StreamServe™ Persuasion™

StreamServe Persuasion SP5 Service Broker

User Guide

Rev A

StreamServe Persuasion SP5 Service Broker User Guide
Rev A

StreamServe Web Site
http://www.streamserve.com

Contents

About Service Broker

The Service Broker is a mediator of services provided by one or more StreamServers.

Publishing services

A StreamServer can publish services in several Service Brokers. Each service is identified by a name and version. Any type of StreamServer task can be published as a service. For example, conversion of XML input to page formatted output, and delivery of the converted output to a printer. A service can also be configured to respond with processed output.

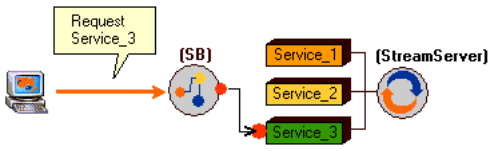
See *Server configuration* on page 7.

Invoking services

A client invokes a service by requesting the service name from a Service Broker.

Example 1 *Invoking a service*

1. A client requests the service `Service_3` from a Service Broker.
2. The Service Broker finds `Service_3` in its list of services.
3. The client accesses the `Service_3` and sends input, via the Service Broker, to the appropriate StreamServer.



See *Client configuration* on page 11.

Multiple StreamServers publishing the same service

Several StreamServers can publish the same service. When a client requests a service, the Service Broker decides which StreamServer to invoke. The client does not have to know anything about server ID, host, port, etc.

Encryption and authentication

Service providing StreamServers, Service Brokers, and Service Broker clients can be configured as SSL clients and SSL servers that use encryption and authentication for communication over HTTPS channels.

See *Encryption and authentication* on page 27

Web services

StreamServer services can be exposed via web services through the SOAP protocol.

See *Web services* on page 21

Service Broker properties in the Control Center

The Service Broker is automatically added as a 4.x Services node service to the Control Center when you install StreamServe Tools. You can edit the Service Broker properties. See *Service Broker settings in the Control Center* on page 37.

Publishing and invoking services

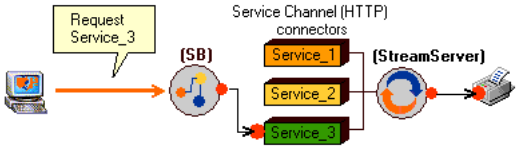
This section describes how to configure a StreamServer as a server that publishes services, or as a client that invokes services via a Service Broker.

Server configuration

A StreamServer can be configured as a service providing server. A service providing StreamServer offers two types of services:

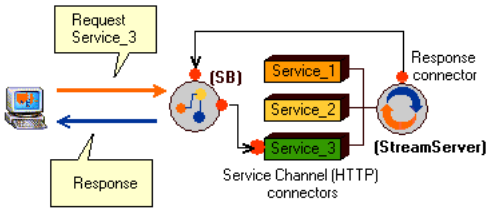
- Services forwarding the processed output

  This type of service includes reception and processing of the input from the client. The processed output is sent to a printer, via fax, email, etc.



- Services returning the processed output

  This type of service includes reception and processing of the input from the client. The processed output is sent back to the client.



Creating services that forward the processed output

The Event, Process, and output connector configurations are the same as in a scenario where a client sends input directly to a StreamServer. The key feature in a Service Broker scenario is the Service Channel (HTTP) input connector:

- With this connector you specify the name and version of the service to publish.
- The client sends the input via the Service Broker to this connector.

To configure a Service Channel (HTTP) input connector

Create and configure the *Service Channel (HTTP) input connector* according to standard procedures. Specify the following settings:

- **Service description** – The name of the service published in the Service Broker. This name will be used in the client requests when requesting this specific service. You can define several names, separated by comma, for the same service.
- **Version** – The version of the service. There can be several versions of the same service. A client can request a specific version of a service. If the client does not request a specific version, it will get the highest available version of the service.

Examples

See *Forwarding output to a printer* on page 15.

Creating services that return the processed output

The Event and Process configurations are the same as in a scenario where a client sends input directly to a StreamServer. The key features in a Service Broker scenario are the Service Channel (HTTP) input, and the Service Channel Response (HTTP) output connectors:

- With the Service Channel (HTTP) input connector you specify the name and version of the service to publish.
- The client sends the input via the Service Broker to the Service Channel (HTTP) input connector.
- With the Service Channel Response (HTTP) output connector you specify the content type of the response to the client.
- The response to the client is sent via the Service Channel Response (HTTP) output connector and the Service Broker to the client that issued the request.

