions.

Sametime 8.5.x, Sametime 9.0– Features of the previous release plus point-to-point and n-way audio/video conferencing,

Enhanced TCSPI interfaces for third-party video enablement, including support of SIP-enabled Multi-point Control Units (MCU) acting as conference-aware audio/video endpoint.

Prerequisite skills and recommended reading

Skills

Skills in the following areas are desirable.

- Java programming skill.
- Using the Eclipse integrated development environment (IDE).
- Telephony theory and experience.
- Communications and Networking.

Background References

- RFC 3261 - “SIP: Session Initiation Protocol”
<http://www.ietf.org/rfc/rfc3261.txt>
- RFC 4235 - “An INVITE-Initiated Dialog Event Package for the Session Initiation Protocol (SIP)”
<http://www.ietf.org/rfc/rfc4235.txt>
- RFC 4353 - “A Framework for Conferencing with the Session Initiation Protocol (SIP)”
<http://www.ietf.org/rfc/rfc4353.txt>
- RFC 5389 - “Session Traversal Utilities for NAT - (STUN)”
<http://tools.ietf.org/html/rfc5389>

The above documents also list many references that are relevant to Sametime conferencing.

Terminology

Discussions of video and voice conferencing use specialized terminology and technologies, which are described in this section.

Commonly used technologies

Table listing and describing commonly used technologies in Sametime.

<i>Term</i>	<i>Meaning</i>
PSTN	Short for Public Switched Telephone Network, which refers to the international telephone system based on copper wires carrying analog voice data. This is in contrast to newer telephone networks base on digital technologies, such as ISDN and FDDI. Telephone service carried by the PSTN is sometimes called plain old telephone service (POTS).
VoIP	Short for Voice over Internet Protocol. This enables people to use the Internet as the transmission medium for telephone calls by sending voice data in packets using IP rather than by traditional circuit transmissions of PSTN. One advantage of VoIP is that the telephone calls over the Internet generally do not

	incur a surcharge beyond what the user is paying for Internet access, much in the same way that the user does not pay for sending individual e-mails over the Internet.
SIP	<p>Short for Session Initiated Protocol, or Session Initiation Protocol. This is an application-layer control protocol, which is a signaling protocol for Internet Telephony. SIP can establish sessions for features such as audio or videoconferencing, interactive gaming, and call forwarding to be deployed over IP networks, thus enabling service providers to integrate basic IP telephony services with Web, e-mail, and chat services.</p> <p>SIP offers user authentication, redirect and registration services. Also, a SIP server supports traditional telephony features such as personal mobility, time-of-day routing and call forwarding based on a person's geographical location.</p>

Conferencing terms

Table listing and describing conferencing terms.

Term	Meaning
Conference Server	As defined in RFC 4353, "A conference server is a physical server that contains, at a minimum, the focus. It may also include a conference policy server and mixers."
Focus	<p>A SIP component that manages multipoint conferences by maintaining a dialog with each participant and ensuring that all media flows between those participants either directly or through appropriate mixers.</p> <p>Mixer – A component that processes and redistributes media streams in a multipoint session. This can run on either a server or on one or more clients in a session.</p>
SIP	The Session Initiation Protocol, which is used to negotiate a media session between two endpoints.
Bridge	Another word for a mixer, especially when referring to audio.
MCU	Multi-point Control Unit. Another name for a mixer, especially when referring to audio / video.
Packet-switching MCU	A type of MCU that simply forwards packets rather than processing the media streams. All required media processing occurs on the clients.
ICE	ICE (Interactive Connectivity Establishment) defines a standard method of using STUN and TURN to establish connectivity between peers. ICE can be used by any protocol utilizing the offer/answer model, such as the Session Initiation Protocol (SIP).
SDP	The Session Description Protocol, which is used to define the characteristics of the media streams that will be exchanged in a session.
STUN	A STUN (Simple Traversal of UDP through NAT) server allows clients to find out their public address, the type of NAT they are behind and the internet side port associated by the NAT with a particular local port. This information is used to set up connection between the client and the provider.

TURN	TURN (Traversal Using Relay NAT) is a protocol that allows an entity behind a NAT or firewall to receive incoming data over TCP or UDP connections.
RTP / SRTP	The Real time Transport Protocol (and its secure counterpart), which is used to transport audio / video within a session.

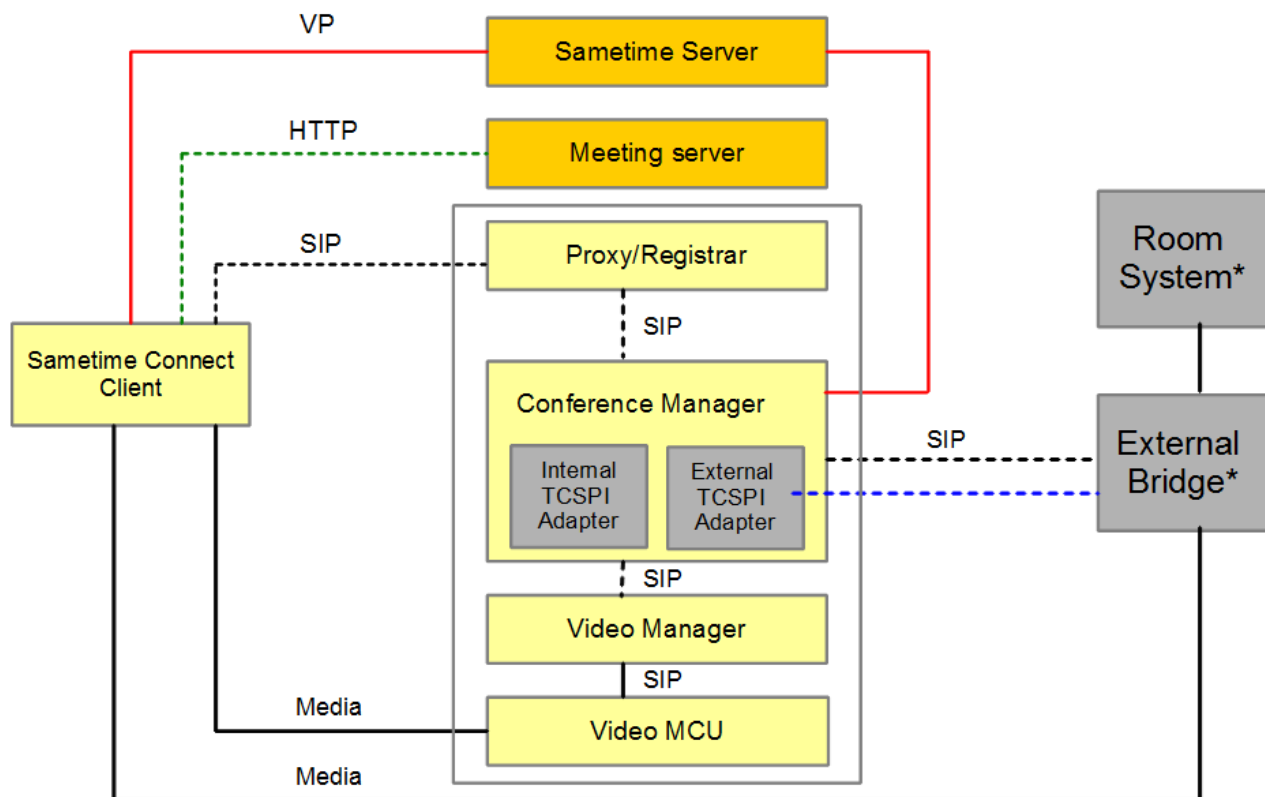
Chapter 2. Overview of the Sametime Media Manager

Introduction

The architecture for Sametime 8.5 audio and video feature departs significantly from the previous releases on the server side. The Sametime Media Manager builds its audio/video capability on top of WebSphere Application Server version 7.0 (WAS) as a converged (SIP and HTTP) application. This will enable it to leverage the WebSphere server infrastructure including scalability, security, and high availability with clustering. The entire programming on the server will use the J2EE model.

Topology

The figure below is a depiction of the overall architecture of the Sametime Media Manager with illustration of potential third party external voice and video bridge integration.



Sametime Media Manager server architecture

The Media Manager requires the Sametime Community Server since it works as a Service Application to provide voice and video capability to Sametime users within the community. The Community Server provides the presence status of audio and video and other standard Sametime features such as

administrative policies and configurations. The Connect client will discover the Media Manager upon log-in and establish the communication endpoint with the Media Manager by registering itself with the Proxy/Registrar.

Sametime continues to use VP channels between the Connect client and the Community Server for all the existing services; however, most new voice and video features will use the standard protocols SIP and RTP. TCSPI uses the VP channel for call management and event notification. TCSPI support in the Media Manager allows backward compatibility. An existing (pre-8.5 release) TCSPI adapter can be deployed on the Media Manager instead of on the Community server. There are new APIs and properties added to the TCSPI to enhance call options for video and SIP-enabled bridges.

The Meetings server is optional. Voice and video are integrated with Meetings on the Connect client and will require the presence of Meetings server and Media Manager to function. Using voice and video in chat for both point-to-point and multipoint will not require the Meetings server.

Main Components

There are four components making up the Media Manager: Proxy Registrar, Conference Manager, Video Manager (VMGR) and Video MCU.

Proxy Registrar: This component consists of two SIP applications: One is the Registrar responsible for location service, and the other is the Proxy forwarding SIP messages to the destination or to another Proxy. The Proxy Registrar implements the standard SIP RFC3261.

Conference Manager: This component is a SIP application that manages all conferences or calls, including point-to-point. It works with the client to establish the SIP session for the conference. It hosts the internal TCSPI adapter and optionally external TCSPI adapter(s).

Video Manager :The Video Manager manages the scaling and distribution of audio and video conferences, through MCU pools and cascading. It also manages attributes for conferences, such as maximum line-rate. The Video Manager must be installed on its own machine.

Video MCU: The Video MCU (Multipoint Control Unit) enables multi-way, audio and video conferences with continuous presence and multiple client layouts. Serves as a switch for scalable audio- and video-streams, delivering to different clients the streams that have been requested. The Video MCU must be installed on its own machine.

Partners extend Sametime via TCSPI adapter

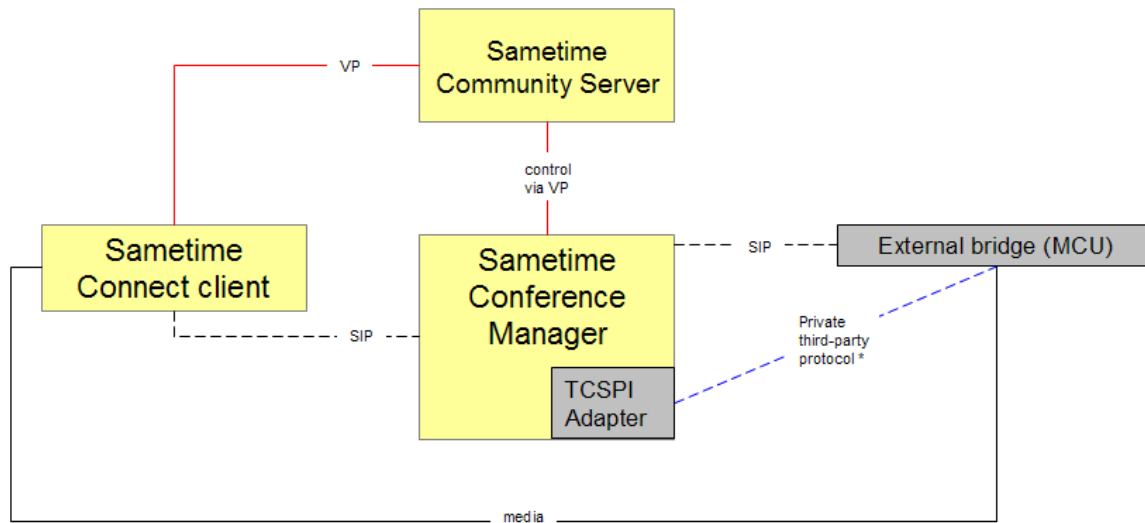
The external TCSPI adapter and MCU are optional installs. They will be provided by partners to integrate Sametime with their equipment, like room systems and telephony devices (hardware or software). The Sametime 8.5 Connect client can work with a SIP-based external MCU via the Media Manager. When the external TCSPI adapter is installed, the Connect client can communicate with the external bridge to establish a conference. Voice and video data will flow directly from the external bridge to the Connect client via RTP. The Connect client supports standard audio and video codecs, including SAC, Siren-LPR, G722.1C, G.722.1, G.711, iSAC, iLBC, H.264, H264-SVC and H263. If the external bridge works with any of these codecs, it can leverage the Connect client as an audio/video endpoint.

The key integration point is the TCSPI adapter to allow the Connect client to manage the calls on the external bridge. The rest of this document will provide detail information for implementing a TCSPI adapter.

Chapter 3. Implementing the Telephony Service Provider, Part I

Since you will be using the TCSPI to implement a service provider of your own, Part I provides an introduction to what a typical service provider is expected to do, followed by an overview of the sample adapter implementation, MyAV. Part II introduces more advanced topics you will need to consider while implementing your own adapter.

