

Service integration bus security

or

How to get onto the security bus



Alasdair Nottingham
WebSphere Messaging Security
not@uk.ibm.com

16-Sep-08

© 2006, 2008 IBM Corporation

Objectives

Equip the WebSphere Application Server, WebSphere ESB or WebSphere Process Server practitioner with the skills required to secure a Service Integration Bus and its clients

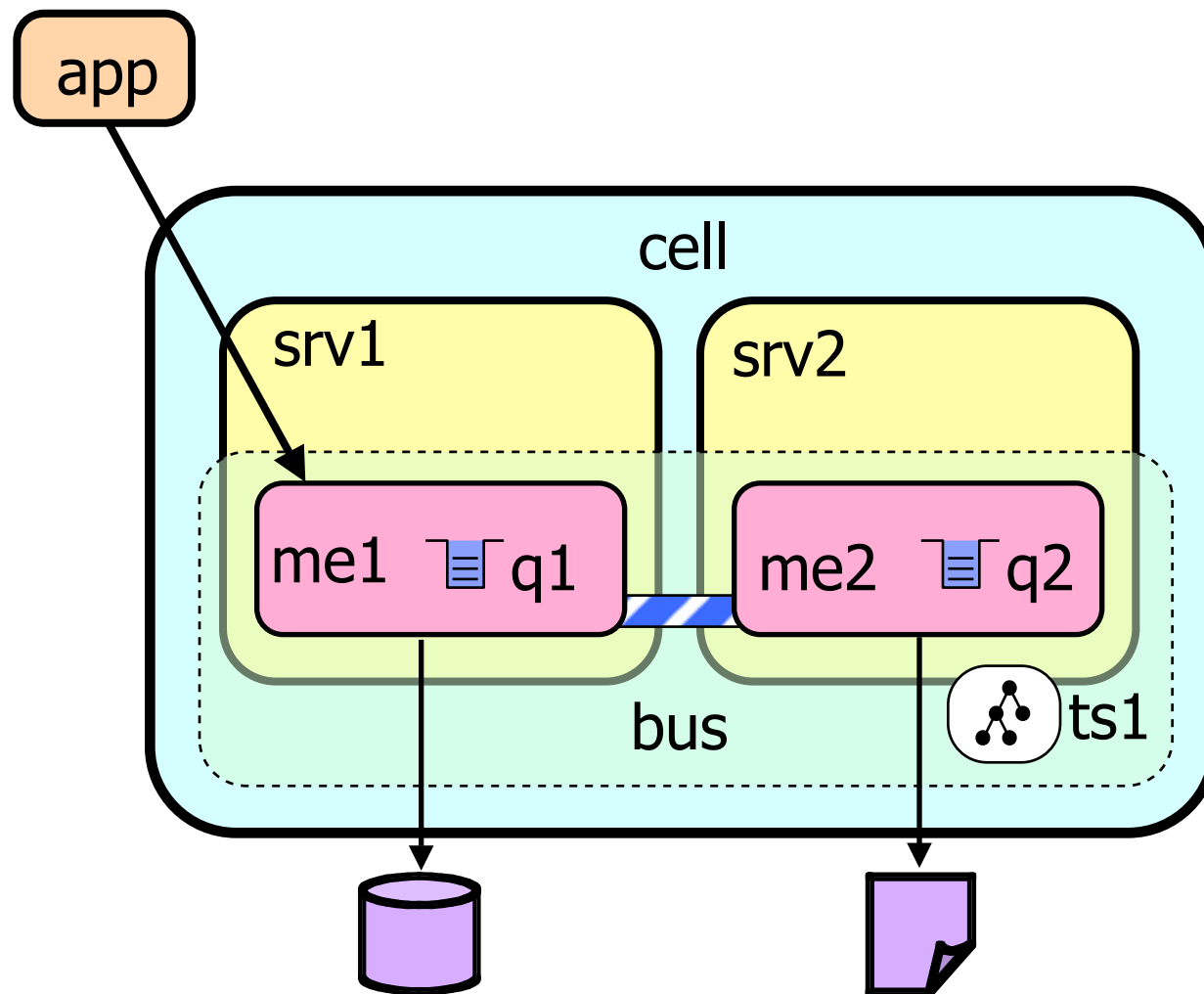
Agenda

- Basic concepts
- Securing a bus
- Securing a messaging engine
- Connecting to a bus
- Destination authorization
- Topic authorization
- What is new in WAS v7

Agenda

- **Basic concepts**
- Securing a bus
- Securing a messaging engine
- Connecting to a bus
- Destination authorization
- Topic authorization
- What is new in WAS v7

Basic concepts



The concept of the bus is largely a logical one – it provides a namespace for destinations and is the entity to which clients connect. In reality, the bus is made up of one or more interconnected messaging engines created by the addition of a server or cluster as a member of the bus. It is these messaging engines that manage the flow of messages through the bus and provide the localizations of destinations. Their associated message stores are where messages are persisted. Within the bus, every messaging engine implicitly trusts every other messaging engine. Destinations defined on the bus act as the point of addressability for messages. Different types of destination are defined such as queue, topic space and alias. A topic space consists of a hierarchy of topics.

Before you begin

- Ensure there are no in-doubt transactions
- Stop the bus

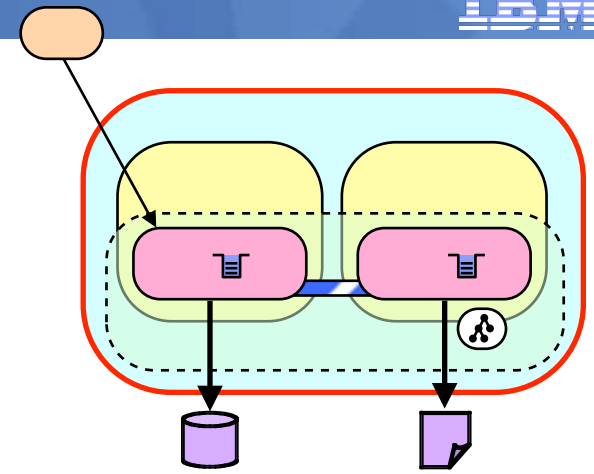
Before configuring security for a service integration bus, it is necessary to ensure that there are no in-doubt transactions involving a messaging engine as transaction recovery will fail to complete once the bus has been secured. Next, all servers should be stopped to ensure that they have a consistent view of the bus security configuration when they restart

Agenda

- Basic concepts
- **Securing a bus**
- Securing a messaging engine
- Connecting to a bus
- Destination authorization
- Topic authorization
- What is new in WAS v7

Enabling global security

- Authentication uses the user registry



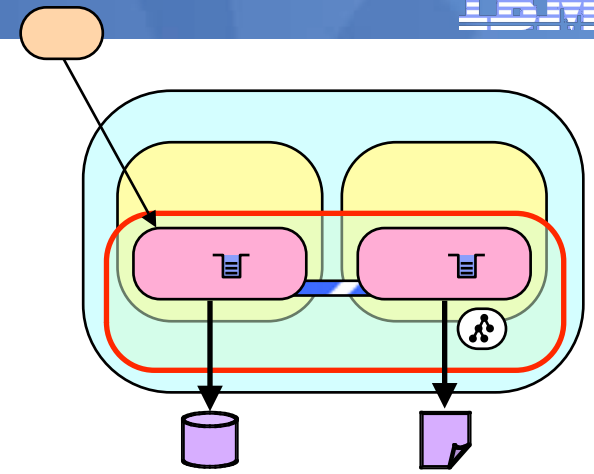
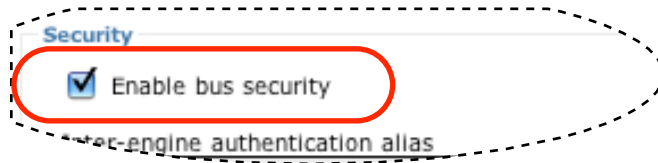
The screenshot shows the 'Administrative security' configuration page. The 'Enable administrative security' checkbox is checked and highlighted with a red circle. Below it, there are links for 'Administrative User Roles' and 'Administrative Group Roles'. The 'Application security' section has an unchecked 'Enable application security' checkbox. The 'Java 2 security' section has three checkboxes: 'Use Java 2 security to restrict application access to local resource' (unchecked), 'Warn if applications are granted custom permissions' (checked), and 'Restrict access to resource authentication data' (unchecked). The 'User account repository' section has a 'Current realm definition' dropdown menu set to 'Federated repositories', which is also highlighted with a red circle. Below this, there is a list of 'Available realm definitions' with 'Federated repositories' selected, and buttons for 'Configure' and 'Set as current'.

The service integration bus uses the WebSphere Application Server user registry to perform authentication. Consequently, global security must be enabled in the application server cell and a user registry configured in order to secure a bus. More generally though, a messaging engine is only as secure as the application server in which it runs and consequently it would not make sense to secure the bus without also securing the application server cell. For Version 6.1, only administrative security need be enabled. This provides the greatest flexibility as it is still possible to disable bus security if it is not required.

Panel path:
Security -> Secure administration and applications

Enabling bus security

- Enabled by default



If admin security is enabled, clients connecting to the bus will be authenticated and authorization checks performed.

- Does not prevent unencrypted messaging traffic
- Does not protect message store

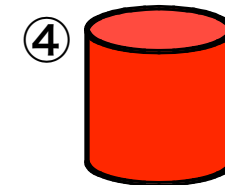
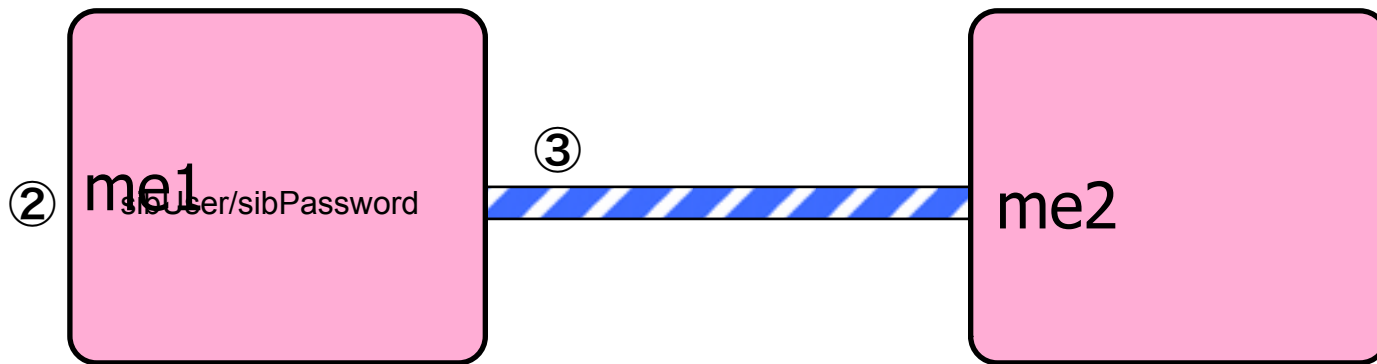
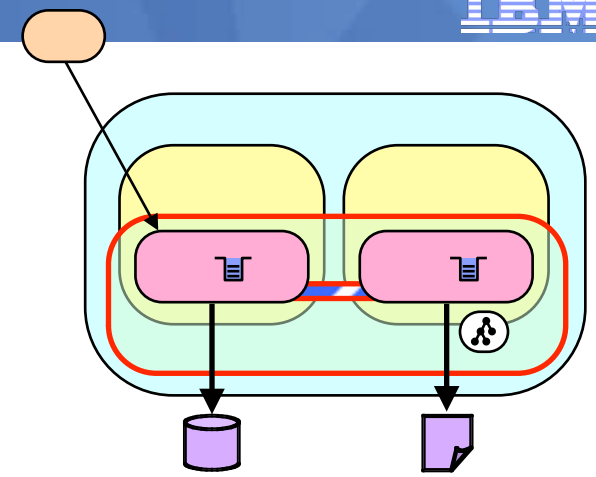
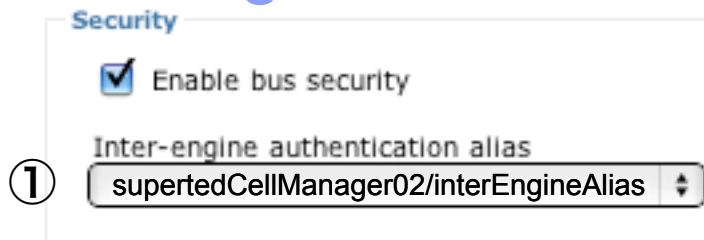
The secure flag on a bus definition is checked by default. In 6.0.x this flag really indicates is that, if global security is enabled, clients connecting to the bus will be authenticated and then authorization checks performed when subsequently accessing bus resources. In 6.1 this will only appear checked if admin security is enabled. Note that in 6.1 the buses collection shows enabled even if admin security has been disabled. Enabling bus security does not affect the security of the client's connection to the bus (in Version 6.0) nor the messages in the data store.

Panel path:
Buses > {bus name} > Security

Script:

```
AdminTask.modifySIBus(['-bus isswBus -busSecurity true'])
```

Securing intra-bus comms



When a messaging engine starts, it will attempt to connect to all of the other available messaging engines in the cell. This initiating messaging engine will lookup the inter-engine authentication alias (see 1) and resolve it to the userid and password (see 2). It will then be sent along the communication channel (see 3) to the target messaging engine. The recipient of the connection will check that the credentials authenticate against the user registry (see 4) and that the user name is the same as that defined in its own copy of the inter-engine authentication alias. (This is why all the servers in the cell must be stopped to ensure that they have consistent security configuration.) Note that there is no mutual authentication. The initiating messaging engine must rely on SSL configuration to ensure that it has really connected to another messaging engine.

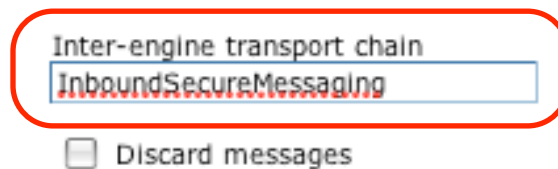
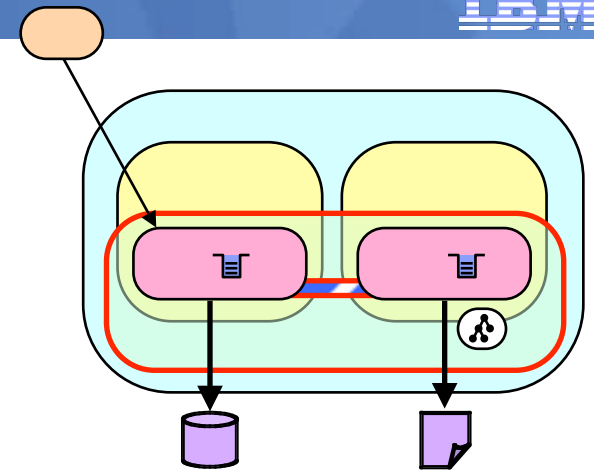
Panel path:
Buses > {bus name} > Security

Script:

```
AdminTask.modifySIBus(['-bus isswBus -interEngineAuthAlias supertedCellManager02/interEngineAlias'])
```

Securing intra-bus comms

- Encrypt traffic between messaging engines
- Set inter-engine transport chain



- Must be available to all bus members

6.0

Explicitly to InboundSecureMessaging

6.1

InboundSecureMessaging default for secure bus

The transport chain that the messaging engines use to communicate with one another is specified on the bus configuration. Chains are defined at the application server scope and consequently the named chain must be defined on all servers that may host a messaging engine on the bus. In version 6.0 when no chain is specified then InboundBasicMessaging is assumed. To secure inter-engine communication, specify the InboundSecureMessaging chain. In Version 6.1 this is the default behaviour for a secure bus

Panel Path:
Buses > {bus_name}

Script:

```
AdminTask.modifySIBus(['-bus isswBus -protocol InboundSecureMessaging'])
```

Bus security panel

Buses

Buses > Security for bus isswBus

Configure the security settings for your service integration bus.

