Default Application Router

Table of Contents

The Application Router is called by the container to select a SIP Servlet application to service an initial request. It embodies the logic used to choose which applications to invoke.

Role of the Application Router

An Application Router is required for a container to function, but it is a separate logical entity from the container. The Application Router is responsible for application selection and does not implement application logic. For example, the Application Router cannot modify a request or send a response.

There is no direct interaction between the Application Router and applications, only between the SIP Servlets Container and the Application Router.

The SIP Servlets container is responsible for passing the required information to the Application Router within the initial request so the Application Router can make informed routing decisions. The Application Router is free to make use of any information or data stores, except for the information passed by the container. It is up to the individual implementation how the Application Router makes use of the information or data stores.

The deployer in a SIP Servlet environment controls application composition by defining and deploying the Application Router implementation. Giving the deployer control over application composition is desirable because the deployer is solely responsible for the services available to subscribers.

Furthermore, the SIP Servlets specification intentionally allows the Application Router implementation to consult arbitrary information or data stores. This is because the deployer maintains subscriber information and this information is often private and valuable.

Restcomm Default Application Router

Restcomm SIP Servlets provides an implementation of the Default Application Router (DAR) as defined in the SIP Servlets 1.1 specification, Appendix C.

The DAR Configuration File

The Default Application Router (DAR) obtains its operational parameters from a configuration text file that is modeled as a Java properties file. The configuration file contains the information needed by the Application Router to select which SIP Servlet application will handle an incoming initial request.

In the case of Restcomm SIP Servlets, it is also possible to configure the DAR through the *server.xml* configuration file (see [_bsssc_configuring_the_service_element_in_the_containers_server.xml] and [_wwtssmc_working_with_the_sip_servlets_management_console]).

The properties file has the following characteristics and requirements:

- It must be made available to the DAR.
- It must allow the contents and file structure to be accessible from a hierarchical URI supplied as a system property *javax.servlet.sip.ar.dar.configuration*.
- It is first read by the container when it loads up and is refreshed each time an application is deployed and undeployed.
- It has a simple format in which the name of the property is the SIP method and the value is a comma-separated string value for the `SipApplicationRouterInfo` object.

```
INVITE: (sip-router-info-1), (sip-router-info-2)..
SUBSCRIBE: (sip-router-info-3), (sip-router-info-4)..
ALL: (sip-router-info-5), (sip-router-info-6)..
```

Restcomm SIP Servlets defines a new keyword called `ALL`. The keyword allows mapping between the sip-router-info data, and all methods supported by the container (for example, INVITE, REGISTER, SUBSCRIBE). This mapping can save time when configuring an application that listens to all incoming methods.

If `ALL`, and a specific method are defined in the DAR file, the specific method takes precedence over `ALL`. When the specific method no longer has applications to serve, `ALL` is enabled again.

The sip-router-info data specified in the properties file is a string value version of the `SipApplicationRouterInfo` object. It consists of the following information:

- The name of the application as known to the container. The application name can be obtained from the element of the *sip.xml* deployment descriptor of the application, or the `@SipApplication` annotation.
- The identity of the subscriber that the DAR returns. The DAR can return any header in the SIP request using the DAR directive `DAR:SIP_HEADER`. For example, `DAR:From` would return the SIP URI in the `From` header. The DAR can alternatively return any string from the SIP request.
- The routing region, which consists of one of the following strings: `ORIGINATING`, `TERMINATING` or `NEUTRAL`. This information is not currently used by the DAR to make routing decisions.
- A SIP URI indicating the route as returned by the Application Router, which can be used to route the request externally. The value may be an empty string.
- A route modifier, which consists of one of the following strings: `ROUTE`, `ROUTE_BACK` or `NO_ROUTE`. The route modifier is used in conjunction with the route information to route a request externally.
- A string representing the order in which applications must be invoked (starts at 0). The string is removed later on in the routing process, and substituted with the order positions of sip-router-info data.
- An optional string that contains Restcomm -specific parameters. Currently, only the `DIRECTION` and `REGEX` parameters are supported.

The field can contain unsupported `key=value` properties that may be supported in future releases. The unsupported properties will be ignored during parsing, until support for the attributes is provided.

The syntax is demonstrated in [_example_dar_direction_example].

- The `DIRECTION` parameter specifies whether an application serves external(INBOUND) requests or initiates (OUTBOUND) requests.

  If an application is marked `DIRECTION=INBOUND`, it will not be called for requests initiated by applications behaving as UAC. To mark an application as UAC, specify `DIRECTION=INBOUND` in the optional parameters in the DAR.

  Applications that do not exist in the DAR list for the container are assumed to be `OUTBOUND`. Because undefined applications are incapable of serving external requests, they must have self-initiated the request. The Sip Servlets Management Console can be used to specify the `DIRECTION` parameter.

  If an application is marked `DIRECTION=UAC_ROUTE_BACK`, the Application that acted as UAC will be called back if it present in the list of applications to be called for that particular message SIP method.

- The `REGEX` parameter specifies a regular expression to be matched against the initial request passed to the Application Router.

  If the regular expression matches a part of the initial request, the application is called. If it does not, it is skipped.

  For example, in the following sip-router-info data:

```
INVITE - ("org.mobicents.servlet.sip.testsuite.SimpleApplication", "DAR:From",
"ORIGINATING", "", "NO_ROUTE", "0", "REGEX=From:.*sip:.*@sip-servlets\.com")
```

only incoming initial requests with a From Header with a SIP URI that belongs to the sip-servlets.com domain will be passed to the SimpleApplication.