The diagram below depicts a simplified view of the TCSPI adapter's relationship to other components of the Sametime Media Server:



The TCSPI adapter's relationship to other components of the Sametime Media Manager

The light gray boxes represent the third-party specific components when a telephony conferencing bridge or MCU is supplying audio/video media to the Sametime Connect client. A third-party adapter could also interface with desk conference phones (not shown), enabling a Sametime Connect user to select a single contact and call them directly, or select several contacts and with a single click, start a conference call using the third-party solution.

This chapter presents the steps necessary to create the TSCPI adapter as an extension of the Sametime Conference Manager above; the next chapter, *Sametime Connect interaction with a third-party MCU*, will elaborate on the messages represented by the dashed SIP and “private third-party protocol” lines above.

A typical service provider has the following attributes.

Table listing service provider capabilities with descriptions.

Capability or function	Description
Audio/video call management	Interfaces with third-party bridge (MCU) to supply audio/video data to the Sametime Connect client or Meeting client.
Control and management of conference calls	Security, call existence, call creation, call moderating are some examples.
Telephone call management (legacy)	Using a combination of PSTN, VoIP, and other telephony technologies.
Valid runtime conference properties	<p>Conferences support these runtime properties:</p> <ul style="list-style-type: none"> • Available – The telephony service is available. • Valid – Once the conference is marked as “invalid” is cannot ever be marked as valid again. • Muted – Muted is a conference property that affects user properties. Therefore, the service provider should set this on users. • Locked – a locked conference prevents additional persons from attending. A value of "false" indicates the service provider should allow joining providing other attendance policies have been considered. • AutoMute – It is a conference property that affects user properties. Therefore, the service provider should set this on users.
Valid runtime user properties	<p>The user model supports these runtime properties. Reference the TCSPI Javadoc to learn more about these properties.</p> <ul style="list-style-type: none"> • Connected – Connected to a conference. • Muted – The user does this by pressing the mute button. • Muted video – The user does this by pausing the video. • Silenced – If the user’s audio and/or video are muted by the moderator (cannot be un-muted). • Voice Level – Percent of volume, for voice level (if supported). • Talking – Indicates if the person is talking. • Role – Moderator or Participant. There is a one-moderator requirement. You cannot have multiple moderators on a call.

You decide how your adapter and bridge communicate

This type of integration does not dictate any particular protocol between the TCSPI adapter and the third-party devices. It requires implementation of a high level set of Java interfaces. The implementation of these interfaces is loaded at runtime by the Sametime Conference Manager to create and control audio and video conference calls. The interfaces and default (abstract) class implementations are provided by the toolkit and summarized in the next section.

Summary of key classes and interfaces

Conference Service Interface

The TCSPI provides Java interfaces, abstract classes, and utility classes that help service providers implement a complete solution. Technically, a service provider implementation is a Java class that implements the Java interface:

`com.ibm.telephony.conferencing.spi.ConferenceService`

However, service providers should not implement the Java interface directly, but instead should extend its abstract Java implementation:

`com.ibm.telephony.conferencing.spi.AbstractConferenceService` (legacy)

`com.ibm.telephony.conferencing.spi.AbstractSipConferenceService` (SIP-controlled)

while implementing the interface:

`com.ibm.telephony.conferencing.spi.ConferenceServiceRequests`

The abstract class provides an abundance of helper methods and default method implementations that are necessary for a successful service provider implementation. It is the starting point for all service provider implementations.

Conference and User Data Model

This document often refers to the Conference and User model from a service provider's point of view. This is always accessed from the `DefaultUser` and `DefaultConference` helper classes.

Your implementation of `ConferenceService`, a subclass of `AbstractConferenceService` or `AbstractSipConferenceService`, will create these in-memory model of conference and users as required to fulfill an incoming request from the Sametime Connect client or your own bridge notifications. Changes to the properties of these instances via the `User.setProperty` and `Conference.setProperty` are forwarded to the `ConferenceService`'s `setUserProperty` and `setConferenceProperties` methods. These common properties are automatically transmitted to the Sametime Connect clients registered to the respective conference. For example, setting the muted attribute of a `User` instance notifies its associated `ConferenceService.setUserProperty` method; the corresponding `deliverUserPropertyResponse` superclass method in `AbstractConferenceService` transmits the new status to each associated client, updating the user interface accordingly.

See the Javadoc and comments of the `processXxx` service request methods of the `MyAVConferenceService` example for further details.

ConferenceServiceRequests interface

This interface defines the methods your ConferenceService implementation must provide in order to complete the functionality defined by the superclass AbstractConferenceService/AbstractSipConferenceService.

Note: Methods that only apply to SIP-controlled conferences are denoted by **. Methods that only apply to non-SIP controlled conferences are denoted by *. This distinction will be further clarified in later chapters.

Conference Lifecycle

processCreateConference

processDeleteConference

processEditConference

processValidateConference

Actions

processDial*

processDialExtended*

processMediaControl

processStartConferenceControl

processStopConferenceControl

processTimeoutError

User management

processAssociate

processDisconnectUser

processRegisterUser

processRegisterUserSip**

processUnregisterUser

Property setting and notification

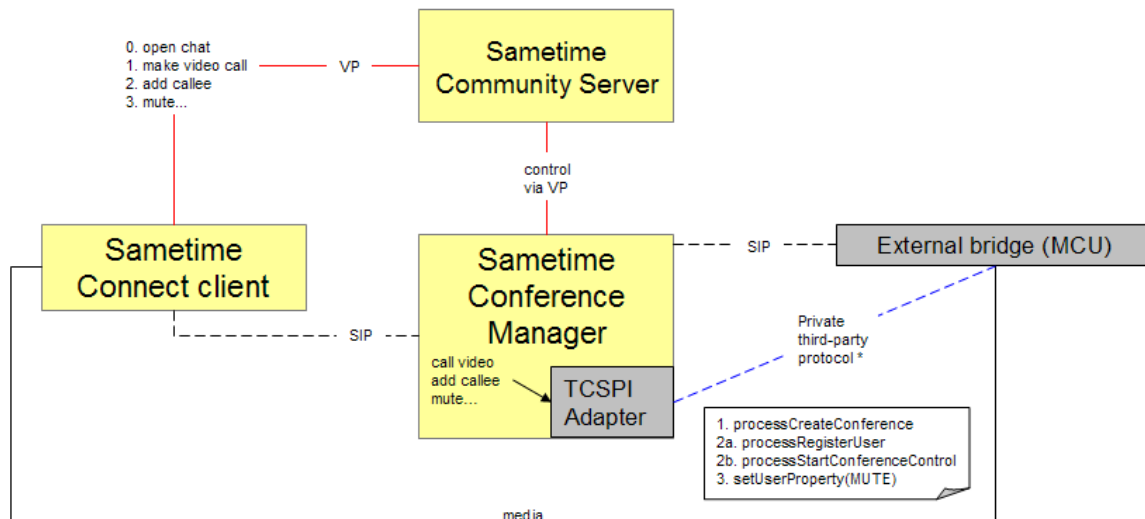
processSetConferenceProperty

processSetUserProperty

Interaction of key classes in sample of user scenarios

Sametime Connect as audio/video endpoint

Below is a pictorial overview of the components involved in an audio/video call initiated from Sametime Connect where the endpoint is the Connect client. This scenario is essentially the same as the “out of the box” audio/video experience provided with the base product, except of course the audio and video are served by a third-party bridge instead of Sametime Media Server’s VMGR. To choose whether calls are hosted by the VMGR or a third-party provider, the Sametime user preferences specify the external service provider that should be used along with the provider’s required information such as conference ID and passcode.



The components involved in an audio/video call initiated from Sametime Connect where the endpoint is the Connect client

When the user selects several users and clicks Call, the Sametime Connect client transmits the request to the Sametime Community Server, which then relays it to the Sametime Conference Manager. Since the caller has identified it as an external (service provider) call, the Conference Manager invokes the register adapter’s ConferenceServiceRequest methods like processCreateConference, processRegisterUser, and processStartConferenceControl in sequence to create, register, and begin the call as requested. The third-party TCSPI adapter communicates with the provider’s bridge (MCU) to reserve the resources necessary for a conference in response to the requests from the Conference Manager. For example, the sequence of requests the service provider adapter should expect for creating and starting a conference are:

processCreateConference

processStartConferenceControl

processRegisterUser (for each participant)

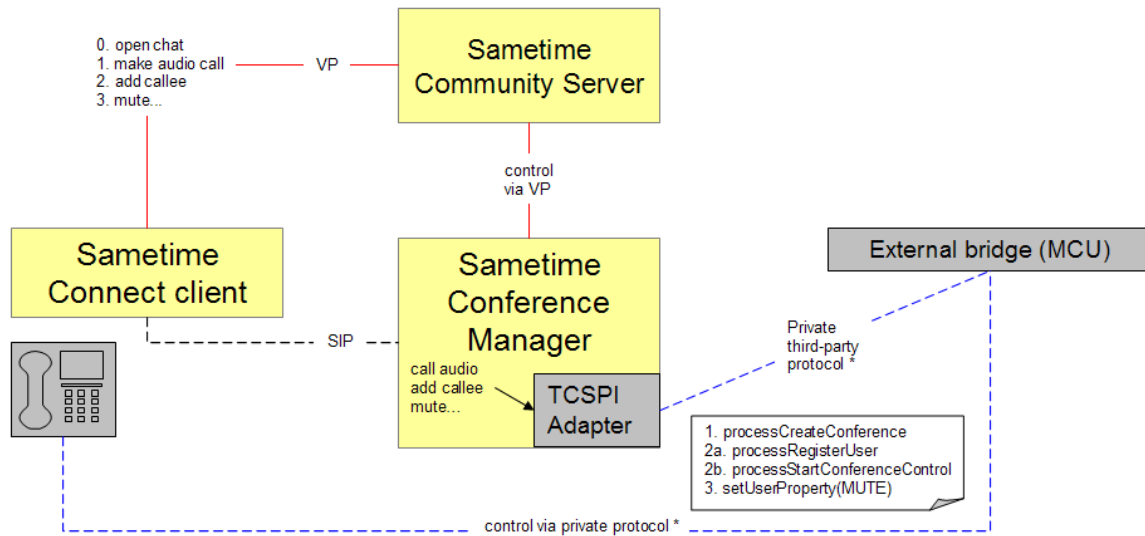
processSetConferenceProperty (sets various values like CallEndedValue = false)

processSetUserProperty (various values like connected status, role, etc.)

For SIP-enabled conferences, each participant is joined into the call with standard SIP messages (i.e., INVITE, OK, and ACK). For legacy non-SIP controlled conferences, the service provider will receive processDial requests for each participant.

Third-party provided non-SIP endpoint (Legacy)

Below is an overview of the components involved in a click-to-call audio call initiated from Sametime Connect where the endpoint is an external device such as a desk-side phone. In this case, the Sametime user preferences specify an external service provider should be used for non-computer audio calls, the preferred number for the device supported by the third party and the provider's required information such as conference ID and passcode.



The components involved in a click-to-call audio call initiated from Sametime Connect where the endpoint is an external device such as a desk-side phone

When the user selects another user and clicks Call, the Sametime Connect client transmits the request to the Sametime Community Server, which then relays it to the Sametime Conference Manager who then forwards the request to the provider's TCSPI adapter. It joins the user into the conference and signals the third-party endpoint. Changes to the status of the conference like user added/removed/connecting/connected/muted are communicated from the bridge to the TCSPI adapter, which updates the User/Conference properties accordingly. The Conference Manager monitors changes to the active instances of User/Conference and forwards them to Sametime Connect client.

Conference management and creation

Conference Management and User Management are the two main areas that the service provider must understand. This section describes conference management and describes the TCSPI methods involved with creating, editing and deleting conferences.

Conference creation

When a conference is created, a serialized XML representation of the initial conference data is passed into the processCreateConference() method.

Conference management process

The conference management process begins when the processCreateConference method is called. Thereafter, it continues during real-time control of the conference. After this, the conference is available for real-time control by the collaboration server.

Your implementation

The service provider processing is shown here in stages. Model your design using this as a guide.

Table describing the stages in server provider processing.

<i>Stage</i>	<i>Item</i>	<i>Description</i>
1	Parse the XML file.	
2	Reserve any resources required to support the conference.	
3	Determine conference-specific information needed.	For example, passcodes and telephone numbers, and ConferenceId.
4	Determine feature support.	
5	Return a full XML representation in deliverCreateConferenceResponse().	This representation should have all conference characteristics.

Conference creation

A fully created conference requires the following attributes and elements. The logs from the sample adapter, MyAV, provide examples of what data types to expect during create as well as what needs population after creating the conference. These are XML files with no length restrictions. However, some of the fields are displayed in the user interface.

Table describing elements of a conference and their functions.

<i>Element</i>	<i>Function</i>
Restricted (true/false)	This indicates if the conference is restricted to allow only registered user participation.
Expected users	This is only a hint to the service provider; the expected number of participating users.
Maximum users	Another hint for the service provider. This is the maximum number of users that should be allowed in the conference. When reached, the service provider may choose to reject additional participants.
Title	A short name for the conference.
Owner	The canonical name of the user on whose behalf the conference should be created. This is not necessarily a participant in the conference. Note: An owner that is also a participant will be explicitly registered through a call to registerUser() at a later time.
Schedule	The conference scheduled times.
	Attributes

	<p>Elements should have an attribute for start time, called <i>time</i>. The element value is represented in milliseconds, as the number of milliseconds since January 1, 1970 00:00:00.000 GMT.</p> <p>Time resolution</p> <p>Service providers are only expected to support time resolution down to minutes. The <i>duration</i> attribute specifies a time in minutes that the conference should be kept active, but is only a hint since the application above the TCSPI will not call stopConferenceControl() until all application participants have left the conference.</p> <p>Note: This field is inherited from previous Sametime releases. The Media Manager conferences are created on demand, thus the notion of a scheduled conference doesn't apply to version 8.5 and beyond.</p>
Features	A list of features that the service provider is either required to support or can optionally support, specified by the 'required' attribute.
ClientID	Tip: ClientID can be used as ConferenceId for a user who may have a dedicated Conference call number.

