



EXERCISE GUIDE
API MANAGEMENT WITH
WEBMETHODS PLATFORM

456-71E

Software AG
Internal Use Only!

This publication is protected by international copyright law. All rights reserved. No part of this publication may be reproduced, translated, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of Software AG.

Software AG and all Software AG products are either trademarks or registered trademarks of Software AG. Other product or company names mentioned herein may be the trademarks of their respective owners.

TABLE OF CONTENTS

Introduction: Current Situation of the SAGTours Company	5
Exercise 1: How to set up API Gateway	7
Exercise 2: Creating a REST API using a Swagger File	15
Exercise 3: Identify Business Objects and configure REST APIs.....	19
Exercise 4: Creating Rest API from Scratch	21
Exercise 5: Adding a Logging Policy.....	37
Exercise 6: Enforcing Security Policy	43
Exercise 7: Using a WSS Security Token.....	47
Exercise 8: Creating a Consumer Application	55
Exercise 9: Enforcing a Consumer Relationship using API Key	61
Exercise 10: Managing API Threat Protection	73
Exercise 11: Manage Fine Granular Exposure of APIs	87
Exercise 12: Monitoring the Virtual API	95
Exercise 13: Using a Routing Policy Action	99
Exercise 14: Updating Content Using XSLT Stylesheet.....	105
Exercise 15: Setting up Integration with API Portal Objectives	109
Exercise 16: Publishing API to API-Portal	113
Exercise 17: Publishing SAGTours APIs to API-Portal.....	117
Exercise 18: Using API Portal as Consumer	127
Exercise 19: Send Events to API Portal.....	137
Exercise 20: Managing API Packages and Plans.....	143
Exercise 21: Try out API secured with OAuth2 token.....	153
Exercise 22: Editing APIs in API Portal.....	157

This page intentionally left blank

Software AG
Internal Use Only!

Introduction: Current Situation of the SAGTours Company

Overview

SAG Tours is a leading provider of yachting holidays offering worldwide yachting trips. It offers pre-arranged tours that can be booked by individuals (Sailing Cruises) as well as yacht chartering (Charter Cruises). SAG Tours primarily operates through its retail agency business.

SAG Tours Future Goals

SAG Tours has recognized the opportunity that APIs allow to generate additional revenue.

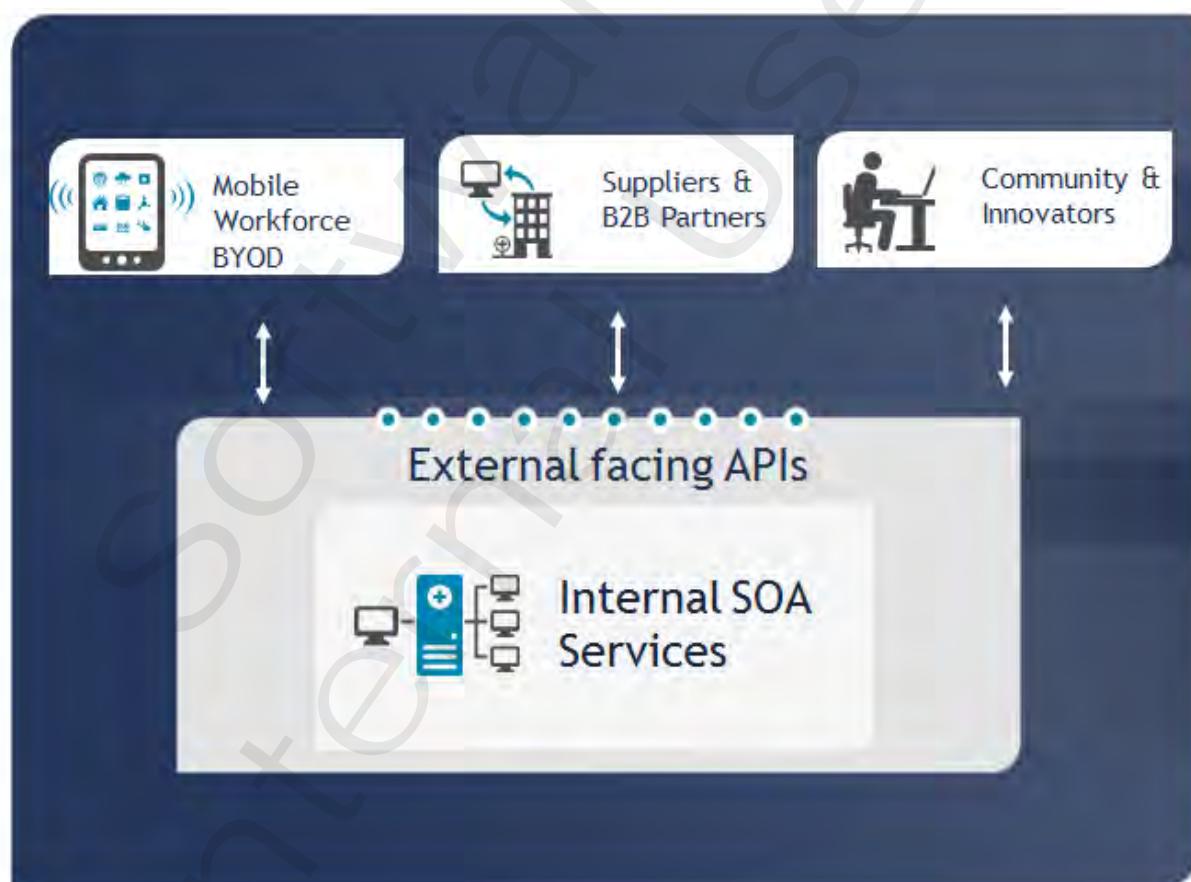
- Support Mobile initiatives and use APIs to serve data to mobile devices
- Leverage APIs for supply-chain and ecommerce initiatives to simplify B2B integration and expand channel outreach
- Encourage external innovation through the community and create incremental revenue

Therefore SAGTours has decided to start an open API initiative along with its traditional sales channels. SAGTours expects that these parties will create innovative new travel applications and mobile applications with the content coming from SAGTours provided through its APIs. Through APIs they want to allow third parties to access booking capabilities that SAGTours uses in its current channels.

To support their API initiative, SAGTours plan to host an API developer portal, to expose their APIs to consumers.

The service/APIs of the booking application should be registered/defined within CentraSite, which is the SAGTours registry and repository. For Runtime Governance SAGTours will use webMethods API Gateway for aspects like logging, security, consumer based monitoring,

The API developer portal will be built using webMethods API-Portal. All APIs SAGTours wants to expose to consumer will be published to the API Portal, so that internal and external consumers can search for APIs using API Portal.



Exercise 1:

How to set up API Gateway

Objectives

In this exercise, you will set up the environment for API Gateway and the SAGTours scenario.

Steps

1. Open **Windows Services UI** to double check that the following services, needed for API Gateway, are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1 (default)**
 - i. **Make sure to start Software AG Integration Server and not the one which is named Internal Integration Server**
 - ii.i. **It is important to verify that the Event Data Store is started, as API Gateway requires that**
 - b.c. **Software AG Runtime 10.1**
 - i. **This service is important so that the SAGTours services are available on port 53307**
 2. Login to **Integration Server Administration** by opening **Firefox** and clicking the **IS Administration** link (just below the URL field). Login as Administrator (**Administrator / manage**).
Note: The Integration Server takes a few minutes to start, therefore you may have to refresh the Administration page until you are asked to login.



3. Select **Packages -> Management** and enable the following packages:
 - a. **Bookstore**
 - b. **SAGTours**
 - c. **TestInvokeESB**
 - d. **MediatorECRWebservices**

Exercise 1: How to set up API Gateway

Package Name	Home	Reload	Enabled	Loaded	Archive	Safe Delete	Delete
Bookstore							
Default							
EmployeeManagement							
OrderService							
RESTServicePkg							
SAGTours							
TestInvokeESB							
websocketDemoPackage							

4. Login to **API Gateway** by opening **Firefox** and clicking the **API Gateway** link (just below the URL field). Login as Administrator (**Administrator / manage**).
5. We want to define different users who have different responsibilities in API Gateway like
 - a. Create APIs in API Gateway
 - b. Publish APIs to API Portal
 - c. Approve Consumer

Navigate to **Administration > User Management**.

6. Within the **Users** tab click **Add User**. Provide the following properties:

Exercise 1:
How to set up API Gateway

- a. **Login ID:** Sumala
 - b. **First Name:** Sumala
 - c. **Last Name:** Sumus
 - d. **Password:** manage
 - e. **Email Address:** sumala@company.com
- i. Click **+Add**

The screenshot shows the 'Create User' page in the WEBMETHODS API Gateway. The 'User details' tab is active. Under 'Basic information', the 'Login ID' is 'Sumala'. In the 'Groups' section, 'API-Gateway-Providers' is listed and selected. At the bottom right, there are 'Cancel' and 'Save' buttons, with 'Save' being highlighted.

- f. Click **Continue to associate Groups >**
- i. Add to group: **API-Gateway-Providers**

The screenshot shows the 'Create User' page with the 'Groups' tab selected. A group named 'api' is selected in the 'Find groups' dropdown. The 'Save' button is highlighted with a red box.

Click Save.

7. Create a new user with the following properties:
 - a. **Login ID:** Andy
 - b. **First Name:** Andy
 - c. **Last Name:** Roth
 - d. **Password:** manage
 - e. **Email Address:** andy@company.com
 - i. Click **+Add**
 - f. Click **Continue to associate Groups >**
 - i. Add to group: **API-Gateway-Administrators**
- Click Save.**
8. Create a new user who is responsible with the following properties:
 - a. **Login ID:** Peter
 - b. **First Name:** Peter

Exercise 1:
How to set up API Gateway

- c. **Last Name:** Green
d. **Password:** manage
e. **Email Address:** peter@company.com

i. Click **+Add**

Click **Save**. Without adding Peter to a group.

9. Later on in the exercises we need a group of users who are approvers of specific processes. Navigate to **User Management > Groups**. Click **Add group**.

The screenshot shows the 'User Management' section of the WEBMETHODS API Gateway interface. Under the 'Groups' tab, there is a list of three groups: 'Administrators', 'API-Gateway-Administrators', and 'API-Gateway-Providers'. Each group has a small icon, a name, a description, and an 'Action' column. In the top right corner of the page, there is a blue button labeled '+ Add group' with a red box drawn around it.

a. **Name:** Approvers

b. **Description:** group of users responsible for approval process within application management

The screenshot shows the 'Create Group' dialog. Under the 'Basic information' tab, the 'Name' field is filled with 'Approvers' and the 'Description' field contains 'group of users responsible for approval process within application management'. At the bottom of the dialog, there is a blue button labeled 'Continue to associate Users' with a red box drawn around it.

Click **Continue to associate Users**. Provide character P for **Login ID**. From the list of users select **Peter**. Click **+**.

The screenshot shows the 'Create Group' dialog. In the 'Find users' section, the 'Login ID' field has 'P' entered. Below the field, a list of users is shown, with 'Peter' highlighted by a red box.

Click **Save**.

10. Now we need to set up an appropriate access profile for the approver group. Navigate to **User Management > Access profiles**. Click **Add access profile**.

The screenshot shows the 'User Management' section of the WEBMETHODS API Gateway interface. Under the 'Access profiles' tab, there is a list of two access profiles: 'Administrators' and 'API-Gateway-Providers'. Each profile has a small icon, a name, a description, and an 'Action' column. In the top right corner of the page, there is a blue button labeled '+ Add access profile' with a red box drawn around it.

Provide the following properties:

a. **Basic Information:** Approvers

b. **Description:** approve Application management process (create, register, update)

Exercise 1:
How to set up API Gateway

The screenshot shows the 'Create Access Profile' page. In the 'Access profile details' section, there are tabs for 'Basic information', 'Functional privileges', and 'Groups'. The 'Basic information' tab is selected, showing fields for 'Name*' (Approver) and 'Description' (approve application management process (create, register update)). Below these fields is a link 'Continue to assign Functional privileges >'. The 'Functional privileges' tab is visible but not selected.

Click **Continue to assign Functional privileges**. Select the following **Functional privileges > APIs, Policies, and Applications**

c. **Manage APIs**

d. **Manage Applications**

The screenshot shows the 'Create Access Profile' page with the 'Functional privileges' tab selected. Under 'APIs, Policies, and Applications', several checkboxes are listed: 'Manage APIs' (selected), 'Manage aliases', 'Manage policy templates', 'API Portal Management', 'Manage packages and plans', 'Administration configurations', 'View administration configurations', 'Execute service result cache APIs', 'Manage user administration', 'Export or Import assets and Purge and Archive events', 'Import assets', 'Activate / Deactivate APIs', 'Manage global policies', 'Manage threat protection configurations', 'Activate / Deactivate packages', 'Manage general administration configurations', 'Manage destination configurations', 'Export assets', 'Publish to API Portal', 'Manage security configurations', 'Manage system settings', and 'Manage purge and restore runtime events'. The 'Manage APIs' and 'Manage applications' checkboxes are highlighted with red boxes.

Click **Continue to associate Groups**. Provide character A for **Name**. From the list of groups select **Approvers**. Click **+**.

The screenshot shows the 'Create Access Profile' page with the 'Groups' tab selected. A search bar 'Find groups' with 'Name' field containing 'A' is shown. A list of groups is displayed: 'Approvers' (selected and highlighted with a red box), 'API-Gateway-Administrators', 'API-Gateway-Providers', and 'Administrators'. The 'Approvers' group is highlighted with a red box.

Click **Save**.

11. Navigate to **Administration > Destinations**.

Exercise 1:
How to set up API Gateway

The screenshot shows the WEBMETHODS API Gateway Administration interface. The top navigation bar includes links for APIs, Policies, Applications, and Packages. On the right, there's a user icon labeled "Administrator". Below the navigation, a breadcrumb trail shows "Home > Administration". The main content area has tabs: "General" (selected), "Security", "Destinations", and "System settings". The "General" tab contains sections for "Event types" and "Performance metrics data". Under "Event types", checkboxes are selected for "Error", "Lifecycle", and "Policy violation". Under "Performance metrics data", the "Report performance data" checkbox is selected, and the "Publish interval (minutes)" field is set to "60". A "Save" button is visible at the bottom.