To configure a Service Channel (HTTP) input connector

Create and configure the *Service Channel (HTTP) input connector* according to standard procedures. Specify the following settings:

- **Service description** – The name of the service published in the Service Broker. This name will be used in the client requests when requesting this specific service. You can define several names, separated by comma, for the same service.
- **Version** – The version of the service. There can be several versions of the same service. A client can request a specific version of a service. If the client does not request a specific version, it will get the highest available version of the service.

To configure a Service Channel Response (HTTP) output connector

Create and configure the *Service Channel Response (HTTP) output connector* according to standard procedures. Specify the following settings:

- **Content-type** – The content-type of the output. For example `text/html`.

Examples

See *Sending a response to a client* on page 16.

Publishing services

All services are automatically published/unpublished when you start/stop the StreamServer. This automatic publishing/unpublishing works if the Service Broker is up and running, and if you have enabled publishing of services.

To enable publishing of services

1. In the Design Center, activate the physical Platform layer you want to configure.
2. Right-click the Platform view and select **Configure Platform**. The Configure Platform dialog opens.
3. Select the **Service Broker** tab and configure the following:
   - **Service Broker host** – Specify the host where the Service Broker is running.
   - **Service Broker port** – Specify the port number used by the Service Broker.
   - **Enable services** – Select to enable publishing of services in the Service Broker.
   - **Port** – Specify the port to use to publish services, and to receive input via the Service Broker.

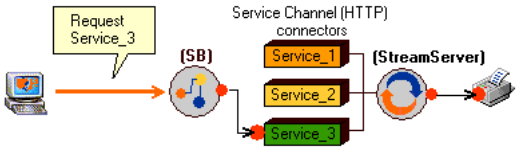
For more information about Service Broker settings, see *Service Broker settings in the Design Center* on page 35.

Client configuration

A StreamServer can invoke services via a Service Broker. There are two types of services:

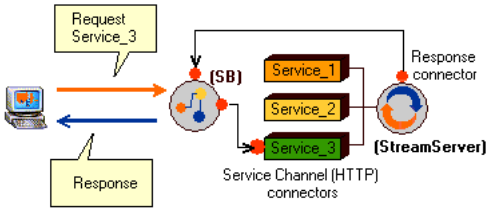
- Services forwarding the processed output

  The Service Broker forwards the output from the client to a StreamServer that has published the service. The StreamServer processes the data, and sends the processed output to a printer, via fax, email, etc.



- Services returning the processed output

  The Service Broker forwards the output from the client to a StreamServer that has published the service. The StreamServer processes the data, and sends the processed output back to the client.



Configuring a client – no response required

The Event, Process, and input connector configurations are the same as in a scenario where the client sends data directly to a StreamServer. The key feature in a Service Broker scenario is the Service Channel Submit (HTTP) output connector. With this connector you specify:

- Which Service Broker to connect to.
- Which service to request.
- The content-type of the data sent, via the Service Broker, to the server.
- That the server should forward the processed output to some kind of destination (no response required).

To configure the Service Channel Submit (HTTP) output connector

Create and configure the *Service Channel Submit (HTTP) output connector* according to standard procedures. Specify the following settings:

- **Service Broker Host** – The Service Broker host.
- **Port** – The Service Broker port.
- **Service Description** – The name of the requested service.

- **Version** – The version of the service. If you leave this empty, the highest registered version of the service is invoked.
- **Content-type** – The content-type of the submitted output. For example `application/pdf`.
- **Command** > **SendDocument**.

Examples

See *Forwarding output to a printer* on page 15.

Configuring a client – response required

The Event and Process configurations are the same as in a scenario where the client sends data directly to a StreamServer. The key features in a Service Broker scenario are the Service Channel Submit (HTTP) output and Service Channel (HTTP) input connectors.

With the Service Channel Submit (HTTP) output connector you specify:

- Which Service Broker to connect to and send the output to.
- Which service to request.
- The content-type of the data sent via the Service Broker to the server.
- That the server should return the processed output.
- The input connector that will receive the response.

The Service Channel (HTTP) input connector is the connector that receives the response.

To configure the Service Channel Submit (HTTP) output connector

Create and configure the *Service Channel Submit (HTTP) output connector* according to standard procedures. Specify the following settings:

- **Service Broker Host** – The Service Broker host.
- **Port** – The Service Broker port.
- **Service Description** – The name of the requested service.
- **Version** – The version of the service. If you leave this empty, the highest registered version of the service is invoked.
- **Content-type** – The content-type of the submitted output. For example `application/pdf`.
- **Command** > **SendReceiveDocument**
- **Response Service Channel** – The service name of the Service Channel (HTTP) input connector that will receive the response.