More on features

For more on features, see the Javadoc for `com.ibm.telephony.conferencing.ConferenceFeature`.

Conference data retrieval

Here are guidelines for conference data retrieval.

Table describing guidelines for conference data retrieval.

<i>Service provider requirements</i>	<i>Description</i>
Return a conference XML document in the response to a processCreateConference() call.	All of the relevant elements and attributes must be filled in.
Externally store any information needing retrieval.	<p>Reason.</p> <ul style="list-style-type: none"> IBM Sametime persists the data, but there is no API to retrieve it. It is passed back to the editConference() and startConferenceControl call.

Persistence

The conference document is persisted by IBM Sametime. It is not possible to call a method to retrieve this data. However, the conference document is passed in on later edits, creates, startConferenceControl, etc.

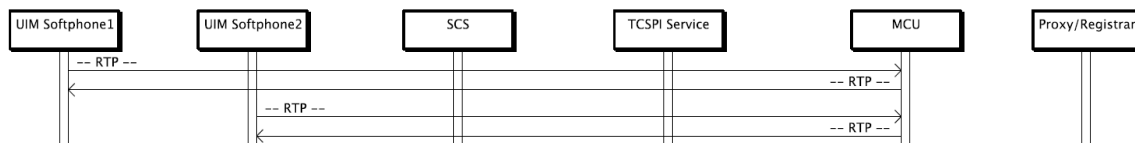
Chapter 4. Sametime Connect interaction with a third-party MCU

In the versions of Sametime prior to 8.5, the Connect client enabled users to initiate conference calls, but the call itself was received on an external device, such as a phone or conferencing system. In versions 8.5 and beyond, the Sametime Connect client can initiate calls as before to third-party external devices, but it can also act as a SIP-enabled endpoint for receiving audio and video data via RTP. This section explains the control flow and introduces how the TCSPI APIs have been extended so the service provider can define SIP-controlled audio/video calls where the Connect client acts as one endpoint and the third-party MCU acts as the other endpoint.

Sametime Connect and SIP-enabled MCU

The prior section defined the components that comprise the Sametime Media Server. The diagram below, *RTP Data Flow Established*, shows the final result of a SIP-negotiated call among these components and a third-party MCU, namely the exchange of audio/video data via RTP between one or more Connect clients, a third-party MCU, and other clients types it supports.

Note: Although two clients could flow data directly between them (a point-to-point call), the diagrams in this section will only show two clients for simplicity; the same control and data flow would apply whether there were two participants or twenty. In the case of an external MCU, the media for point-to-point flows to/from the MCU, not directly between clients.



RTP Data Flow Established

Sequence of Events

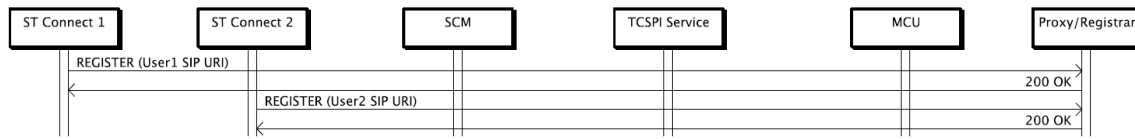
This chapter ends with a full-page diagram, *Start Conference with External MCU*, which shows all the steps and components involved in setting up a SIP-enabled conference. To make it easier to understand, this section will present each major step in a piecemeal fashion.

The steps to starting a SIP-enabled conference are outlined below:

1. Client endpoint registration.
2. Call initiation and conference creation.
3. Conference start and endpoint registration.
4. Dial request and media negotiation (INVITE / OK / ACK).
5. Data flow establishment via RTP.

Repeat previous two steps with other conference participants.

These steps will be depicted in the following series of diagrams, beginning with endpoint registration as shown in *Client Endpoint Registration*:



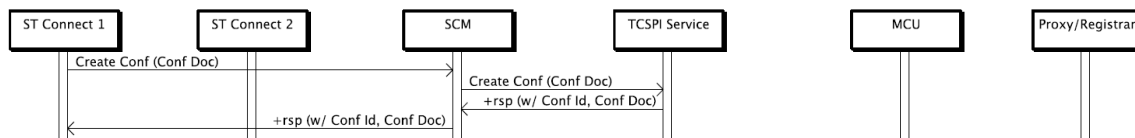
Client Endpoint Registration

Client endpoint registration

In order for SIP-enabled clients to correctly address each other, they register with a common contact point, namely the Sametime Media Server's Proxy/Registrar, where they record their address of record (AOR). When the Sametime Connect client launches, it registers itself with the Proxy/Registrar, whose configuration is defined by the Sametime System Console.

Note: The MCU and Conference Manager themselves are not registered with the Proxy/Registrar, but rather the conferences they manage, as will be explained momentarily.

After the clients register themselves, the user can start an audio/video conference by choosing one or more participants in their Sametime contact list and selecting **Call** or **Video Call**. Information about the newly created conference is stored in an XML format conference document and transmitted to the Sametime Conference Server, as shown in *Call Initiation and Conference Creation*:

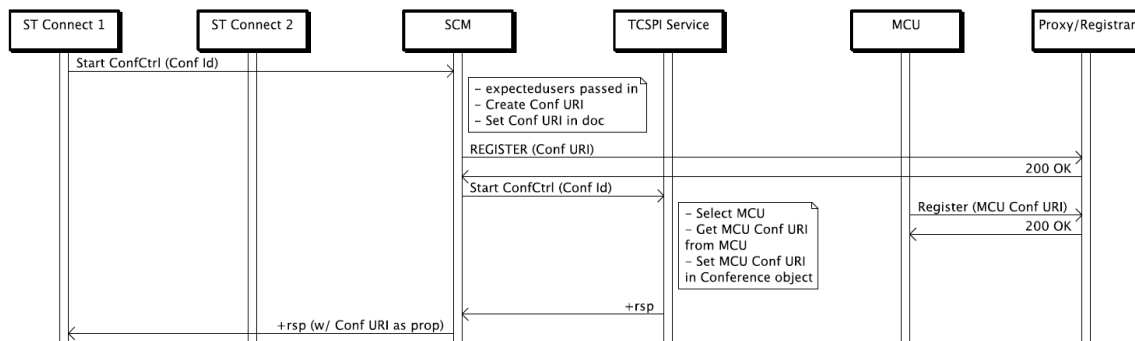


Call Initiation and Conference Creation

Call Initiation and Conference Creation

The Service Provider receives the details of the conference participants, requested features, and similar information via the conference document supplied to its processConferenceCreate method. At this point the service provider defines the conference identifier and takes whatever additional steps are necessary to prepare for the conference (e.g., reserve resources on its MCU). The conference includes among its properties whether it is SIP-controlled; since this scenario describes the interaction between the Sametime Connect client and a SIP-enabled MCU, the service provider should set this property accordingly.

Once conference creation is complete, the service provider reports its success or failure. In the case of failure, the service provider can specify the error message to be shown to the end users. Assuming no errors, the next step is starting the conference itself and registering the necessary conference endpoints, as shown in *Conference Start and Endpoint Registration*:

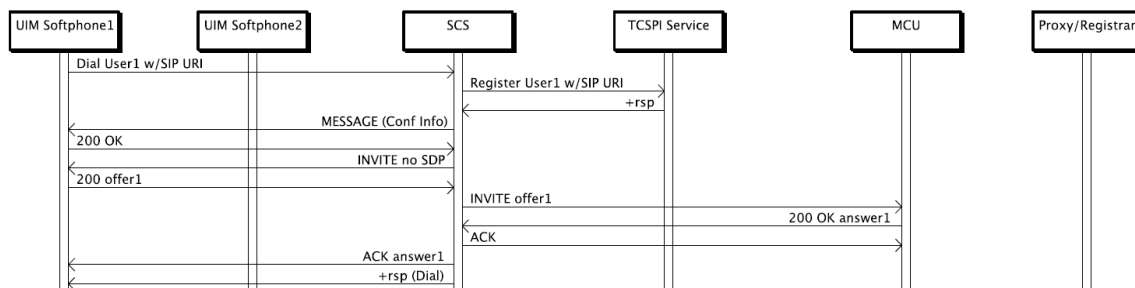


Conference Start and Endpoint Registration

Conference Start and Endpoint Registration

The Service Provider is notified that the conference should begin via its `processStartConferenceControl` method. In response, the service provider should inform its MCU to allocate conference resources and return the assigned MCU SIP URI representing the conference. This endpoint URI will be used for the subsequent INVITE media negotiation between the Sametime Connect clients, Conference Manager, and the third-party SIP-enabled MCU. The service provider is expected to set the MCU conference URI in the conference instance as a result of its `processStartConferenceControl` method so it can be recorded by the Conference Manager.

As before, once conference allocation and registration are complete, the service provider reports the success or failure. Assuming no errors, the next step is initiating the call and subsequent media negotiation between the Sametime Connect client, Conference Manager, and third-party MCU, as shown in *Dial Request and Media Negotiation*:



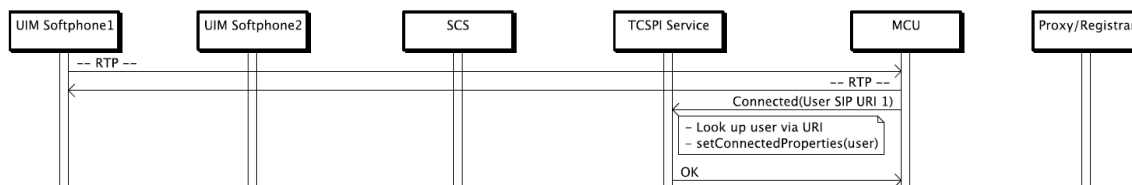
Dial Request and Media Negotiation (INVITE/OK/ACK)

Dial Request and Media Negotiation

The sequence begins with the Sametime Connect client asking the server to “dial” the first user into the conference. For each participant in the conference, the Service Provider’s `processRegisterUserSip` is invoked so it can create a corresponding User instance and inform its MCU, if necessary.

From this point forward, the remaining interaction is via SIP messages between the endpoints with the Conference Manager acting as a third-party call controller. The Conference Manager responds to the initial Connect client’s request to start the call by querying the client’s supported media via an INVITE message without session descriptor protocol (SDP) content specified. The softphone’s response is then proxied to the MCU conference URI previously provided by the service provider during the conference start step.

Its offer, if accepted, is returned to the Sametime Connect client via an OK message, indicating that it should signal its agreement by sending an ACK message and then establishing an RTP session with the other endpoint, as shown in *Data Flow Establishment via RTP*:



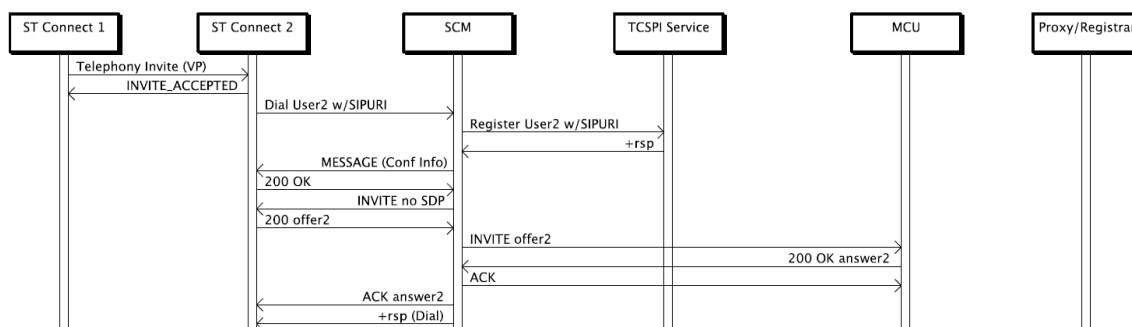
Data Flow Establishment via RTP

Data Flow Established via RTP

The two endpoints, namely Sametime Connect client and the third-party MCU, have now been formally introduced and agreed upon media type and protocol—its time to let the data flow. The MCU should also inform the service provider that the user is joined into the conference so it can convey this status to the Conference Manager and other users in the conference.

Note: TCSPI does not dictate what protocol is used between the third-party service provider and its MCU. That is, the “OK” arrow in the above diagram should not be interpreted as a SIP message, but rather an acknowledgement via the private protocol these two external components use to communicate. For example, in the SDK’s TCSPI sample, TCP/IP sockets are used with the service provider acting as the socket client and its “fake MCU” acting as the socket server.