Configuration

General Properties

Security

☒ Enable bus security

Inter-engine authentication alias
supertedCellManager02/InterEngineAlias

Permitted transports

☐ Allow the use of all defined transport channel chains

☒ Restrict the use of defined transport channel chains to those protected by SSL

☐ Restrict the use of defined transport channel chains to the list of permitted transports

Mediations authentication alias
supertedCellManager02/mediationAlias

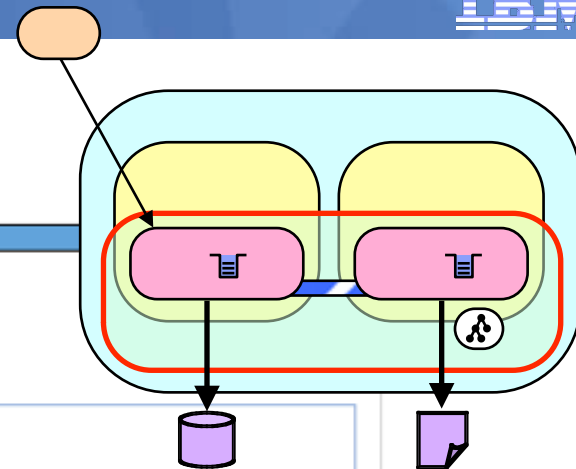
Apply OK Reset Cancel

Additional Properties

- Users and groups in the bus connector role
- Permitted transports

Related Items

- JAAS - J2C authentication data
- Secure Administration and Applications

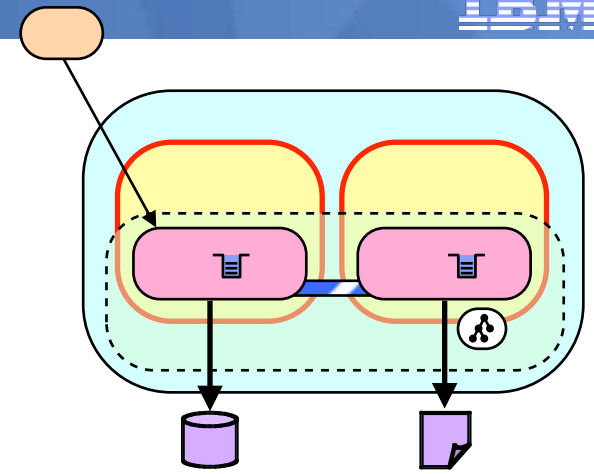


In Version 6.1 there is a new panel containing all of the security information for a bus. In particular, note the “Permitted transports” section which, in this case, indicates that only transport chains protected via SSL may be used. This is now the default when bus security is enabled.

Panel Path:
Buses > {bus name} > Security

DCS encryption

- Messaging resource locations shared via DCS
- By default DCS does not use SSL



The final step in securing the bus is secure the Distribution and Consistency Services (DCS). DCS is used to share the location and availability of messaging resources and consequently an attack on this protocol could insert or divert messages. By default, DCS is secured using the server LTPA token but does not use SSL. The DCS-Secure transport chain may be selected for the core group if SSL is required.

Panel Path:
Core groups > {core_group_name}

Script:

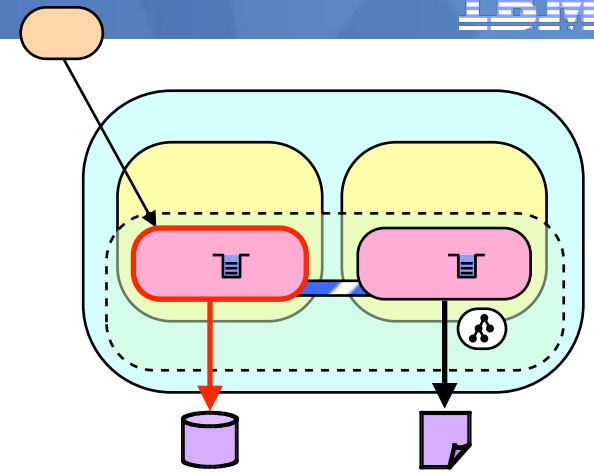
```
coreGroups = AdminConfig.list('CoreGroup')
AdminConfig.modify(coreGroups, [{"channelChainName", "DCS-Secure"}])
```

Agenda

- Basic concepts
- Securing a bus
- **Securing a messaging engine**
- Connecting to a bus
- Destination authorization
- Topic authorization
- What is new in WAS v7

Securing the data store

- Messaging engine can specify credentials when connecting to the database
- Stored on the datastore as an auth alias
- Do not set component alias on data source



General Properties

UUID

7A4E853F2B6E4FD7

* Data source JNDI name

jdbc/com.ibm.ws.sib/superted.server1-isswBus

Schema name

IBMWSSIB

Authentication alias

supertedCellManager02/dataStoreAlias

Securing the bus does not cause the persisted messages to be secured. The bus makes no statements about how this data is secured, and leaves it up to the database to protect the data. If the database is secured then messaging engine can be configured to send a userid and password when creating a connection. This can, and should, be specified via an authentication alias defined on the data store of the messaging engine. Do not define an alias on the data source itself as this will make the authenticated resource available to all applications with access to JNDI.

This needs to be done separately for each messaging engine in the cell.

Panel path:

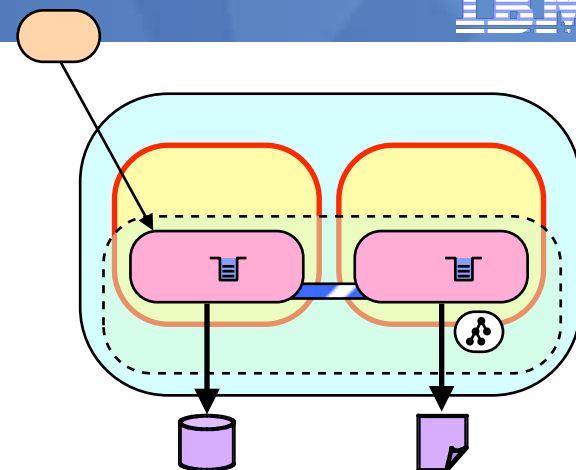
Buses > {bus_name} > Messaging engines > {me_name} > Data store

Script:

```
engines = AdminTask.listSIBEngines(['-bus isswBus'])
datastore = AdminConfig.list('SIBDatastore', engines)
AdminConfig.modify(datastore, [['authAlias', "supertedCellManager02/dataStoreAlias"]])
```

Disabling insecure transports

- Enabling bus security does not prevent the use of insecure transports
- InboundBasicMessaging should be disabled or deleted
- Not possible for secure and insecure buses to share a server



General Properties

* Name

InboundBasicMessaging

☐ Enabled

Transport Channels

• TCP Inbound channel (SIB_TCP_JFAP)

Host	*
Port	7281
Thread pool	SIBFAPInboundThreadPool

• JFAP Inbound channel

Apply

OK

Reset

Cancel

By default, an application server is configured with both secure (SSL) and insecure transport chains for messaging. Enabling bus security does not prevent communication via insecure transport chains. It is necessary to explicitly delete or disable the insecure transport chain (i.e. InboundBasicMessaging). As a consequence, it is not possible for secure and insecure servers to both have messaging engines on the same server.

This procedure is not necessary in version 6.1

This needs to be repeated for each server that has the sib service enabled.

Panel Path:

Application servers > {server_name} > Transport Chain > InboundBasicMessaging

Script:

```
chains = AdminConfig.list('Chain').splitlines()
```

```
for chain in chains:
```

```
    chainName = AdminConfig.showAttribute(chain, 'name')
```

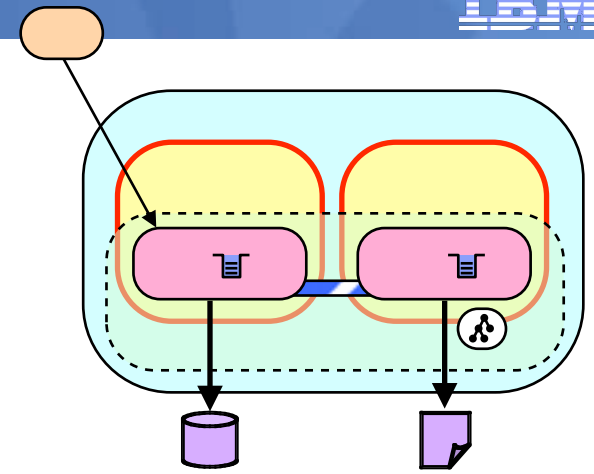
```
    if chainName == "InboundBasicMessaging":
```

```
        AdminConfig.modify(chain, [{"enable", "false"}])
```


6.1

Disabling insecure transports

- Permitted chains
 - ▶ Any defined chain
 - ▶ Chains protected by SSL
 - ▶ Chain with a specified name
- Secure and insecure buses can share a server



Permitted transports

- ☐ Allow the use of all defined transport channel chains
- ☒ Restrict the use of defined transport channel chains to those protected by SSL
- ☐ Restrict the use of defined transport channel chains to the list of permitted transports

As touched on earlier, the new bus security panel enables you to configure the transport chains that are permitted for connection to a given bus. There are three options: any defined chain, chains protected by SSL, and a list of chains specified by name. This removes the need to explicitly disable chains on a per server basis and also enables a secure and insecure bus to both have messaging engines on the same server.

Note that this setting does not prevent the ports being opened, however it will prevent the bus using that transport for connections. If ports should be closed it is still important to follow the procedure on slide 16.

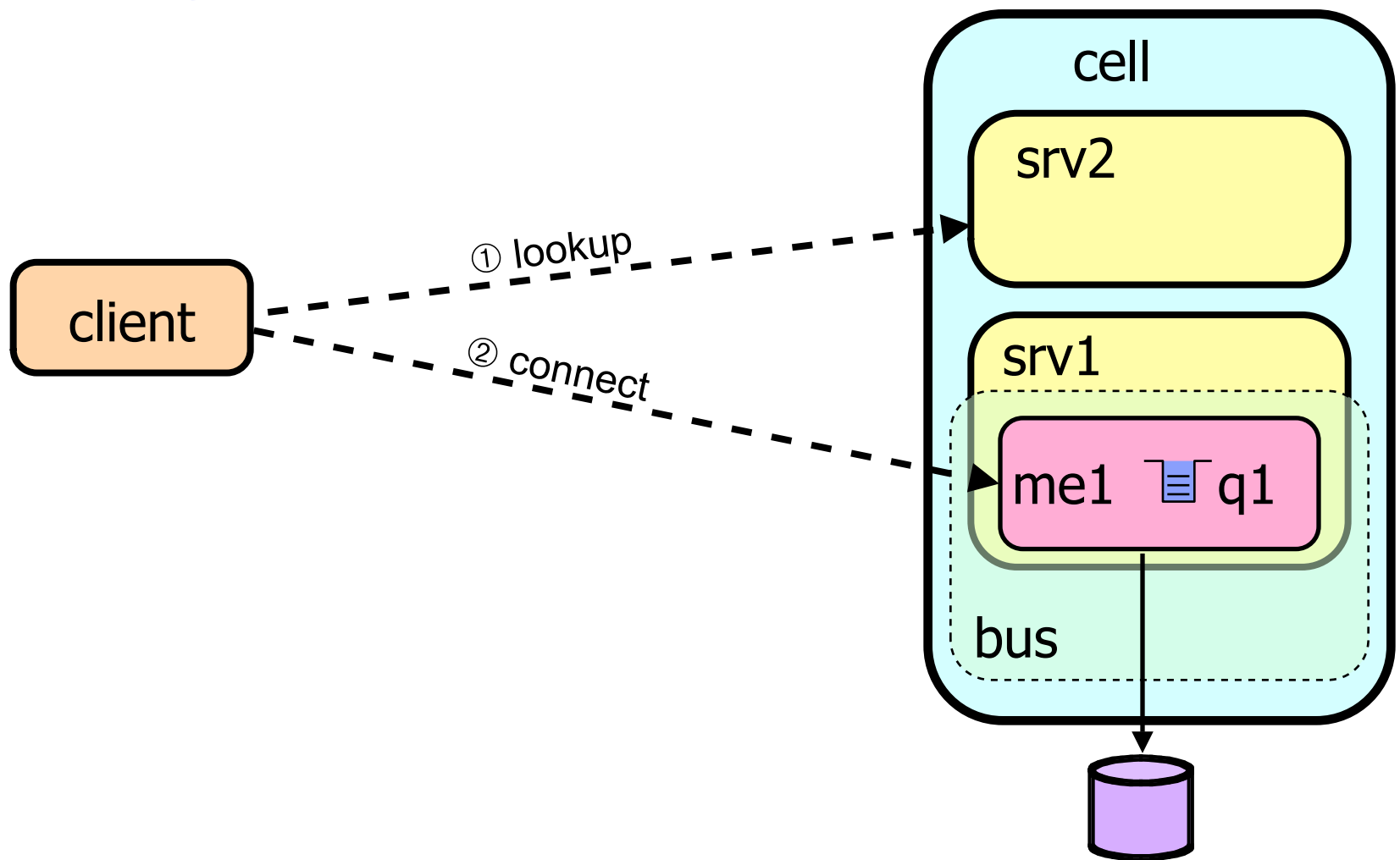
Panel Path:
Buses > {bus name} > Security

Script:
AdminTask.modifySIBus(['-bus isswBus -permittedChains SSL_ENABLED'])

Agenda

- Basic concepts
- Securing a bus
- Securing a messaging engine
- **Connecting to a bus**
- Destination authorization
- Topic authorization
- What is new in WAS v7

Connecting to the bus

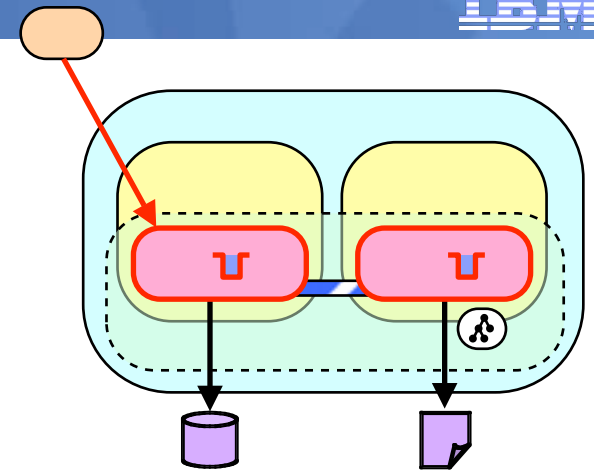


Connecting to the bus is based on a two-stage protocol. The first step is to lookup (or bootstrap) the location of a suitable messaging engine. The second step is to connect to the messaging engine. Both stages of the protocol are authenticated and authorized. So a client cannot determine messaging engine locations unless it is permitted to connect to the bus.

If the client is running inside the cell hosting the bus then the lookup can occur locally without needing to call another server.

Authorization to connect

- Client must be authorized
- Specified using bus connector role



6.0 Must use wsadmin

6.1 Can use admin console

New Delete		
<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>		
Select	Name	Type
<input type="checkbox"/>	Server	Group
<input type="checkbox"/>	bruce	User
<input type="checkbox"/>	isswHursley	Group
Total 3		

The authenticated user must then have authorization to connect to the bus. This is specified via the bus connector role.

In 6.0 the bus connector role can only be configured using the wsadmin command line tool. In 6.1 there are panels in the admin console.

Panel Path:

Buses > {bus name} > Security > Users and groups in the bus connector role

Script:

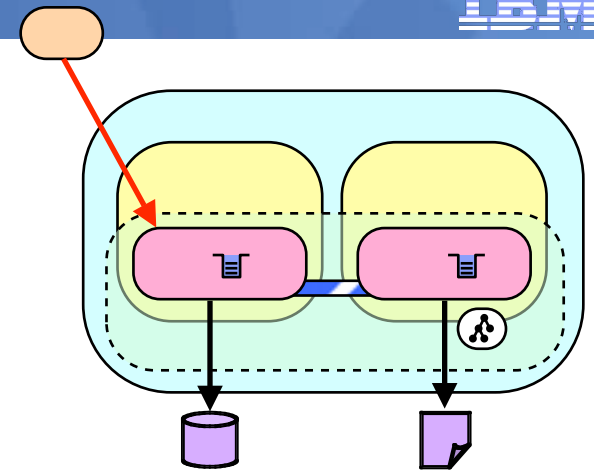
```
AdminTask.addGroupToBusConnectorRole(['-bus isswBus -group isswHursley'])
AdminTask.addUserToBusConnectorRole(['-bus isswBus -user bruce'])
```

```
AdminTask.listGroupsInBusConnectorRole(['-bus isswBus'])
AdminTask.listUsersInBusConnectorRole(['-bus isswBus'])
```

```
AdminTask.removeGroupsFromBusConnectorRole(['-bus isswBus -group isswHursley'])
AdminTask.removeUsersFromBusConnectorRole(['-bus isswBus -user bruce'])
```

Specifying connection credentials

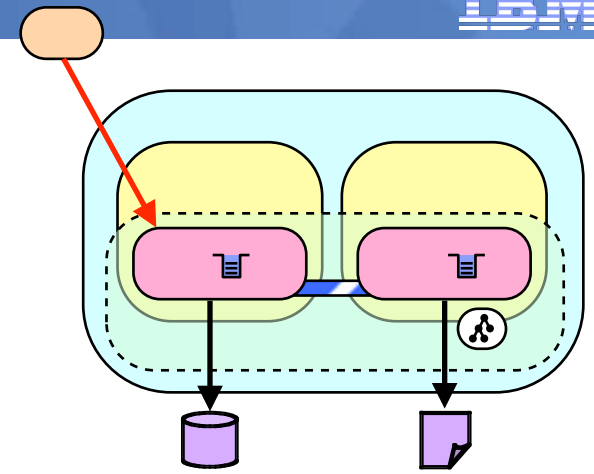
- Authentication mode set in resource reference *res-auth*
 - ▶ *Application*
 - Credentials passed on *createConnection* call
☹ *credentials hard-coded into application*
 - Component-managed authentication alias
☹ *server based applications with JNDI access share credentials*
 - ▶ *Container*
 - Credentials provided by container
 - Authentication alias set on resource-reference
😊 *only this application can use the credentials*



So where do we specify the username and password with which to authenticate? J2EE and WebSphere provide a number of options. The resource-reference used to define the connection factory contains a *res-auth* element which specifies the authentication mechanism. If this is set to *Application*, then either the application specifies the credentials on calling *createConnection* (which is bad because the credentials are hard-coded in to the application) or they are specified on the connection factory using a component-managed authentication alias (which is also bad as anyone with JNDI access can retrieve the authenticated resource). If the *res-auth* is set to *Container*, then any credentials specified by the application on *createConnection* are ignored. Instead, the credentials are specified via a container-managed authentication alias defined on the bindings for the resource-reference. In this way, only this application can access the authenticated resource, anyone else performing a JNDI lookup will get back an unauthenticated connection factory.