Note: The connector must be queue enabled, and the number of threads on the selected queue must be at least 2.

To configure the Service Channel (HTTP) input connector

Create and configure the *Service Channel (HTTP) input connector* according to standard procedures. **Service description** should be the same as `Response Service Channel` specified for the Service Channel Submit (HTTP) output connector above.

To enable reception of the response

To enable the reception of the response via a Service Broker, you must specify the Service Broker host and port. You must also specify on which client port to receive the response.

1. In the Design Center, activate the physical Platform layer you want to configure.
2. Right-click the Platform view and select **Configure Platform**. The Configure Platform dialog opens.
3. Select the **Service Broker** tab and configure the following:
   - **Service Broker host** – Specify the host where the Service Broker is running.
   - **Service Broker port** – Specify the port number used by the Service Broker.
   - **Port** – Specify the port to use to publish services, and to receive input via the Service Broker.

For more information about Service Broker settings, see *Service Broker settings in the Design Center* on page 35.

Examples

See *Sending a response to a client* on page 16.

Attaching documents to a response

A StreamServer can use a Service Channel Submit (HTTP) output connector to attach documents to a response. The Service Broker takes the documents and the response, creates a multipart MIME document, and sends it to the client. A typical scenario is to add status information to the response.

To configure the Service Channel Submit (HTTP) output connector

Create and configure the *Service Channel Submit (HTTP) output connector* according to standard procedures. Specify the following settings:

- **Service Broker Host** – The Service Broker host.
- **Port** – The Service Broker port.
- **Content-type** – The content-type of the submitted output. For example `application/pdf`.
- **Command** > **AddDocument**
- **Doc Group ID** – The response to add documents to. The instance that issued the `SendReceiveDocument` request is the instance that will receive the response. The instance that issued the `AddDocument` request is the instance that will add the document to the above response. The document will be added to the response if either:
  - Both instances use the same Doc Group ID.
  - No Doc Group ID is specified, and the response and document to add have the same Strs ID.
- **LastDoc** – Defines the current document as the last document to add.

Examples

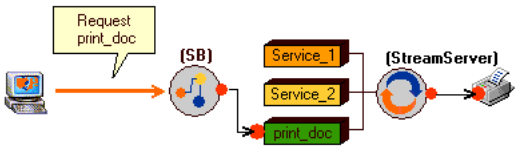
See *Adding status information to the response* on page 19.

Examples

Forwarding output to a printer

- The Service Broker runs on the address `134.123.6.87:1717`.
- A StreamServer has published the service `print_doc` version 2. It uses port `1616` to communicate with the Service Broker.
- Another StreamServer invokes `print_doc` version 2 via the Service Broker.

The client requests `print_doc` version 2. The Service Broker forwards the document to the server. The server processes the document and sends the output to a printer.



Service Broker configuration

Control Center GUI	
IPport	1717

Server configuration

Configure Platform dialog – Service Broker tab	
Enable publishing of services, and specify the interface between the service providing server and the Service Broker.	
Service Broker host	134.123.6.87
Service Broker port	1717
Enable services	Select
Port	1616

Service Channel (HTTP) input connector settings
Specify the name and version of the service.

Service Channel (HTTP) input connector settings	
Service description	print_doc
Version	2

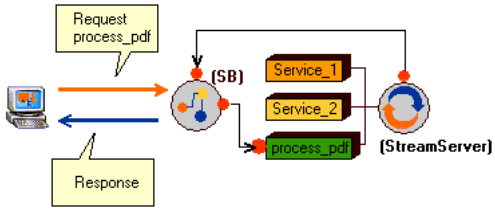
Client configuration

Service Channel Submit (HTTP) output connector settings	
Specify: • *The interface between the client output channel (request) and the Service Broker.* • *Which service and version to invoke.* • *That no response is required.*	
Service Broker Host	134.123.6.87
Port	1717
Service Description	print_doc
Version	2
Command	SendDocument

Sending a response to a client

- The Service Broker runs on the address `134.123.6.87:1717`.
- A StreamServer has published the service `process_pdf` version 2. It uses port `1616` to communicate with the Service Broker.
- Another StreamServer invokes `process_pdf` version 2 via the Service Broker.

The client requests `process_pdf` version 2. The Service Broker forwards the document to the server. The server processes the document, and sends the output back to the client.



Service Broker configuration

Control Center GUI	
IPport	1717

Server configuration

Configure Platform dialog – Service Broker tab	
Enable publishing of services, and specify the interface between the service providing server and the Service Broker.	
Service Broker host	134.123.6.87
Service Broker port	1717
Enable services	Select
Port	1616

Service Channel (HTTP) input connector settings	
Specify the name and version of the service.	
Service description	process_pdf
Version	2

Service Channel Response (HTTP) output connector settings	
Specify the format of the response. The response will be sent, via the Service Broker, to the client that issued the request.	
Content-type	application/pdf
Device driver	Select PDF as the device driver for this connector.