The remaining steps are similar to the last two steps: The next Sametime Connect client is invited into the conference and establishes its own media stream with the MCU, as shown in *Repeat Media Negotiation and Data Flow*:

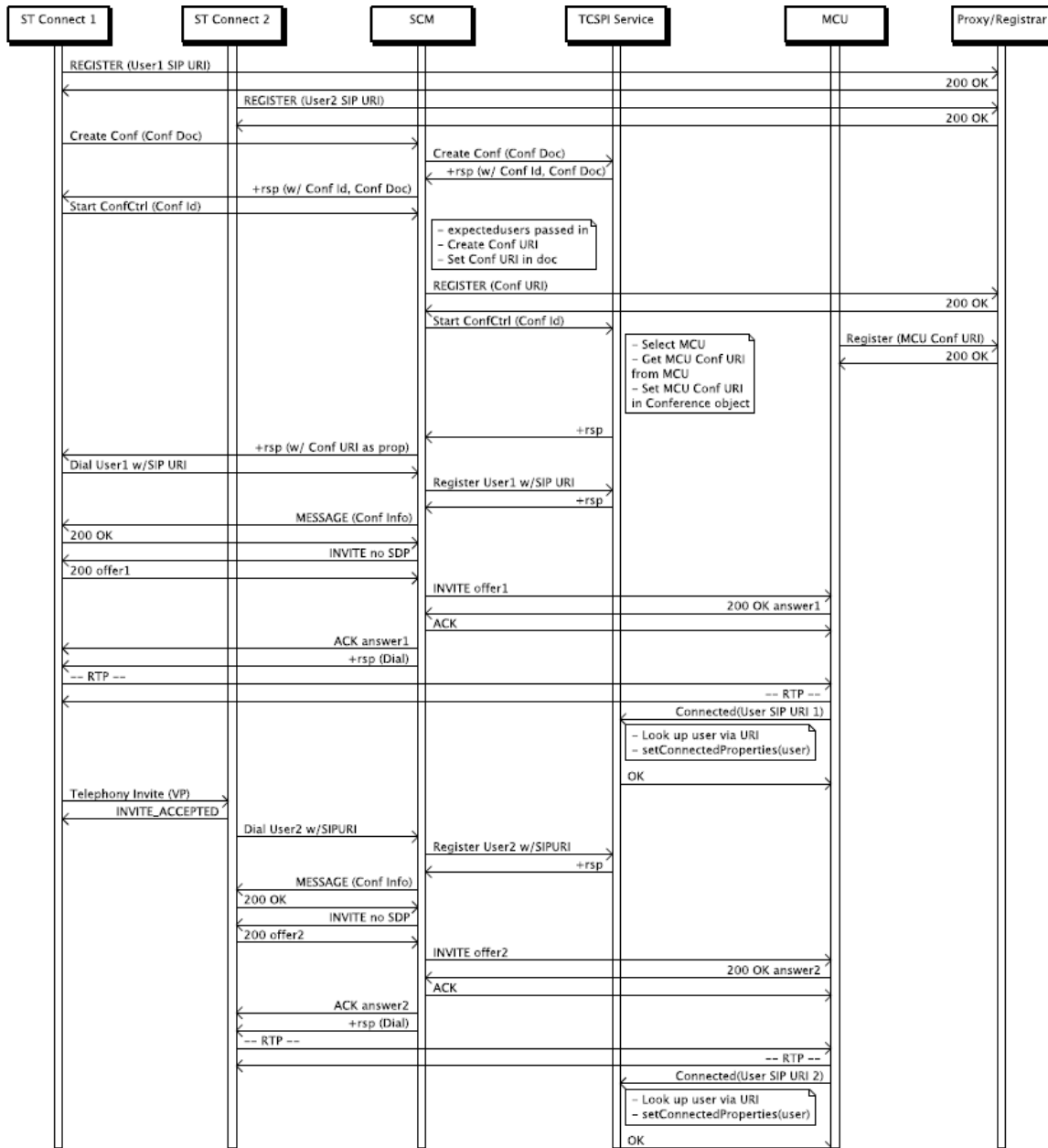


Repeat Media Negotiation and Data Flow

Summary

Working together with its MCU, a service provider can take advantage of Sametime Connect’s internal softphone for conference control and media sharing using SIP control flow and RTP media transfer standards. The next section introduces the SDK’s sample, MyAV, which acts as a template implementation of a SIP-controlled TCSPI adapter.

Interaction diagram: Start conference with an external MCU



Notes:

- Success responses to TSCPI messages like processDial(...) are ignored by the client; only failure responses are processed.
- Typically a two-participant call implies a point-to-point call, but the above shows it as server-hosted for simplicity.
- Actual messages between the TSCPI service adapter and MCU are not shown since this vendor/implementation unique
- Dial requests from the client are treated as processRegisterUser and then processDial by the Conference Server. This is existing behavior from previous releases.

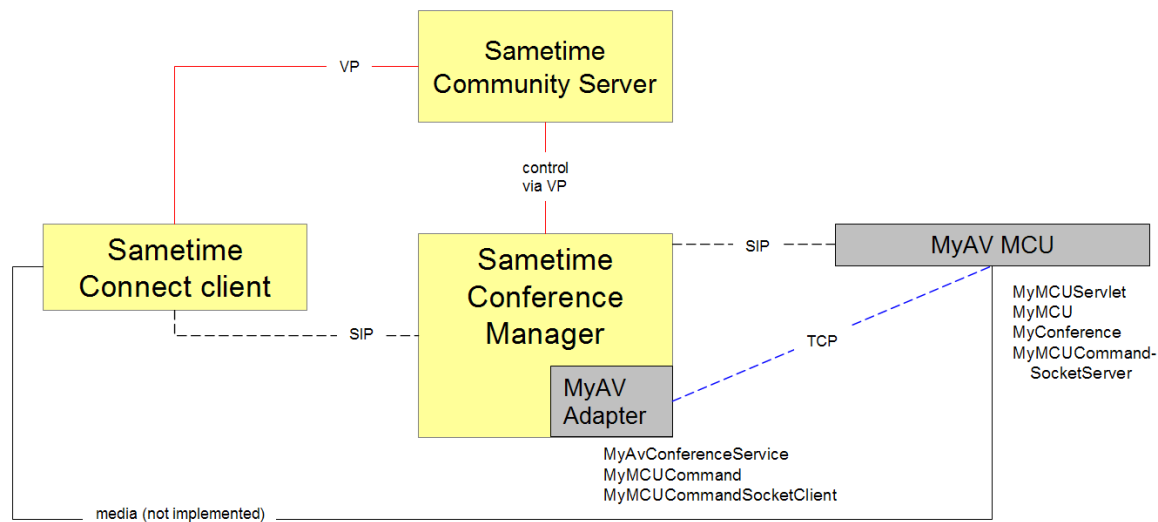
Chapter 5. Introduction to the MyAV sample service provider

The Service Provider interface was discussed in Part I and an overview of the steps in creating a SIP-enabled conference was discussed in the previous section; this section covers the SDK sample that demonstrates how to use these APIs.

My A/V sample

The TCSPI provides a sample implementation, called the “MyAV Conference Service” service provider. This section describes the MyAV service provider supplied with the TCSPI. You can refer to this sample implementation when writing your own service provider. The MyAV service provider is not a fully functional service provider; it is a template with “todo” flags indicating where your implementation-specific code should be inserted. Although the adapter itself doesn’t actually transfer media to the client, the structure of the code and the traces it generates demonstrates the code flow you will need to understand to complete your own implementation. It can help you learn to model the use cases and methods that you believe will be needed for your own service provider. The source code for the MyAV service provider is included with the Sametime SDK.

The diagram below shows the classes that comprise the sample overlaying a simplified topology similar to the one shown in the architecture introduction:



The classes that comprise the sample overlaying a simplified topology

Note the dashed “private third-party protocol” line between the adapter and MCU has been replaced with TCP. The MyAV uses a TCP socket client/server to communicate requests from the adapter to the MCU; similarly, the MCU uses a TCP socket to communicate notifications from it to the MyAV adapter.

Service Adapter Classes

Below is a summary of the key classes of the sample adapter code that is installed on the Conference Manager server (com.ibm.telephony.conferencing.myav.jar).

MyAVConferenceService – subclasses the telephony toolkit framework **AbstractSipConferenceService**, acting as the central handler for telephony requests (creating conference, registering users, updating user properties, etc.)

MyMCUCommand – encapsulation of the requests the MCU carries out at the request of **MyAVConferenceService**.

MyMCUCommandSocketClient – TCP transport handler for **MyMCUCommand** requests to **MyMCU** and incoming notifications from **MyMCU**; it runs on a dedicated thread created during the service startup.

CommandDispatcher – defined by the Telephony Services framework, an instance of this class dispatches the execution of **MyMCUCommand** instances; it runs on a dedicated thread created during startup.

The central class, **MyAVConferenceService**, handles service requests in its **processXxx** methods from the Conference Manager (e.g., **processCreateConference**, **processStartConferenceControl**, **processRegisterUser**). These **processXxx** methods are invoked as part of the processing of requests to the Conference Manager's own (internal) service provider. For example, when the Conference Manager's internal service provider receives a **ConferenceService.createConference** request, it handles the preliminary setup and then calls the registered external service adapter's **processCreateConference** method to complete the desired functionality.

MCU Classes

Below is a summary of the key classes of the sample MCU code that is installed as a J2EE Enterprise Application (myav.ear). It can be co-located as an application on the same server as the Conference Manager, or can be installed on a separate server.

MyMCU – simple “fake MCU” that coordinates the handling of requests from and notifications to **MyAVConferenceService**.

MyMCUServlet – **Servlet** responsible for routing appropriate SIP messages to **MyMCU** or **MyMCUConference** instance.

MyMCUConference – simple “fake conference” that manages conference identity and lifecycle.

MyMCUCommandSocketServer – TCP transport handler for incoming requests from **MyAVConferenceService** and outgoing notifications.

Note: The controller class **MyMCU** and its data model class **MyMCUConference** are minimal implementations; however, they should be sufficient to serve as initial prototypes for interfacing with your specific MCU's APIs.

Chapter 6. Implementing the Telephony Service Provider, Part II

Extending the Default Conference and User

The implementations of the User and Conference interfaces, DefaultConference and DefaultUser, provide the data required by the Sametime Conference Manager such as the user's name, PIN, speaking status, and so forth. However, your implementation may wish to extend the data associated with these instances for your own use. The Conference.setDelegate/getDelegate and User.setDelegate/getDelegate methods are provided for this purpose. Your implementation can set these values for the user/conference instances created in your adapter's processCreateConference and processRegisterUser methods and later query them as required in other methods of your ConferenceService implementation.

Defining your supported required and optional features

Some features are required for your implementation. There are optional features. This section describes a few of them, and presents an example.

Conference features are specified as XML

When a conference is created and started, a conference document in XML format is provided to describe the features it will support. Below is an example conference document template with several features highlighted; values shown in braces ({1}, {2}, etc.) would be replaced at runtime with the actual values provided by the user.

```
<conference restricted="false" expectedusers="2" maxusers="20">
  <title><![CDATA[{0}]]></title>
  <owner>
    <user>
      <name><![CDATA[{1}]]></name>
    </user>
  </owner>
  <feature required="true">com.ibm.telephony.conferencing.feature.DialIn</feature>
  <feature required="true">com.ibm.telephony.conferencing.feature.SpeakingIndication</feature>
  <feature required="true">com.ibm.telephony.conferencing.feature.MuteUser</feature>
  <feature required="true">com.ibm.telephony.conferencing.feature.DisconnectUser</feature>
  <feature required="true">com.ibm.telephony.conferencing.feature.DialUser</feature>
  <feature required="false">com.ibm.telephony.conferencing.feature.LockConference</feature>
  <feature required="false">com.ibm.telephony.conferencing.feature.MuteConference</feature>
  <feature required="false">com.ibm.telephony.conferencing.feature.UserVolumeControl</feature>
  <property readonly="true" type="com.ibm.telephony.conferencing.property.SystemPropertyType">
    <name>com.ibm.telephony.conferencing.property.ServiceLocation</name>
    <value>{2}</value>
  </property>
  <property readonly="true" type="com.ibm.telephony.conferencing.property.SystemPropertyType">
    <name>com.ibm.telephony.conferencing.property.Passcode</name>
    <value>{3}</value>
  </property>
  <property readonly="true" type="com.ibm.telephony.conferencing.property.SystemPropertyType">
    <name>com.ibm.telephony.conferencing.property.ConferenceOptionOne</name>
    <value><![CDATA[{4}]]></value>
  </property>
  <property readonly="true" type="com.ibm.telephony.conferencing.property.SystemPropertyType">
    <name>com.ibm.telephony.conferencing.property.ConferenceOptionTwo</name>
    <value><![CDATA[{5}]]></value>
  </property>
  <property readonly="true" type="com.ibm.telephony.conferencing.property.SystemPropertyType">
    <name>com.ibm.telephony.conferencing.property.ConferenceOptionThree</name>
    <value><![CDATA[{6}]]></value>
  </property>
  <property readonly="true" type="com.ibm.telephony.conferencing.property.SystemPropertyType">
    <name>com.ibm.telephony.conferencing.property.ConferenceOptionFour</name>
    <value><![CDATA[{7}]]></value>
  </property>
</conference>
```

Example conference document

Your telephony hardware

Feature evaluation must be done in the context of your hardware capabilities. Remember that IBM Sametime has features that must be supported by your telephony hardware. So check to make sure that your hardware minimally supports required features as well as the optional ones that you plan to use.

Guidelines for choosing or using features

You must use certain features for your implementation of the TCSPI. However, other features are not required. As you design your implementation, you should evaluate optional features. Use this table to help guide your design process.