12. The destination **API Gateway** is preselected. Edit the configuration and set the following values: Some of the settings are already done, and some need to be adapted.

- a. **Error:** - select -
- b. **Lifecycle:** - select -
- c. **Policy violation:** - select -
- d. **Report Performance data:** - select -
- e. **Publish interval (minutes):** 2

This screenshot is identical to the one above, showing the WEBMETHODS API Gateway Administration interface with the "General" tab selected. It displays the same configuration options for event types and performance metrics, with the "Publish interval (minutes)" set to "2".

Click **Save**.

Software AG
Internal Use Only!

This page intentionally left blank

Software AG
Internal Use Only!

Exercise 2: Creating a REST API using a Swagger File

Objectives

In this exercise, you will add a REST Service to API Gateway. You will run the service and look for the performance metrics information in API Gateway

To create and manage services in API-Gateway, users must belong to the following group in Integration Server:

- API-Gateway-Provider

We will keep our exercise as simple as possible. **Sumala** will be the service provider.

Steps

1. Open **Windows Services UI** to double check that the following services, needed for API Gateway are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1**
 - b. **Software AG Event Data Store 10.1 – will be started with Integration Server**
2. Start **Mozilla Firefox** and logon to **API Gateway** as user **Sumala/manage**.
3. Create a new asset of type **REST** with name **Bookstore** using the import functionality of API Gateway.

Click **Manage API** within the Home Page

WEBMETHODS
API Gateway

APIs Policies Applications Packages

Sumala

Welcome to webMethods API Gateway

End-to-end API management solution powered by WEBMETHODS
Helps you leverage your internal assets by exposing them as simple APIs in a secure fashion, provides relief from threat protection and complex routing and traffic management problems, provides extensive mapping and transformation support, monitors performance metrics, and provides API analytics.



Manage policies

Define and manage policies that perform security-related activities, auditing/logging tasks, and performance reporting functions.

LEARN MORE



Analyze APIs

Get more insight about APIs using relevant KPIs and keep track of key metrics.

LEARN MORE



Manage APIs

View and manage APIs.

LEARN MORE

Version: 10.1.0.0.508

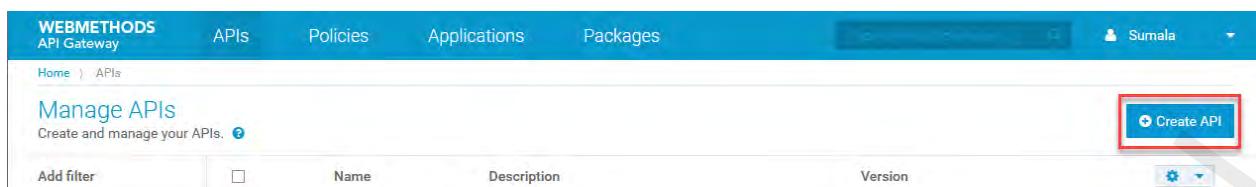
software

or navigate to **APIs** in the header section.

Exercise 2:
Creating a REST API using a Swagger File

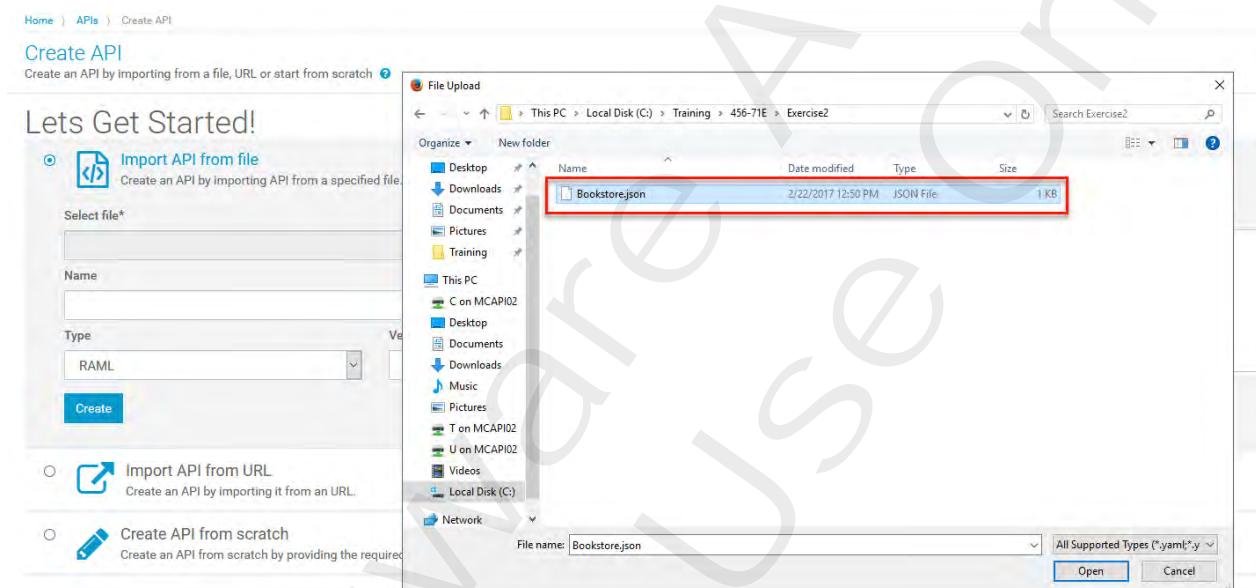


4. Click Create API.



A shortcut to this would be using the **Create API** on the Home page.

5. We would like to import a REST API from file. Therefor we leave the default selection. Within the **Import API from File** section click **Browse**. Navigate to **C:/Training/456-71E/Exercise2** and select **Bookstore.json**.



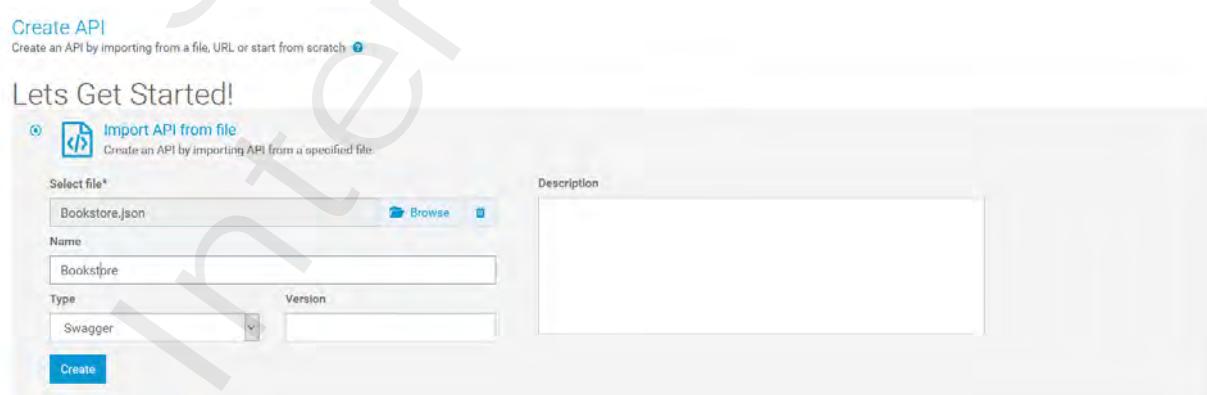
Click **Open**.

Provide the following properties:

Name: Bookstore

Type: Swagger

Version: 1.0



Click **Create**.

Exercise 2:
Creating a REST API using a Swagger File

6. Review the properties of the generated API.

The screenshot shows the 'Bookstore' API details page. At the top, there are tabs for 'API details', 'Scopes', 'Policies', 'Applications', and 'Analytics'. Below the tabs, the 'Basic information' section is expanded, showing details like Name: Bookstore, Version: 1.0, Owner: Sumala, Active: No, Maturity state: Beta, and Created: 2017-12-29 13:46:25 GMT. The 'Description' field contains a detailed explanation of the service. Below this, the 'Technical information' section is shown, with the native endpoint listed as http://localhost:5555/rest/bookstore.

7. Click **Activate**. Click **Yes**.

8. Review the API details. Scroll down to **Technical Information** to get the gateway endpoint .

The screenshot shows the 'Bookstore' API details page after activation. The 'Basic information' section is identical to the previous screenshot. In the 'Technical information' section, the 'Gateway endpoint(s)' field is highlighted with a red box, showing the URL http://daivintrain24496:5555/gateway/Bookstore/1.0.

9. Open **Resources and methods** to review the registered Resources. Select the resource /books to view the resource information and the assigned methods.

10. Explore consuming this API:

- Copy the Gateway URL as shown, then click on the REST Test Client icon;

Exercise 2:
Creating a REST API using a Swagger File

The screenshot shows the WebMethods API Gateway interface. In the top navigation bar, there are tabs for WEBMETHODS, APIs, Policies, Applications, and Packages. On the right side, there are buttons for 'Sumala' and 'Create new version'. Below the navigation, the URL is 'Home > APIs > Bookstore'. The main content area is titled 'Bookstore' with the sub-section 'View API details, basic and technical information, resources and methods available, and API documentation'. There are several tabs: 'API details' (selected), 'Scopes', 'Policies', 'Applications', and 'Analytics'. Under 'API details', there are sections for 'Basic information', 'Technical information' (which is currently selected and highlighted in blue), 'Resources and methods', 'API mocking', and 'Specifications'. The 'Technical information' section displays the 'Native endpoint(s)' as 'http://localhost:5555/rest/bookstore' and the 'Gateway endpoint(s)' as 'http://daeltrain26051-5555/gateway/Bookstore/1.0'. A red circle highlights the gateway endpoint URL.

- b. Paste the URL into the REST client and append "/books" to the URL; then click "SEND".

The screenshot shows a REST Client tool window. At the top, it says 'Request' and 'Response' tabs. The 'Request' tab is active, showing a 'Method: GET' dropdown set to 'GET' and a 'URL: http://daeltrain26051-5555/gateway/Bookstore/1.0/books'. A red circle highlights the URL. To the right of the URL is a 'SEND' button, which is also highlighted with a red circle. Below the request tab, there is a 'Body' section with a text area containing placeholder JSON. The 'Response' tab is shown below, with 'Headers' and 'Response' sub-tabs. The 'Response' sub-tab is active and displays an XML response. The XML content includes the following structure:

```
<?xml version="1.0"?>
<booklist>
  <book isbn="9781584033970">
    <author>Eric Weisstein</author>
    <title>CRC Concise Encyclopedia of Mathematics</title>
    <rating>3.5</rating>
    <votes>2</votes>
    <detailurl>http://localhost:5555/rest/bookstore/books/9781584033970</detailurl>
  </book>
</booklist>
```

Exercise 3:

Identify Business Objects and configure REST APIs

Objectives

In this exercise, you will discuss and think about which Business Objects we have in our current application and how we would define the REST APIs for our consumers

Steps

1. Design the APIs which **SAGTours** would create in order to expose their Business to outside of the organization
 - a. Which business services exist and which are useful to expose as APIs?
 - b. How to configure REST APIs using best practices?
 - i. Which resources do we have, Path Definition
 - ii. Which HTTP methods would we like to support
 - iii. Which Parameters would we like to provide
 1. Path parameters
 2. Header Parameter
 3. Query parameter
 - c. Think about using best practices like
 - i. Partial Responses
 - ii. Limit Results
 - iii. Pagination
 - iv. Server Side Sorting
 - v. Design for Versions
 - d. Think about roles and permission
 - i. API Administrator
 - ii. API Developer
 - iii. API Provider
 - iv. API Consumer
2. This is a Sample Result which we will use in our workshop to expose the SAGTours application. We will add these entities to API Gateway and expose to Consumers:
 - a. Following resources
 - i. Search just using HTTP Methods: GET
 1. /cruises
 2. /cruises?
 3. /cruises/{cruiseID}
 - ii. Booking
 - iii. SignUp
 - b. Following Groups, Users, Roles
 - i. Andy as API as owner of Runtime Environment

This page intentionally left blank

Software AG
Internal Use Only!

Exercise 4: Creating Rest API from Scratch

Objectives

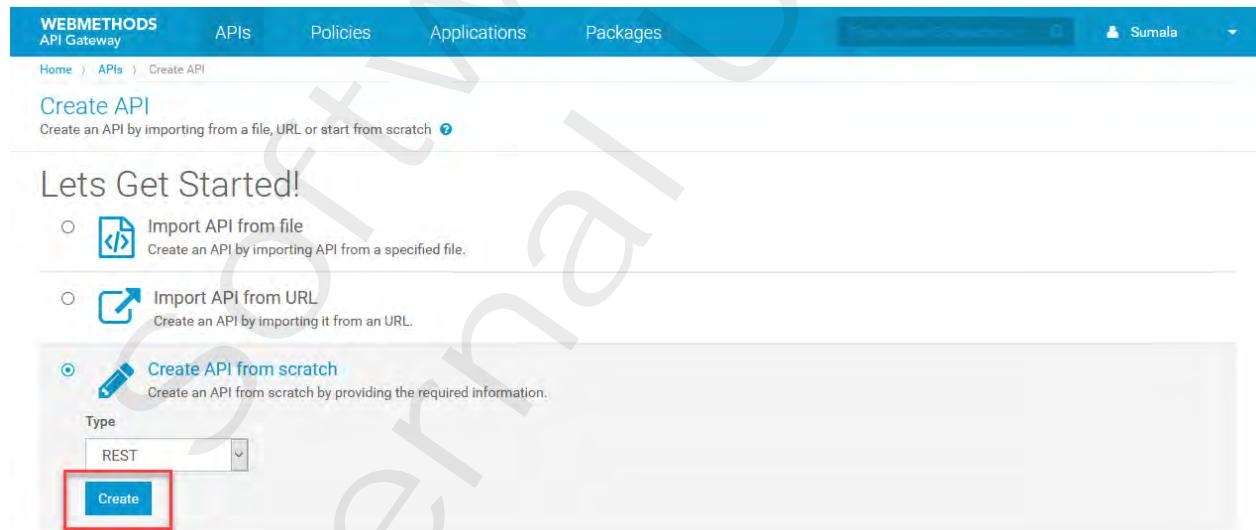
Create a REST API from Scratch.