Client configuration

Service Channel Submit (HTTP) output connector settings	
Specify: • *The interface between the client output channel (request) and the Service Broker.* • *Which service and version to invoke.* • *That a response is required*. • *On which connector to receive the response*.	
Service Broker Host	134.123.6.87
Port	1717
Service Description	process_pdf
Version	2
Command	SendReceiveDocument
Response Service Channel	service_response
Queue settings	The connector is connected to the queue `Output`. This queue uses two threads (Advanced tab in the Manage Queues dialog box).

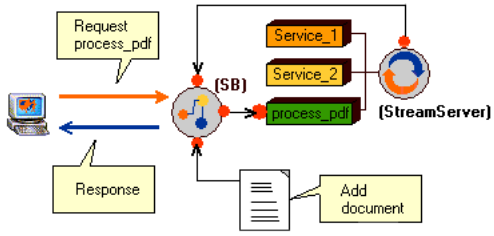
Configure Platform dialog – Service Broker tab	
Specify the interface between the client input channel (response) and the Service Broker.	
Service Broker host	134.123.6.87
Service Broker port	1717
Enable services	Do not select
Port	1616

Service Channel (HTTP) input connector settings
Specify the connector that will receive the response.

Service Channel (HTTP) input connector settings	
Service description	service_response
Version	1

Adding status information to the response

This example is an addition to *Sending a response to a client* on page 16. In this example, the server is also configured to add status information to the response sent to the client.



The status message is created using a Status Messenger input connector, a MessageIN Event, and a PageOUT Process. See the *Status Messenger* documentation for information on how to configure status messages.

The status message is added to the response using a Service Channel Submit (HTTP) output connector.

Service Channel Submit (HTTP) output connector settings	
Specify: • *The interface between the "Add Document" channel and the Service Broker.* • *Which service and version (response) to add the document to.* • *The client connector that will receive the response.*	
Service Broker Host	134.123.6.87
Port	1717
Service Description	process_pdf
Version	2
Command	AddDocument
Response Service Channel	service_response

Web services

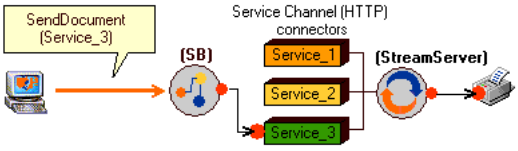
StreamServer services that are registered in a Service Broker can be exposed via web services through the SOAP protocol. A client that needs to invoke these services can import the Service Broker WSDL description and use this information to create the necessary infrastructure.

The exposed web services are not the same as the services published in the Service Broker. The Service Broker acts as a SOAP gateway that exposes two web services: `SendDocument` and `SendReceiveDocument`. Both web services accept the following parameters:

- `ServiceDescription` – The name of the published service.
- `ServiceVersion` – The version of the published service.
- `ServerID` – The server ID of a specific StreamServer.
- `DocData` – The data to send to the service.

Example 2 *Client application calling a SendDocument web service*

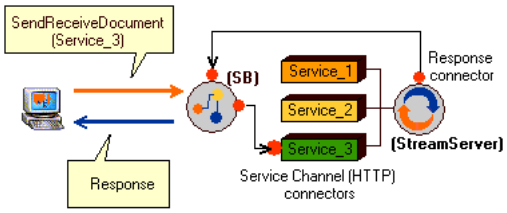
The client makes the SOAP call `SendDocument("Service_3", "", "", soapRequestData)`, and the Service Broker invokes the service `Service_3` in the appropriate StreamServer instance.



The StreamServer processes the `soapRequestData`, and sends the output to a printer.

Example 3 *Client application calling a SendReceiveDocument web service*

The client makes the SOAP call `SendReceiveDocument("Service_3", "", "", soapRequestResponseData)`, and the Service Broker invokes the service `Service_3` in the appropriate StreamServer instance.



The StreamServer processes the `soapRequestResponseData`, and sends the output in the response to the client.

Calling web services

It is possible to call Service Broker web services from applications written in languages that have support for sockets, HTTP, and SOAP (e.g. Java, C#, and VB.NET). Details on how to create and configure this type of applications is beyond the scope of this document, since it can be done in many different ways. To understand how to use the information described here, you must be experienced in the area of web services and WSDL.

