

## B.3 Messaging Infrastructure

**Role(s):** Service Customer, Service Provider

**Component(s):** Gateway (GW)  
Service Instance Registry (SIR)  
Security Token Service (STS)

**License:** MS-PL

The gateway toolkit supports dynamic and extensible message policy enforcement – this ranges from routing (virtualisation & encapsulation) over authorisation and access restriction (in combination with e.g. the STS) to message transformation (interoperability). This toolkit enhances the messaging channels of (web service) based transactions, allowing to enforce any amount of policies upon in- and outgoing messages.

### B.3.1 Installation

#### B.3.1.1. Installation Requirements

- Windows Server 2003 / XP or higher
- .NET Framework 3.5 or higher
- Geneva Identity Framework CTP2

#### B.3.1.2. Deployment Tips

Through using “double-blind virtualisation” mechanisms, i.e. by deploying the gateway on both consumer *and* service provider side, it is possible to alter resources *and* providers without affecting the calling applications. Figure 62 presents a sample deployment of the Gateway infrastructure. The dashed line denotes an indirect link (as the gateway extends the message channel).

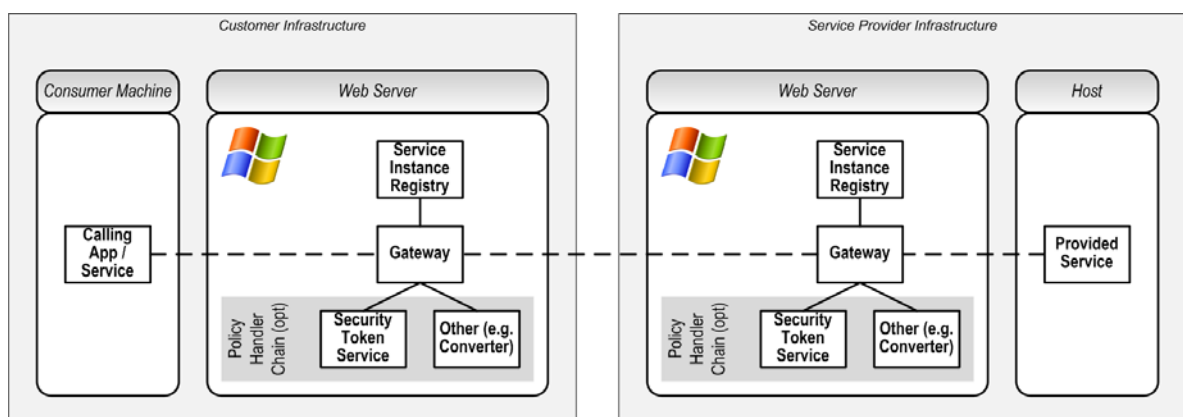


Figure 62: Sample deployment of the Gateway infrastructure

- The Gateway is always necessary.
- The Service Instance Registry is highly recommended, but not required
- Policy Handlers (including the Service Instance Registry) can be deployed on separate machines – however, since the communication between Gateway and

Policy Handler are not secured, this is only recommended within protected environments

## B.3.2 Software Installation

Unload and Unzip the “Messaging Infrastructure Installers.rar” file into the server where you will install the software. Execute BREIN Messaging Infrastructure Setup.exe.

The installer will install the following components:

- Gateway
- SIR
- STS

### Gateway

#### Prerequisites

The Gateway relies on the Microsoft .NET Framework 3.5. Necessary components for a successful operation of the Gateway are the SIR and the STS.