Steps

1. Open **Windows Services UI** to double check that the following services, needed for API Gateway, are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1**
 - b. **Software AG Event Data Store 10.1 – will be started with Integration Server**
2. Logon to **API Gateway** as user **Sumala/manage**.
3. Create a new asset of type **REST** from Scratch. The service enables users to search for tours offered by **SAGTours**. The Search service provides the following search options:
 - Show all available cruises
 - Show cruises based on filter criteria
 - Show a specific cruise based on its cruise ID.

If an error occurs, the API returns an HTTP status code and, when appropriate, a plain text message in the message body providing the reason for the error.

Navigate to **Manage APIs**, click **Create API**. Select **Create REST API from scratch** and click **Create**.



The screenshot shows the 'Create API' interface. It has three options: 'Import API from file', 'Import API from URL', and 'Create API from scratch'. The 'Create API from scratch' option is selected and highlighted with a red box. Below it, there is a dropdown menu set to 'REST' and a blue 'Create' button, also highlighted with a red box.

4. Add properties of new REST service – you can copy and paste content out of file **CruisesAPIResourceCruise.txt** in folder **C:\Training\456-71E\Exercise4**
 - c. Basic Information:
 - i. **Name:** SearchCruise
 - ii. **Version:** 1.0
 - iii. **Maturity state:** Beta
 - iv. **API grouping:** Search

Exercise 4:
Creating Rest API from Scratch

- v. **Description:** Searches for all cruises, or searches for a cruise that matches the specified criteria.

The screenshot shows the 'API details' page for a new API named 'SearchCruise'. The 'Basic information' section is filled out with the following details:

- Version:** 1.0
- Maturity state:** Beta
- API grouping:** (empty)
- Description:** Searches for all cruises, or searches for a cruise that matches the specified criteria

A large watermark 'Software AG Only' is diagonally across the page.

Click **Continue to provide Technical information for this API**.

d. **Technical information:**

- Protocol:** HTTP
- Host:** localhost
- Base path:**

The screenshot shows the 'API details' page with the 'Technical information' section highlighted. The 'Protocol' field has 'HTTP' checked and 'HTTPS' unchecked. The 'Host' field contains 'localhost'. The 'Base path' field is empty.

Click **Continue to provide Resource & Methods for this API**.

e. **Resources and Methods:**

- Resource name:** cruises
- Resource path:** /cruises
- Description:** cruise information like destination, ships, ports, ...
- Supported Methods:** GET

Exercise 4:
Creating Rest API from Scratch

The screenshot shows the 'API details' tab selected in the left sidebar. A modal window titled 'Add resource' is open, prompting for a 'Resource name*' (cruises) and a 'Resource path*' (/cruises). Under 'Supported methods*', 'GET' is checked, while 'POST', 'PUT', 'DELETE', and 'PATCH' are unchecked. Below the modal is a link to 'Continue to provide Mocking information for this API >'.

Click **Add**.

- f. Scroll down to **GET** methods and provide the following properties.
- i. **Description:** Search for all cruises, or searches for a cruise that matches the specified criteria

The screenshot shows the 'SearchCruise' API details page. The 'GET' method is selected in the 'Supported methods' section. The 'Description' field contains the text: 'Search for all cruises, or searches for a cruise that matches the specified criteria'. The 'Expose to consumers' toggle is turned on.

- g. Click **Add Method Parameter** section and provide the following properties
1. **Name:** startDate
 2. **Description:** Cruise start date, in the format YYYYMMDD. Selects cruises that start on this date. Combined with endDate, this parameter filter ...
 3. **Type:** Query-string
 4. **Data Type:** String
 5. **Required:** - leave empty –
 6. **Array:** - leave empty –

7. **Value:** - leave empty –

The screenshot shows a REST API configuration interface. At the top left is a green 'GET' button. To its right is a blue toggle switch labeled 'Expose to consumers'. Below the method is a 'Description' field containing the placeholder 'Search for all cruises, or searches for a cruise that matches the specified criteria'. Underneath is a table titled 'Add Method Parameter'. The table has columns for Name*, Description, Type, Data type, Required, Repeat, and Value. A row is currently selected for 'startDate', which is described as 'THIS date. Combined with endDate, this parameter filter ...'. The 'Type' dropdown shows 'Qi' and the 'Data type' dropdown shows 'St'. The 'Required' and 'Repeat' checkboxes are unchecked. The 'Value' column contains an empty input field with a '+' button and a red-bordered 'Add' button. Below the table are two buttons: 'Add Request' and 'Add Response'.

Click +Add.

- h. Click **Add Method Parameter** section and provide the following properties

1. **Name:** endDate
2. **Description:** Cruise end date, in the format YYYYMMDD. Selects cruises that start on this date. Combined with startDate, this parameter filter ...
3. **Type:** Query-String
4. **Data Type:** String
5. **Required:** - leave empty –
6. **Array:** - leave empty –
7. **Values:** - leave empty –

Click + Add.

Add Method Parameter						
Name*	Description	Type	Data type	Required	Array	Value
<input type="text"/>	<input type="text"/>	<input type="button" value="▼"/>	<input type="button" value="S1"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/> + <input type="button" value="Add"/>
endDate	Cruise end date, in the format YYYYMMDD. Selects cruises that start on this date. Combined with startDate, this parameter filter ...	Query-string	String	No	No	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
startDate	Cruise start date, in the format YYYYMMDD. Selects cruises that start on this date. Combined with endDate, this parameter filter ...	Query-string	String	No	No	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

- i. Click **Add Request** link to define the supported content type of the API. Select **application/json** from the list and click **Add**.
- i. Switch to sample and copy and paste the sample request from the txt-file **CruisesAPIResourceCruise.txt** available in **C:\Training\456-71E\Exercise4**

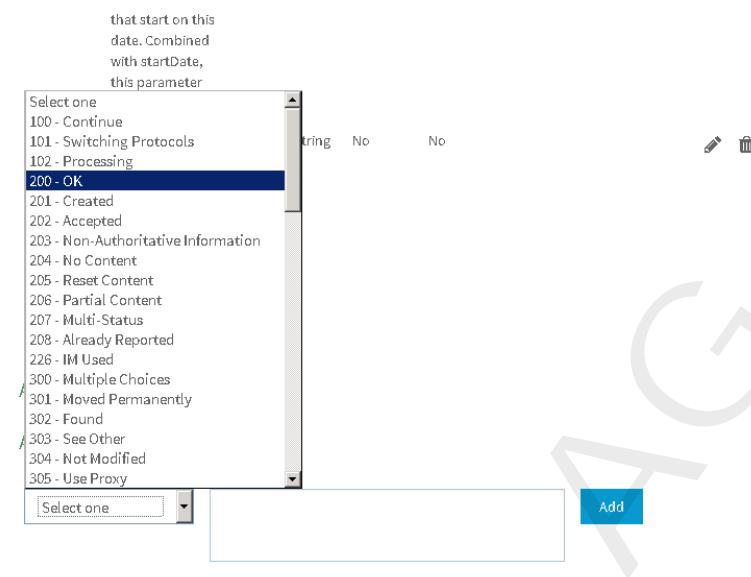
Click **Add**.

- j. Click **Add Response** link to add HTTP Status Code. Use the + icon.

1. **Status Code:** 200 OK

Exercise 4:
Creating Rest API from Scratch

2. **Description:** Search (with filter) was successful. GET method returned a list of all cruises that match the specified criteria, in JSON format.



Click **Add**.

- k. Add another Response Code:

1. **Status Code:** 400 Bad Request
2. **Description:** Query failed because the filter parameters are not valid. Check the resulting message for detail

Click **ADD**.

11. Click **Save**

The screenshot shows the 'SearchCruise' API details page. The top navigation bar includes 'Enable mocking', 'Update', 'Create new version', 'Edit', and 'Activate' buttons. The main content area has tabs for 'API details', 'Scopes', 'Policies', 'Applications', and 'Analytics'. The 'Basic information' tab is active, displaying details like Name: SearchCruise, Version: 1.0, Owner: Sumala, Active: No, Maturity state: Beta, API grouping: Search, and Created: 2017-12-28 14:35:32 GMT. The 'Description' field contains the text: 'Searches for all cruises, or searches for a cruise that matches the specified criteria.' The 'Technical information' tab shows the Native endpoint(s) as http://localhost:53307/SAGTours. The 'Resources and methods' tab lists the /cruises endpoint. The 'API mocking' tab is also visible.

12. Switch over to Edit mode and navigate to **Resources and methods**. Select the response code 200 to provide a sample response document.

Exercise 4: Creating Rest API from Scratch

The screenshot shows the 'Add Response' dialog for a 200 status code. The 'Description' field contains the text: "Search (with Filter) was successful. GET methods returns a list". The 'Content type' dropdown is set to "Select one". The 'Schema' tab is selected, showing a large empty text area for defining the schema.

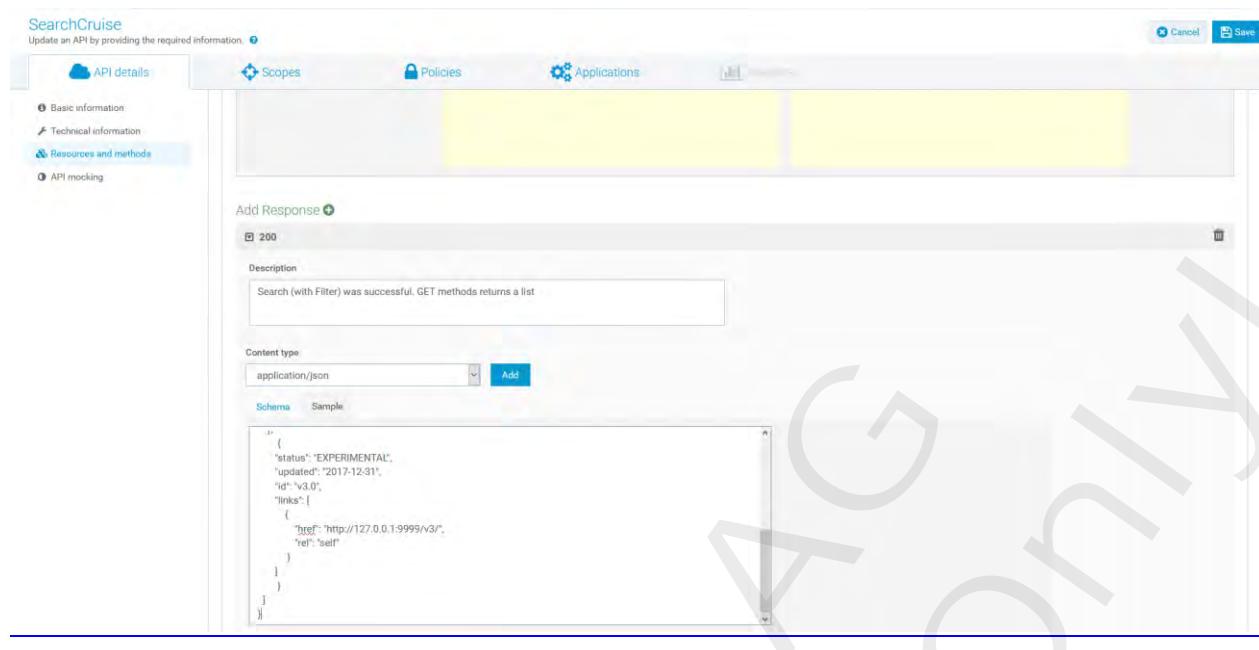
a. Provide the following properties:

- i. Content Type: application/json
- ii. Sample: - copy and paste 200 response code sample from txt file mentioned above.

The screenshot shows the 'Add Response' dialog for a 200 status code. The 'Content type' dropdown is set to "application/json". The 'Schema' dropdown is set to "Select one". The 'Sample' text area contains the following JSON response code:

```
{  
  "versions": [  
    {  
      "status": "CURRENT",  
      "updated": "2016-12-31",  
      "id": "v2.0",  
      "links": [  
        {  
          "href": "http://127.0.0.1:9999/v2/",  
          "rel": "self"  
        }  
      ]  
    }  
  ]  
}
```

Exercise 4: Creating Rest API from Scratch



The screenshot shows the 'API details' tab selected in the left sidebar. A 'Resources and methods' section is open, showing a '200' response. The 'Content type' is set to 'application/json'. The 'Schema' tab is selected, displaying a JSON schema for a cruise object:

```
        "status": "EXPERIMENTAL",
        "updated": "2017-12-31",
        "id": "v3.0",
        "links": [
            {
                "href": "http://127.0.0.1:9999/v3/",
                "rel": "self"
            }
        ]
    ]
```

Click **Add**.

13. Create another Resource using the cruiseID as Path parameter. You can copy and paste content out of file **CruisesAPIResourceCruiseWithID.txt** in folder **C:\Training\456-71E\Exercise4**. Click **Edit** to change to edit mode. Select **Resources and methods** from the **API details** menu on the left hand side. Click **+ Add Resources**.



The screenshot shows the 'API details' tab selected. The 'Resources and methods' section is open, and the 'Add Resources' button is highlighted with a red box.

Provide the following details

a. **Add Resource**

- i. **Resource name:** cruisesWithCruiseID
- ii. **Resource path:** /cruises/{cruiseID}
- iii. **Description:** Returns a specific cruise with information like destination, ships, ports, ...
- iv. **Supported Methods:** GET

Click **Add**.

- b. Open the new created resource. The Path parameter is already created.