Table listing guidelines for using TCSPI features.

<i>If a feature is</i>	<i>Then you must</i>	<i>Comments</i>
Required	Use this feature as part of your implementation.	
Optional	Evaluate using the feature	Evaluation criteria should include user experience, expected behavior, and telephony hardware capabilities.

Features have an impact on the user interface, or visualization. Here are some of the required features for a typical conference, with corresponding information showing the visualization impact (if any). At minimum, service providers must implement required features.

Table describing required features for a typical conference.

<i>Feature</i>	<i>Required or Optional</i>	<i>How this affects the IBM Louts Sametime visualization in the UI</i>
MUTE_CONFERENCE	Optional	IBM Sametime visualization has no UI for this feature.
MUTE_USER	Required	“Muted” shown in user interface.
DIAL_IN	Required	The dial in numbers are strings. The sample service provider has dummy numbers. If your PBX does not deal with dial in numbers, you may decide to substitute some text such as dial in not supported.
DIAL_OUT	Required	Dial out is pervasive in the application and must be supported for telephone conferenes.
PIN_COLLECTION	Optional	IBM Sametime visualization has no UI for this feature.
RECORD_CONFERENCE	Optional	IBM Sametime visualization has no UI for this feature.
SIP_DIAL	Optional	IBM Sametime visualization has no UI for this feature. Assumes a string data type for the telephone number.

SPEAKING_INDICATION	Required	If this is not used, then IBM Sametime cannot send events. Visualization will not display talking icons.
USER_VOLUME_CONTROL	Optional	User interface displays volume control dialog if this is supported.

Handling failure scenarios

This section provides guidance on handling failure scenarios. There are four scenarios presented here. Use these to help plan aspects you need to plan for into your service provider implementation.

System failure scenarios

The following failure scenarios deserve consideration when designing and developing your implementation. There may be other cases as well dependent upon the hardware software, and feature decisions.

In brief, you must at least cover these failure scenarios:

- IBM Sametime server failure
- Service provider failure
- Audio/video bridge failure
- Network connection cannot be restored.

Scenario 1. The IBM Sametime server fails

Consider that the IBM Sametime server could potentially fail in multiple ways. Some failures are serious enough to require a server restart. The guidelines are as follows:

- If the IBM Sametime server is restarted then the service provider will also be constructed and started.
- Conferences will be started and controlled as web conferences are started.
- Service providers should respond to multiple restarts without affecting the state of an audio/video conference.

Scenario 2. The service provider fails

In scenario 2, the expected behavior is that service providers are equipped to handle internal exceptions and recover from network connection drops without affecting the integrity of the conference in user model. Network connections should be retried until they are restored. Once network connections are restored, the model should be updated to reflect the actual state of running audio/video conferences.

Table listing expected behavior in case of failure.

<i>Asynchronous request potential failure</i>	<i>What your service provider should do</i>
During the initial method call.	The service provider should throw a conference exception inside the process method indicating a failed asynchronous request.
After the initial process, method called has succeeded.	If this happens the service provider should call the helper method, <code>deliverFailureResponse()</code> with an appropriate error.

Scenario 3. The audio bridge fails

The audio bridge fails In failure scenario 3, the expected behavior is that web conferences should be able to continue even though the audio bridge has failed. Conference objects should be marked as unavailable until the audio bridge recovers. Follow these guidelines.

- If users are disconnected from the conference, then user objects should be marked as disconnected.
- Once the bridge recovers, the users will then be able to dial back into the conference.

Scenario 4. The network connection cannot be restored

In the case where the service provider fails and is not able to restore connection.

- `deliverFailureResponse()` for any subsequent request from conferences that are running. For example, the conferences for which `startConferenceControl()` has succeeded and the model has been built.
- This does not affect the conference and the user state, Connection, Mute, Lock, Silenced etc. This should be updated (refreshed) if the connection is restored. This state can be cleaned up if `stopConferenceControl` is called.
- `deliverFailureResponse()` for any calls to `startConferenceControl()` and everywhere it makes sense, for example, such as create, edit, and delete.

Asynchronous response handling

When writing a service provider, create a java class that extends `AbstractConferenceService` or `AbstractSipConferenceService`. This is done by implementing the `ConferenceServiceRequests` interface.

The TCSPI does not collect events from telephony hardware

The TCSPI does not provide any special mechanism for collection of asynchronous events from telephony hardware. This is completely up to the implementation of the service provider.

Your implementation could accomplish this with a web service or with something as simple as a protocol over a TCP socket established with the telephony server/bridge.

Correlate with the Conference ID

Whatever the implementation choice, a response or incoming event should correlate to a specific IBM Sametime conference. This should be done by means of the conference ID. The conference ID is established when IBM Sametime calls the service provider with `processCreateConference`.

Example 1

During a conference call, the telephony hardware publishes events with a unique conference or call id. In this example, you could use this ID as the conference ID you return when `processCreateConference` is called on your service provider. The model allows your service provider to lookup this Conference object by its ID whenever it is appropriate to do so.

Example 2

Another option is helpful if you have additional information that needs association with the conference object. In this example, you could to use the delegate object on the Conference.

The process

When you receive a `processCreateConference`, you should fill in all of the details of the conference you know about at that time. Sametime persists this data. It is then presented on subsequent calls to `processCreateConference` or `processEditUpdate`.

Process and deliver methods are paired

There are method pairings in place to support the asynchronous nature of all service provider calls.

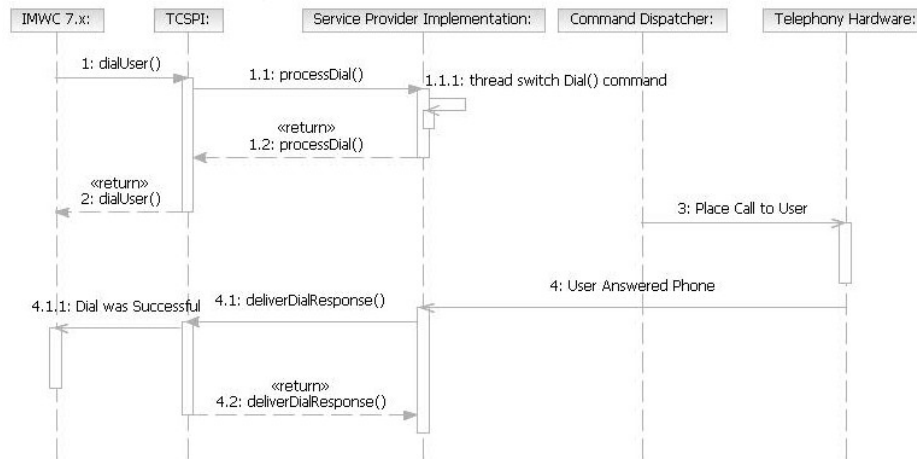
- Most of the names of the abstract methods start with the word, “process”.
- Most methods have a matching helper method whose name starts with the word, “deliver”.

Process and deliver sequences

The process and deliver sequences do not need to be synchronous. For example, a process delivered does not mean the user is connected when complete. This only indicates that the dial process has been started. Later on, an attribute to update will result in the final state.

Example: Dial methods

The abstract method to dial a user, “processDial”, has a matching helper method, “deliverDialResponse”. Within this context, a service provider implements the “processDial” method to dial a user. Once the dial has completed successfully, the service provider calls “deliverDialResponse”.



Methods for dialing

Understanding callbacks

The abstract “process” methods cannot return results directly. Instead, they all have an argument called the “ConferenceResponseHandler”. This serves as a callback receive the results.

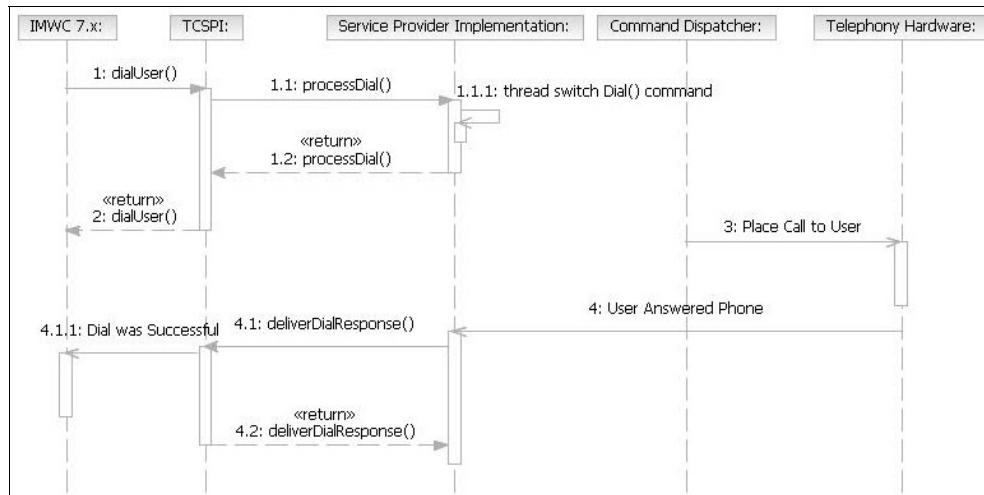
For each request, these handler objects are passed in as new instances. Service providers can use the handler instance as a key to match requests to responses.

Deliver methods

To help deliver these results through the “ConferenceResponseHandler” callback class, each abstract “process” method has a corresponding “deliver” helper method. Either the “deliver” methods can be called inside the “process” methods, or later, hence the asynchronous nature of the method calls.

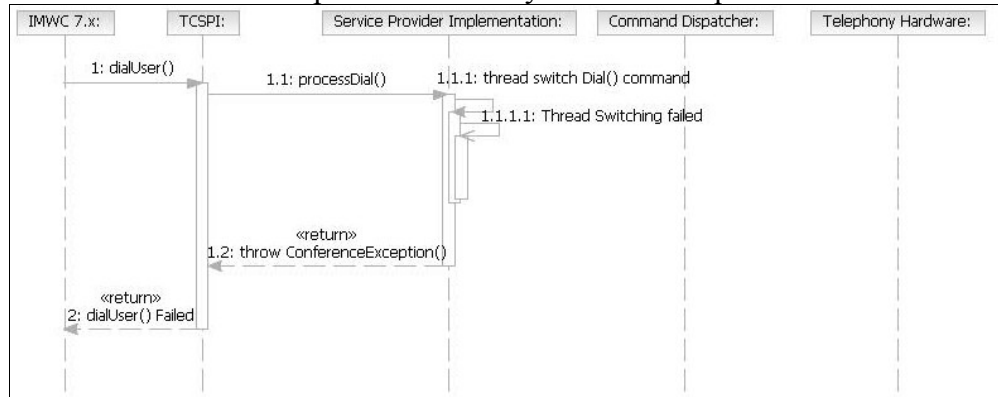
processDial() Comparing success to failure

This diagram illustrates a successful call to abstract method, process dial.



Successful call to abstract method, process dial

Example of a failed asynchronous request.



Failed asynchronous request

Handling request timeouts

This section provides guidelines on request timeouts. Although timeouts can occur for a number of reasons, you can build timeout controls into your implementation of the service provider. Each asynchronous request in the TCSPI has a configurable timeout value specified in the `ConferenceManager.properties` file.

Service provider responsibilities

Service providers must call a corresponding “deliver” method for every “process” request within the timeout period or the operation is considered timed out.

Timeout process

If a timeout occurs, the method `processTimeoutError()` will be called with the `ConferenceResponseHandler` object that was passed into the request.

Service providers must then call the `requestFailed()` method and cancel the pending request. Additionally, service providers should always call the `listenerIsActive()` method before delivering a response to determine whether a timeout has been reached for a request.

Important note, if a service provider tries to deliver a response to a timed out request, the “deliver” method throws a `TimeoutException`.

Canceling pending requests

The service provider implementation of TCSPI sends the request to its telephony hardware. Model your implementation so that it includes telephony hardware notification.

Tips on notifying users

When timeouts occur users should be notified by displaying something meaningful on their screen.