Using container managed authentication

- Alias defined in admin console has a prefix
- Alias defined in wsadmin does not
- Specify on resource reference



Name:	jms/factory
Description:	
Type:	javax.jms.ConnectionFactory
Authentication:	Container
Sharing scope:	Shareable

WebSphere Bindings

The following are binding properties for the WebSphere Application Server.

JNDI name:	
JAAS Login Configuration:	
<input type="radio"/> None <input checked="" type="radio"/> Use Default Method:	
Authentication Alias:	jmsAuthAlias

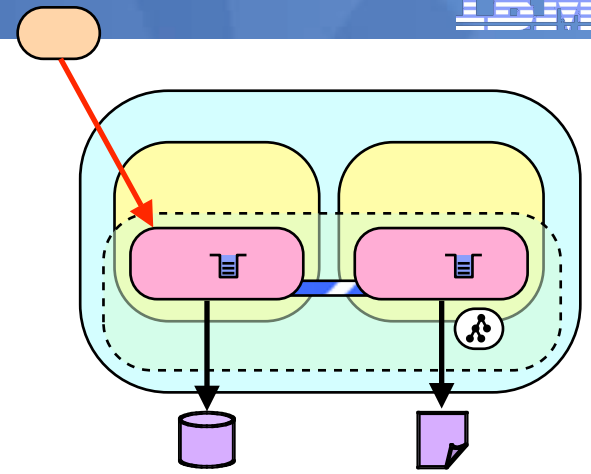
When creating an authentication alias use wsadmin to avoid the node name being prefixed by administration console (which would then require the node name to be included in the application). Then, on the references tab of the EJB Deployment Descriptor editor, select the resource reference, choose Container authentication and then, under the WebSphere bindings, specify that the default authentication method should be used with the given authentication alias.

Script:

```
AdminConfig.create("JAASAuthData", AdminConfig.list("Security"), [{"alias", "jmsAuthAlias"}, {"userId", "David"}, {"password", "DavidsPwd"}])
```

Specify an XA recovery alias

- Credentials defined on resource-reference are not available during transaction recovery
- Specify XA Recovery alias on ConnectionFactory
- Alias is used during recovery
- User only needs bus connector role



Advanced Administrative

Component-managed authentication alias
(none)

☐ Log missing transaction contexts

☐ Manage cached handles

☐ Share data source with CMP

XA recovery authentication alias
supertedCellManager02/jmsRecoveryAlias

As the container will only pass these credentials to the JMS provider when a connection is created, they are not available for the provider to use during transaction recovery. Consequently, a separate XA recovery authentication alias should be defined on the connection factory. The credentials for this alias are only used during transaction recovery. The associated user only needs authorization to connect to the bus.

Panel Path:

Resources > JMS > [Queue|Topic] connection factories > {connection_factory_name}

Script:

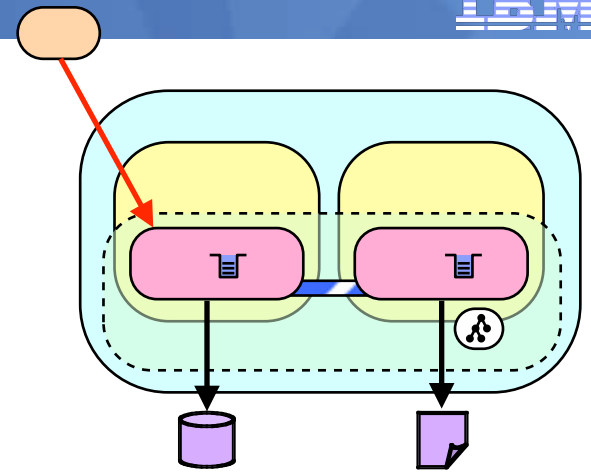
```
server = AdminConfig.list('Server')
```

```
cf = AdminTask.listSIBJMSConnectionFactory(server)
```

```
AdminTask.modifySIBJMSConnectionFactory(cf, '[-xaRecoveryAuthAlias supertedCellManager02/jmsRecoveryAlias]')
```

Message-driven bean credentials

- Define in the message-driven bean binding
- Used for initial connection and XA recovery



WebSphere Bindings
The following are binding properties for the WebSphere Application Server.

☐ Listener Port

ListenerPort name:

☒ JCA Adapter

ActivationSpec JNDI name:	ims/ActivationSpec
ActivationSpec Authorization Alias:	jmsAuthAlias
Destination JNDI name:	<input type="text"/>

Similarly, when using a message-driven bean, the authentication alias should be defined in the bindings for the bean, rather than in the activation specification. This alias will be used for both the initial connection attempt and transaction recovery.

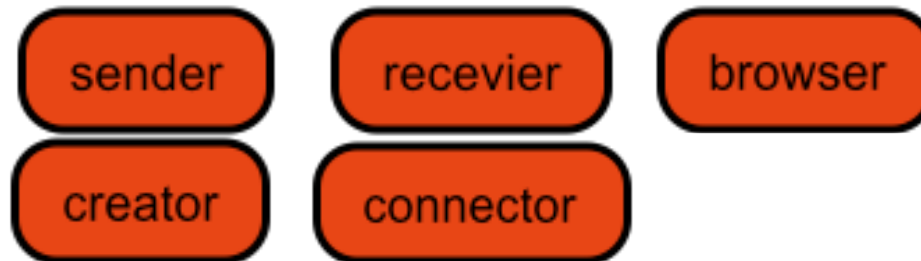
Bean tab of EJB Deployment Descriptor editor

Agenda

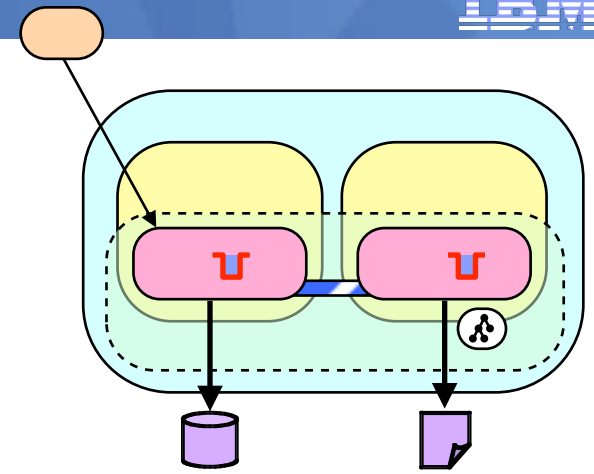
- Basic concepts
- Securing a bus
- Securing a messaging engine
- Connecting to a bus
- **Destination authorization**
- Topic authorization
- What is new in WAS v7

Role-based authorization

- Role-based policy applies to bus
- Role Types

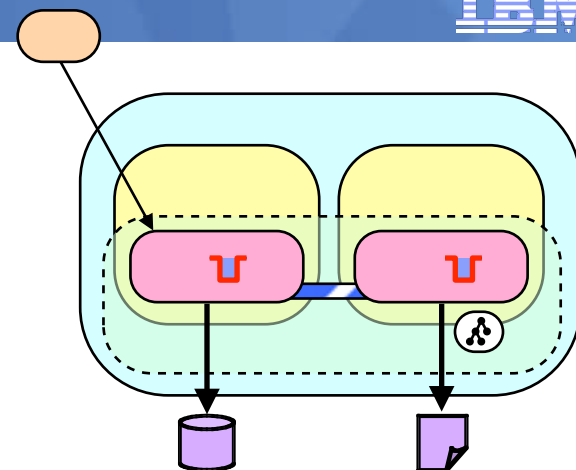


6.1 Special groups: AllAuthenticated, Everyone, Server.



Authorization for the Service Integration Bus is defined using a role-based policy defined at the bus scope. Users and groups from the user registry, along with the special groups AllAuthenticated, Everyone and Server (not supported by 6.0) are assigned to roles. The special Server entity was added in Version 6.1 and can be used with message-driven beans running under the server identity. There are five different role types applicable to applications: connector, browser, receiver, sender and creator. Which shall look at these role types in more detail later. Unfortunately, the authorization policy for a bus is neither editable nor viewable via the administration console – wsadmin must be used to access it. In Version 6.1, the bus connector role can be modified in the administrative console.

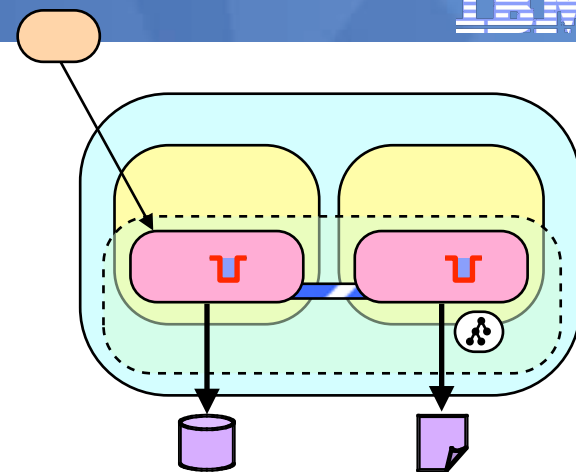
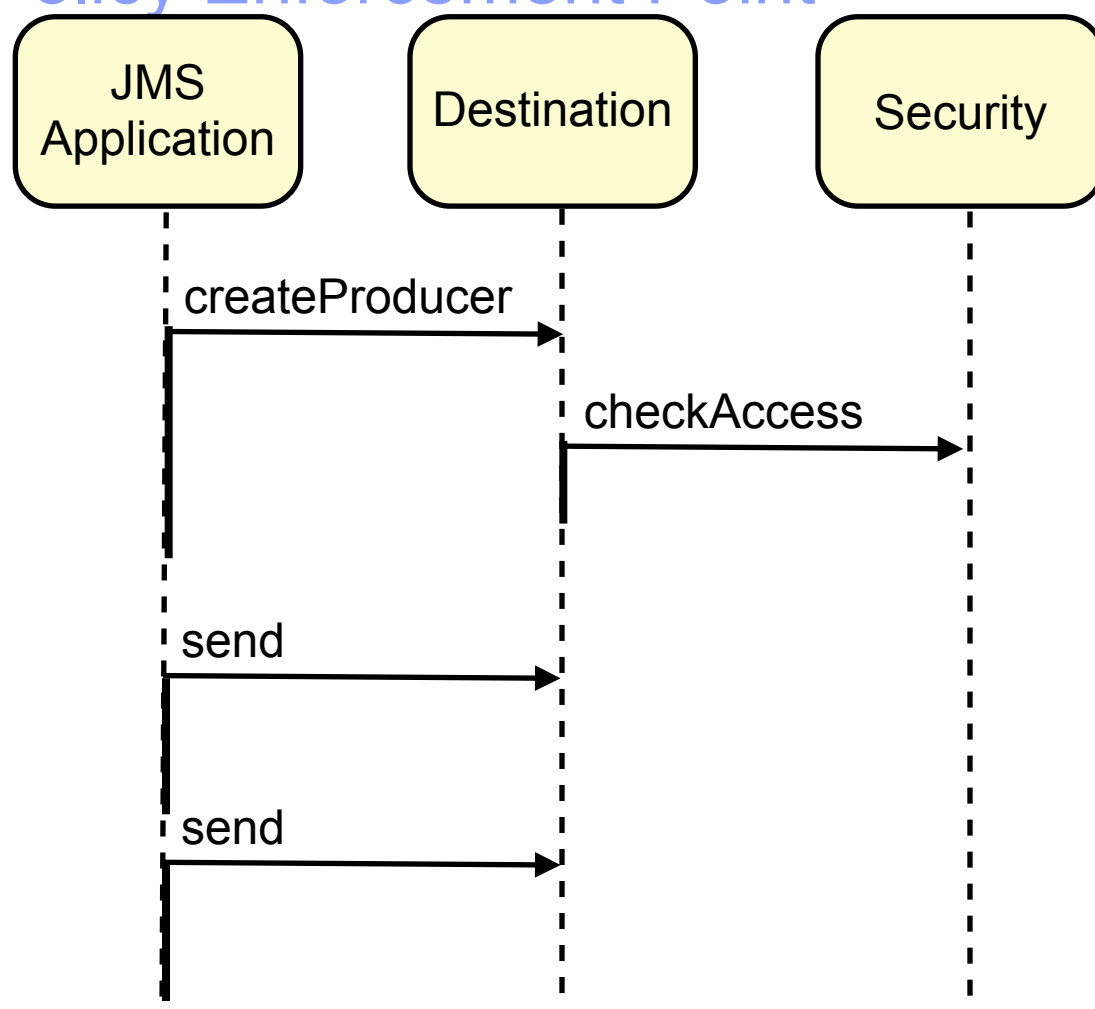
Role applicability



		Destinations						
		Queue	Topic Space	Alias	Foreign	Web service	Port	Temporary destination prefix
Role Type	sender	✓	✓	✓	✓	✓	✓	✓
	receiver	✓	✓	✓			✓	
	browser	✓		✓			✓	
	creator							✓

This chart shows which roles are applicable to which destination types. This presentation will only talk about Queues, Topic Spaces (and Topics) and temporary destination prefixes.

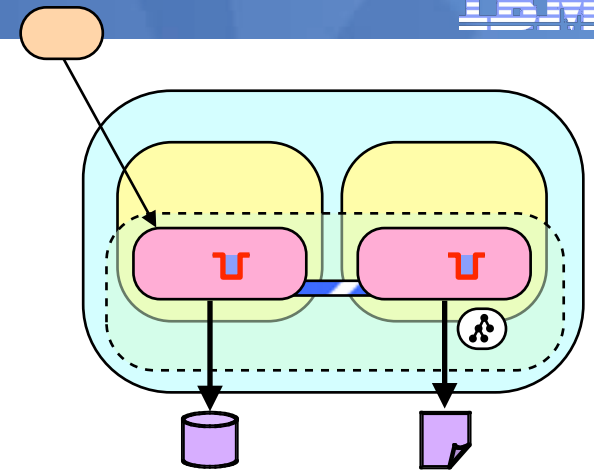
Policy Enforcement Point



The access decision is made at the time a JMS message producer (or consumer) is created, not at send time. This means that the overhead for security is low, but it also means that a messaging engine restart is required to disconnect clients that no longer have access.