Exercise 4:
Creating Rest API from Scratch

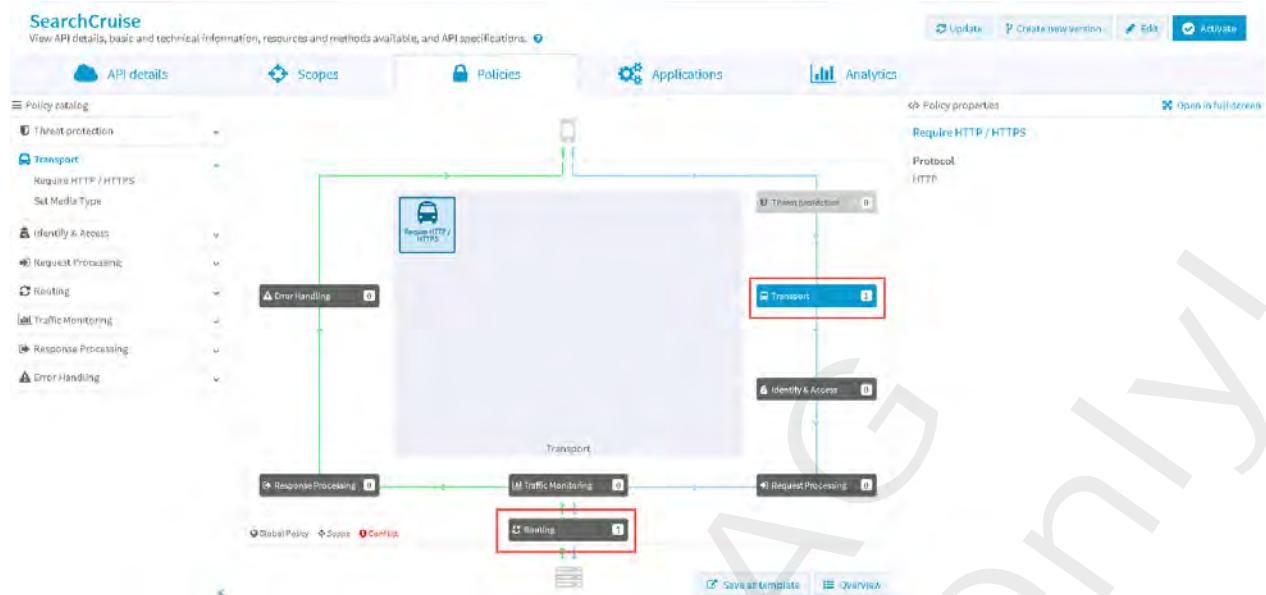
- c. Scroll down to **GET** methods and provide the following properties.
 - i. **Description:** Returns a specific cruise based on the specified cruise ID
 - d. Click **Add Request** link to provide Content type application/json. Copy and paste the sample request out of the text file mentioned above. Click **Add**.
 - e. Click **Add Response** link to add HTTP Status Code. Use the + icon.
 1. **Status Code:** 200 OK
 2. **Description:** Search with cruiseID was successful. GET methods returns a specific cruise..
- Click ADD.**
- f. Add another Response Code:
 1. **Status Code:** 400 Bad Request
 2. **Description:** Cruise ID is not valid: <cruiseID>
- Click ADD.**

Status code	Description
Select one	

Click Save.

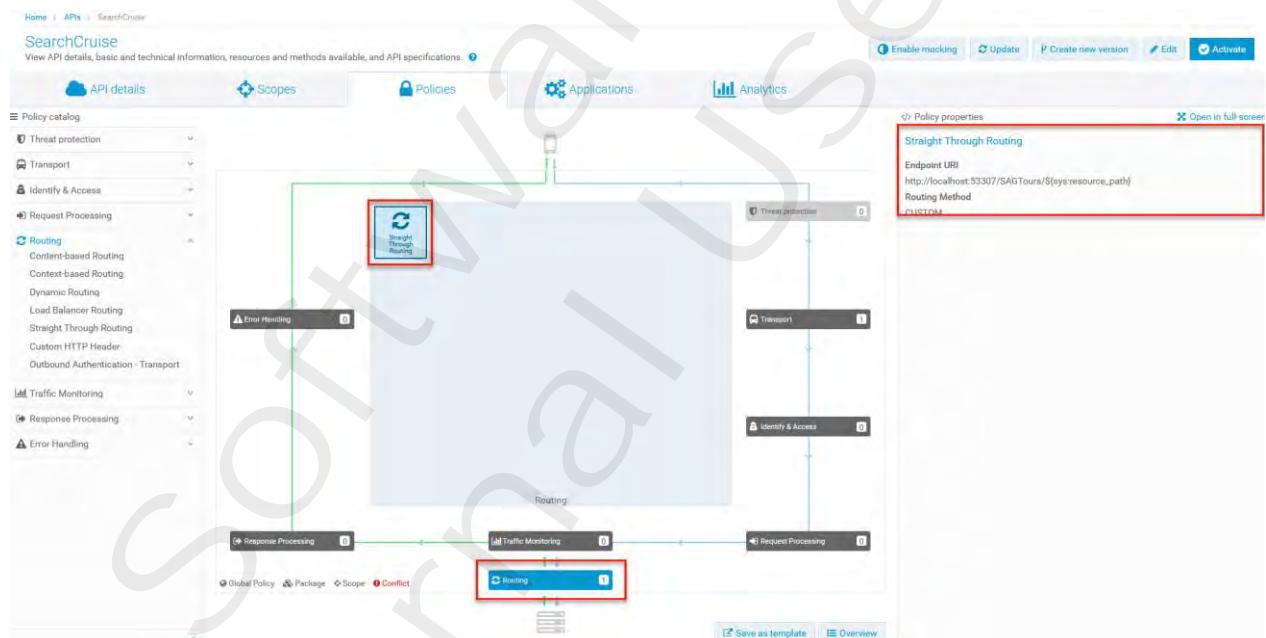
14. Review the default Policy Definitions. Select the **Policies** tab. The only policies in place are inside the section **Transport** and inside the section **Routing**.

Exercise 4: Creating Rest API from Scratch



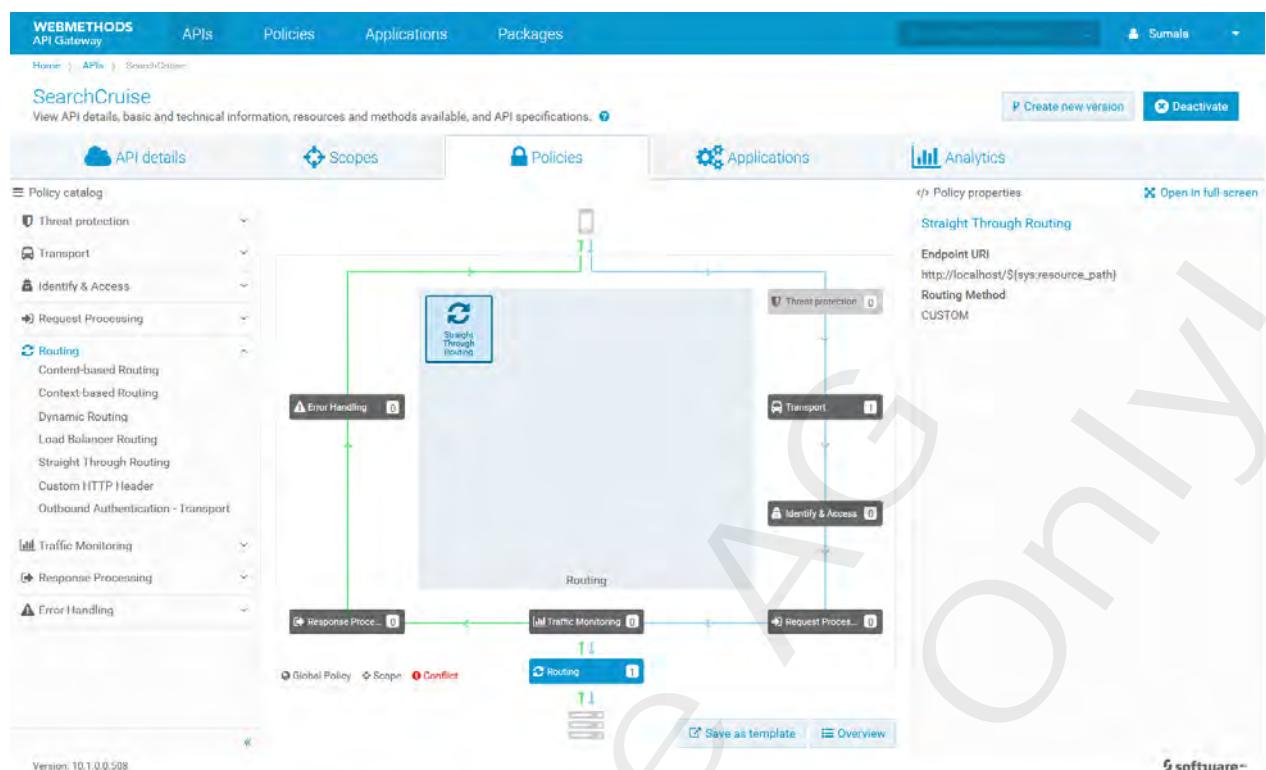
The Transport section is highlighted. The **Require HTTP / HTTPS** policy action is selected by default and configured, so that Protocol is set to **HTTP**.

15. Review the Transport policy **Require HTTP / HTTPS**. Review the Routing policy by clicking the **Routing** box in the section Configure Policies. The **Straight Through Routing** policy action is selected by default and pre-configured. Click the policy action.



16. Click **Activate**. Click **Yes**. The service will now be virtualized and created in **API Gateway**. Select the **API details** tab.

Exercise 4: Creating Rest API from Scratch



17. Navigate to **API Details** and open the profile **Technical information**. You will find a reference to the native REST endpoint and the URL to access the Gateway endpoint(s)

The screenshot shows the "SearchCruise" API details page. The left sidebar has tabs for API details, Scopes, Policies, Applications, and Analytics. The "Basic information" tab is selected. It shows the Maturity state as Beta, API grouping as Search, Created on 2017-12-29 10:37:03 GMT, Last updated on 2017-12-29 10:57:59 GMT, and a Description: "Searches for all cruises, or searches for a cruise that matches the specified criteria." Below this, the "Technical information" section is expanded, showing Native endpoint(s) as http://localhost/ and Gateway endpoint(s) as http://daeitrain24496:5555/gateway/SearchCruise/1.0. The "Resources and methods" section shows a single resource named "/cruises". The bottom of the screen shows the version "Version: 10.1.0.0.508" and the Software AG logo.

Exercise 4: Creating Rest API from Scratch

The screenshot shows the API Gateway interface for the 'SearchCruise' API. The top navigation bar includes 'Create new version' and 'Deactivate' buttons. Below the header are tabs for 'API details', 'Scopes', 'Policies', 'Applications', and 'Analytics'. The 'API details' tab is selected, displaying basic information like maturity state (Beta), created date (2018-03-05 10:35:52 GMT), and last updated date (2018-03-05 10:51:44 GMT). A description states: 'Searches for all cruises, or searches for a cruise that matches the specified criteria.' The 'Technical information' section shows native and gateway endpoints. The 'Resources and methods' section lists a single resource '/cruises' with a GET method. The method details show a description: 'Search for all cruises, or searches for a cruise that matches the specified criteria.' Method parameters include 'Name' and 'Description' columns.

18. Open **Mozilla Firefox REST client** and configure a GET request against our newly created virtual service.

- Methods:** GET
- URL:** Copy and paste the Access URL out of the **Technical information** profile and append the **/cruises** resource from the **Resources and methods** profile
- Custom Headers:**
 - Accept:** application/json

Run the request pressing the **Send** button.

The screenshot shows the Mozilla Firefox REST Client interface. The 'Request' tab is active, displaying a GET method with the URL 'http://daeitrain24496:5555/gateway/SearchCruise/1.0/cruises'. The 'Headers' section shows 'Accept: application/json'. The 'Body' section is empty. The 'Response' tab shows an error message: 'Exception: API Gateway encountered an error. Error Message: Downtime exception: Connection refused: connect. Request Details: Service - SearchCruise, Operation - /cruises, In.'

19. The error response for the request shows a downtime exception because we haven't provided a real endpoint for the native service. Therefor we want to use the Mocking feature of API Gateway.

20. Within **API Gateway** select the API and Click **Deactivate**. Click **Yes**.

The screenshot shows the API Gateway interface for the 'SearchCruise' API. The top navigation bar includes 'Create new version' and a red-highlighted 'Deactivate' button. Below the header are tabs for 'API details', 'Scopes', 'Policies', 'Applications', and 'Analytics'. The 'API details' tab is selected, displaying basic information like maturity state (Beta), created date (2018-03-05 10:35:52 GMT), and last updated date (2018-03-05 10:51:44 GMT). A description states: 'Searches for all cruises, or searches for a cruise that matches the specified criteria.' The 'Technical information' section shows native and gateway endpoints. The 'Resources and methods' section lists a single resource '/cruises' with a GET method. The method details show a description: 'Search for all cruises, or searches for a cruise that matches the specified criteria.' Method parameters include 'Name' and 'Description' columns.

Exercise 4:
Creating Rest API from Scratch

21. Click the **Enable Mocking** button at the top of the **SearchCruise** page.

The screenshot shows the 'SearchCruise' API details page. At the top right, there are several buttons: 'Enable mocking' (highlighted with a red box), 'Update', 'Create new version', 'Edit', and 'Activate'. Below these buttons are five tabs: 'API details', 'Scopes', 'Policies', 'Applications', and 'Analytics'. The 'API details' tab is selected.

22. Navigate to **API Details > API Mocking** information for this API and switch over to **Edit** mode.

- Open the resource within the section **Mocked responses**
- Review the Mock payloads for the GET methods and the response **200**.

