




WSO2 API Manager 4.1.0 Fundamentals - Integration Profile

Triggering Messages - Part I



WSO2 Training

CC by 4.0



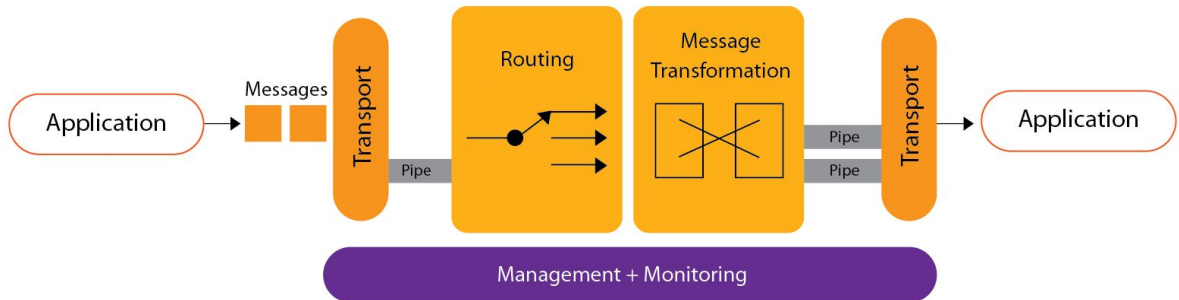
Module Objective

At the end of this module, attendees will be able to:

- Understand how messages are triggered in WSO2 Micro Integrator.
- Understand APIs and Proxy Services.



Messaging Architecture



Triggering Messages

Messages can be injected into sequences through the following triggers:

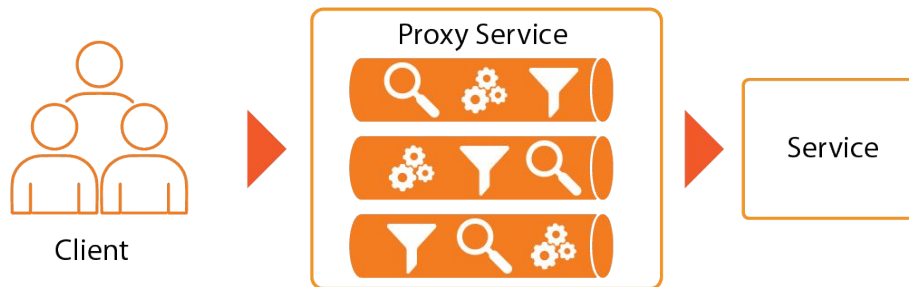


Messages can be injected into sequences through one of the following 4 ways:

- Proxy: The Micro Integrator can act as a virtual service in front of the real back-end service, allowing messages to transform before sending a message to the actual service.
- Main sequence: All messages that are not sent to a proxy service are sent to the main sequence.
- API: Accepts REST messages that allow clients to provide additional information on how to manage the message.
- Scheduled Tasks: Triggers sequences with messages for processing.

Proxy Service

Acts as a virtual service. Receives messages and mediates them before sending them to the endpoint.



Proxy services are virtual services that receive messages and optionally process them before forwarding them to a service at a given endpoint. This approach allows you to perform necessary transformations and to introduce additional functionality without changing your existing service.

Responses returning from services can be further processed and forwarded to clients. Resembles traditional HTTP proxy servers.

For example, if you want a service to handle messages that are in different formats, you can create a Transformer proxy service to transform requests and responses based on specified XSLT configurations.

Any available transport can be used to receive and send messages from the proxy services. Proxy services are made up of sequences.

Main Sequence

Default sequence for all messages that are not dispatched to an API, a proxy service, or an inbound endpoint.

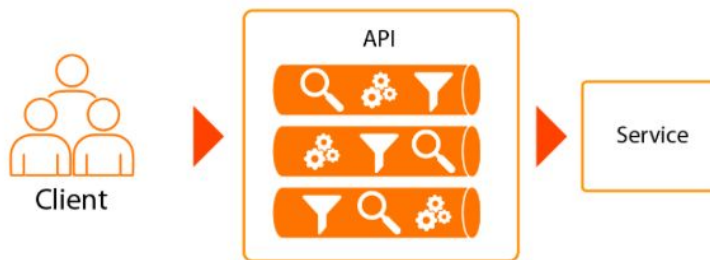


All messages that are not sent to an API, a proxy service, or an inbound endpoint are sent through the main sequence. By default, the main sequence simply sends a message without mediation. To add message mediation, you add mediators and/or named sequences in the main sequence.