#### Installation

The Gateway can be adapted via the corresponding app.config file. This file can be found in the corresponding installation directory. The entries to be adapted are introduced with the comment "`<!-- DEPLOYMENT`" followed by a description what kind of information has to be provided.

```
<!-- Gateway -->
<configuration>
  ...
  <microsoft.identityModel>
    <service>
      <issuerNameRegistry ...>
        <trustedCertificates>
          <!-- DEPLOYMENT: Add trusted issuers here. -->
          <add name="CN=ca.breincompany1.com"
findValue="CN=ca.breincompany1.com" storeLocation="LocalMachine"
storeName="Root" x509FindType="FindBySubjectDistinguishedName" />
          <!-- DEPLOYMENT: Add STS certificate location here -->
          <add name="CN=sts.breincompany1.com"
findValue="CN=sts.breincompany1.com" storeLocation="LocalMachine"
storeName="My" x509FindType="FindBySubjectDistinguishedName" />
        </trustedCertificates>
      </issuerNameRegistry>
    </service>
  </microsoft.identityModel>
  ...
  <system.serviceModel>
    ...
    <bindings>
      <wsFederationHttpBinding>
        <binding name="OwnSTS">
          <security mode="Message">
            <message>
              <issuer
address="http://breinbase.hlrs.de/BREIN/C1/STS/WSTrust.svc/wsHttpBindin
g" binding="wsHttpBinding" bindingConfiguration="HomeRealmSTS">
              <!-- DEPLOYMENT: Add an identity section if the service
does not use his real hostname in the certificate. -->
```

```

        <identity> <dns value="sts.breincompany1.com" />
</identity>
    </issuer>
</message>
</security>
</binding>
<binding name="externalEndpointBindingConfiguration">
    <security mode="Message">
        <message>
            <issuerMetadata
address="http://breinbase.hlrs.de/BREIN/C1/STS/mex">
                <!-- DEPLOYMENT: Add an identity section if the service
does not use his real hostname in the certificate. -->
                <identity> <dns value="sts.breincompany1.com" />
                </identity>
            </issuerMetadata>
        </message>
    </security>
</binding>
</wsFederationHttpBinding>
    ...
</configuration>

```

## Service Instance Registry (SIR)

### Prerequisites

The SIR relies on the Microsoft .NET Framework 3.5. There are no other components needed to run the SIR.

### Installation

The SIR does not need any adaptations of the corresponding app.config file.

## Security Token Service (STS)

### Prerequisites

The STS relies on the Microsoft .NET Framework 3.5. There are no other components needed to run the STS.

### Installation

The STS can be adapted via the corresponding app.config file. This file can be found in the corresponding installation directory. The entries to be adapted are introduced with the comment "`<!-- DEPLOYMENT`" followed by a description what kind of information has to be provided.

```

<!-- STS -->
<configuration>
    ...
    <microsoft.identityModel>
        <service>
            <issuerNameRegistry type=...>
                <trustedCertificates>
                    <!-- DEPLOYMENT: Add trusted issuers here. -->
                    <add name="CN=ca.breincompany1.com"
findValue="CN=ca.breincompany1.com" storeLocation="LocalMachine"
storeName="Root" x509FindType="FindBySubjectDistinguishedName" />
                </trustedCertificates>
            </issuerNameRegistry>
            <securityTokenHandlers name="OnBehalfOf"/>
        </service>
    </microsoft.identityModel>

```

### B.3.3 Usage Instructions

An essential functionality of the gateway consists in message routing – actually it depends on the usage scenario whether this functionality is considered *required*. As we foresee the main use for the gateway consisting in virtualisation & encapsulation of resources / services, routing is required as an integral part of this capability, the Service Instance Registry maintains routing endpoints and their exposed virtual address, so that any message going over the gateway will be redirected from the virtual address to its accordingly stored endpoint in the SIR. This allows the calling application to maintain a static endpoint which is resolved in the SIR using information provided by an administrator or by exploiting additional context information contained in the message header.

**Note:** The Gateway *can* be used without the SIR, in which case the user will have to write their own policy handlers.

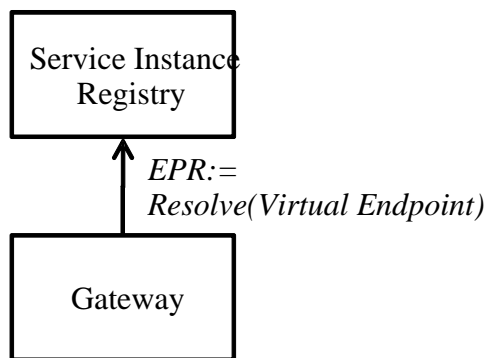


Figure 63: Functional relationship between Gateway and Service Instance Registry

The gateway can be extended by a security policy handler – in particular the Security Token Service (STS) by EMIC. The STS is a specific policy handler to ensure secure and authenticated message exchange between (trusted) parties. To do this, the STS requires that dedicated collaboration tokens have been created and registered. Therefore, the security token service is responsible for three topics.