The screenshots show the 'SearchCruise' API details page with the 'API mocking' tab selected. In the 'Mocked responses' section for the '/cruises' endpoint, a 'GET' method is selected. A '200' response is defined with 'Add Response Header' and 'Add Content-type' options. The 'Content type' is set to 'application/json'. The 'Mocked responses' area contains a JSON payload:

```
    "cruises": [
        {
            "cruiseID": "00001",
            "title": "Alaska Whale Watching Photo Expedition",
            "description": "Photographing the whale in Alaska is a unique experience. Join us on this tour to this unique area. The cruise is a luxury tour for those who want to spend time with others like for see these unique animals of the sea. The tour will be conducted by a professional photographer who will work with each of the members of the tour as needed."
        }
    ]
```

In the second screenshot, the 'Mock payloads' section for the same endpoint is shown. It also contains a JSON payload:

```
    "http://127.0.0.1:9999/v3/",
    "ref": "self"
}
```

- Click **Save**. Click **Activate**.

23. Move back to **REST API** and rerun the request. The Mocking response is shown.

Exercise 4:
Creating Rest API from Scratch

The screenshot shows a REST client interface. In the 'Request' section, a GET method is selected with the URL `http://daetrain24496:5555/gateway/SearchCruise/1.0/cruises`. The response is displayed in the 'Response' section, showing a JSON object:

```
1: {  
2:   "version": 1  
3:   "status": "CURRENT",  
4:   "updated": "2016-12-31",  
5:   "id": "v2.0",  
6:   "links": [  
7:     {  
8:       "href": "http://127.0.0.1:9999/v2/",  
9:       "rel": "self"  
10:    }  
11:  ]  
12: }  
13: 
```

24. Change the request and append the cruiseID 000001 to the URL

- a. ResponseCode shows 200, but no response body. We haven't provided a response payload.

The screenshot shows a REST client interface. A GET method is selected with the URL `http://daetrain24496:5555/gateway/SearchCruise/1.0/cruises/00001`. The response is empty, indicated by a single character '}' in the response body.

25. Within **API Gateway** deactivate the API and switch to edit mode. Navigate to **API Mocking**. Within the mocked responses for `/cruises` and the **GET** method click **Add Response**.

The screenshot shows the AWS API Gateway 'SearchCruise' API configuration. The 'Basic Information' tab is selected. In the 'Mocked responses' section for the `/cruises` endpoint and `GET` method, there is a red box around the 'Add Response' button. Below it, two response codes are listed: 200 and 400.

Exercise 4:
Creating Rest API from Scratch

- a. Select **Status code 201 – Created** and click **Add**.
- b. Select the status code **201**. This will open the section to provide the Mock Payload. Click **Add Content-type**. Provide the following properties:
 - i. **Content-type:** text/plain
 - ii. **Mock payloads:** This is the mocking response for status code 201.

Mocked responses

The screenshot shows the 'Mocked responses' section for a 'GET /cruises' endpoint. Under 'Add Response', '201' is selected. Below it, 'Content type' is set to 'text/plain' and 'Mock payloads' contains the text 'This is the mocking response for status code 201'. A blue 'Add' button is visible at the bottom right of the payload area.

Click **Add**.

26. Save your changes and activate the API.
27. Move back to **REST Client**. Provide the following changes.
 - a. Within the URL definition replace the appendix **/cruiseID** with **?expectedStatusCode=201**
 - b. Change Customer Header **Accept** from **application/json** to **text/plain**.
28. Rerun the request. The mocked response is shown.

The REST Client interface shows a successful request. In the 'Request' tab, the URL is http://dawtfan24496:5555/gateway/SearchCruise/1.0/cruises?expectedStatusCode=201, and the 'Accept' header is set to 'text/plain'. In the 'Response' tab, the response body is 'This is the mocking response for status code 201.' Both the URL and the response body are highlighted with red boxes.

29. Now we want to point to the implementation of the native API and disable the mocking of the API. Within **API Gateway** deactivate the API and switch to edit mode. Disable mocking by clicking the **Disable mocking** button.
30. Switch over to edit mode and navigate to **Technical Information**. Provide the following properties:
 - a. **Host:** localhost:53307
 - b. **Base path:** /SAGTours

Exercise 4:
Creating Rest API from Scratch

The screenshot shows the 'SearchCruise' API configuration screen. The 'Policies' tab is selected. Under 'Routing', the policy action 'StraightThroughRouting' is chosen. The 'Protocol' is set to 'HTTP' and 'Host' is 'localhost:53307'. The 'Base path' is '/SAGTours'. There are tabs for 'Basic information', 'Technical information', 'Resources and methods', and 'API mocking'. Buttons for 'Add parameter' and 'Add schema' are visible. A link 'Continue to provide Resource & Methods for this API' is at the bottom.

31. Select the **Policies** tab. Within **Routing** select the policy action **StraightThroughRouting**. Provide the change concerning the endpoint of the native service in Endpoint URI
 - a. **Endpoint URI:** http://localhost:53307/SAGTours/\${sys:resource_path}
32. Save the changes and activate the API.
33. Move back to **RESTClient**. Provide the following changes.
 - a. Within the URL definition delete the appendix?expectedStatusCode=201
 - b. Change Customer Header **Accept** from **text/plain** to **application/json**.
34. Verify the response **Response Body**.
35. In order to view the response body in correct JSON format use the **Preview** tab.

The screenshot shows the RESTClient interface. A GET request is made to 'http://daetrain25856:5555/gateway/SearchCruise1/cruises'. The 'Headers' section includes 'Accept: application/json'. The 'Body' section shows an empty JSON object. The 'Response' section shows a JSON response with one cruise entry. The JSON is displayed in a 'Preview' tab, showing the raw JSON code and a formatted tree view. The cruise details include: cruiseID: "000001", title: "Alaska Whale Watching Photo Expedition", description: "Photographing the whale in Alaska is a unique experience. Join us on this tour to this unique area. The cruise is a luxury tour for those who want to spend time with others with a like for seeing these unique animals of the sea. The tour will be conducted by a pro photography who will work with each of the members of the tour as needed.", startPort: "Juneau, Alaska", endPort: "Seattle, Washington", startDate: "20150520", endDate: "20150527", roomType: ["100P", "100D"], roomDetails: [{"roomType: "Inside Premier Double", title: "Inside Premier Double cabin", description: "An eco-friendly double room with desk, TV and shower", maxOccupants: "2", totalRoomsOfType: "3"}].

Exercise 5: Adding a Logging Policy

Objectives

In this exercise, you will learn how to provide a global runtime policy which enforces the API Gateway to log the message request body.

Steps

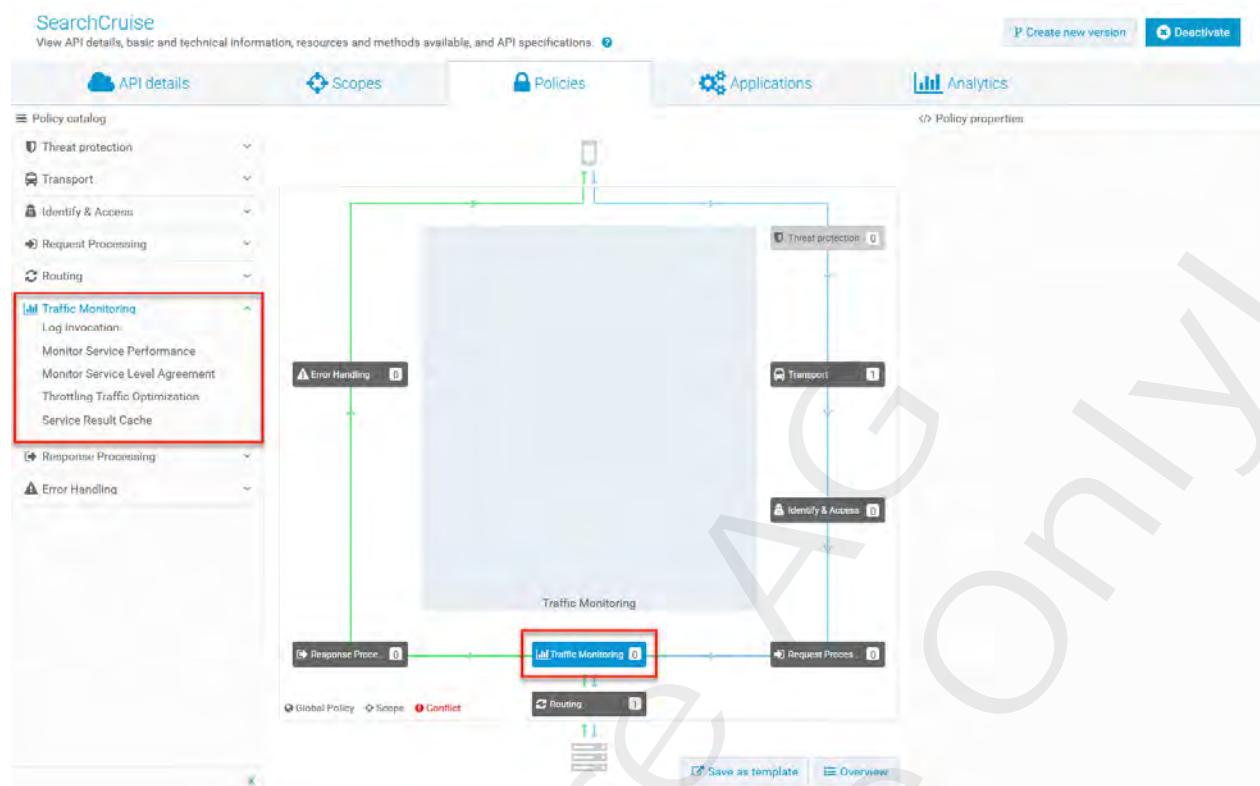
1. Open Windows Services UI to double check that the following services, needed for API Gateway, are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1**
 - b. **Software AG Event Data store 10.1 – should be started together with IS**
2. Login to the **API Gateway UI** as user **Sumala/manage** and configure monitoring policies for the service **SearchCruise**. On the **Manage APIs Page** select the service **SearchCruise**. This will open the Detail View of the **SearchCruise** service.

The screenshot shows the 'Manage APIs' page of the WEBMETHODS API Gateway. The top navigation bar has tabs for 'WEBMETHODS API Gateway', 'APIs' (which is highlighted with a red box), 'Policies', 'Applications', and 'Packages'. A user 'Sumala' is logged in. Below the tabs, there's a breadcrumb 'Home > APIs' and a 'Create API' button. The main area is titled 'Manage APIs' with the sub-instruction 'Create and manage your APIs...'. There are filters for 'Add filter', 'Type' (REST, SOAP), and 'Activation status' (Active, Inactive). Two APIs are listed: 'Bookstore' and 'SearchCruise'. The 'SearchCruise' row is highlighted with a red box. The 'SearchCruise' row details are as follows:

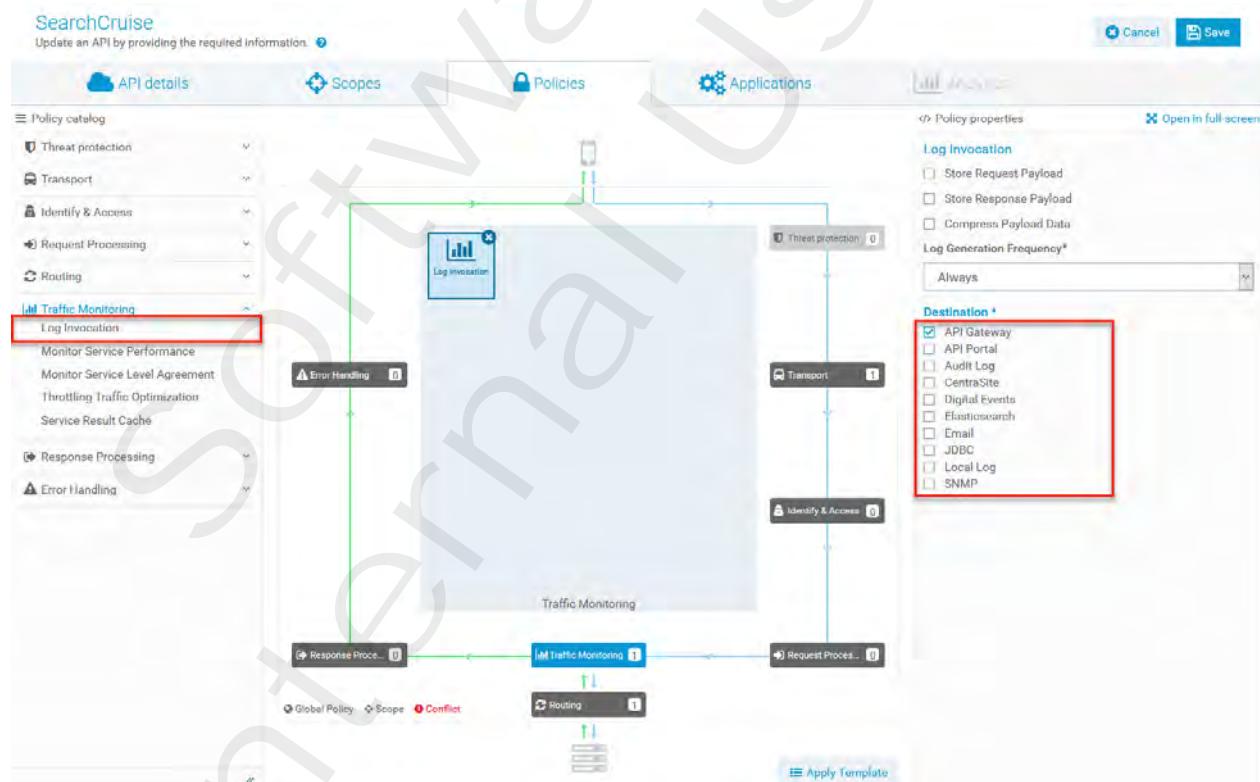
Name	Description	Version	Action
Bookstore	This service is a RESTful service which deals resources of type Book. You can use GET requests to retrieve information about books available in the Bookstore. The information is available in two forma...	1.0	
SearchCruise	Searches for all cruises, or searches for a cruise that matches the specified criteria. The GET method has two forms: -GET /cruises returns all cruises that are currently available. - GET /cruises?<...	1.0	

- a. Deactivate the API in order to change the policy configuration. Click the **Deactivate** button. Click **Edit** to open API in Edit mode.
3. Navigate to **Policies**. Select **Traffic Monitoring**.