Default authorization policy

- Groups that can connect
 - 6.0 AllAuthenticated
 - 6.1 Server
- All connected users have full access
- Not suitable if application isolation is needed

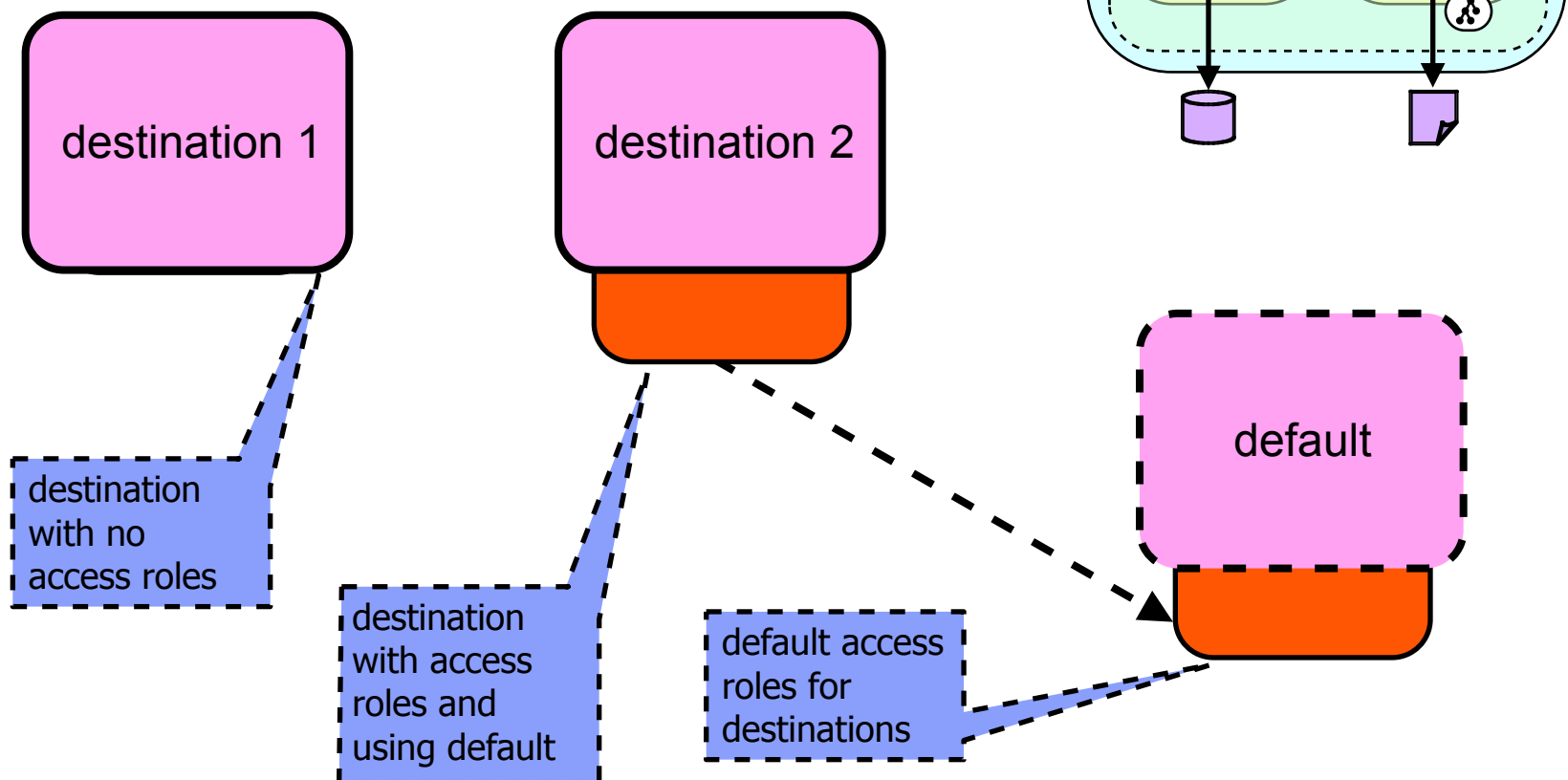


In version 6.0, by default, all users defined to the user registry can access the bus. In version 6.1 this was changed so only application servers in the cell can access the bus (specified using the Server group). It is important to note that the Server group includes the application servers themselves, and does not imply any access for applications running inside the application server.

Once connected all users have access to all destinations (even in 6.1). If application isolation is desired this may be all that is needed, but in most cases these defaults will not be appropriate. The following admin command can be run to remove the default roles:

```
AdminTask.removeDefaultRoles(['-bus isswBus'])
```

Default roles



Once connected, the application will attempt to access one or more destinations. It is possible to control access for all destinations on the bus using the default destination permissions. Inheritance of these defaults may then be turned off on a destination by destination basis. Such default permissions should be used cautiously and only where appropriate, for example to allow the user associated with the SCA_Auth_Alias in WebSphere Process Server to send and receive from automatically generated destinations on the SCA system bus.

Script:

```
AdminTask.removeDefaultRoles(['-bus isswBus'])
```

```
AdminTask.listGroupsInDefaultRole(['-bus isswBus'])
```

```
AdminTask.listUsersInDefaultRole(['-bus isswBus'])
```

```
AdminTask.addGroupToDefaultRole(['-bus isswBus -group isswUsers -role sender'])
```

```
AdminTask.addUsersToDefaultRole(['-bus isswBus -user bruce -role sender'])
```

```
AdminTask.removeGroupFromDefaultRole(['-bus isswBus -group isswUsers -role sender'])
```

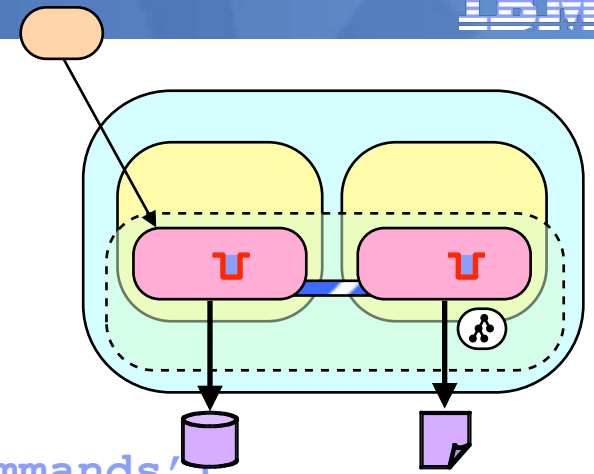
```
AdminTask.removeUsersFromDefaultRole(['-bus isswBus -user bruce -role sender'])
```

Administration

- Only via wsadmin

```
AdminTask.help('SIBAdminBusSecurityCommands')
```

- Anatomy of addGroupToDestinationRole



Parameter	Description
type	The type of the destination, one of: Queue, Port, Webservice, TopicSpace, ForeignDestination, Alias
bus	The bus name
destination	The destination name
role	The role type, one of: Sender, Receiver, Browser, Creator
group	The group name
foreignBus	The name of a foreign bus the destination is defined to. Mandatory for foreign destinations, optional for alias destinations.

All the admin commands are contained in a group called SIBAdminBusSecurityCommnds.

The commands all follow a similar convention and the addGroupToDestinationRole command is described on this foil.

Script:

```
AdminTask.addGroupToDestinationRole(['-bus isswBus -type Queue -destination isswQueue -role sender -group isswUsers'])
AdminTask.addUserToDestinationRole(['-bus isswBus -type Queue -destination isswQueue -role sender -user bruce'])
```

```
AdminTask.removeGroupFromDestinationRole(['-bus isswBus -type Queue -destination isswQueue -role sender -group isswUsers'])
AdminTask.removeUserFromDestinationRole(['-bus isswBus -type Queue -destination isswQueue -role sender -user bruce'])
```

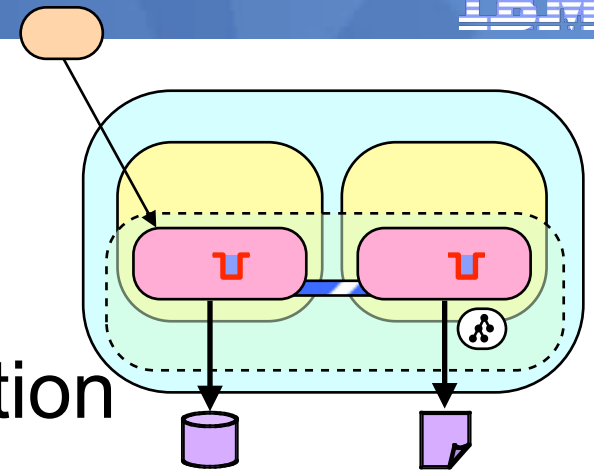
```
AdminTask.listGroupsInDestinationRole(['-bus isswBus -type Queue -destination isswQueue -role sender'])
AdminTask.listUsersInDestinationRole(['-bus isswBus -type Queue -destination isswQueue -role sender'])
```

```
AdminTask.isInheritDefaultsForDestination(['-bus isswBus -type Queue -destination isswQueue'])
AdminTask.setInheritDefaultsForDestination(['-bus isswBus -type Queue -destination isswQueue -inherit false'])
```

Temporary destinations

- Names are generated on creation

_Q **<prefix>** **<unique value>**



- Security configured using just the prefix

Advanced Messaging

Read ahead
Default

Temporary queue name prefix
prefix

Temporary topic name prefix

Share durable subscriptions
In cluster

- Prefix set on JMS Connection Factory

Temporary destinations have generated names, which means that the existing mechanism we use to configure destination security does not, by itself work. Temporary destinations are named with three parts. A constant “_Q” that denotes them as temporary queue (a temporary topic is indicated with “_T”), a client specified prefix and then a uniquely generated value. This produces a unique name. The prefix specified by the client is then used to configure security.

The prefix is set on the JMS connection factory, and by default is the empty string.

Panel Path:

Resources > JMS > [Queue|Topic] connection factories > {connection_factory_name}

Script:

```
server = AdminConfig.list('Server')
cf = AdminTask.listSIBJMSConnectionFactory(server)
AdminTask.modifySIBJMSConnectionFactory(cf, '[-tempQueueNamePrefix prefix]')
```


Temporary destination authorization

- Only two roles relevant

sender

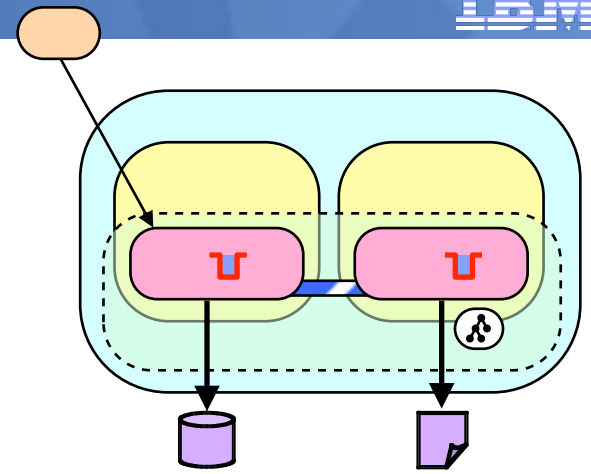
creator

- Example:

```
AdminTask.addGroupToDestinationRole('[ -type Queue
    -bus isswbus -destination "isswTmpDest" -role
Creator
    -group isswUsers]')
```

- Resultant queue name

_Q **isswTmpDest** **_A334C30000001**



Only two of the roles are relevant for temporary destinations. Sender, which dictates who can send messages to the destination, and creator which dictates who can create temporary destinations using the prefix.

Consumers can only be created for a temporary destination using the connection that created them, so the receiver role is not required.

How does this work in practice? Well the script shown in the slide defined a temporary destination prefix of `isswTmpDest`. In this case only `isswUsers` can create temporary destinations with that prefix. When a temporary destination is created by a client in that group a destination with the name `_QisswTmpDest_A334C30000001`.

Script:

```
AdminTask.addGroupToDestinationRole('[ -bus isswBus -type Queue -destination prefix -role sender -group isswUsers]')
AdminTask.addUserToDestinationRole('[ -bus isswBus -type Queue -destination prefix -role sender -user bruce]')
```

```
AdminTask.removeGroupFromDestinationRole('[ -bus isswBus -type Queue -destination prefix -role sender -group isswUsers]')
AdminTask.removeUserFromDestinationRole('[ -bus isswBus -type Queue -destination prefix -role sender -user bruce]')
```

```
AdminTask.listGroupsInDestinationRole ('[ -bus isswBus -type Queue -destination prefix -role sender]')
AdminTask.listUsersInDestinationRole ('[ -bus isswBus -type Queue -destination prefix -role sender]')
```

```
AdminTask.isInheritDefaultsForDestination('[ -bus isswBus -type Queue -destination prefix]')
AdminTask.setInheritDefaultsForDestination('[ -bus isswBus -type Queue -destination prefix -inherit false]')
```

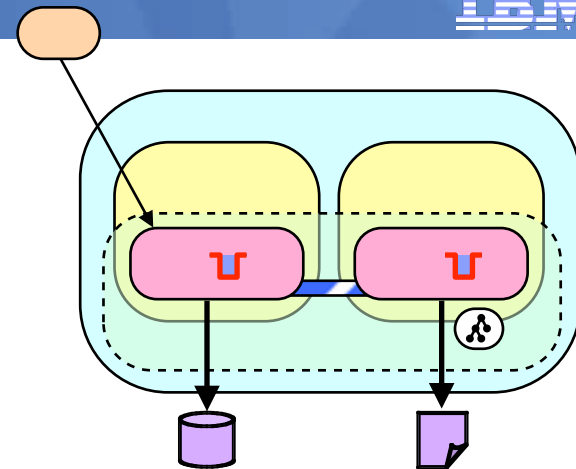
Default prefix

- Default prefix is the empty string
- Security can be applied to default:

```
AdminTask.addGroupToDestinationRole('[ -type Queue  
-bus isswbus -destination -role Creator  
-group isswUsers]')
```

- Prevent it being used:

```
AdminTask.setInheritDefaultsForDestination('[  
-type Queue -bus isswbus  
-destination -inherit false]')
```



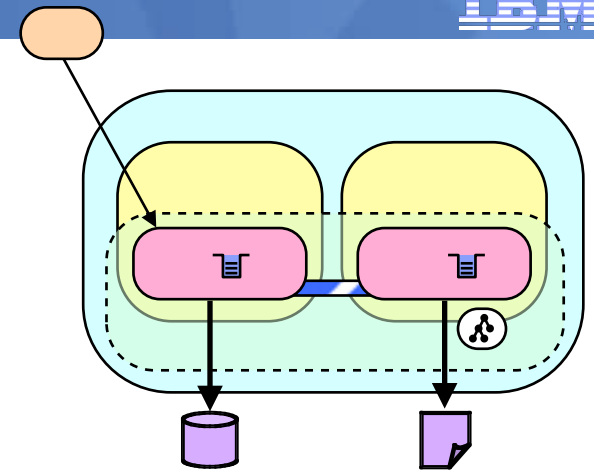
As mentioned previously the default prefix used by connection factories is the empty string. Security can still be configured for this prefix using wsadmin.