APIs

APIs in WSO2 Micro Integrator can accept REST messages that allow clients to provide additional information on how to manage the message:

- Can handle multiple URLs
- Can handle parameters in the URL



Each API is anchored at a user-defined URL context, much like how a web application deployed in a servlet container is anchored at a fixed URL context.

An API will only process requests that fall under its URL context. For example, if a particular API is anchored at the “/test” context, only HTTP requests that have URL paths starting with “/test” will be handled by that API.

It is also possible to bind a given API to a user-defined hostname and/or a port number.

For example, if your API is anchored at the ‘http://your.host.name/test’ URL, the following requests will be handled by your API:

- <http://your.host.name/test/getNumbers>
- <http://your.host.name/test/calculation/getRate>

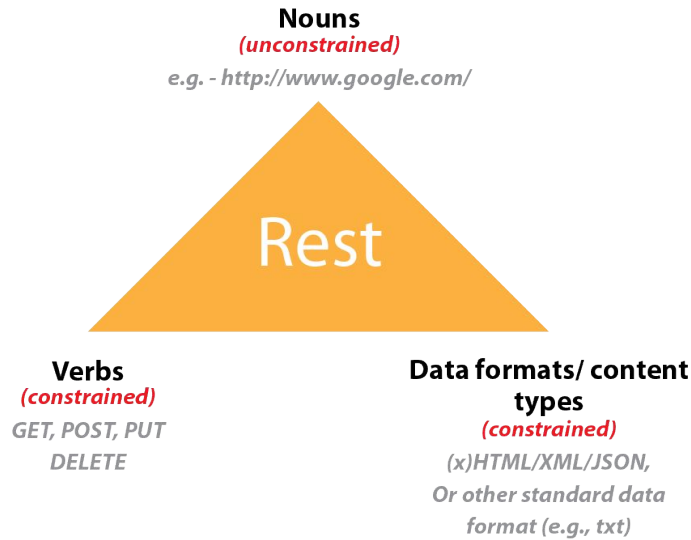


RESTful Integration

- Expose REST interfaces with APIs.
- Expose HTTP services.
- Pinned to a given URL context and can have multiple resources.
- Can be considered as a light-weight, unmanaged API.



RESTful Integration



- Verbs: API Resources and HTTP Endpoints
- Nouns: HTTP Endpoints and URI Templates
- Data formats/types: XML/JSON (Payload Factory mediators, JSON content based routing)

RESTful Integration Support

- JSON Transformations:
 - ◉ PayloadFactory mediator supports all possible combinations of JSON and XML transformations.
- JSON content-based routing:
 - ◉ JSONPath expressions
- High-performance JSON processing:
 - ◉ Staxon library
- No canonicalization (No conversion back and forth JSON->SOAP->JSON).

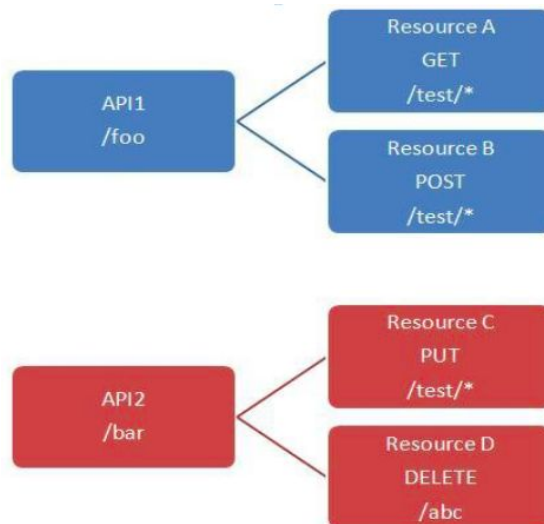


Creating APIs

- RESTful integration with APIs
- API anchored at URL context:
 - ◉ GET /customer/name/{customerName}
 - ◉ GET /customer/id/{customerId}
- Resources:
 - ◉ API component accessed through an HTTP call.
 - ◉ Similar to a proxy service (in, out, and fault sequences).
 - ◉ Restricts resource scope (using URL patterns and URI templates).