Exercise 5: Adding a Logging Policy



- Click on Log Invocation in Policy Actions catalog section. This will move the policy action to the section </> Policy properties.



- Provide the following properties

Store Request Payload: - select -
Store Response Payload: - select -
Log Generation Frequency: always

Exercise 5:
Adding a Logging Policy

Destination: API Gateway

Log Invocation

Store Request Payload
 Store Response Payload
 Compress Payload Data

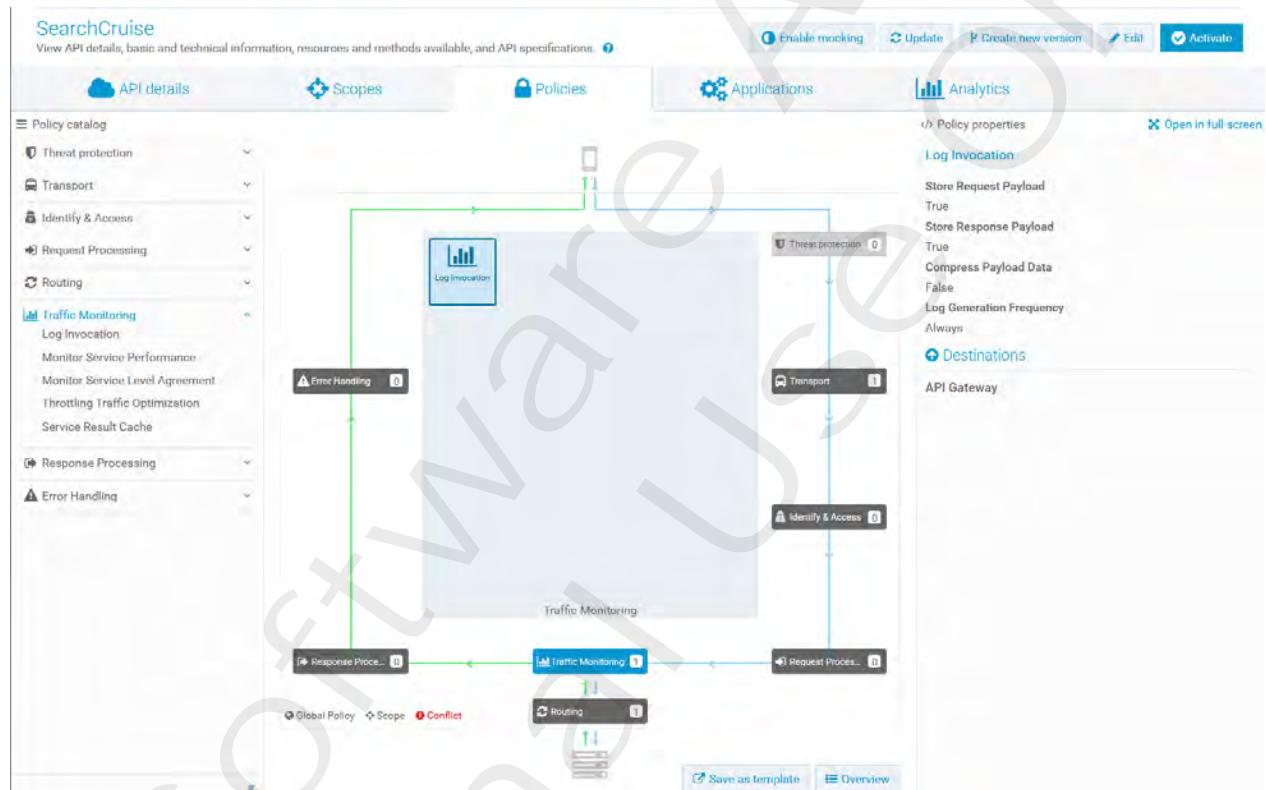
Log Generation Frequency*

Always

Destination *

API Gateway
 API Portal
 Audit Log
 CentraSite
 Digital Events
 Elasticsearch
 Email
 JDBC
 Local Log
 SNMP

Click Save.

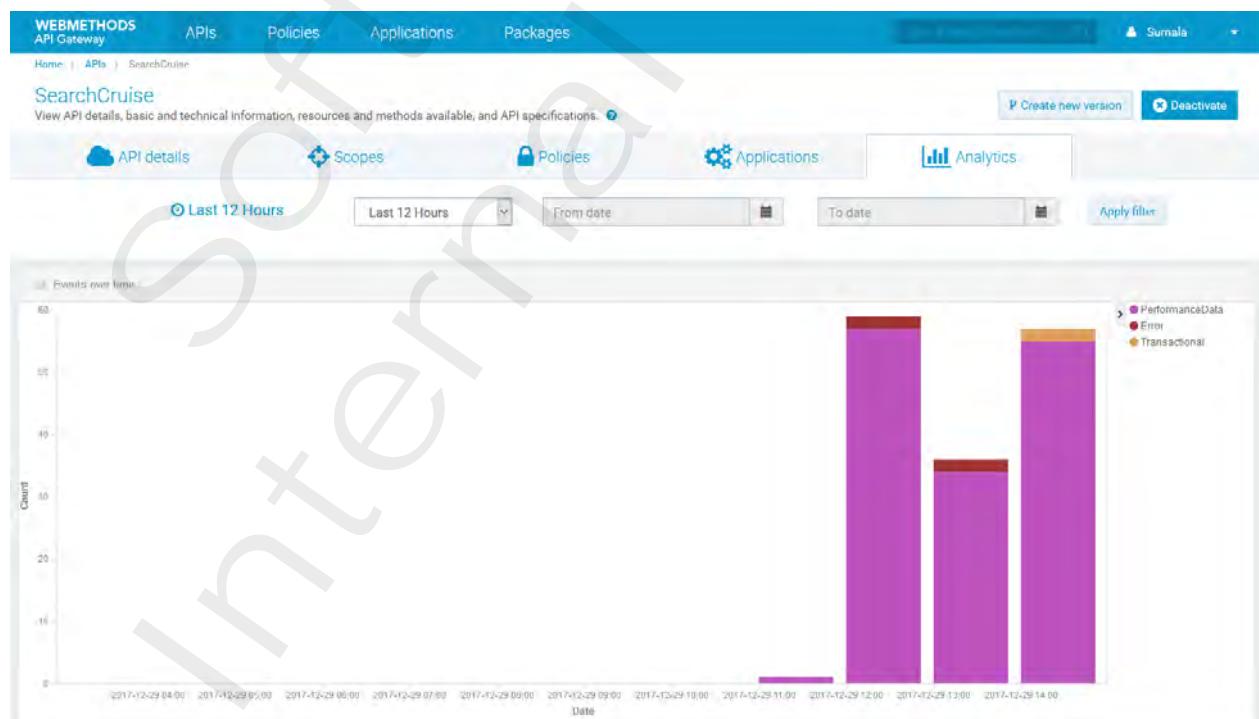


6. Click Activate, click Yes to enable the changes at runtime
7. Open API Details view to get Gateway endpoint.

Exercise 5: Adding a Logging Policy

The screenshot shows the API Gateway interface for the SearchCruise API. The top navigation bar includes 'Create new version' and 'Deactivate' buttons. Below the navigation are tabs for 'API details', 'Scopes', 'Policies', 'Applications', and 'Analytics'. The 'Policies' tab is active, showing a description: 'Searches for all cruises, or searches for a cruise that matches the specified criteria.' It details two GET methods: one for all available cruises and another for cruises matching specific criteria. It also specifies a start date for filtering cruises. The 'Technical information' tab shows native and gateway endpoints. The 'Resources and methods' tab lists '/cruises' and '/cruises/{cruiseID}'. The 'API mocking' tab indicates no mocked responses have been created.

8. Open Mozilla Firefox REST client and configure a GET request against our newly created virtual service.
 - a. **Methods:** GET
 - b. **URL:** Copy and paste the Gateway endpoint out of the **API Gateway**
Basic information profile and add resource name:
<http://DAETRAIN00749:5555/gateway/SearchCruise/1.0/cruises>
9. Run the request pressing the **Send** button. In case of an error, make sure that you have set the desired custom header for Accept.
10. Now we look for the log data **API Gateway**. Go back to **API Gateway** and select the API **SearchCruise**. Open the tab **Analytics**.



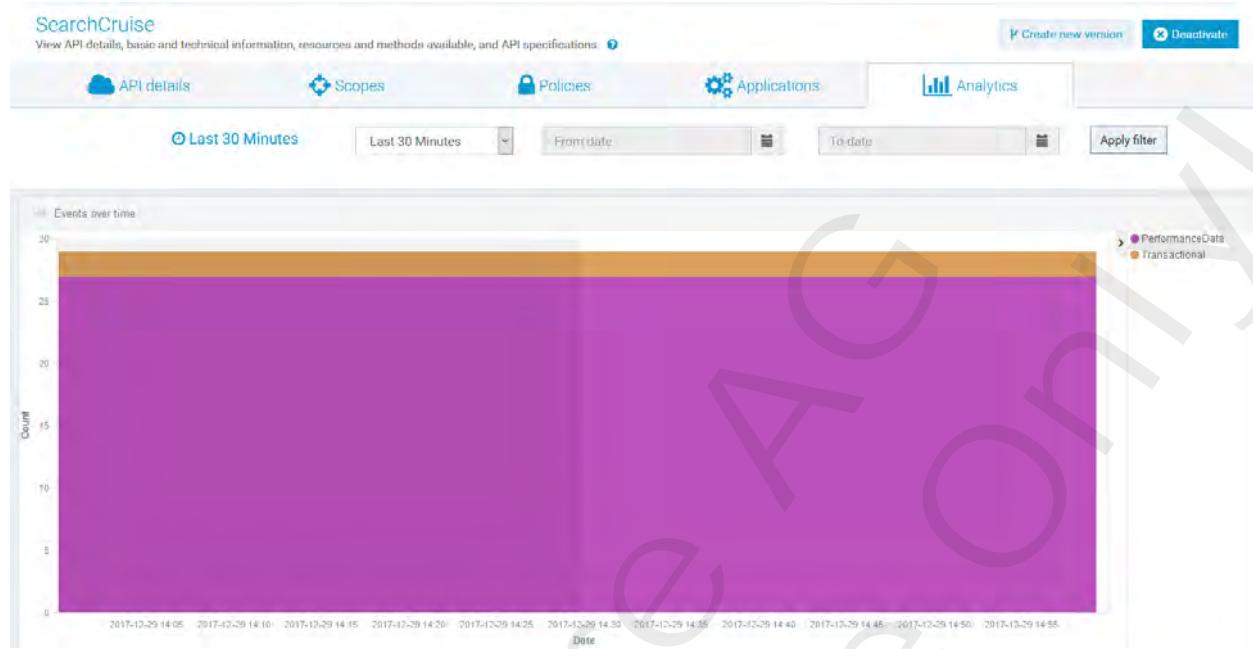
Your Graph probably will look somewhat different.

Exercise 5: Adding a Logging Policy

11. Refine the filter conditions

a. Time Range: last 30 minutes

Press Apply Filter Your Graph probably will look somewhat different.



12. Scroll down to view the provided data in the dashboards like API trend by response, Native service performance, ...

13. Navigate to Runtime Events. Select an event of type Transactional Data.

Time	apiName	applicationName	eventType	alertDesc	eventSource
2017-12-29 14:59:44.677	SearchCruise	-	PerformanceData	-	-
2017-12-29 14:58:44.677	SearchCruise	-	PerformanceData	-	-
2017-12-29 14:57:44.677	SearchCruise	-	PerformanceData	-	-
2017-12-29 14:57:43.924	SearchCruise	Unknown	Transactional	-	-
2017-12-29 14:57:41.364	SearchCruise	Unknown	Transactional	-	-
2017-12-29 14:56:44.676	SearchCruise	-	PerformanceData	-	-
2017-12-29 14:52:44.675	SearchCruise	-	PerformanceData	-	-
2017-12-29 14:51:44.685	SearchCruise	-	PerformanceData	-	-
2017-12-29 14:50:44.677	SearchCruise	-	PerformanceData	-	-
2017-12-29 14:49:44.673	SearchCruise	-	PerformanceData	-	-
2017-12-29 14:48:44.675	SearchCruise	-	PerformanceData	-	-
2017-12-29 14:47:44.673	SearchCruise	-	PerformanceData	-	-
2017-12-29 14:46:44.671	SearchCruise	-	PerformanceData	-	-