Importing the Service Broker WSDL definition

The WSDL description of the web services exposed by the Service Broker (`SendDocument` and `SendReceiveDocument`) is located in:

`<StreamServe installation>\Applications\StreamServer\<version>\Server\servicebroker.wsdl`

You must use the appropriate tools to convert this file to the proxy code needed to make a SOAP call. For example in a .NET environment you just have to import `servicebroker.wsdl` to the Visual Studio development environment.

Creating the function calls

When the proxy code is generated, you must write the function calls that you need to pass all necessary parameters to the web service.

Methods

The following methods are available:

- `SendDocument(ServiceDescription, ServiceVersion, ServerID, DocData)`
- `SendReceiveDocument(ServiceDescription, ServiceVersion, ServerID, DocData)`

Parameter	Description
`ServiceDescription`	A string specifying the name of the published service to invoke.
`ServiceVersion`	A string specifying the version of the published service. If omitted, the Service Broker invokes the highest available version of the service.
`ServerID`	A string specifying the service name of a specific StreamServer. Specify this only if you want to send the data to a specific StreamServer.

Parameter	Description
`DocData`	The document to send to the service. In the WSDL description, this is declared as `soap-enc:base64` encoded data. The document can be anything ranging from plain ASCII to PDF. It must therefore be base64 encoded. In a VB.NET environment, the actual base64 conversion is done by the generated VB.NET web service proxy code, and is therefore transparent to the calling application.

Example 4 Simple VB.NET client call – SendDocument

Use `SendDocument` to send the text "Test data" to the service `Service_3`, version 1.

```
Dim soapSendDocument As SOAPReference.SendDocumentSOAP
Dim soapRequestData As Byte()
Dim unicodeEncoder As System.Text.UnicodeEncoding

'Create the SOAP method wrapper.
soapSendDocument = New SOAPReference.SendDocumentSOAP
soapSendDocument.Url = "http://wghot06:1717"

'Convert input data from a unicode encoded string to a byte array.
unicodeEncoder = New System.Text.UnicodeEncoding
soapRequestData = unicodeEncoder.GetBytes("Test data")
unicodeEncoder = Nothing

'Make the SOAP call.
soapSendDocument.SendDocument("Service_3","1","",
soapRequestData)
soapSendDocument = Nothing
```

Example 5 Simple VB.NET client call – SendReceiveDocument

Use `SendReceiveDocument` to send the text "Test data" to the service `Service_3`, version 1, and to receive a response from the service.

```
Dim soapSendReceiveDocument As
ServiceBroker.SendReceiveDocumentSOAP
Dim soapRequestResponseData As Byte()
Dim decoder As System.Text.ASCIIEncoding
Dim unicodeEncoder As System.Text.UnicodeEncoding
Dim soapResponseString As String

'Call SendRevceiveDocument method.
soapSendReceiveDocument = New
ServiceBroker.SendReceiveDocumentSOAP
soapSendReceiveDocument.Url = "http://wghot06:1717"
unicodeEncoder = New System.Text.UnicodeEncoding
soapRequestResponseData = unicodeEncoder.GetBytes("Test data")
unicodeEncoder = Nothing
soapSendReceiveDocument.SendReceiveDocument("TestService", "1",
"", soapRequestResponseData)

'Convert the received ascii byte array to a unicode string.
'This depends on the returned data. Assume that return data is a
Unicode string.
decoder = New System.Text.ASCIIEncoding
soapResponseString = decoder.GetChars(soapRequestResponseData)
decoder = Nothing
```

Client call scenario

When the Service Broker receives a call, it will parse the SOAP package to make sure that it conforms to the SOAP standard. Then it starts to extract all known parameters for a given SOAP action (`SendDocument` or `SendReceiveDocument`). If any parameter is missing or unrecognized, the Service Broker returns a SOAP error to the client. If all parameters are present and valid, the Service Broker uses an internal protocol mapping to convert the SOAP call into an HTTP call. It then forwards the HTTP call to a service that matches the `ServiceDescription` and `ServiceVersion` parameters in the SOAP call.

When the StreamServer has processed the data, and returned a response to the Service Broker (`SendReceiveDocument` only), the Service Broker creates a SOAP response package, and returns the processed data to the calling client.

Encryption and authentication

A Service Broker scenario includes the following components:

- One or more Service Broker clients.
- One or more Service Brokers.
- One or more service providing StreamServers.

These components can be configured as SSL clients and SSL servers that use encryption and authentication for communication over HTTPS channels. This section describes the interfaces involved, the different types of security configurations that can be defined in a Service Broker scenario, and how to configure the security settings for the different components.