Creating APIs



To further understand how request dispatching works with APIs and resources, let's consider the object hierarchy in the diagram. Here we have two APIs anchored at "/foo" and "/bar" contexts respectively. Each API is made of two resources.

A GET request to the /foo/test path will be received by the first API since it is anchored at the /foo context. Then it will be handed over to resource A, which is configured to process the GET request with the /test/* URL pattern. Similarly a POST request to the /foo/test path will be received by resource B in the same API. The PUT requests to the /bar/test path will be dispatched to resource C in the second API, and DELETE requests to the /bar/abc path will be processed by resource D.

The above example demonstrates the usefulness of APIs.

In a nutshell, it provides a simple, yet powerful, approach to breaking down a stream of HTTP calls based on HTTP methods, URL patterns, and various other parameters. Once the HTTP calls have been filtered out and dispatched to appropriate APIs and resources, we can subject them to the routing and mediation capabilities of the Micro Integrator using mediators, sequences, and endpoints.

Creating APIs

- URL mapping
 - ◉ Path mappings (eg: /test/*, /foo/bar/*)
 - ◉ Extension mappings (eg: *.jsp, *.do)
 - ◉ Exact mappings (eg: /test, /test/foo)
- URI template
 - ◉ /order/{orderId} would process /order/A0001
 - ◉ /dictionary/{char}/{word} would process /dictionary/c/cat

As stated earlier, a resource can be associated with a URL mapping or a URI template. A URL mapping could be any valid servlet mapping. Hence, as stated in the servlet specification, there are three types of URL mappings:

Path mappings - For example, /test/*, /foo/bar/*

Extension mappings - For example, *.jsp, *.do

Exact mappings - For example, /test, /test/foo

When a resource is defined with a URL mapping, only those requests that match the given URL mapping will be processed by the resource. Alternatively, you could configure a resource with a URI template. A URI template represents a class of URIs using patterns and variables. Some examples of valid URI templates are given below.

/order/{orderId}

/dictionary/{char}/{word}

All the identifiers within curly braces are considered variables. For example, the template /order/{orderId} would process /order/A0001

Creating APIs

Example 1: Use the <api> tag with a unique name and URL context.

```
<api name="API_1" context="/order">  
  <resource url-mapping="/list"  
    methods="POST" inSequence="RequestSeq"  
    outSequence="DisplaySeq"/>  
</api>
```

An API definition is identified by the <api> tag. Each API must specify a unique name and a unique URL context. One or more tags can be enclosed within an API definition.

Creating APIs

Example 2: Use 'url-mapping' and 'uri-template' attributes to define resource paths.

```
<api name="API_2" context="/user">  
  <resource url-mapping="/list/*" methods="GET"  
    inSequence="ReqUsersSeq"outSequence="ReturnUsersSeq"/>  
  <resource uri-template="/edit/{userId}"  
    methods="PUT POST" inSequence="InsertUpdateSeq"  
    outSequence="ReturnStatusSeq"/>  
</api>
```

An API definition is identified by the <api> tag. Each API must specify a unique name and a unique URL context. One or more tags can be enclosed within an API definition.

Creating APIs

Example 3: Define default resource.

```
<api name="API_3" context="/payments">
  <resource url-mapping="/list" methods="GET" inSequence="seq1" outSequence="seq2"/>
  <resource uri-template="/edit/{userId}" methods="PUT POST" outSequence="seq3">
    <inSequence>
      <log/>
      <send>
        <endpoint key="BackendService"/>
      </send>
    </inSequence>
  </resource>
  <resource inSequence="seq5" outSequence="seq6"/>
</api>
```

Note the last resource definition in API_3. It does not specify a URL mapping nor a URI template. This is called the default resource of the API. Each API can have at most one default resource. Any request received by the API that does not match any of the enclosed resource definitions will be dispatched to the default resource of the API. For example, a DELETE request on the "/payments" URL will be dispatched to the default resource as none of the other resources in API_3 are configured to handle DELETE requests.

Let's try it out!

Configuring a REST API





Let's try it out!

Configuring a Proxy Service