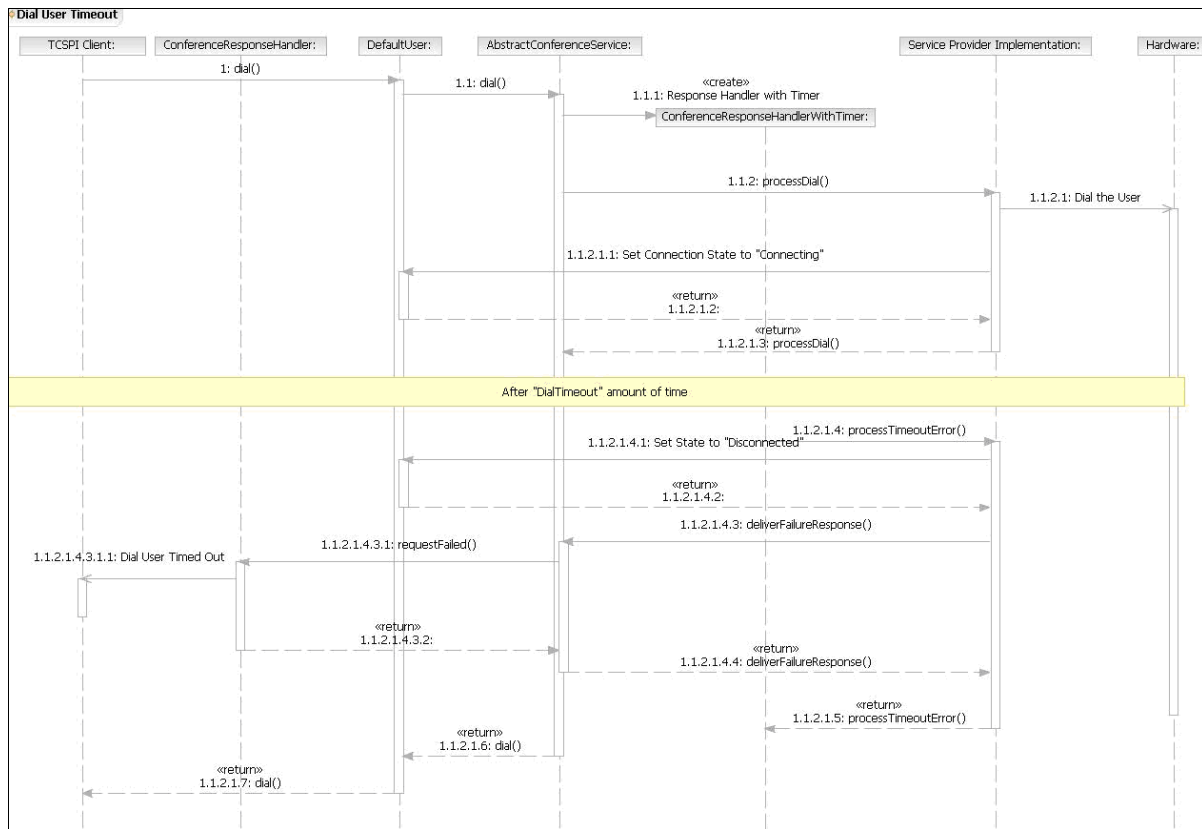
Fields

You may use the *Error Code* and *Error Text* fields defined within *AbstractConferenceService* to notify users using the Sametime Connect client.

Testing

To test `deliverFailureResponse()`, try to return a failure response on `processDial()`. This prompts the user with a dial error.

This UML diagram shows the process flows corresponding to a dial timeout.



Process flows corresponding to a dial timeout

Appendix A: Setting up your development environment

Install Rational Application Developer; it is included in the WebSphere Application Server v7.0 distribution and trial versions are available for download from IBM.



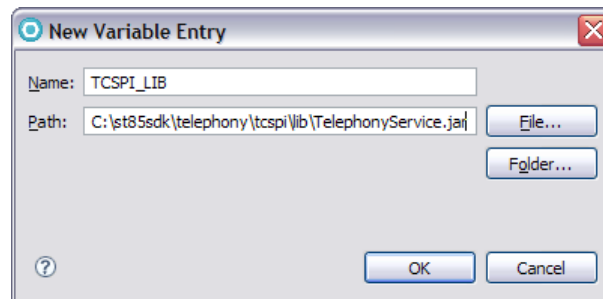
IBM Rational Application Developer

Copy the TelephonyService.jar from the SDK telephony\tcspi\lib directory or your Sametime Media Server installation to your local development environment. The service jar is located here:

C:\IBM\WebSphere\AppServer\profiles\STMSAppProfile\installedApps\<server>MediaCell\ConferenceFocus.ear\ConferenceFocus.war\WEB-INF\lib\TelephonyService.jar (Windows)

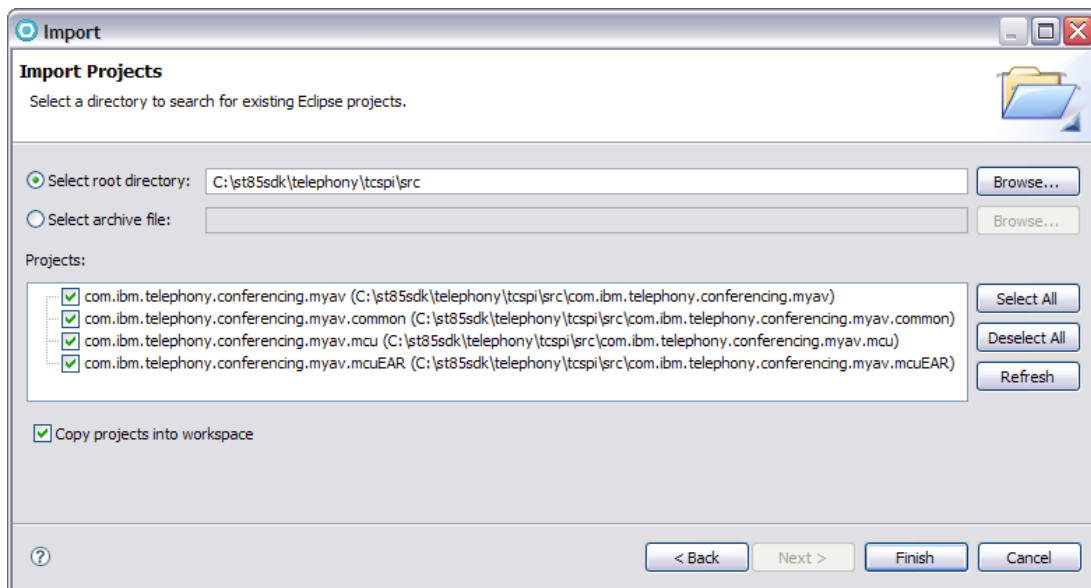
/opt/IBM/WebSphere/AppServer/profiles/STMSAppProfile/installedApp/.../TelephonyService.jar (Unix)

Define the classpath variable TCSPILIB (Window > Preferences > Java > Build Path > Classpath Variables > New):



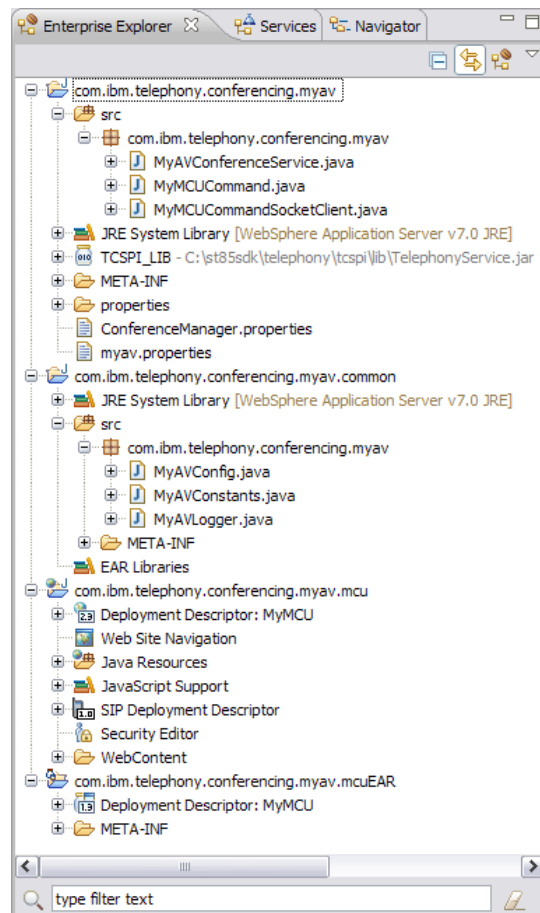
New Variable Entry dialog box

Import the sample projects (**File > Import > Existing Projects into Workspace**). Remember to select the checkbox “Copy projects into workspace”:



Import dialog box

The code should compile without errors. If it does not compile cleanly, double-check the WAS JRE is correctly checked as the default JRE (**Window > Preferences > Java > Installed JREs > WebSphere Application Server v7.0 JRE**) and the classpath variable for TCSPI_LIB is valid.



Checking the WAS JRE in Enterprise Explorer

Appendix B: Installing the MyAV sample

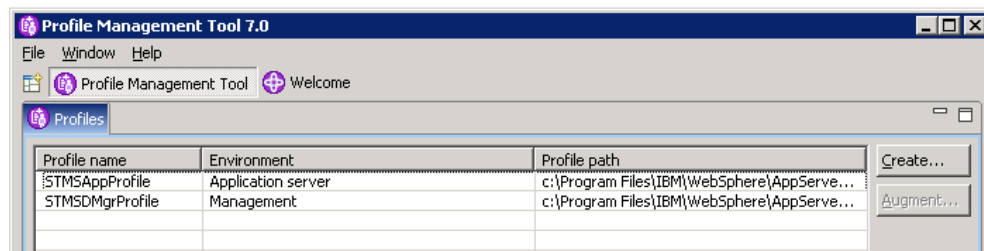
The MyAV sample is comprised of two components that can be installed on separate servers:

- TCSPI Service Provider Adapter (com.ibm.telephony.conferencing.myav.jar) – must be installed on the Conference Manager server,
- MyAV MCU application (com.ibm.telephony.conferencing.myav.mcu.ear) – can be installed on Conference Manager server or a separate WebSphere Application Server.

These instructions assume the MyAV MCU application (com.ibm.telephony.conferencing.myav.mcu.ear) will be installed on the Conference Manager server in a separate WebSphere profile.

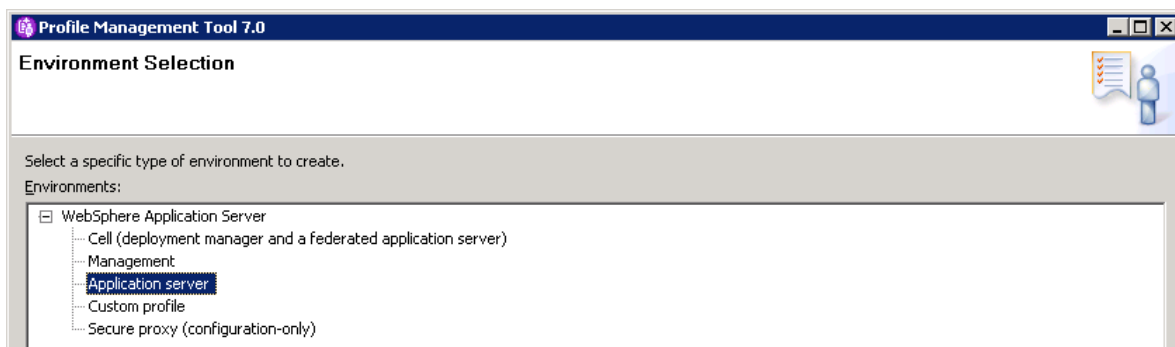
Creating a new WebSphere Server Profile

Start the WebSphere Profile Management tool (**Start > All Programs > IBM WebSphere > Application Server Network Deployment > Profile Management Tool** on Windows and `/opt/IBM/WebSphere/AppServer/bin/ProfileManagement/pmt.sh` on Unix) and then select **Create**.

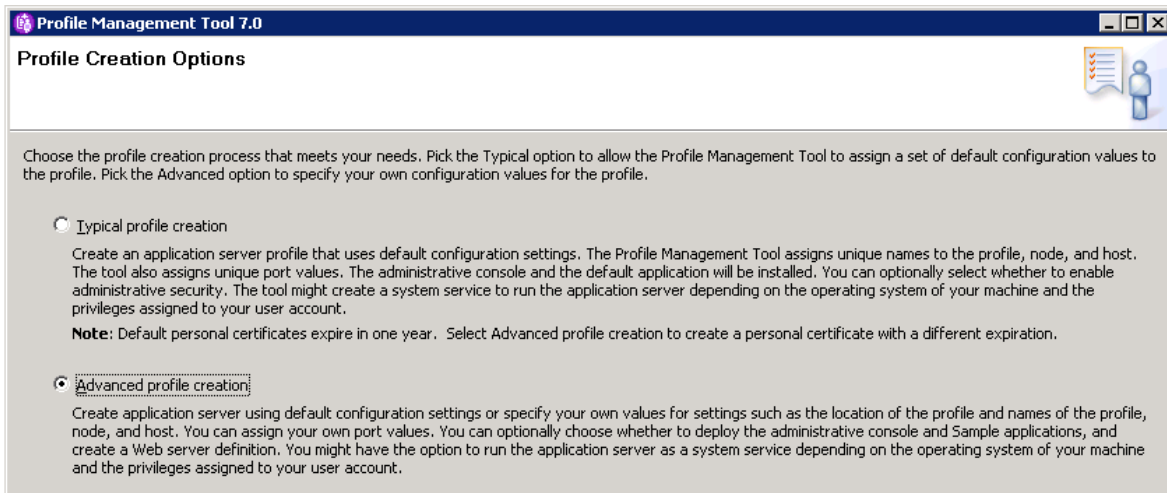


WebSphere Profile Management Tool

Select **Application server** and then **Next**:

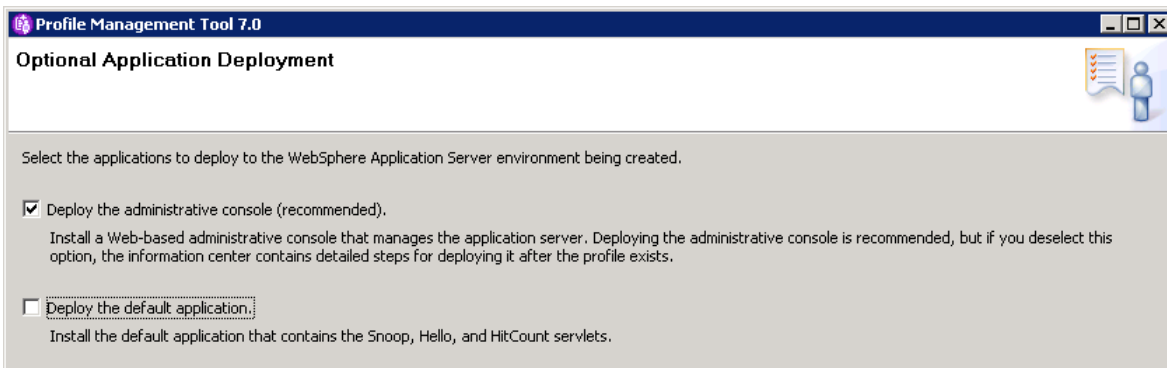


Choose Environment



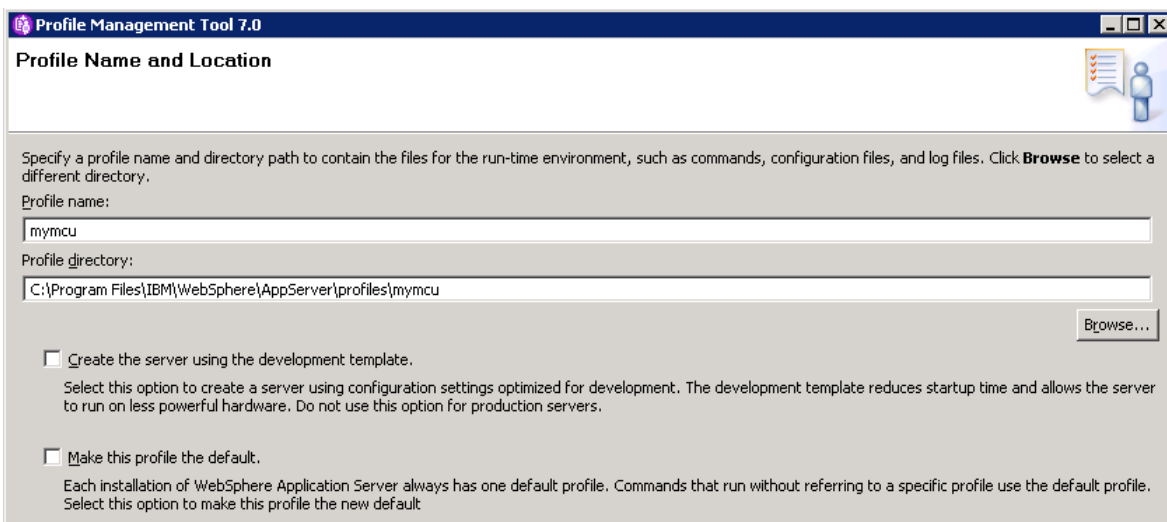
Advanced profile creation options

Uncheck **Deploy the default application** since there is no need for it and then select **Next**.



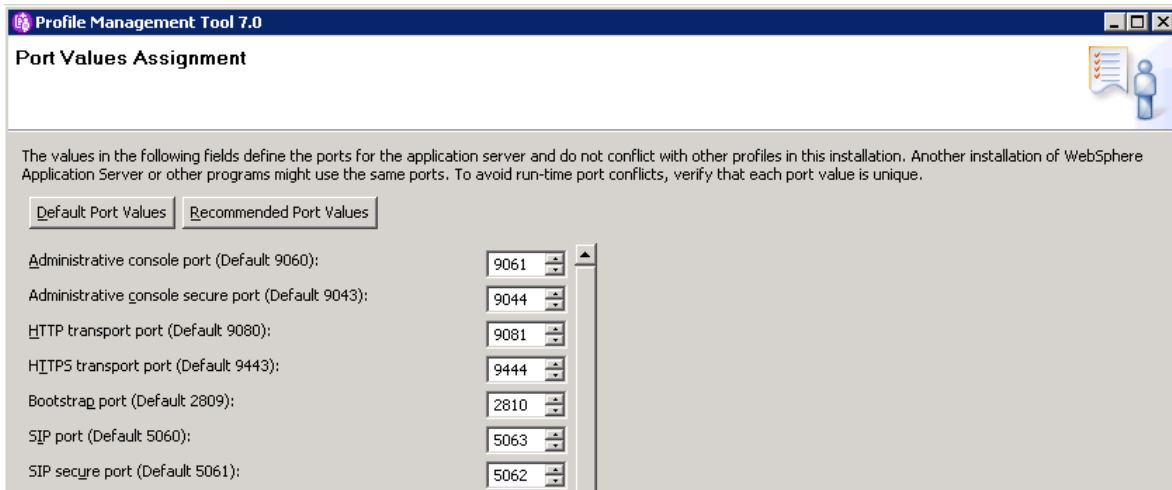
Optional application deployment – default applications are unneeded

Update the profile name and directory as shown below and select **Next**.



Profile name and location “mymcu”

Accept the defaults for the **Node and Host Names**, **Administrative Security**, and **Security Certificate** (Part 1 and 2) pages. Make note of the assigned ports on the **Port Values** page, especially the administrative console and SIP ports since you will need them later to complete the configuration. For the remaining pages, the default choices are acceptable. Continue until the Profile Creation Summary page and select **Create**.



Default Port Values	Recommended Port Values
Administrative console port (Default 9060):	9061
Administrative console secure port (Default 9043):	9044
HTTP transport port (Default 9080):	9081
HTTPS transport port (Default 9443):	9444
Bootstrap port (Default 2809):	2810
SIP port (Default 5060):	5063
SIP secure port (Default 5061):	5062

Note port value assignments, especially Administrative console and SIP ports

The last page offers to launch the First Steps application; launch it and then select its **Start the server** choice. Or you can start it from the command line:

```
"C:\Program Files\IBM\WebSphere\AppServer\profiles\mymcu\bin\startServer.bat"
server1 (Windows)
```

```
/opt/IBM/WebSphere/AppServer/profiles/mymcu/bin/startServer.sh server1 (Unix)
```

The next section describes how to install the service provider JAR, `com.ibm.telephony.conferencing.myav.jar`, after which you will install the `com.ibm.telephony.conferencing.myav.mcu.ear` enterprise application MyMCU in the server profile you just created.

Installing the MyAV service provider

Copy the contents of the SDK's `telephony\tcspi\sametime_tcspi` directory to a new subdirectory, `sametime_tcspi`, located in the Conference Manager root directory:

```
C:\IBM\WebSphere\AppServer\profiles\STMSAppProfile (Windows)
```

```
/opt/IBM/WebSphere/AppServer/profiles/STMSAppProfile (Unix)
```

The contents of this directory are briefly described below:

- `ConferenceManager.properties` – specifies service provider class and configuration
- `myav.properties` – sample-specific configuration such as ports used and debug output
- `com.ibm.telephony.conferencing.myav.jar` – compiled Java code for service provider classes

- properties directory – translated messages used by audio-video component

The myav.properties file defines the sample's configuration. These should be updated to match your installation:

MYMCU_HOST_NAME=your.mymcu.server

MYMCU_HOST_COMMAND_PORT=50001

MYMCU_HOST_SIP_PORT=5063

MYMCU_HOST_SIP_PORT_SECURE=5062

MYMCU_HOST_SIP_TRANSPORT=TCP

Note: The variable MYMCU_HOST_NAME must be the fully qualified server name. The variable MYMCU_HOST_SIP_PORT and MYMCU_HOST_SIP_TRANSPORT must be set equal to the SIP port and transport associated with the host defined for the server indicated by MYMCU_HOST_NAME.

The remaining values for myav.properties are set to the same values defined in the Sametime System Console as shown below:

The screenshot shows the 'Server Integration' section of the Sametime System Console. Under the 'Community server' tab, the following configuration is visible:

- *Host name: stcommunity.lotus.com
- *Port: 1516
- SIP proxy registrar
 - *Host name: STProxyRegistrar.lotus.com
 - *Port: 5080
 - *Transport protocol: TCP
 - Session Expiry: 150

Below the Session Expiry field, a note states: 'Used to control how often messages are sent to clients to verify connection (30s-300s)'.

Sametime System Console

Note: The Sametime System Console copies these values to the server in a file called stavconfig.xml. Specifically, this file contains an XML representation of the configuration values used by the Media Server:

```
<configuration lastUpdated="..." name="SIPProxyServerHost" value="
your.proxyreg.server"/>
<configuration lastUpdated="..." name="SIPProxyServerPort" value="5080"/>
<configuration lastUpdated="..." name="SIPProxyServerTransportProtocol" value="TCP"/>
```

These values correspond to similarly named keys for myav.properties. Set these values defined in myav.properties equal to those specified for the **SIP proxy registrar** the Sametime System Console:

SIP_PROXY_HOST=your.proxyreg.server

SIP_PROXY_PORT=5080

`SIP_PROXY_PORT_SECURE=5081`

`SIP_PROXY_TRANSPORT=TCP`

Note: Port 5060 is the default unsecured SIP port, but it is frequently reassigned to other port numbers, depending on the installation. In the case of the default Conference Manager installation, it is assigned to port 5080 with transport of TCP and port 5081 with transport of TLS. Confirm the `SIP_PROXY_*` values from `myav.properties` match the values specified in the Conference Manager `stavconfig.xml` file, otherwise the MCU conference will fail to register and Sametime Connect clients will be not be able to join a conference. Also confirm that the `MYMCU_HOST_SIP_PORT` value matches the SIP port defined in the MyAV MCU server and that there are no port conflicts.

Once the `myav.properties` file is configured, copy it to the `mymcu` server profile created in the previous section to new subdirectory, `sametime_tcspi`, located in the server root directory:

`C:\IBM\WebSphere\AppServer\profiles\mymcu (Windows)`

`/opt/IBM/WebSphere/AppServer/profiles/mymcu (Unix)`

This file has the same content as `myav.properties` configuration file used by the MyAV adapter. For easier debugging, you may change the options for separate logging as shown below:

```
# OPTION 1: Consolidated log for MyAV adapter
DEBUG_LOG_LOCATION=C:/MyAV.log

# OPTION 2: Consolidated log for MyMCU enterprise application
DEBUG_LOG_LOCATION=C:/MyMCU.log

# OPTION 3: For separate logs per class, set to directory (e.g., C:/)
DEBUG_LOG_LOCATION=C:/
```

Otherwise, you can disable `myav.properties` logging option:

```
# OPTION 4: For no separate file logging, set to an empty string
DEBUG_LOG_LOCATION=
```

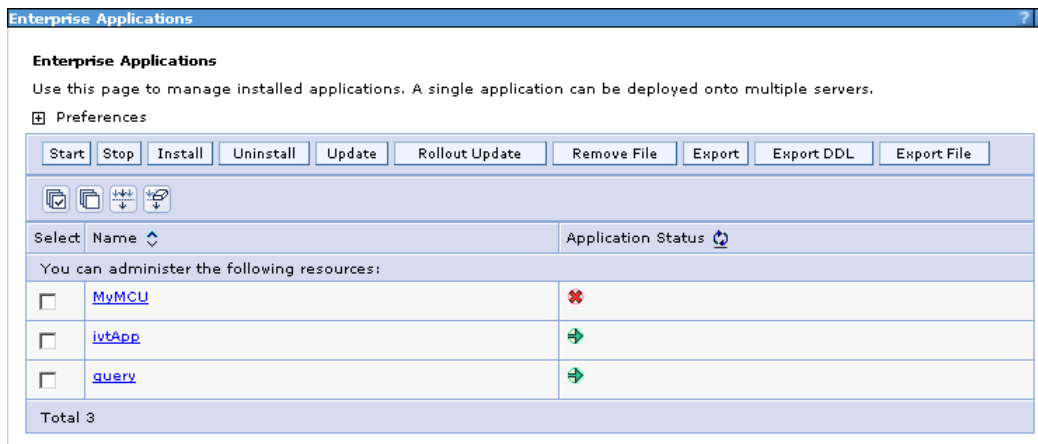
In this case, logging will only appear in the WebSphere `trace.log` file.

Installing the MyAV MCU enterprise application

The instructions below assume the MyAV service provider and the MyAV MCU enterprise application will be installed on the same server as the Conference Manager, but with the MyAV enterprise application being managed in a separate WebSphere server profile. While the MyAV MCU enterprise application is not required to be installed on the same server as the Sametime Conference Manager, having it and the MyAV service provider on the same server simplifies the installation for development purposes.