Agenda

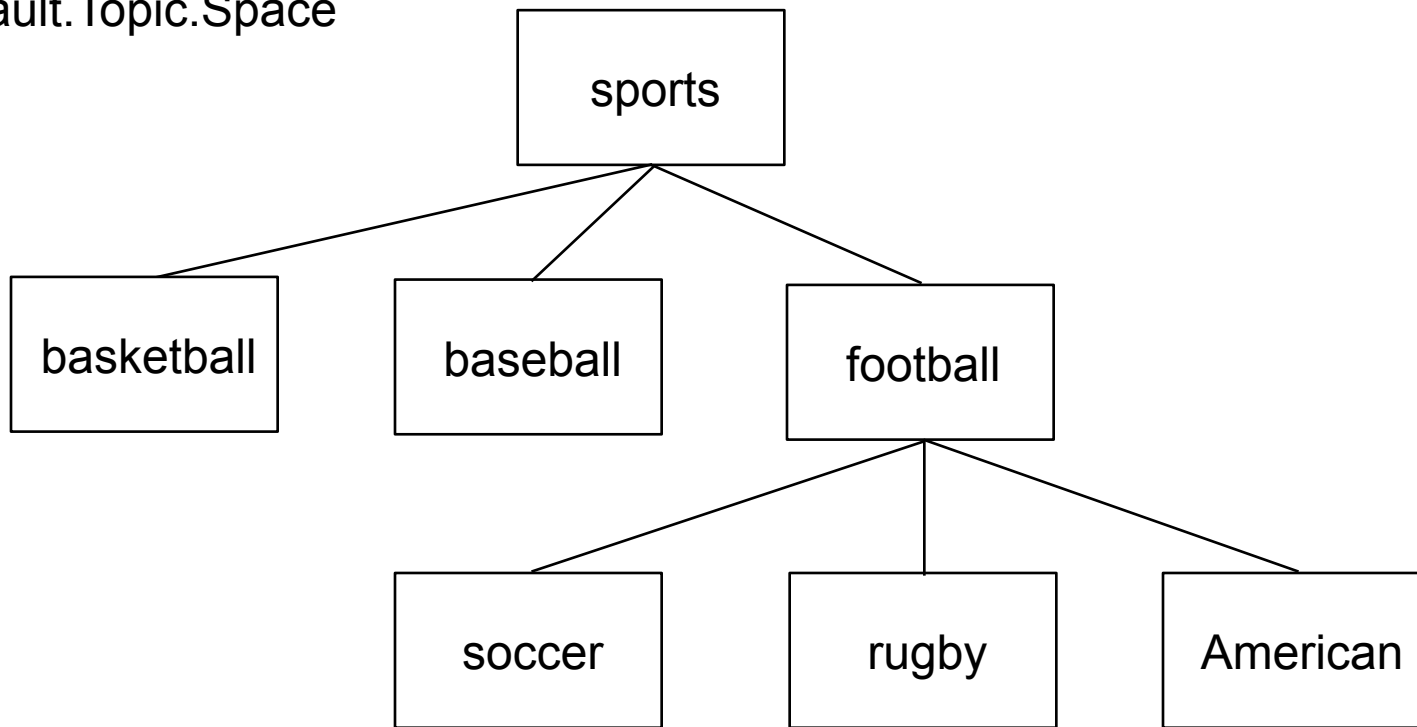
- Basic concepts
- Securing a bus
- Securing a messaging engine
- Connecting to a bus
- Destination authorization
- **Topic authorization**
- What is new in WAS v7

Topics

- Dynamic hierarchy of topics
- Contained in a Topic Space



Default.Topic.Space

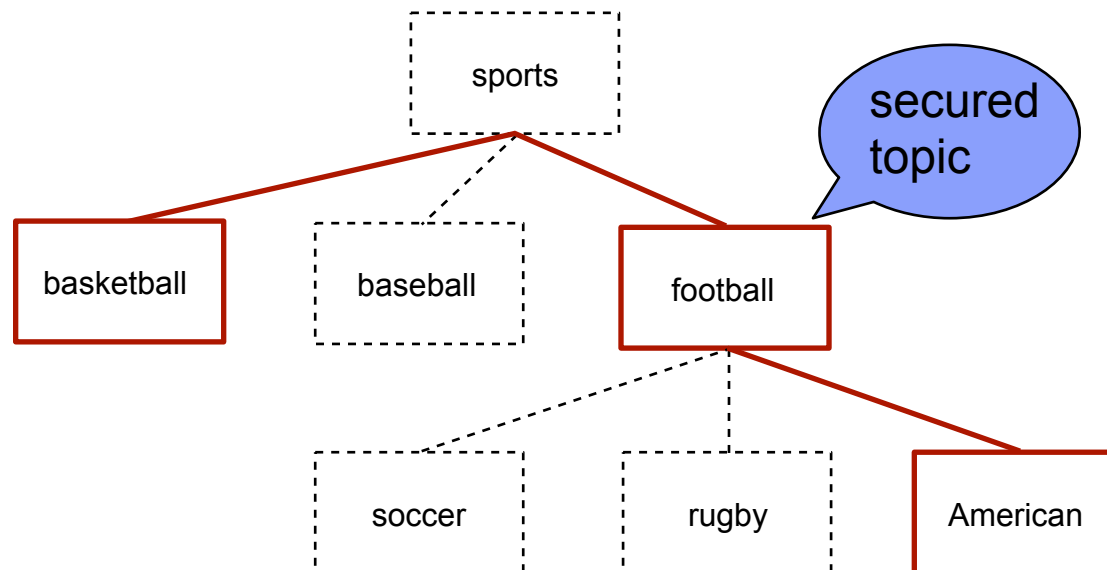
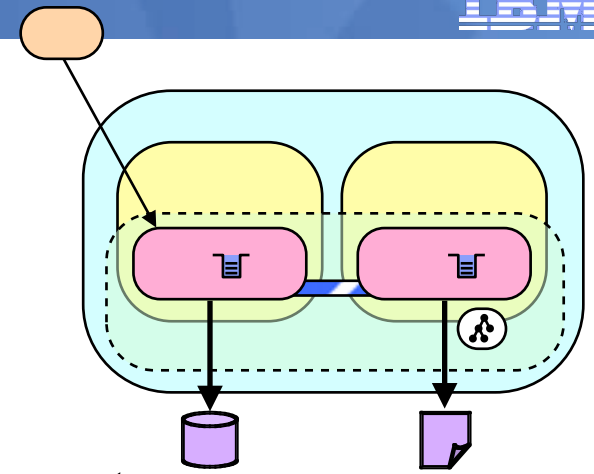


Topics exist in a hierarchical structure within a topic space. It is possible for a bus to host two identical topic hierarchies, but treat them separately by using multiple topic spaces (a topic space is a type of destination).

Topics are not managed administratively (although JMS topics may be). They are created when a subscription is created and exist only as long as there are subscribers.

Topics (2)

- Topics are pre-declared
- Child topics inherit roles from parents



So the question then becomes how do you apply security to the topics in a topic space. To do this security allows you to pre-declare topics that applications will use. This does not cause the topics to exist, but when a client application comes to use that topic the roles defined on that topic will be evaluated. Child topics (those shown with dashed boxes) inherit the authorizations of the parent in the hierarchy.

Script:

```
AdminTask.addGroupToTopicRole(['-bus isswBus -topicSpace Default.Topic.Space -topic sports/basketball -role sender -group isswUsers'])
```

```
AdminTask.addUsersToTopicRole(['-bus isswBus -topicSpace Default.Topic.Space -topic sports/basketball -role sender -user bruce'])
```

```
AdminTask.removeGroupFromTopicRole(['-bus isswBus -topicSpace Default.Topic.Space -topic sports/basketball -role sender -group isswUsers'])
```

```
AdminTask.removeUsersFromTopicRole(['-bus isswBus -topicSpace Default.Topic.Space -topic sports/basketball -role sender -user bruce'])
```

```
AdminTask.listGroupsInTopicRole(['-bus isswBus -topicSpace Default.Topic.Space -topic sports/basketball -role sender'])
```

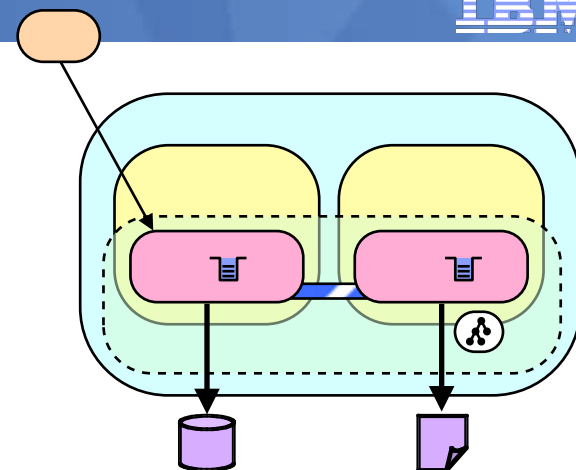
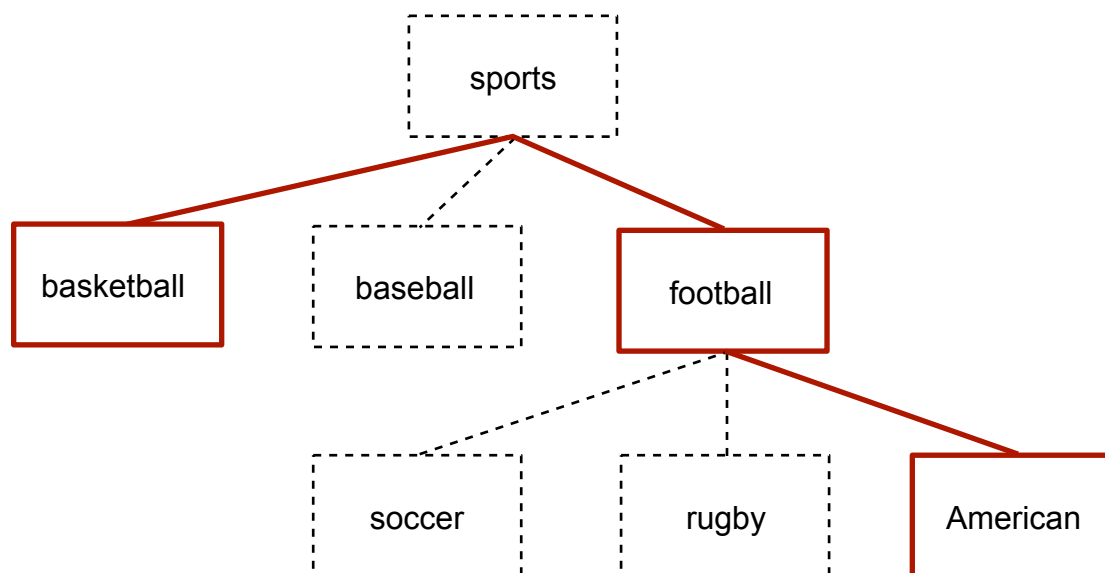
```
AdminTask.listUsersInTopicRole(['-bus isswBus -topicSpace Default.Topic.Space -topic sports/basketball -role sender'])
```

Topics (3)

- Applicable Roles

sender

recevier

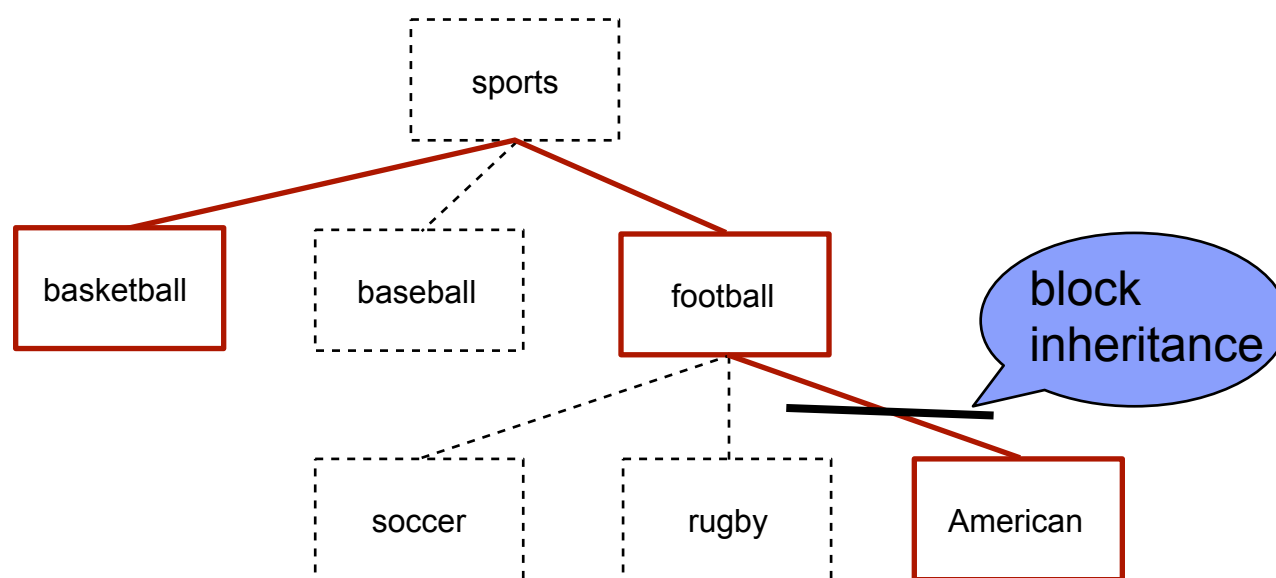
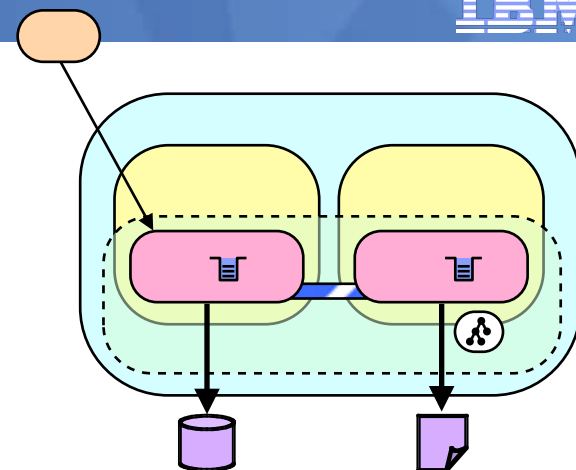


Topics only support the sender and receiver roles.

Any client must have sender or receiver access to the topic space that contains the topic before topic level security is applied. Topic level security can be disabled.

Topics (4)

- Topics inherit role assignments from their parent
- Inheritance can be blocked



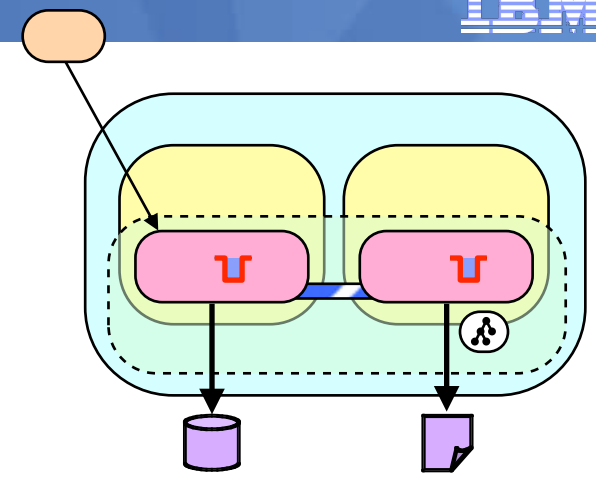
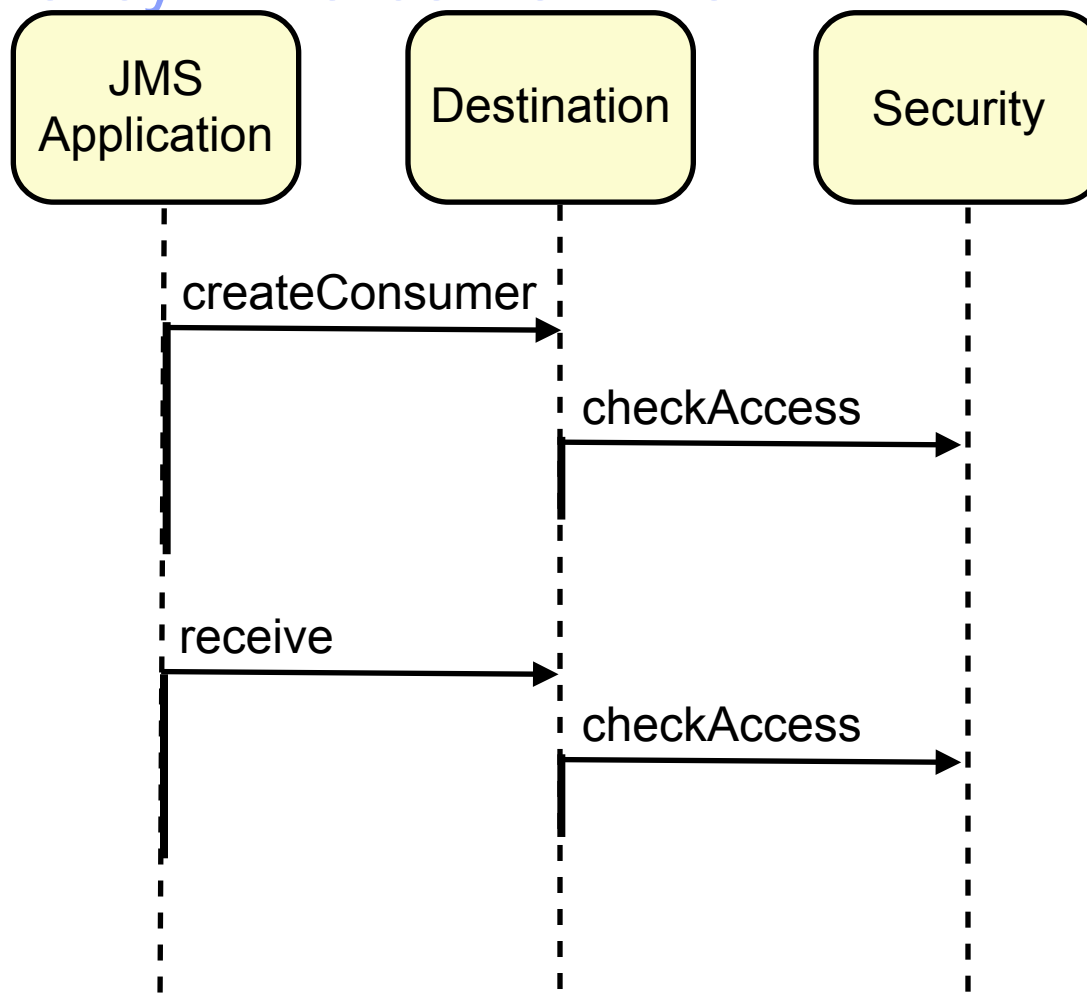
A topic at any point in the hierarchy inherits permissions from its parent topic. This can be blocked at any point. A block can be applied to only one of the roles, so you can block sender, but inherit receiver.