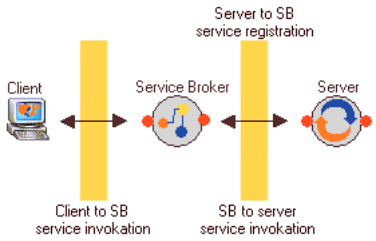
How to create security configurations, and how to handle SSL for the StreamServer in general, is described in the *Encryption and authentication* documentation.

Communication interfaces

There are three different communication interfaces involved when dealing with service registration and invocation via a Service Broker:

- Service registration interface (server to Service Broker).
- Service invocation interface (Service Broker to server).
- Service invocation interface (client to Service Broker).

Each interface can be set up as either secure (via HTTPS) or insecure (via HTTP).



Service registration – server to Service Broker

A StreamServer can publish services as secure or insecure. When a StreamServer publishes a service as secure, it also informs the Service Broker that this service must be invoked as secure. This means that it is not possible to publish a service as secure and then let the Service Broker invoke the service as insecure, or vice versa.

The same security status for all services

A StreamServer must publish all its services as either secure or insecure.

Multiple registrations of the same service

If several StreamServers publish the same service in a Service Broker, they must all publish it with the same security status. For example, if CS1 and CS2 host the same service, both must publish the service with the same security status. If CS1 published the service as secure, and CS2 tries to publish the same service as insecure, the service registration for CS2 will fail.

Service invocation – client to Service Broker

The client can be a StreamServer, or any type of client that can use the Service Broker invocation protocol.

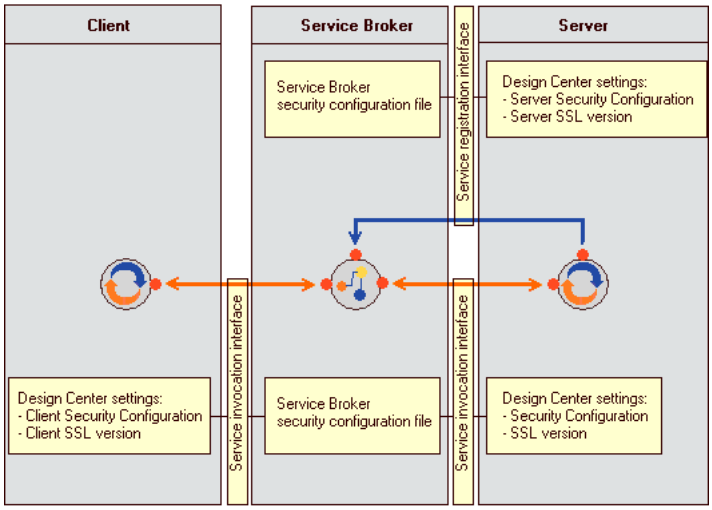
The Service Broker has one HTTP listener for insecure communication, and one HTTPS listener for secure communication. Both listeners can be run at the same time, enabling both secure and insecure communication. The configuration parameters for the listeners – port, time-outs, etc. – are specified at different locations. The parameters for the HTTP listener are the Service Broker properties specified in the Control Center (see *Service Broker settings in the Control Center* on page 37). The parameters for the HTTPS listener are specified in a security configuration file. The path to this file is specified in the Control Center GUI.

Service invocation – Service Broker to server

When a client sends a request to the Service Broker, the Service Broker tries to find a matching service. If a matching service is found, the Service Broker checks the security status of the service. If the security status is secure, the Service Broker invokes the service via HTTPS, and if the security status is insecure, the Service Broker invokes the service via HTTP.

Security configurations

You must specify different security configurations for the components involved when you are dealing with secure service registration and invocation via a Service Broker.



Service Broker

You can specify security configurations for the following functions:

- Service registration from the service providing server, where the Service Broker is running as the SSL server.
- Service invocations from the client, where the Service Broker is running as the SSL server.
- Service invocations towards the service providing server, where the Service Broker is running as the SSL client.

See *Configuring SSL for a Service Broker* on page 30.

Service providing StreamServer

You can specify security configurations for the following functions:

- Service registration towards the Service Broker, where the StreamServer is running as the SSL client.
- Service invocations from the Service Broker, where the StreamServer is running as the SSL server.

See *Configuring SSL for service providing StreamServers* on page 32.

StreamServer running as Service Broker client

You can specify a security configuration for service invocation towards the Service Broker, where the StreamServer is running as the SSL client.

See *Configuring SSL for StreamServers running as Service Broker clients* on page 33.