Example 1. DIRECTION Example

In this example, two applications are declared for the `INVITE` request. The `LocationServiceApplication` is called for requests coming from outside the container, but it will not be called for the requests initiated by the UAC application `Click2DialApplication`.

```
INVITE: ("org.mobicents.servlet.sip.testsuite.Click2DialApplication", "DAR:From",
"ORIGINATING", "", "NO_ROUTE", "0", "DIRECTION=OUTBOUND"), \
("org.mobicents.servlet.sip.testsuite.LocationServiceApplication", "DAR\:From",
"ORIGINATING", "", "NO_ROUTE", "0", "DIRECTION=INBOUND")
```

This type of configuration is useful in cases where different application must be responsible for both requests initiated by the container, and external requests received by the container.

Example 2. ORIGINATING/TERMINATING DAR Example

In this example, the DAR is configured to invoke two applications on receipt of an INVITE request; one each in the originating and the terminating halves. The applications are identified by their application deployment descriptor names.

```
INVITE: ("OriginatingCallWaiting", "DAR:From", "ORIGINATING", "", "NO_ROUTE",
"0"), ("CallForwarding", "DAR:To", "TERMINATING", "","NO_ROUTE", "1")
```

For this example, the returned subscriber identity is the URI from each application's `From` and `To` headers respectively. The DAR does not return any route to the container, and maintains the invocation state in the `stateInfo` as the index of the last application in the list.

Routing of SIP Messages to Applications

Initial Requests and Application Selection Process

Initial Requests are those that can essentially be dialog creating (such as, `INVITE`, `SUBSCRIBE` and `NOTIFY`), and not part of an already existing dialog.

Initial requests are routed to applications deployed in the container according to the SIP Servlets 1.1 specification, Section 15.4.1 Procedure for Routing an Initial Request.

There are some other corner cases that apply to initial requests. Refer to Appendix B, Definition of an Initial Request in the SIP Servlets 1.1 specification.

Example 3. INVITE Routing

The following example describes how the DAR routes an INVITE to two applications deployed in a container. The applications in this example are a Location Service and a Call Blocking application.

In the example, the assumption of a request coming to the server is described. However, applications can act as a UAC, and generate initial requests on their own. For routing purposes, it is not necessary for the specified application initiating the request to have an entry in the DAR file.

The DAR file contains the required information for the two applications to be invoked in the correct order.

```
INVITE: ("LocationService", "DAR:From", "ORIGINATING", "", "NO_ROUTE", "0"),
("CallBlocking", "DAR:To", "TERMINATING", "","NO_ROUTE", "1")
```

Processing occurs in the following order:

1. A new `INVITE` (not a re-INVITE) arrives at the container.

   The `INVITE` is a dialog creating request, and is not part of any dialog.

2. The Application Router is called.

   From the `INVITE` information, the first application to invoke is the Location Service.

3. The Application Router returns the application invocation order information to the container (along with the rest of the sip-router-info data) so the container knows which application to invoke.
4. The container invokes the LocationService that proxies the `INVITE`.

   The proxied `INVITE` is considered as a new `INVITE` to the known IP Address of the registered user for the Request URI

   For further information regarding `INVITE` handling, refer to "Section 15.2.2 Sending an Initial Request" in the SIP Servlets 1.1 Specification.

5. Because the `INVITE` has been proxied, the container invokes the Application Router for the proxied `INVITE` to see if any more applications are interested in the event.
6. From the proxied invite, the Application Router determines that the second application to invoke is the Call Blocking application.
7. The Application Router returns information regarding the Call Blocking application to the container (along with the rest of the sip-router-info data) so the container knows which application to invoke.
8. The container routes the `INVITE` for the Call Blocking application to the next application in the chain.
9. The Call Blocking application determines that the user that initiated the call is black listed. The application rejects the call with a "Forbidden" response.
10. Because the Call Blocking application acts as a UAS, the Application Selection Process is stopped for the original `INVITE`.

The path the `INVITE` has taken (that is, LocationService to CallBlocking) is called the

application path. The routing of the responses will now occur as explained in the next section.

Response Routing

Responses always follow the reverse of the path taken by the corresponding request. In our case, the Forbidden response will first go back to the LocationService, and then back to the caller. This is true for responses to both initial and subsequent requests. The application path is a logical concept and as such may or may not be explicitly represented within containers.

Another possible outcome could have been that the Call Blocking application, instead of sending a Forbidden response, allowed the call and proxied the INVITE to the same Request URI chosen by the Location Service. Then when the callee sends back the 200 OK Response, this response goes back the same way through the application path (so in the present case Call Blocking, then Location Service, then back to the caller).

The Call Blocking application cannot just do nothing with the request and expect the container to route the request in its place (either to a next application in chain if another one is present or to the outside world if none is present). The Application has to do something with request (either proxy it or act as a UAS).

Subsequent Requests

Subsequent requests are all requests that are not Initial.

The second scenario, where the Call Blocking application allowed the call, will be used in this section to showcase subsequent requests. The caller has received the 200 OK response back. Now, according to the SIP specification (RFC 3261), it sends an ACK. The ACK arrives at the container, and is not a dialog creating request and is already part of an ongoing dialog (early dialog) so the request is detected as a Subsequent request and will follow the application path created by the initial request. The ACK will go through Location Service, Call Blocking, and finally to the callee.

Limitations of the Default Application Router

The DAR is a minimalist Application Router implementation that is part of the reference implementation. While it could be used instead of a production Application Router, it offers no processing logic except for the declaration of the application order.

In real world deployments, the Application Router plays an extremely important role in application orchestration and composition. It is likely that the Application Router would make use of complex rules and diverse data repositories in future implementations.