- **Token issuance:** First it issues tokens that can be used to protect outgoing messages, to authenticate the sender and to convey authorization information about the sender to the receiver's side.
- **Policy store:** Second, the STS is the place where the security policy resides. The security policy is kind of the 'rule set' for token issuing and validation of tokens, and for access control decisions. When issuing a token, the STS checks embedded claims about the requestor in the token. Token issuance can be parameterized by the requesting application, e.g. by giving an identifier of the sending application, or by requesting particular types of claims.
- **Token validation and access control decision:** Third, the STS checks incoming security tokens and performs access control. Thus, it acts as a policy decision point (PDP). The STS of the receiving side will not accept any token, but only tokens that are expected. So this again makes the STS to the place where the policy resides, this time on the receiver's side. The configuration of the receiving STS determines who may access the service protected by this particular STS and what criteria have to be fulfilled to grant access to the service. The criteria are expressed by signed tokens and the configuration of the STS what sender-side STS may issue what kind of token.

Any invocation of a consuming application via the gateway will be enhanced with the credentials of the sender according to the collaboration details by resolving the destination EPR. Incoming messages on the provider side will be validated against the known collaboration tokens and its response will again be enhanced with the according details from his / her side.

**Note:** The STS requires the Gateway and the SIR to function properly.

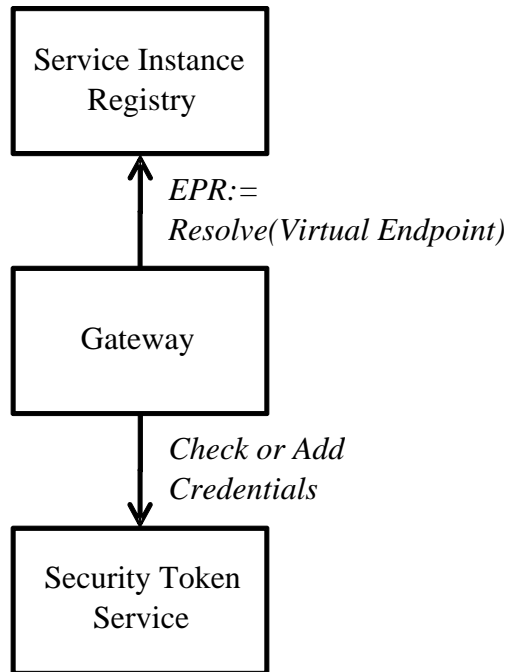


Figure 64: Functional relationship between Gateway, Service Instance Registry and STS

The typical deployment will have at least the Gateway and the Service Instance Registry hosted on a web server which forms the entrance to the resource infrastructure. In order to be able to receive messages, it requires an open port in the firewall on the web server – however, the firewall of the individual service hosts need only be opened for communication with this server, not with the external internet.

Any policy handler extending the gateway could be located on any machine that the web server can access – potentially even located in the internet. Since messaging between gateway and policy handler is not secured, this is only recommended for non-critical requests though.

### Gateway

The Gateway acts as a transparent message interceptor and thus does not provide a “viewable” interface. In order to configure the gateway, the corresponding handler has to be invoked. In particular in that case this refers to the SIR.

### Service Instance Registry (SIR)

The SIR allows for the mapping of virtual endpoint information to concrete service endpoints, Table 20.

**Table 20 - SIR interface**

|        |   |
|--------|---|
| Method | <code>void RegisterEndpoint(EndpointAddress10 virtualName,</code> |
|--------|---|

|                        |  |
|------------------------|--|
|                        | <code>EndpointInformation endpointInformation)</code>  |
| <b>Arguments</b>       | <code>EndpointAddress10 virtualName</code> – The virtual name of a service<br><code>EndpointInformation endpointInformation</code> – A Struct providing the necessary information in order to map a virtual endpoint to a concrete one |
| <b>Return Value[s]</b> |  |
| <b>Description</b>     | Registers a virtual endpoint with the corresponding mapping information to the SIR.  |
| <b>Method</b>          | <code>EndpointInformation GetEndpoint(EndpointAddress10 virtualName)</code>  |
| <b>Arguments</b>       | <code>EndpointAddress virtualName</code> – The virtual name to be resolved.  |
| <b>Return Value[s]</b> | <code>EndpointInformation virtualName</code> – The corresponding mapping information.  |
| <b>Description</b>     | This method allows the mapping of a virtual name to concrete endpoint information. This method is invoked by the gateway in order to resolve the corresponding virtual endpoint.   |
| <b>Method</b>          | <code>void ModifyEndpoint(string virtualName, string newEndpoint);</code>  |
| <b>Arguments</b>       | <code>String virtualName</code> – The virtual name of a service<br><code>String newEndpoint</code> – The URI of the new service endpoint   |
| <b>Return Value[s]</b> |  |
| <b>Description</b>     | This method allows the modification of already registered virtualName – concreteEndpoints pairs.   |