Configuring SSL for a Service Broker

The Service Broker reads all security parameters from a security configuration file (`*.ssc`). You will find a security configuration file template in

`<installation_directory>\Applications\StreamServer\<version>\Common\Templates\keywords.ssc`

All keywords and values are described in

`<installation_directory>\Applications\StreamServer\<version>\Common\Templates\template.ssc`

To configure SSL parameters using the Control Center

1. Create the security configuration file. See *The Service Broker security configuration file* below.
2. In the Control Center, select the **Service Broker** service (4.x Services node).
3. In the Properties view, right-click the **Security configuration file** property and select **Edit property**. The Edit Value dialog box opens.
4. In the **Value** field, specify the path to the security configuration file and click **OK**.

To configure SSL parameters using command line arguments

1. Create the security configuration file. See *The Service Broker security configuration file* below.
2. Use the following command line arguments:

`-securityconfiguration <path>`	Path to the service configuration file. This corresponds to the property `Security configuration file` in the Control Center.
`-workdir <path>`	Path to the Service Broker working directory (WIN32). This corresponds to the property `Working directory` in the Control Center.
`-wd <path>`	Path to the Service Broker working directory (UNIX). This corresponds to the property `Working directory` in the Control Center.

The Service Broker security configuration file

In the Service Broker security configuration file you specify both the server side (client to Service Broker and service registration) and the client side (Service Broker to server) security configurations. You can specify several security configuration sections in this file:

`FILE`	Describes all necessary information needed to set up an SSL client or SSL server. All the certificate and private key data is stored in files.
`XKMS`	Describes all necessary information needed to set up an SSL client or SSL server. All the certificate information is retrieved from an XKMS server (Trust Server). Only the root CA that verifies the identity of the XKMS server is stored in a file.
`SERVER_PORT_BINDING`	The server port binding that connects the data communication parameters (port number, threads, etc.) to a `FILE` or `XKMS` configuration. All server port bindings used by the Service Broker must have the name `SERVICE_BROKER_SSL_SERVER`
`CLIENT_PORT_BINDING`	The client port binding that connects the data communication parameters (port number, threads, etc.) to a `FILE` or `XKMS` configuration. All client port bindings used by the Service Broker must have the name `SERVICE_BROKER_SSL_CLIENT`

Trouble shooting

Subject name and host name

The certificate subject name must be the same as the host name (case sensitive). For example, if the Service Broker runs on the host `xyz`, using a server certificate with subject name `XYZ`, all clients trying to establish an SSL connection using `xyz` as the host name will fail.

Domain name and IP address

The Service Broker cannot contact a StreamServer using domain name or IP address. It can only contact StreamServers using local machine names.

XKMS server specific

- XKMS server is not running.
- The client's trusted XKMS CA is not configured.
- The certificate used by the XKMS server is not issued by a trusted CA.
- The host name used to access the XKMS server does not match subject name (case sensitive) in the received certificate.
- The user name or password is not correct.

- The certificate ID is not registered in the XKMS server.
- The private key ID is not registered in the XKMS server.
- The pass phrase does not match the private key ID.

Parsing the configuration file

There are a number of error scenarios when parsing the configuration file. All of these are related to missing or incorrect information. Parse errors are also logged in the Service Broker log with a description of the problem, and a line number where the problem is located. The location information can be used as a rough pin pointer of the problem, because it only counts handled configuration lines (empty lines and comments are not counted). If the error is a missing or incorrect parameter this will not be detected until the next `END_SECURITY_CONFIGURATION` is found. In these scenarios the row counter will not point to the failing parameter but to the `END_SECURITY_CONFIGURATION` line. The error text can give you more information about the real problem.

Binding names

The Service Broker identifies its SSL server and SSL client through two predefined names. The SSL server port binding must be named `SERVICE_BROKER_SSL_SERVER`, and the SSL client port binding must be named `SERVICE_BROKER_SSL_CLIENT`. See *The Service Broker security configuration file* above.

Configuring SSL for service providing StreamServers

You must specify security configurations for the following functions:

- Service registration towards the Service Broker, where the StreamServer is running as the SSL client.
- Service invocations from the Service Broker, where the StreamServer is running as the SSL server.

Both these functions must have the same security status – if you enable SSL for one of them, SSL is automatically enabled for the other.

Prerequisites

You have created security configurations for both the registration and invocation channels. See the *Encryption and authentication* documentation.

To configure the SSL parameters

1. In the Design Center, activate the physical Platform layer you want to configure.
2. Right-click the Platform view and select **Configure Platform**. The Configure Platform dialog opens.
3. Select the **Service Broker** tab and configure the following:

- **Use SSL for publish/unpublish** – Select to use SSL when publishing/unpublishing services in the Service Broker.
- **Server Security Configuration** – The security configuration to use when publishing/unpublishing services.
- **Server SSL version** – The SSL version to use when publishing/unpublishing services.
- **Security Configuration** – The security configuration to use when the StreamServer is running as an SSL server.
- **SSL version** – The security configuration to use when the StreamServer is running as an SSL server.