Script:

```
AdminTask.setInheritSenderForTopic(['-bus isswBus -topicSpace Default.Topic.Space -topic sports/football/American -inherit false'])
AdminTask.setInheritReceiverForTopic(['-bus isswBus -topicSpace Default.Topic.Space -topic sports/football/American -inherit false'])
```

```
AdminTask.isInheritSenderForTopic(['-bus isswBus -topicSpace Default.Topic.Space -topic sports/football/American'])
AdminTask.isInheritReceiverForTopic(['-bus isswBus -topicSpace Default.Topic.Space -topic sports/football/American'])
```

Policy Enforcement Point

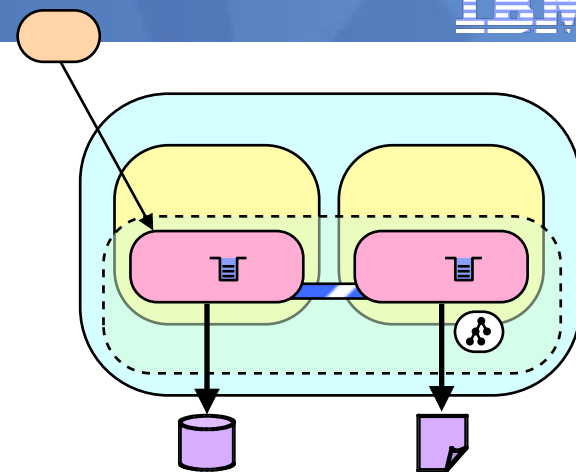
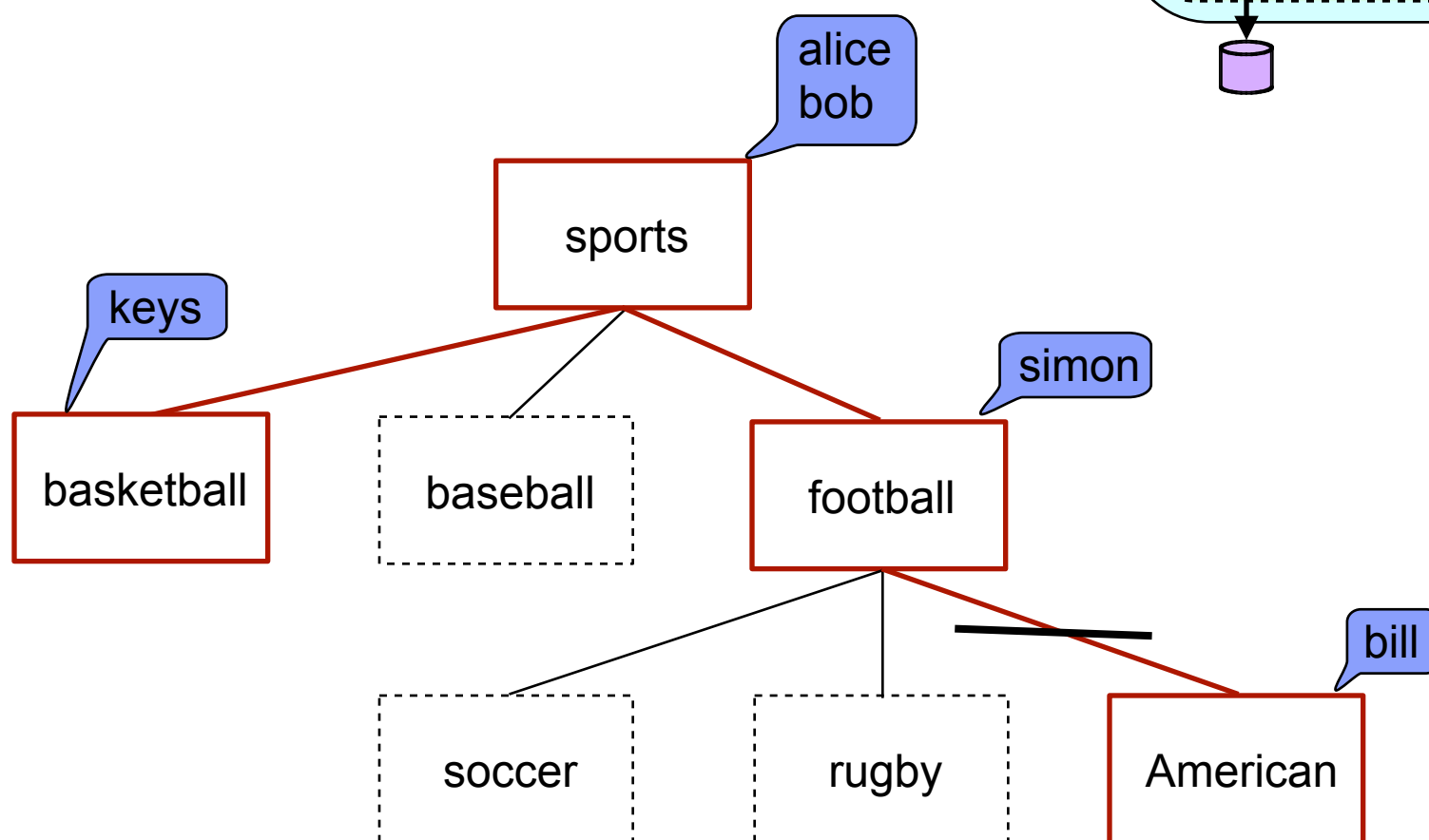


Access checks for topics work differently from destination access checks. When a consumer is created if a single topic is being subscribed to then an access check will occur then, if a wildcarded subscription (.e.g sports/football/*) is used no access check is made at subscription time.

When messages are “queued” up for delivery to a subscription a check is made to see if the subscriber has authority to the topic and if they do not then the message is silently discarded. As a result a subscriber who is not authorized to see messages will not receive feedback to that effect, but will not receive any messages.

From an external perspective the effect of this is that each receive runs a security check (although results are cached for performance reasons meaning that overall performance is not hugely affected).

Examples



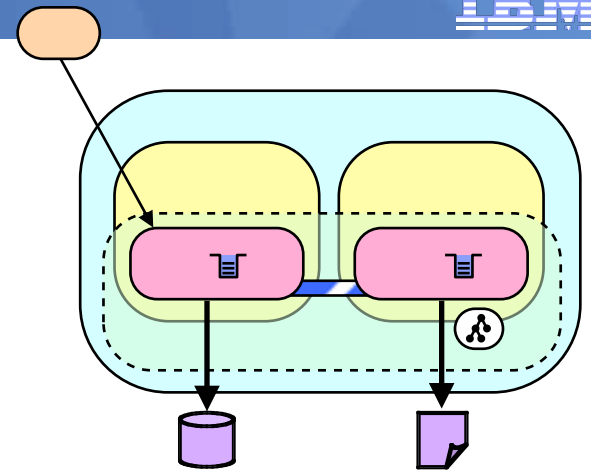
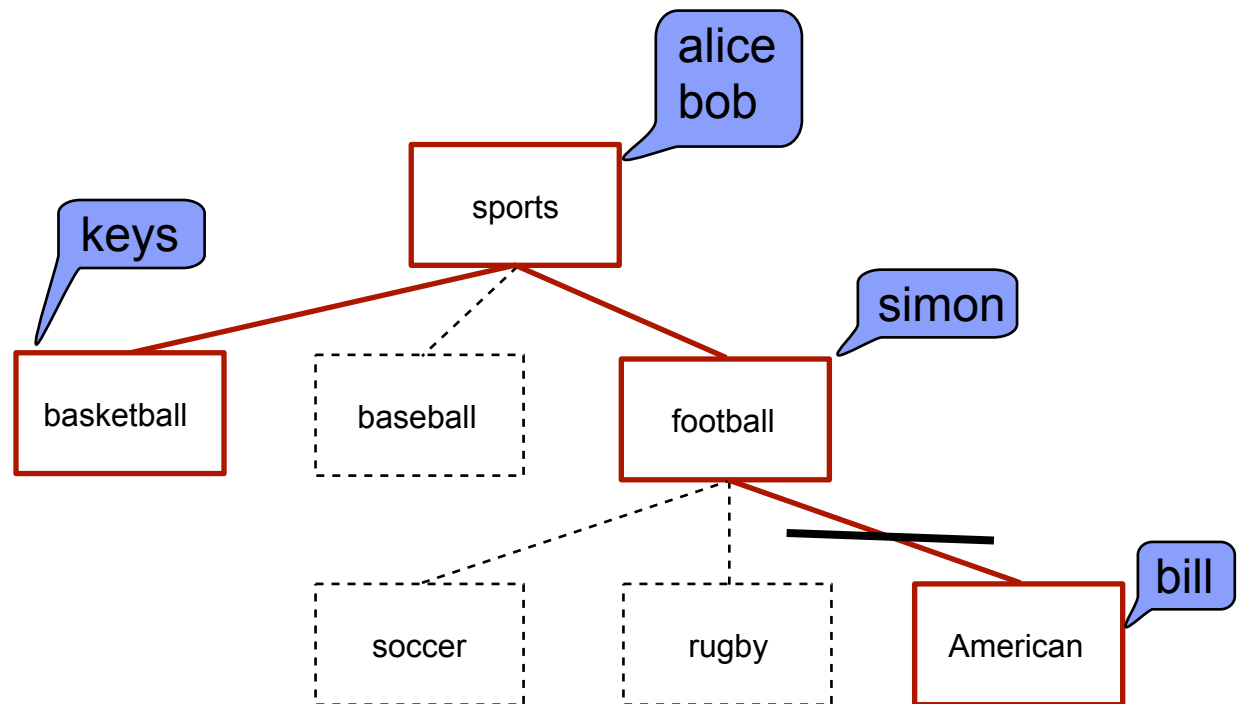
This is quite a complex part of authorization, so we should go through some examples. We will use this topic tree. with the indicated authorizations (for simplicity we will ignore the distinction between roles).

Examples (1)

- Who can access?

sports/basketball

alice
bob
keys



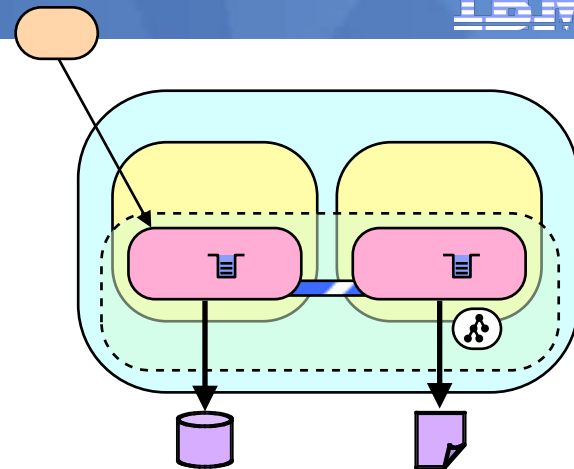
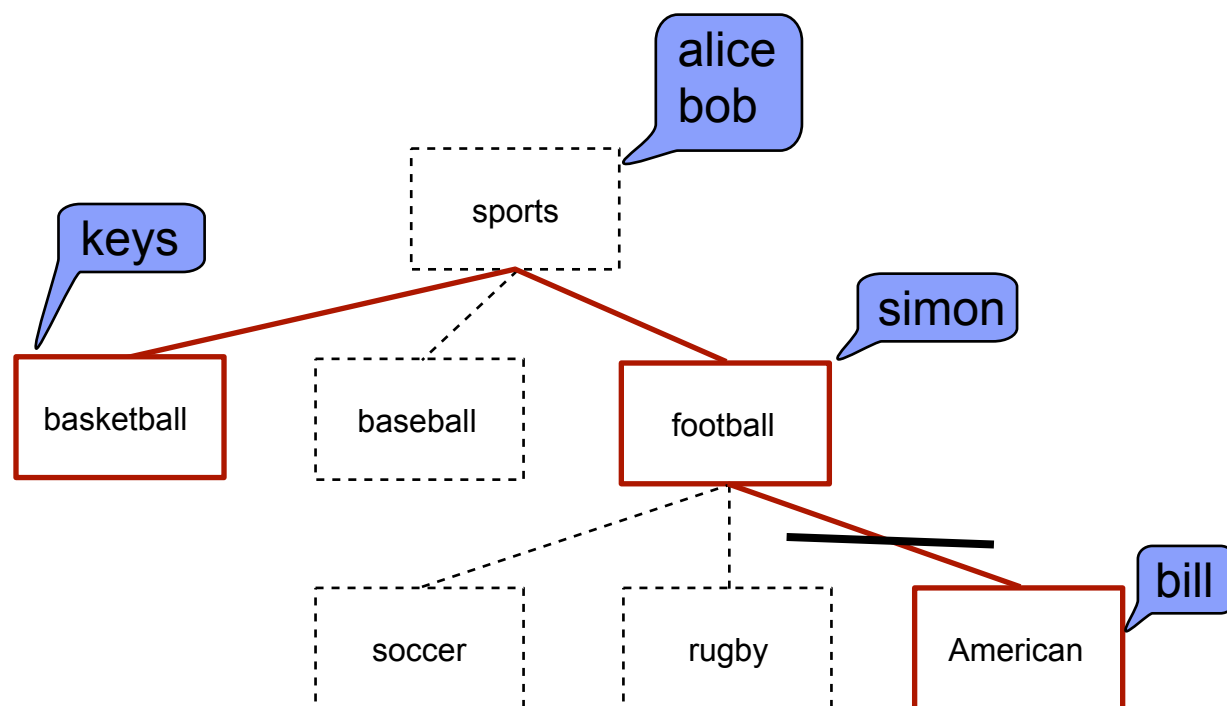
alice, bob and keys can access sports/basketball. keys has access directly on basket ball, and alice and bob have access to the sports topic and their access is inherited.

Examples (2)

- Who can access?

sports/football/soccer

alice
bob
simon



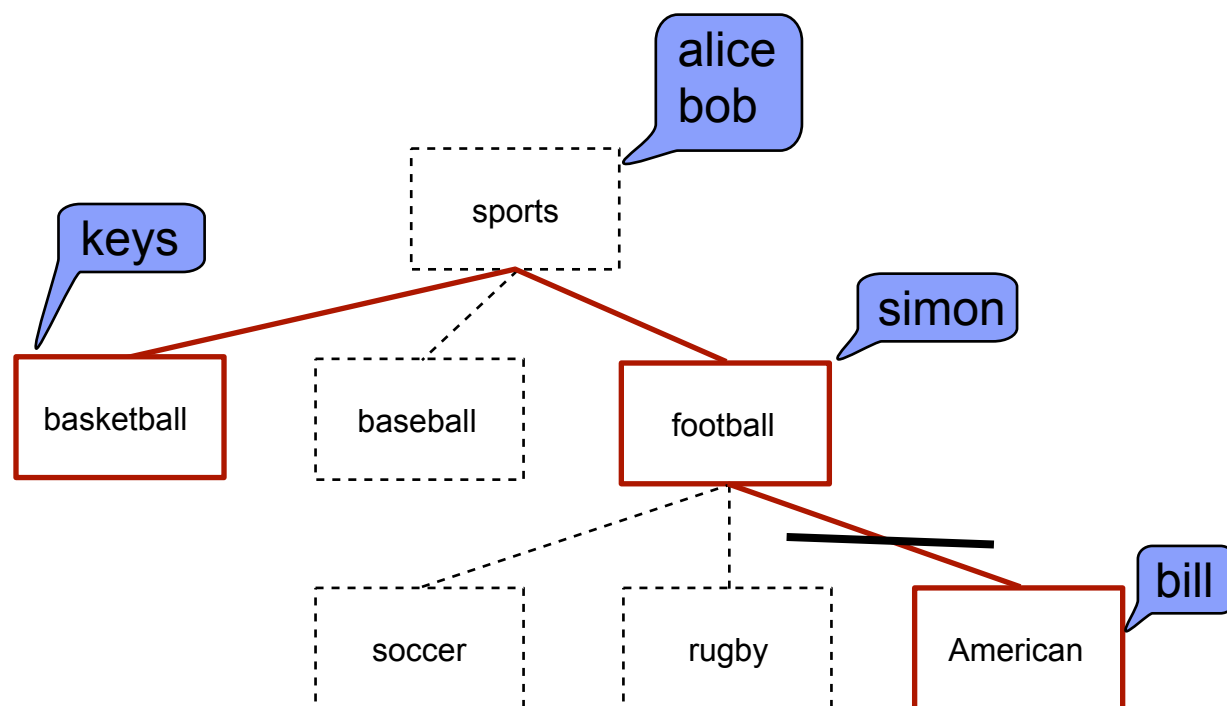
alice, bob and simon call all access sports/football/soccer. Simon can access sports/football and alice and bob can access sports.