14. Click the little triangle at the beginning of the row. This will open the details for this event.

creationDate	2017-12-29 14:57:43.924
eventType	Transactional
httpMethod	get
operationName	/cruises
providerTime	5
resPayload	{ "cruises": [{ "cruiseID": "00001", "title": "Alaska Whale Watching Photo Expedition", "description": "Photographing the whale in Alaska is a unique experience. Join us on this tour to this unique area. The cruise is a luxury tour for those who want to spend time with others like me to see these unique animals of the sea. The tour will be conducted by a pro photographer who will work with each of the members of the tour as needed.", "startPort": "Juneau, Alaska", "numDays": "8", "startDate": "20150820", "endDate": "20150827", "roomTypes": [{ "roomID": "IP0", "roomDetails": "Inside Premier Double", "title": "Inside Premier Double cabin", "description": "An Eco-Friendly double room with desk, TV and Shower", "maxOccupants": "2", "totalRoomsOfType": "3"}, { "roomID": "OP0", "roomDetails": "Seaview Premier Double", "title": "Premier Double cabin with seaview", "description": "An Eco-Friendly double room with seaview, desk, TV and Shower", "maxOccupants": "2", "totalRoomsOfType": "3"}, "numRoomsAvailable": "3" }, { "roomID": "SD0", "roomDetails": "Seaview Deluxe Room", "title": "Deluxe cabin with seaview", "description": "For those wanting the extra bit of luxury. The Deluxe cabins come with a private balcony, desk, TV and Shower", "maxOccupants": "2", "totalRoomsOfType": "2"}, "numRoomsAvailable": "2" }, { "roomID": "DD0", "roomDetails": "Seaview Double Deluxe Room", "title": "Deluxe cabin with 2 double beds and a seaview", "description": "For those wanting that extra bit of luxury. The Double Deluxe cabins come with 2 double beds a private balcony, desk, TV and Shower", "maxOccupants": "4", "totalRoomsOfType": "5"}, "numRoomsAvailable": "5" }, { "roomID": "S15", "roomDetails": "Seaview Junior Suite", "title": "2 bedroom suite with seaview", "description": "The Junior suite cabins come with 2 bedrooms, a separate sitting area, a private balcony, TVs in each room and Shower", "maxOccupants": "4", "totalRoomsOfType": "5"}, "numRoomsAvailable": "5" }, { "roomType": "Seaview Deluxe Suite", "title": "3 bedroom suite with seaview", "description": "The Deluxe suite cabins come with 3 bedrooms, a spacious entertainment room, a private balcony with hot tub, TVs in each room and Bath with Jacuzzi", "maxOccupants": "4", "totalRoomsOfType": "1"}, "numRoomsAvailable": "1" }], "cruiseID": "00002", "title": "Panama canal", "description": "Cruise from San Diego to Mexico and on to the canal, then up to Miami. The canal is 77.1 Km (48 miles) long and connects the Atlantic Ocean with the Pacific Ocean. This cruise shows a side of the canal that is unique for those that want to see it in luxury.", "startPort": "San Diego", "numDays": "14", "startDate": "20150710", "endDate": "20150724", "roomTypes": [{ "roomID": "IP0", "roomDetails": "Inside Premier Double", "title": "Inside Premier Double cabin", "description": "An Eco-Friendly double room with desk, TV and Shower", "maxOccupants": "2", "totalRoomsOfType": "100"}, { "roomID": "OP0", "roomDetails": "Seaview Premier Double", "title": "Premier Double cabin with seaview", "description": "An Eco-Friendly double room with seaview, desk, TV and Shower", "maxOccupants": "2", "totalRoomsOfType": "100"}] }

This page intentionally left blank

Software AG
Internal Use Only!

Exercise 6: Enforcing Security Policy

Objectives

In this exercise, you will define a policy to enforce username and password when calling the API.

Steps

1. Open Windows Services UI to double check that the following services, needed for API Gateway, are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1**
 - b. **Software AG Event Data store 10.1 – should be started together with IS**
 - c. **Software AG Runtime 10.1**
2. Login to the **API Gateway UI** as user **Sumala/manage** and configure security policies for the service **SearchCruise**. On the **Manage APIs Page** select the service **SearchCruise**. This will open the Detail View of the **SearchCruise** service.



The screenshot shows the 'Manage APIs' page of the API Gateway UI. At the top, there are tabs for 'APIs', 'Policies', 'Applications', and 'API Packages'. Below the tabs, there's a search bar and a 'Create API' button. The main area displays a list of APIs with columns for Name, Description, and Version. Two APIs are listed: 'PetStore' and 'SearchCruise'. A red box highlights the 'SearchCruise' row. To the left, there's a sidebar with filters for Type (REST, SOAP) and Activation status (Active, Inactive).

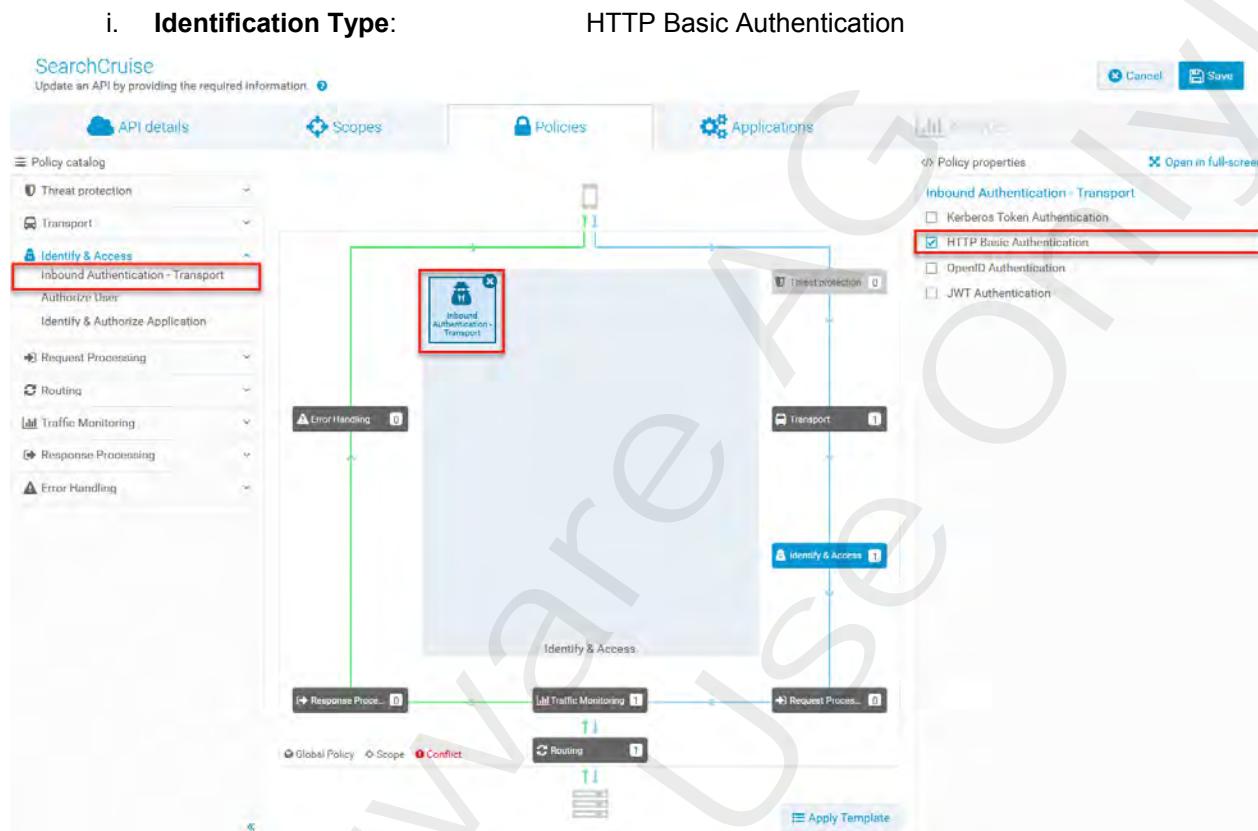
3. Navigate to **Policies**. Select **Identify & Access**.



The screenshot shows the 'SearchCruise' service detail view in the API Gateway UI. The top navigation bar includes 'Create new version' and 'Deactivate' buttons. Below the bar, there are tabs for 'API details', 'Scopes', 'Policies', 'Applications', and 'Analytics'. The 'Policies' tab is selected. On the left, a sidebar lists policy categories: Threat protection, Transport, Identify & Access, Request Processing, Routing, Traffic Monitoring, Response Processing, and Error Handling. A red box highlights the 'Identify & Access' category. The main area shows a policy flow diagram for the 'Identify & Access' category. The flow starts with 'Inbound Authentication - Transport' (green arrow), followed by 'Threat protection' (grey box), 'Transport' (grey box), 'Identity & Access' (blue box), 'Request Process.' (grey box), 'Response Process.' (grey box), 'Traffic Monitoring' (grey box), and 'Routing' (grey box). A red box highlights the 'Identity & Access' policy node in the flow.

Exercise 6:
Enforcing Security Policy

4. Deactivate the API in order to change the policy configuration. Click the **Deactivate** button.
Click **Yes**.
 - a. Click **Edit** to open API in Edit mode.
5. Click on **Inbound Authentication - Transport** in **Policy catalog** section. This will move the policy action to the section **Policy properties**.
 - a. In **Policy properties** select:



6. Click **Save**, click **Activate**. Click **Yes**.
7. Open **Mozilla Firefox REST client** and configure a GET request against our newly created virtual service.
 - a. **Methods:** GET
 - b. **URL:** Copy and paste the Gateway Endpoint and resource from API Gateway UI: <http://DAETRAIN00729:5555/gateway/SearchCruise/1.0/cruises>
 - c. **Custom Headers:**
 - i. **Accept:** application/json
- Run the request pressing the **Send** button.
8. You will get an error response with stats code **401 Unauthorized**. In case an Authentication dialog pop-up, do not provide any credentials.

Exercise 6: Enforcing Security Policy

The screenshot shows the RESTClient interface. In the 'Request' tab, a GET method is selected with the URL <http://daetrain24496:5555/gateway/SearchCruise/1.0/cruises>. The 'Headers' section contains the following entries:

Header	Value
Content-Type	application/json

In the 'Response' tab, the status code is 401 Unauthorized, and the response body is:

```
HTTP/1.1 401 Unauthorized
Basic Realm = Integration Server
```

9. Provide an additional variable for Authentication.

a. **Authentication:**

i. **Basic Authentication:** Sumala/manage

Run the request pressing the **Send** button.

The screenshot shows the RESTClient interface. In the 'Request' tab, a GET method is selected with the URL <http://daetrain25856:5555/gateway/SearchCruise/1.0/cruises>. The 'Headers' section contains the following entries:

Header	Value
Accept	application/json

The 'Send' button is highlighted with a red box. In the 'Response' tab, the status code is 200 OK, and the response body is:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 125
Date: Mon, 12 Mar 2018 10:45:20 GMT
Server: Apache-Coyote/1.1