See *Service Broker settings in the Design Center* on page 35.

Configuring SSL for StreamServers running as Service Broker clients

You must specify a security configuration for service invocation towards the Service Broker, where the StreamServer is running as the SSL client.

Note that it is possible to configure the client's input side insecure, and at the same time configure the service invocation channel secure, and vice versa.

Prerequisites

You have created a security configuration for the invocation channel. See the *Encryption and authentication* documentation.

To configure the SSL parameters

1. In the Design Center, activate the physical Platform layer you want to configure.
2. Right-click the Platform view and select **Configure Platform**. The Configure Platform dialog opens.
3. Select the **Service Broker** tab and configure the following:
   - **Use SSL for output Service Channels** – Select to use SSL for communication between Service Channel Submit (HTTP) output connectors and the Service Broker.
   - **Client Security Configuration** – The security configuration to use when the StreamServer is running as an SSL client to the Service Broker.
   - **Client SSL version** – The SSL version to use when the StreamServer is running as an SSL client to the Service Broker.

See *Service Broker settings in the Design Center* on page 35.

Service Broker GUI reference

Service Broker settings in the Design Center

Service Broker host

The host where the Service Broker is running.

Service Broker port

The port number used by the Service Broker. Must be the same as `IPport` in the Control Center configuration of the Service Broker.

Use SSL for output Service Channels

Select to use SSL for communication between Service Channel Submit (HTTP) output connectors and the Service Broker. This is used when the StreamServer is running as an SSL client requesting services from the Service Broker.

Client Security Configuration

The security configuration to use when the StreamServer is running as an SSL client to the Service Broker. The security configuration must be included in a resource set connected to the Platform. See the *Encryption and authentication* documentation for more information.

Client SSL version

The SSL version to use when the StreamServer is running as an SSL client to the Service Broker.

Enable services

Select to enable publishing of services in the Service Broker.

Port

The port to use to publish services, and to receive input via the Service Broker.

Threads

The maximum number of concurrent connections to the services available from the StreamServer. When all connections are busy, and a new client tries to connect to a service, the connection will fail.

Idle time-out (ms)

This time-out (milliseconds) is applied when the StreamServer has finished processing a request, and no more data related to the request is received or sent. It specifies the maximum time the connection will remain open, and enables the client to send a new request without having to set up a new connection.

Connection time-out (ms)

This time-out (milliseconds) is applied when the StreamServer sends or receives data, and the purpose is to close dead connections. If no data is sent or received during the time specified, the connection is closed.

Response time-out (ms)

This time-out (milliseconds) is applied when the StreamServer has received all data from the client. It specifies the maximum time the client is expected to wait for a response. Set this time-out to `0` if the client does not expect any response.

Use SSL for publish/unpublish

Select to use SSL when publishing/unpublishing services in the Service Broker. This is used when the StreamServer is running as an SSL client, publishing services in the Service Broker.

Server Security Configuration

The security configuration to use when publishing/unpublishing services. The security configuration must be included in a resource set connected to the Platform. See the *Encryption and authentication* documentation for more information.

Server SSL version

The SSL version to use when publishing/unpublishing services.

Use SSL for input Service Channels

Select to use SSL for communication between Service Channel (HTTP) input connectors and the Service Broker. This is used when the StreamServer is running as an SSL server invoked by the Service Broker.

Security Configuration

The security configuration to use when the StreamServer is running as an SSL server. The security configuration must be included in a resource set connected to the Platform. See the Encryption and authentication documentation for more information.

SSL version

The SSL version to use when the StreamServer is running as an SSL server.

Service Broker settings in the Control Center

Executable path

The path to the Service Broker executable. You cannot change this.

Security configuration file

The path to the security configuration file. See *The Service Broker security configuration file* on page 31.

Working directory

The working directory for the Service Broker. All relative paths are relative to this working directory.

Communication time-out

The maximum time (milliseconds) the Service Broker will wait for input during communication.

Connection time-out

The maximum time (milliseconds) the Service Broker will wait to connect to a StreamServer.

Number of threads

The maximum number of concurrent connections to the Service Broker.

IPport

The Service Broker port.

Temporary directory

The path to the directory for temporary files.

Log file

The path to the log file.

Log message file

The path to the log message file.

Log level

A value between 0 and 4. The higher the value, the more information will be displayed in the log.

Language ID

The language for the log messages.