Examples (3)

- Who can access?

sports/football/American

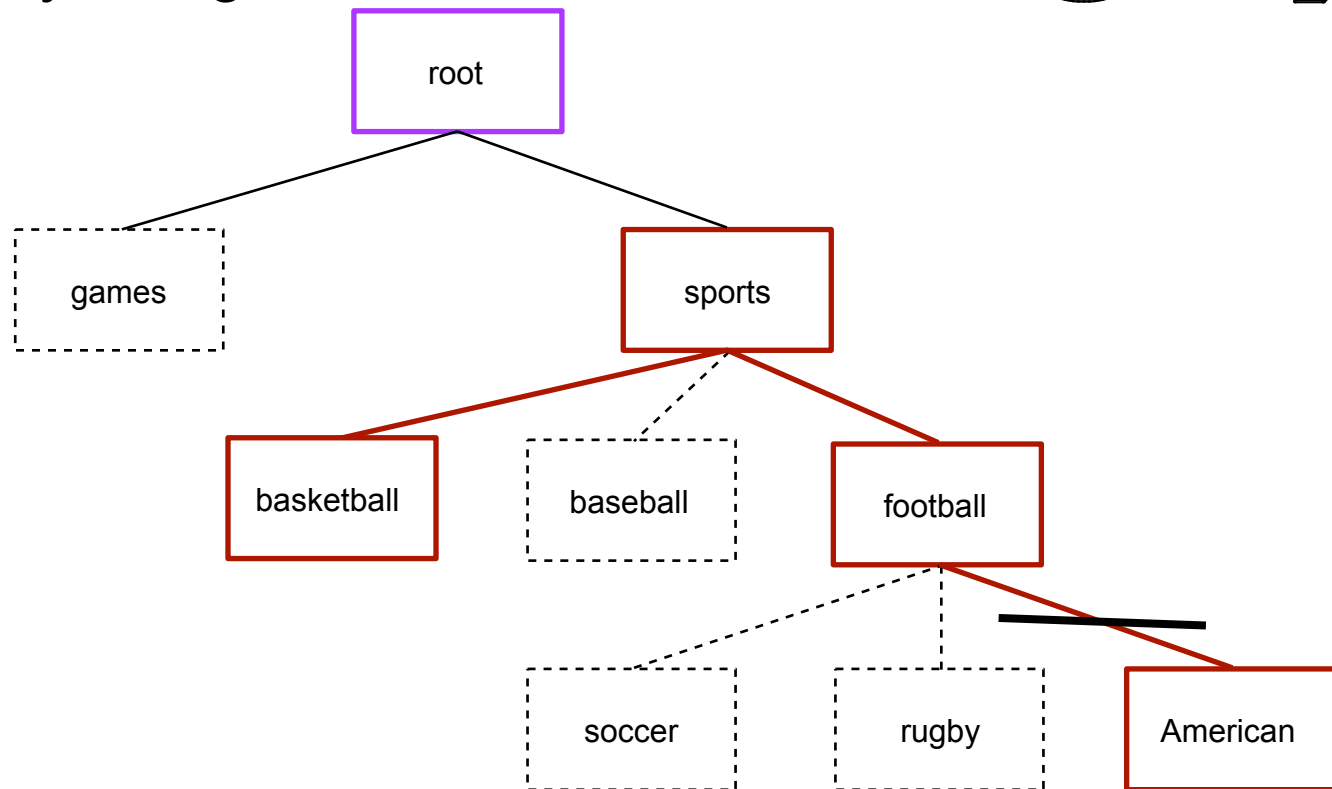
bill



Only bill can access sports/football/American because of the block in place.

Topic space root

- What is to stop a games hierarchy being added?



If we assume for a moment that the topic space is for the use of sports then how do you prevent someone creating a parallel hierarchy for games? This can be controlled using the topic space root object, which represents the logical root of all topics in the tree.

Script:

```
AdminTask.addGroupToTopicSpaceRootRole(['-bus isswBus -topicSpace Default.Topic.Space -role sender -group isswUsers'])
AdminTask.addUsersToTopicSpaceRootRole(['-bus isswBus -topicSpace Default.Topic.Space -role sender -user bruce'])
```

```
AdminTask.removeGroupFromTopicSpaceRootRole(['-bus isswBus -topicSpace Default.Topic.Space -role sender -group isswUsers'])
AdminTask.removeUsersFromTopicSpaceRootRole(['-bus isswBus -topicSpace Default.Topic.Space -role sender -user bruce'])
```

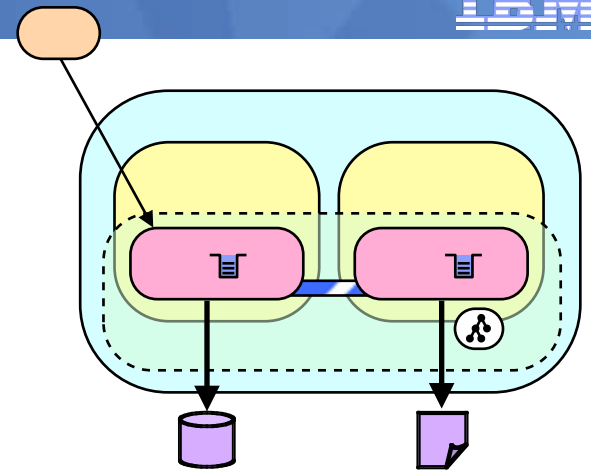
```
AdminTask.listGroupsInTopicSpaceRootRole(['-bus isswBus -topicSpace Default.Topic.Space -role sender'])
AdminTask.listUsersInTopicSpaceRootRole(['-bus isswBus -topicSpace Default.Topic.Space -role sender'])
```

Agenda

- Basic concepts
- Securing a bus
- Securing a messaging engine
- Connecting to a bus
- Destination authorization
- Topic authorization
- **What is new in WAS v7**

Highlights

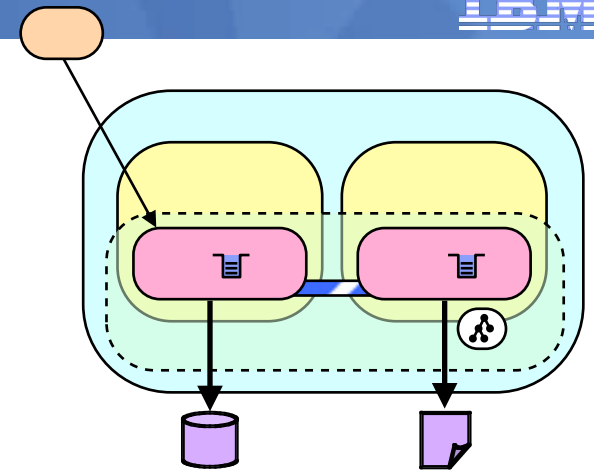
- Admin console support for authorizations
- Multi domain
- Audit
- Simplified admin
- WebSphere MQ



WAS v7 includes the most significant update to security for the service integration bus since WAS v6.0. The following slides go over some of these and most (but not all) are available in the open beta. This page lists some highlights.

Authorization administration

- Authorization model can be fully configured using admin console
- Accessed via bus security panel



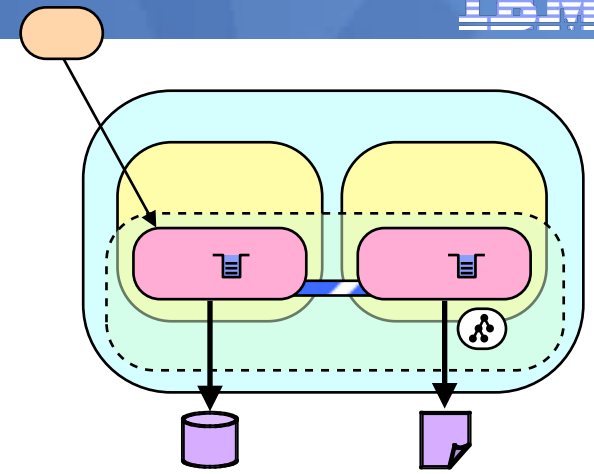
Authorization Policy

- [Users and groups in the bus connector role](#)
- [Manage default access roles](#)
- [Manage destination access roles](#)
- [Manage foreign bus access roles](#)
- [Manage temporary destination prefix access roles](#)
- [Manage topic access roles](#)
- [Manage users and groups not known to the user repository](#)

The authorization model is now fully exposed in the administrative console. It is accessed by the bus security panel. It is important to note that destination access roles, and topic access roles cannot be managed in the open beta.

The following 5 slides will walk through adding a group to the default access roles.

Administer defaults



Buses

Buses > Security for bus isswBus > Default access roles

This panel shows the role type assignments for the default access.

☐ Default access Add Remove

Select	Name	Type	Sender	Receiver	Browser	Creator
<input type="checkbox"/>	AllAuthenticated	Group	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Apply OK Cancel

wizard for adding new users or groups to role

allows access to be allowed or disallowed

When clicking to manage the default access roles you get this summary panel which shows all users and groups and which roles they have (shown with a tick). The destination access roles panel (not shown here) also indicates where roles have been inherited from the default roles. These role assignments can be added or removed (and clicking apply causes them to come into effect).

The add button allows new users and groups to be granted roles, and the remove button will remove selected groups from the resource.

Add wizard

SIB Security Resource Wizard

SIB Security Resource Wizard

→ Step 1: Search for Users or Groups

Step 2: Select Users or Groups

Step 3: Select Role Types

Step 4: Summary

Search for Users or Groups

Use this search to search for the users and groups to give access to for the resource.

Resolve

Default



search groups or users

Groups



Users

* Search pattern

*

search registry

* Maximum number of search results

20

limit the number of results

Next

Cancel

Once the wizard is launched the registry is queried. Only users or groups can be added on any path through the wizard.

Select groups

SIB Security Resource Wizard

SIB Security Resource Wizard

Step 1: Search for Users or Groups

→ **Step 2: Select Users or Groups**

Step 3: Select Role Types

Step 4: Summary

Select Users or Groups

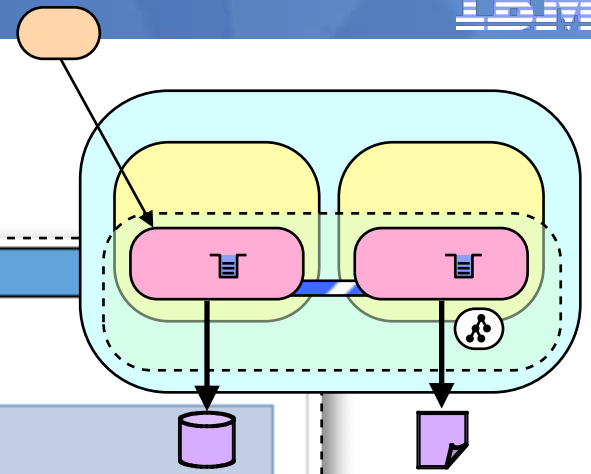
Use this step to select the users and groups to grant access for.

Resource: Default

select the groups of interest

Select	
<input checked="" type="checkbox"/>	Hursley
<input type="checkbox"/>	isswUsers
<input checked="" type="checkbox"/>	isswNordic
<input checked="" type="checkbox"/>	isswUKISA
<input type="checkbox"/>	wasDev
<input type="checkbox"/>	isswUSA

Next



Once the registry has been queried the list of groups can be further narrowed down.

Assign roles

SIB Security Resource Wizard

SIB Security Resource Wizard

Step 1: Search for Users or Groups
Step 2: Select Users or Groups
→ **Step 3: Select Role Types**
Step 4: Summary

Select Role Types

Use this step to select the role types granted to the users and groups.

Resource
Default

Name	Send	ser	Creator
Hursley	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
isswNordic	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
isswUKISA	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

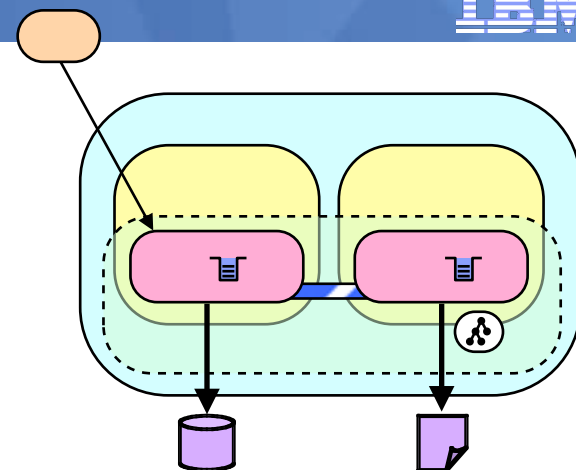
assign the relevant roles

Previous Next Cancel

Finally roles can be assigned to the users or groups.

7.0

Results



☐ Default access Add Remove

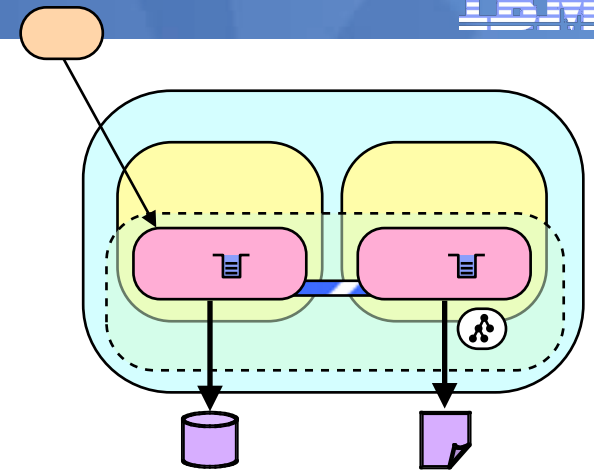
Select	Name	Type	Sender	Receiver	Browser	Creator
<input type="checkbox"/>	AllAuthenticated	Group	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	Hursley	Group	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	isswNordic	Group	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	isswUKISA	Group	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Apply OK Cancel

The collection is updated and there is no need to click apply or ok. Those buttons are there only for changes made directly on that page.

Multi domain

- A bus can be in its own domain
- All messaging engines use one domain.



Bus security domain

- ☐ Use the cell security domain
- ☐ Use the cell default security domain
- ☒ Use the selected domain

IsswBusDomain

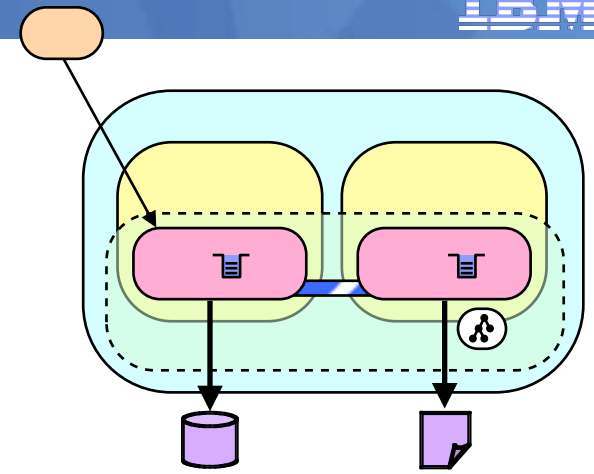
[Configure security domain IsswBusDomain](#)

will read
global domain
when released

A major new feature for WAS v7 is multi domain support. A bus can be configured as a part of a domain. The bus can be in the global (listed as "User the cell security domain" in the beta) which is required for mixed version buses, use the cell default security domain, or use a specific domain. All messaging engines in the bus will authenticate to the same domain.

Audit

- Authentication and authorization
- Using new audit support for WAS
- Can be disabled for a bus
- Can be disabled for a topic space



Audit

☒ Enable the auditing service for this bus

WAS v7 introduces a new audit service. A bus will (if configured) output audit events for authentication and authorization decisions. In the pub/sub world this will result in regular audit messages being output, so this can be disabled on a per topic space basis.