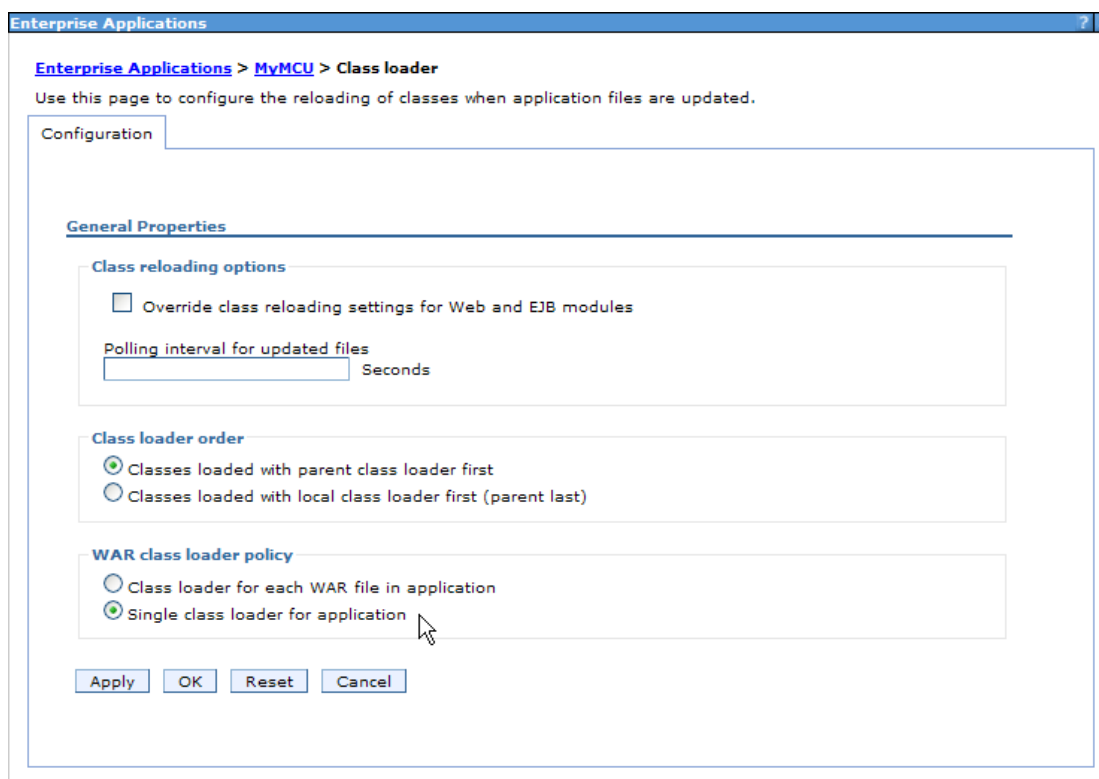
To begin, log into the WebSphere Integrated Solutions Console (<https://server.name:9044/ibm/console>) for the MyMCU profile created in the previous section using your WAS admin ID and password.

Install the myav.ear Enterprise application (**Applications > Application Types > WebSphere enterprise applications > Install > Local file system > Browse**; select the `com.ibm.telephony.conferencing.myav.mcu.ear` and then accept all defaults to install).



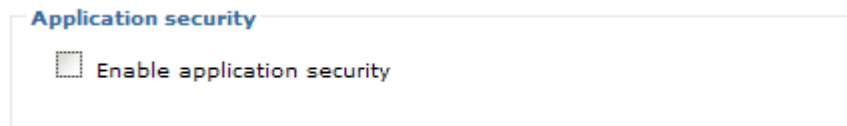
Installing an enterprise application

Set the MyMCU application's class loader policy to single:



Setting the MyMCU application's class loader policy to single

Verify that application security for the server hosting MyMCU matches the setting on the server hosting the MyAV adapter (**Security > Global Security**):



Verify application security settings match

(Optional) Add telephony-related tracing options before restarting the MyMCU server (**Troubleshooting > Logs and trace > [your server] > Change Log Detail Levels**):

```
*=info:
```

```
com.ibm.telephony.conferencing.myav.*=all:  
com.ibm.telephony.conferencing.myav.mcu.*=all:
```

The lines shown above apply to the server hosting MyMCU. To have full trace for the MyAV adapter, add the trace options to the Media Server as well. For example:

```
*=info:
```

```
com.ibm.mediaserver.*=all:  
com.lotus.sametime.telephony.sipfocus.*=all:  
com.ibm.telephony.conferencing=fine:  
com.ibm.telephony.conferencing.spi.*=fine:  
com.ibm.telephony.conferencing.myav.*=all:
```

Once these steps are complete, stop the Media Server and then restart both server profiles, starting with the server hosting MyMCU. Below are the commands for the default location on Windows:

```
cd "C:\Program Files\IBM\WebSphere\AppServer\profiles\mymcu\bin"  
startServer.bat server1
```

Then start the Media Server. Below are the commands for the default locations on Windows:

```
cd "C:\Program Files\IBM\WebSphere\AppServer\profiles\STMSDMgrProfile\bin"  
startServer.bat dmgr  
  
cd "C:\Program Files\IBM\WebSphere\AppServer\profiles\STMSAppProfile\bin"  
startServer.bat nodeagent  
startServer.bat STMediaServer
```

How to enable TLS for SIP transport

The installation instructions for MyAV cover the steps when the SIP transport for the Sametime Media Manager is TCP. To enable TLS (secure SIP) for the MyAV telephony adapter, follow these additional steps:

1. Import the SIP Proxy/Registrar's certificate into MyMCU and vice versa. Because MyMCU opens a TLS connection to the SIP Proxy/Registrar, you need to exchange certificates between the servers by exporting the SIP Proxy/Registrar's certificate to MyMCU and vice versa. The steps are the same as documented in “Exchanging certificates between the Video Manager(VMGR) and the Conference Focus” in the IBM Sametime Installation and Administration online help, substituting SIP Proxy/Registrar for 'Conference Focus' and MyMCU for 'Video Manager' in the outlined steps.
2. If MyMCU is installed on a separate machine, see the section entitled “Adding trusted IP addresses to the Media Manager SIP Proxy and Registrar” in the IBM Sametime Installation and Administration online help for additional steps to enable MyMCU to securely connect to the SIP Proxy/Registrar.

If TLS is enabled and these steps are not carried out, the symptoms will include:

- MyAV call participants will remain in Connecting status, not Connected
- Server logs for the Media Manager and MyMCU include SIP error "503 Service Unavailable"
- FINE level tracing for MyMCU indicates SIP messages failing due to security certificate errors.

Note: Starting with Sametime version 8.5.2, the System Console default is TLS.

Installation verification

Verify the successful startup of the two components. Messages similar to those shown below should be in the trace.log located in the mymcu logs subdirectory:

```
com.ibm.telephony.conferencing.myav.mcu.MyMCUServlet USE_SIP_CONTROL=true
com.ibm.telephony.conferencing.myav.mcu.MyMCUServlet Creating MyMCU instance
com.ibm.telephony.conferencing.myav.mcu.MyMCU Configuration={...}
com.ibm.telephony.conferencing.myav.mcu.MyMCU Starting MCU command server
com.ibm.telephony.conferencing.myav.mcu.MyMCU MyMCU available
com.ibm.telephony.conferencing.myav.mcu.MyMCUCommandSocketServer Configuration={...}
```

If DEBUG_LOG_LOCATION is set in the myav.properties file (e.g., DEBUG_LOG_LOCATION=C:/), then logs will be created for each of the MyAV classes as they log messages (e.g., MyAVConferenceService.log and MyMCUServlet.log). Below are the key messages to check to verify successful startup of the MyAV service provider.

```
com.ibm.telephony.conferencing.myav.MyAVConferenceService
USE_SIP_CONTROL=true
com.ibm.telephony.conferencing.myav.MyAVConferenceService Configuration={...}
com.ibm.telephony.conferencing.myav.MyAVConferenceService Starting command
dispatcher
```



```
com.ibm.telephony.conferencing.myav.MyAVConferenceService Starting MCU
command client

com.ibm.telephony.conferencing.myav.MyAVConferenceService Service available

com.ibm.telephony.conferencing.myav.mcu.MyMCUCommandSocketServer
Configuration={...}

com.ibm.telephony.conferencing.myav.mcu.MyMCUCommandSocketServer

    MCU command server accepting connections at server.name:50001

com.ibm.telephony.conferencing.myav.mcu.MyMCUCommandSocketServer$Notificatio
nHandler

    Start processing notifications
```

Appendix C: Installing and testing your service provider

As discussed in previous chapters, your telephony service provider is packaged as a Java JAR file. This section explains what other configuration files you will need to create and how to install them on the Sametime Media Server. The steps are:

Implement the TelephonyService interface and optionally DefaultUser/DefaultConference delegates (covered in previous chapters).

Export these classes to a JAR file and dependent libraries that are not part of the standard Java library / WebSphere application environment (covered in previous chapters).

Define a ConferenceManager.properties specifying your TelephonyService implementation and configuration parameters.

Optionally create additional localized versions of AudioConference.properties and telephony messages located in \${USER_INSTALL_ROOT}/sametime_tcspi and \${USER_INSTALL_ROOT}/sametime_tcspi/properties.

Copy these files to the Sametime Media Server in the \${USER_INSTALL_ROOT}/sametime_tcspi subdirectory. See “How do external TCSPI adapters work?” and its referenced topics of the *Sametime Information Center* for more installation details.

The following table further describes the configuration parameters specified in ConferenceManager.properties.

Table detailing ConferenceManager.properties.

Part	Function	Description
Conference service class	Adapter interface to TCSPI. For example, ConferenceServiceClass=com.ibm.telephony.conferencing.myav.MyAVConferenceService	Fully qualified implementation name of class extending the AbstractConferenceService or AbstractSipConferenceService class.
Conference service name	Label of adapter interface to TCSPI. For example, ConferenceServiceName=MyAV Conference Service	Name of TCSPI service shown in user interface. Can be overridden by locale-specific translations in AudioConference_*.properties files.
SIP-controlled conferencing supported	Specifies whether adapter supports SIP controlled conferencing via the Sametime Connect softphone user interface. The values are true and false. For example, SIPConferenceEnabled=true	If true, Conference Manager will use SIP signaling to 3rd party MCU, otherwise will use dial(...) interface to the TCSPI adapter.
Video conferences	Specifies whether adapter supports video conferencing.	If true, client will enable video start/stop control.

supported	The values are true and false. For example, VideoConferenceEnabled=true	
Telephone conferences supported	Specifies whether adapter supports telephone conferencing. The values are true and false. For example, TelephoneConferenceEnabled=true	If true, client will allow user to enter phone numbers as endpoints.
Default service provider	If more than one service provider is installed and a Sametime client earlier than version 8.5.2 connects, this flag indicates the one that should be used.	DefaultService=true
Maximum conference participants	Indicates how many audio/video participants are supported for a given conference.	MaximumConferenceUsers=100 MaximumAudioConferenceUsers=60 MaximumVideoConferenceUsers=60 MaximumExternalConferenceUsers=2
Client identifier field	The client ID field specifies whether an ID is required enabled or mutable. ClientID field = required The values are true and false For example: ClientIdEnabled=true	If Domino and the service provider directories are not synchronized, then enabling this field could require that the user enter an ID specific to the telephony system.
Client password	Similar to the client ID field. The values are true and false.	Used to authenticate the client. For example: ClientIdRequired=true
Conference types	The two types of conferences are dynamic and static. Static - this is a reservationless telephone bridge. Dynamic- a bridge that is scheduled for a specific time/date, and unique phone number. The values are true and false DynamicConferencesEnabled=false StaticConferenceEnabled=true. If the conference type is dynamic, option ReuseDynamicConferencesDialIn controls whether the dynamically generated call-in information is persisted in the meeting room at the end of the call (default is false and the call-in	Depending upon whether static or dynamic are enabled, the preferences appropriate options. Static enables a field called "Passcode" for example, allowing the user to identify his or her standing bridge number.

	information is cleared at the end of the call)	
Service locations	<p>The service locations section of this file can be used to specify multiple locations.</p> <p>Type: String with spaces A ";" separated list</p>	<p>For example:</p> <p>ServiceLocations=Location1;Location2;Location3;</p>
Conference options	<p>This section specifies entries for optional features that are shown in the telephony preferences pages. These optional conference properties are enabled by the service provider at deployment time and their values are passed into the service provider when a new conference is created.</p> <p>The values are true and false</p>	<p>For example:</p> <p>ConferenceOption1Enabled=false ConferenceOption1Required=false ConferenceOption1Mutable=true</p> <p>ConferenceOption2Enabled=false ConferenceOption2Required=false ConferenceOption2Mutable=true</p> <p>ConferenceOption3Enabled=false ConferenceOption3Required=false ConferenceOption3Mutable=true</p> <p>ConferenceOption4Enabled=false ConferenceOption4Required=false ConferenceOption4Mutable=true</p>
Help URL	<p>Here is a point where you may specify a URL for the end user help.</p> <p>Type=String</p>	<p>For example:</p> <p>ConferenceServiceHelpURL=http://www.somehelpurl.org/HelpMe</p>
Timeout value	<p>Each a synchronous request has configurable timeout values.</p> <p>The value is time in milliseconds</p>	<p>For example:</p> <p>CreateConferenceTimeout=60000 EditConferenceTimeout=60000 CancelConferenceTimeout=60000 GetConferenceDocumentTimeout=60000 StartConferenceControlTimeout=60000 StopConferenceControlTimeout=60000 RegisterUserTimeout=60000 UnRegisterUserTimeout=60000 SetConferencePropertyTimeout=60000 DialUserTimeout=60000 DialConferenceTimeout=60000 DisconnectUserTimeout=60000 SetUserPropertyTimeout=60000 AssociateUserTimeout=60000 MediaControlTimeout=60000</p>

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you. Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
5 Technology Park Drive
Westford Technology Park
Westford, MA 01886

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only. All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp.

Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

These terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM

AIX

DB2

DB2 Universal Database Domino

Domino

Domino Designer

Domino Directory

i5/OS

iSeries

Lotus

Notes

OS/400

Sametime

System i

WebSphere

AOL is a registered trademark of AOL LLC in the United States, other countries, or both.

AOL Instant Messenger is a trademark of AOL LLC in the United States, other countries, or both.

Google Talk is a trademark of Google, Inc, in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Microsoft, and Windows are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.