Script:

```
AdminTask.modifySIBus(['-bus isswBus -auditAllowed false'])
```

```
AdminTask.modifySIBDestination(['-bus isswBus -auditAllowed false'])
```

7.0

Securing intra-bus c

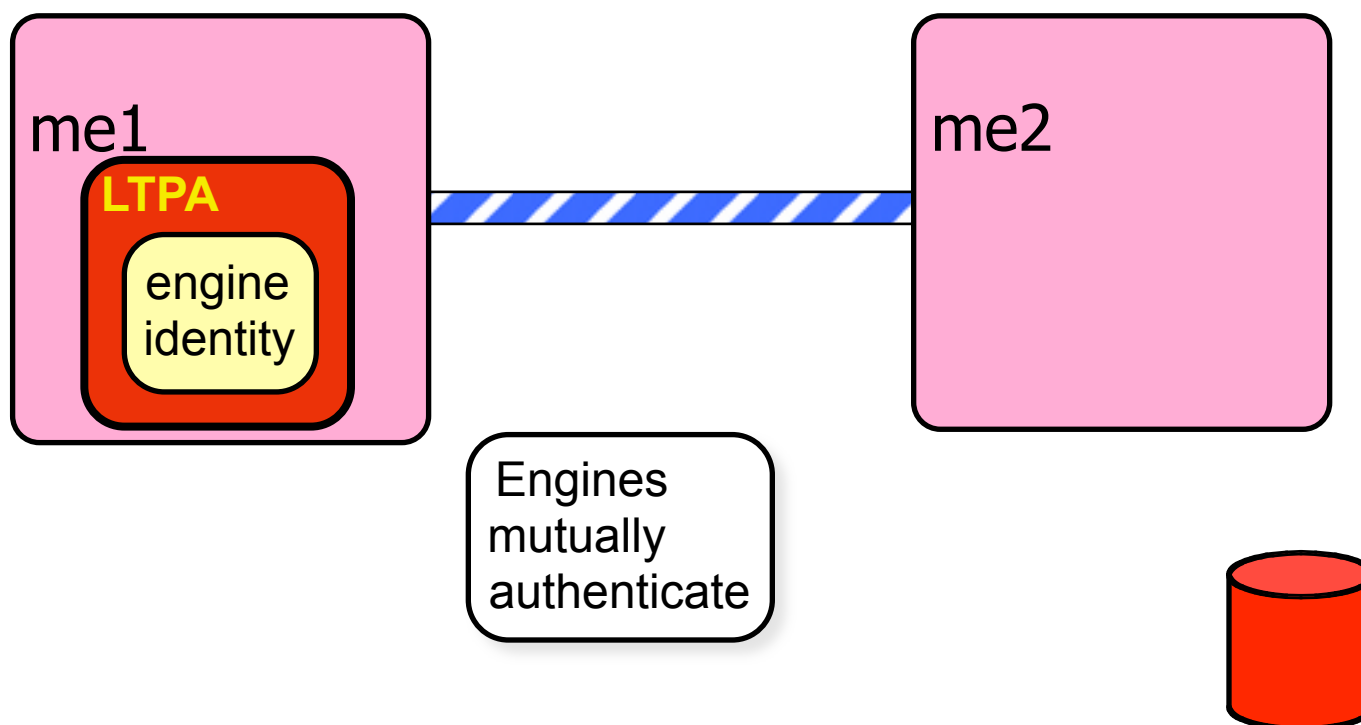
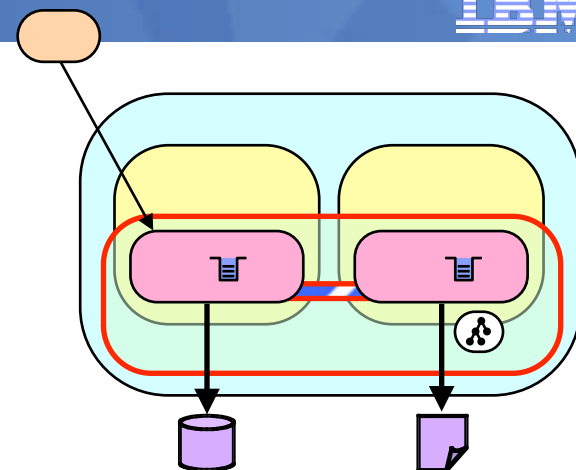
Security

☒ Enable bus security

Inter-engine authentication alias

supertedCellManager02/interEngineAlias

No longer
used in v7
buses

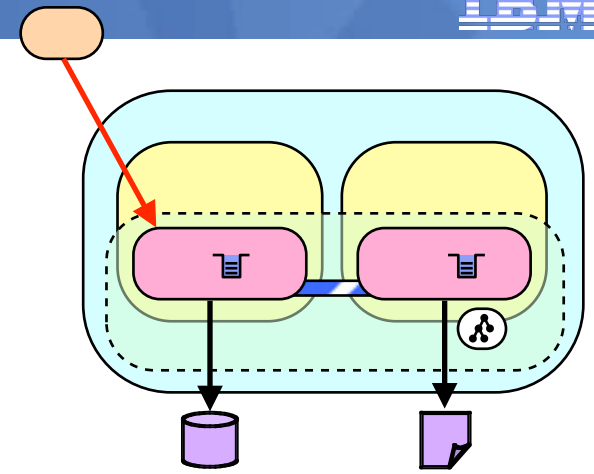


Security of intra-bus comms has also been updated. When peer messaging engines are both at version 7 they will not use the inter-engine authentication alias, but will mutually authenticate making use of specially crafted LTPA tokens.

7.0

XA recovery alias

- No longer required
 - Unless contacting a bus in another cell
- Server group must be in bus connector role

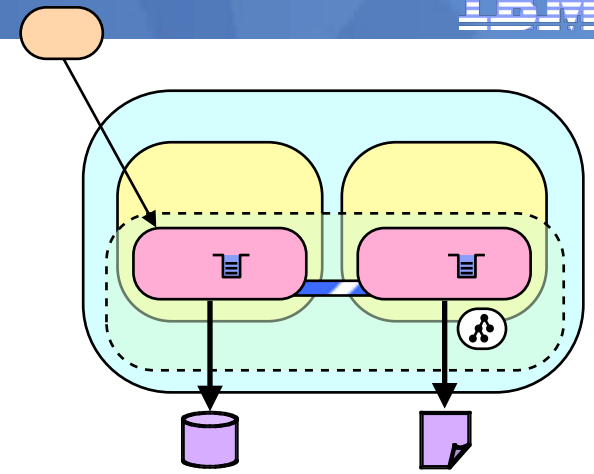


The XA Recovery alias is no longer required in some situations. The situations are:

1. The application server performing recovery and the messaging engine must be sharing LTPA keys (generally this means must be in the same cell).
2. The messaging engine must be running on a 6.1 or greater node.
3. The application server performing recovery must be running on a 7.0 node.
4. The server group must be in the bus connector role.

Permitted chains

- Chains that are not permitted will not be started
- Ports will not be open
- Chains started or stopped when policy changes



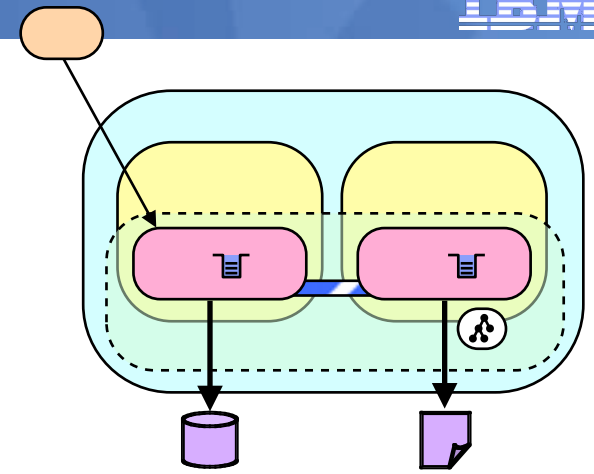
Permitted transports

- ☐ Allow the use of all defined transport channel chains
- ☒ Restrict the use of defined transport channel chains to those protected by SSL
- ☐ Restrict the use of defined transport channel chains to the list of permitted transports

In v7 chains that are not permitted will not be started, so the ports will not be open. Updates to this policy will be dynamic, so making a chain permitted will cause it to start automatically.

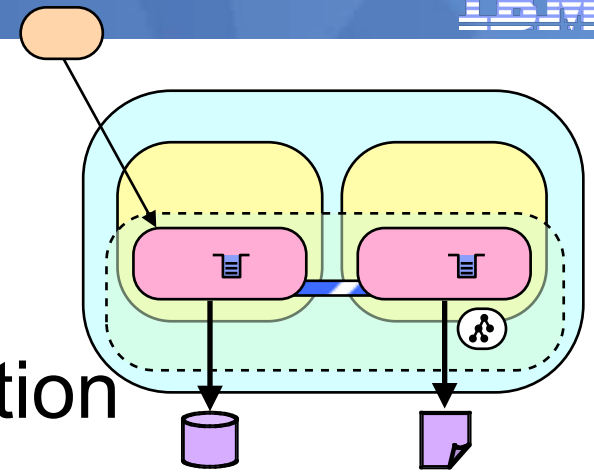
Client SSL authentication

- Supported
- Identity in client certificate used for authorization
- 6.x clients are supported
- Simply configure the client to use mutual SSL and provide no, or null credentials



Mutual SSL authentication is properly supported. In previous versions this could be configured, but a userid/password would still need to be exchanged in order for the bus to make authorization decisions. A v7 messaging engine will automatically support mutual SSL authentication and use the certificate DN for authorization. The certificate DN is mapped to a user in the registry by the user registry using existing mechanisms. v6.x clients can exploit this feature. Clients using this must provide no credentials (or a null userid and password). If a username and password are provided these will take precedence over the identity in the clients certificate.

WebSphere MQ

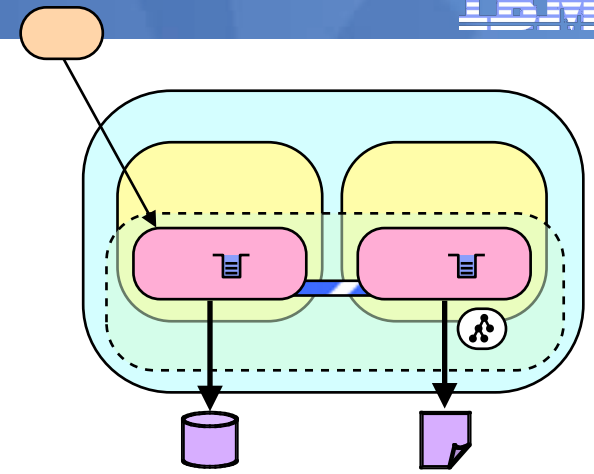


- Major Revamp of JMS integration
- JMS provider uses JVM wide keystore and trust store settings
- Unreliable as WAS changed these
- PK24431 added ssl.repertoire for per connection factory key and trust store

The WebSphere MQ messaging provider has had a significant update to support WebSphere MQ v7. This has included updates to the SSL support. Originally SSL was not well supported and the WebSphere MQ client used the JVM wide keystore and trust store settings. This caused problems because the WAS admin subsystem would change these as it needed to. This meant that getting WebSphere MQ connections using SSL was not reliable, so to help PK24431 was released. This allowed a custom property of `ssl.repertoire` to be added to a connection factory and the keystore and trust store location would be taken from the specified `ssl` configuration.

WebSphere MQ SSL

- Per connection factory SSL configuration supported



Server connection channel
SYSTEM.DEF.SVRCONN

☒ Use SSL to secure communication with WebSphere MQ

☐ Centrally managed

☒ Specific configuration

SSL configuration
WebSphereMQSSLConfiguration

In v7 this support is improved and the connection factory panel now has the ability to configure per connection factory ssl configurations.

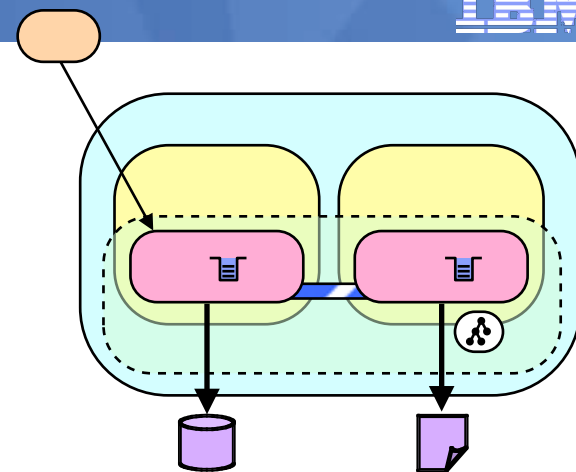
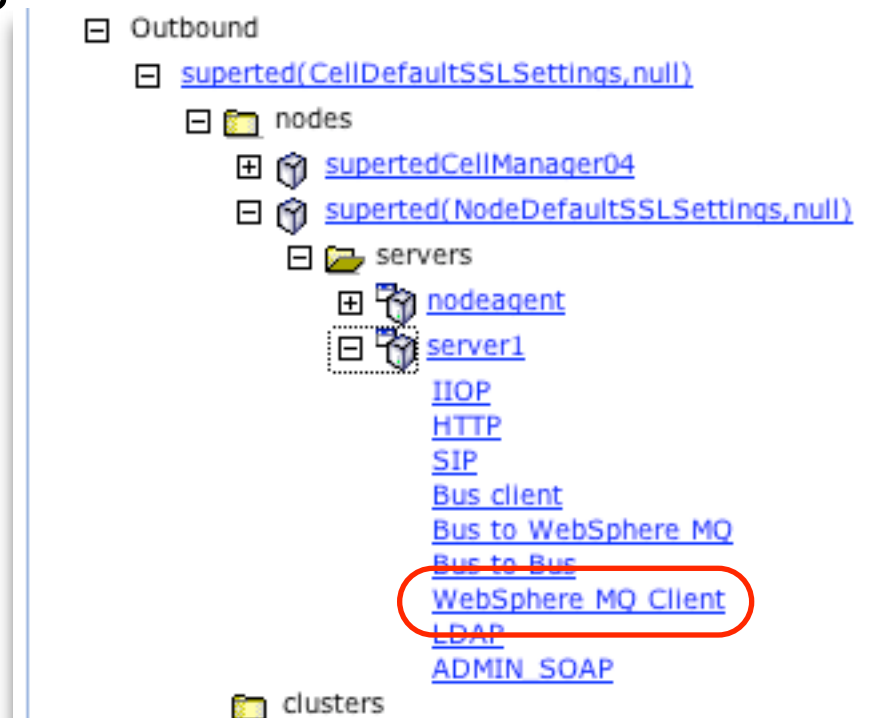
It also allows this to be specified centrally.

Panel Path:

Resources > JMS > [Queue|Topic] connection factories > {connection_factory_name}

WebSphere MQ SSL (2)

- Centrally managed SSL configuration



When centrally configuring SSL it appears under WebSphere MQ Client.

Panel Path:

Security > SSL certificate and key management > Manage endpoint security configurations

WebSphere MQ security

- Additional SSL properties still supported

Additional SSL settings

Certificate revocation list

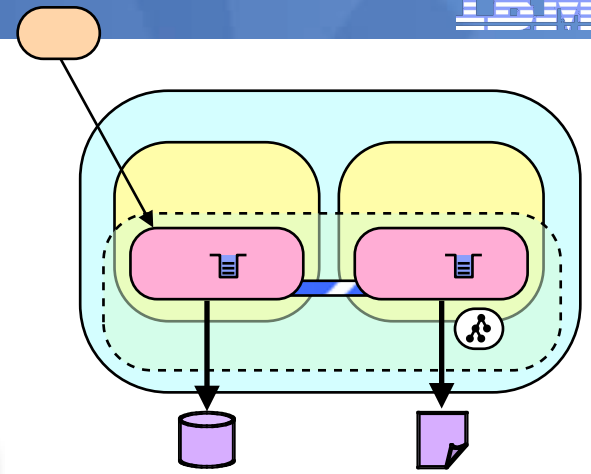
Peer name

Reset count

- Admin console support for security exits

Security exit

Security exit initialization data



It is still possible to configure the a CRL, and SSL peer name. There is also improved support for exists, including security exists.

Panel Path:

Resources > JMS > [Queue|Topic] connection factories > {connection_factory_name} > Client transport properties

Summary

- Basic concepts
- Securing a bus
- Securing a messaging engine
- Connecting to a bus
- Destination authorization
- Topic authorization
- What is new in WAS v7

References

- **WebSphere Application Server v6.1 infocenter**
<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp>
- **WebSphere Application Server V6.1: JMS Problem Determination**
<http://www.redbooks.ibm.com/redpieces/abstracts/redp4330.html?Open>
- **Connect an application to the default message provider using the run-as identity**
http://www.ibm.com/developerworks/websphere/techjournal/0710_col_nottingham/0710_col_nottingham.html

The full urls for this slide are as follows:

WebSphere Application Server v6.1 infocenter
<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp>

WebSphere Application Server V6.1: JMS Problem Determination
<http://www.redbooks.ibm.com/redpieces/abstracts/redp4330.html?Open>

Connect an application to the default message provider using the run-as identity
http://www.ibm.com/developerworks/websphere/techjournal/0710_col_nottingham/0710_col_nottingham.html

WebSphere Application Server v7.0 open beta
<https://www14.software.ibm.com/iwm/web/cc/earlyprograms/websphere/wasndv7/index.shtml>

Questions

- Please remind students to complete their session evaluation form
- Thank you!!