[{"cruise": {"cruiseID": "00001", "title": "Alaska Whale Watching Photo Expedition", "description": "Photographing the Whales in Alaska is a unique experience. Join us on this tour to this unique area. The cruise is a luxury tour for those that want to spend time with others with a like for sea these unique animals of the seas. The tour will be conducted by a pro photography who will work with each of the members of the tour as needed.", "startPort": "Juneau, Alaska", "endPort": "", "startDate": "20160820", "endDate": "20160827", "roomTypes": [{"roomID": "IFD"}]}}
```

This page intentionally left blank

Software AG
Internal Use Only!

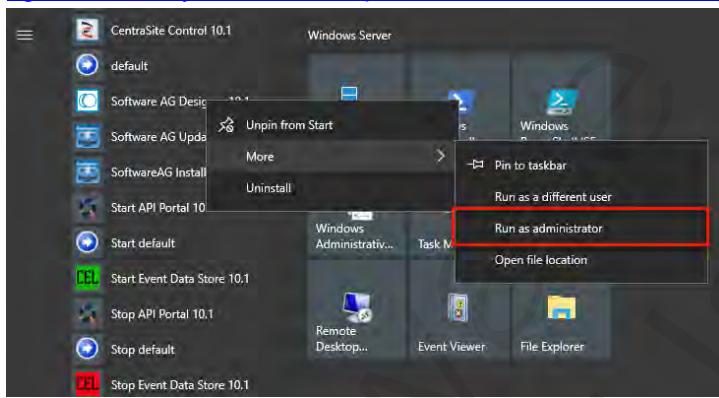
Exercise 7: Using a WSS Security Token

Objectives

In this exercise, you will define a security run-time policy using **WSS Username token** and apply it to a virtual service running on API Gateway.

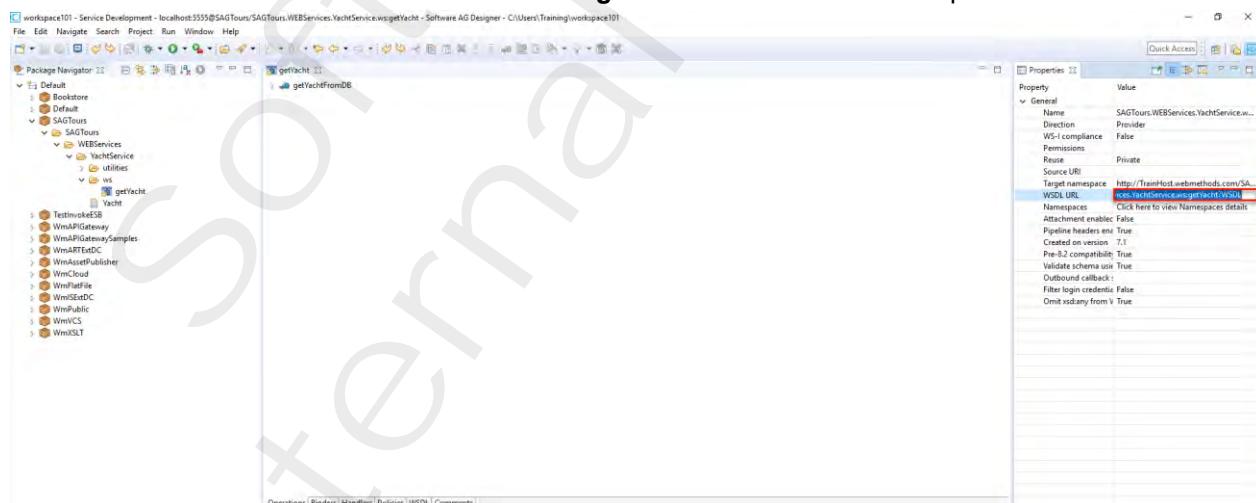
Steps

1. Open Windows Services UI to double check that the following services, needed for API Gateway, are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1**
 - b. **Software AG Event Data store 10.1 – should be started together with IS**
2. Open Software AG Designer 10.1 from the Windows Start Menu > Software AG > Tools. Use the right mouse key and use the option Run as administrator to start the Designer.



In case you are asked for a **secure storage password**, provide the password **manage**.

- 2.3. In the Package Navigator open the package **SAGTours** and edit the **SAGTours.WEBServices.YachtService.ws:getYacht** Web Service Descriptor.



In the Properties, copy the Value of WSDL URL and paste it into the Import URL field in API Gateway. The value pasted should be similar to the following:

**http:// sagbase.eur.ag.sag:5555/ws/SAGTours.WEBServices.YachtService.ws:
getYacht?WSDL**

1. Login to the **API Gateway UI** as user **Sumala/manage** and create a new SOAP based API. Use the option **Import API from URL**.

Exercise 7:
Using a WSS Security Token

- a. **URL:** - as mentioned above -
- b. **Type:** WSDL
- c. **Version:** 1.0

Create API
Create an API by importing from a file, URL or start from scratch 

Lets Get Started!

Import API from file
Create an API by importing API from a specified file.

Import API from URL
Create an API by importing it from an URL.

URL*
r.ad.sag:5555/ws/SACTours.WEBServices.YachtService.ws:getYacht?WSDL

Protected

Name

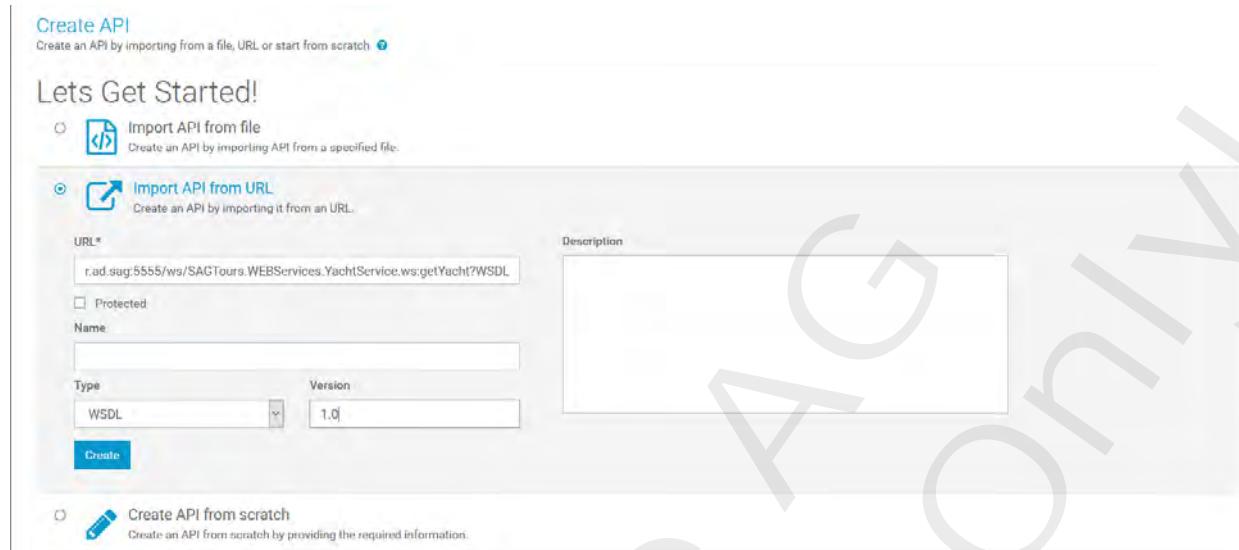
Type
WSDL

Version
1.0

Description

Create

Create API from scratch
Create an API from scratch by providing the required information.



Click **Create**.

2. Naviagte to **Policies > Identify & Access**. Switch to **Edit** mode.
3. Select **Inbound Authentication – Message** navigate to **Token Assertions**
- a. **Require WSS Username token:** - select -

Inbound Authentication - Message

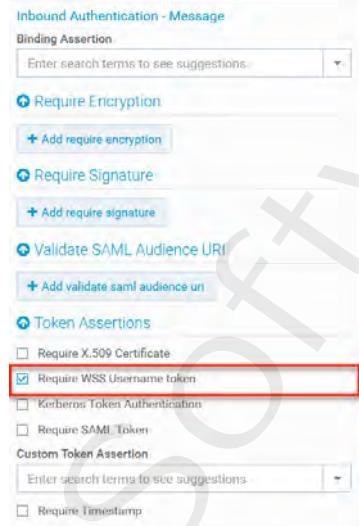
Binding Assertion
Enter search terms to see suggestions...

Require Encryption
+ Add require encryption

Require Signature
+ Add require signature

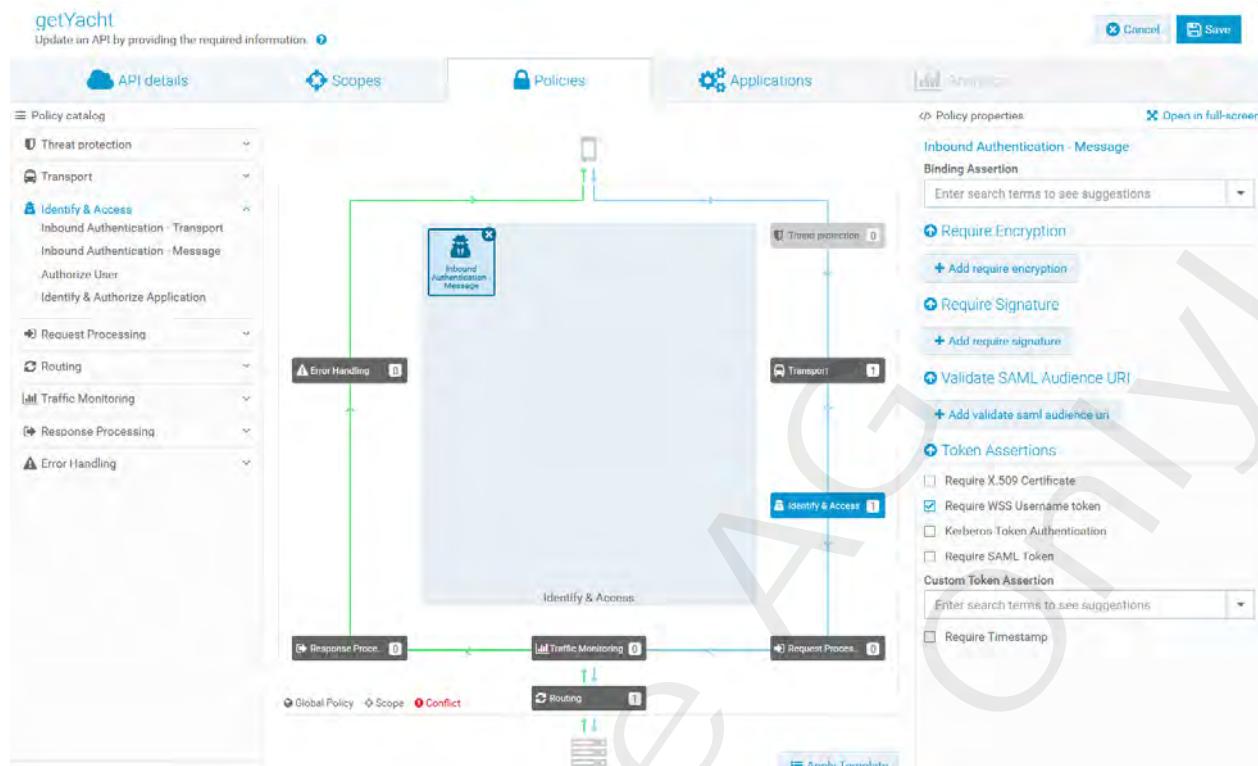
Validate SAML Audience URI
+ Add validate saml audience uri

Token Assertions
Require X.509 Certificate
 Require WSS Username token
Kerberos Token Authentication
Require SAMI Token
Custom Token Assertion
Enter search terms to see suggestions...
Require Timestamp



4. The policy action will identify the consumer from WSS Username

Exercise 7: Using a WSS Security Token



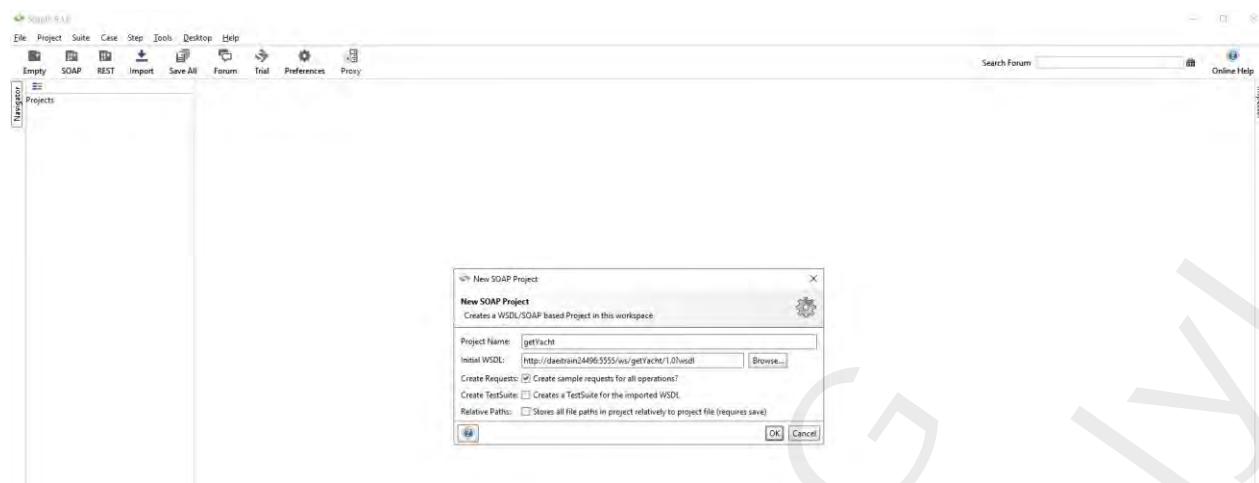
Click Save and Activate.

5. Navigate to **API details** and get the **Gateway endpoint**.

Native endpoint(s)	http://DAETRAIN00994.eu.ad.sag:5555/ws/SAGToursWEBServices/YachtService/waggerYacht/!SAGTours_WEBServices_YachtService_ws_getYacht_Port
Gateway endpoint(s)	http://DAETRAIN00994:5555/ws/getYacht/1

6. Open the **SOAP UI** web service test tool and create a new SOAP project. You will find SOAP UI within **Windows Start Menu > Tools > SOAP UI**. Select Projects, right mouse click, select New SOAP Project.
 - a. **Name:** getYacht
 - b. **Initial WSDL:** Copy the **Gateway Endpoint**. You will get this information out of **API Gateway > APIs > getYacht > Technical information**. And add to the URL: **?wsdl**

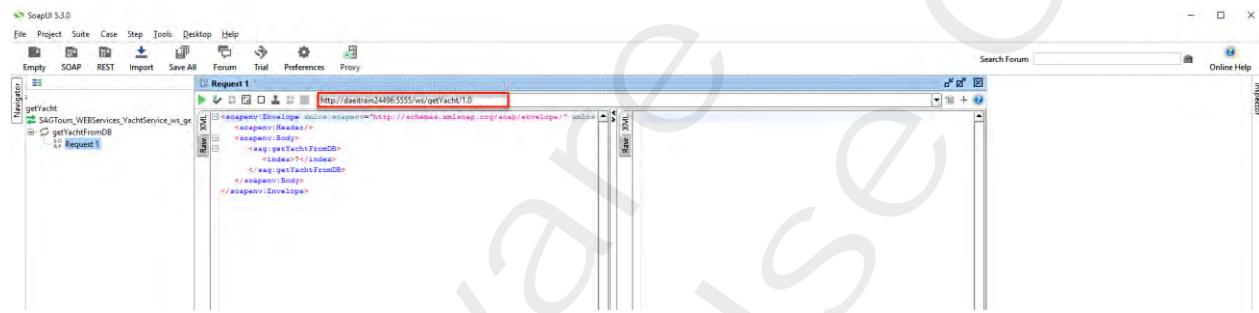
Exercise 7: Using a WSS Security Token



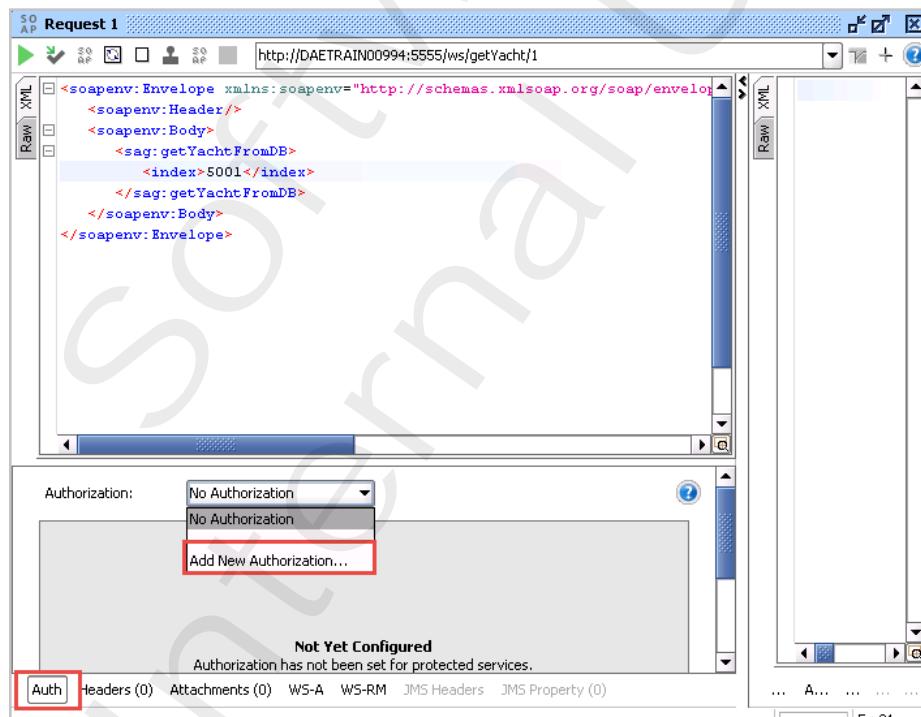
Click **Ok**.

A new SOAP Project will be created. Browse to the operation **getYachtFromDB**. You will see a pre-generated request named **Request 1**. Open the request.

7. A new SOAP Project will be created. Browse to the operation **getYachtFromDB**. You will see a pre-generated request named **Request 1**. Open the request.

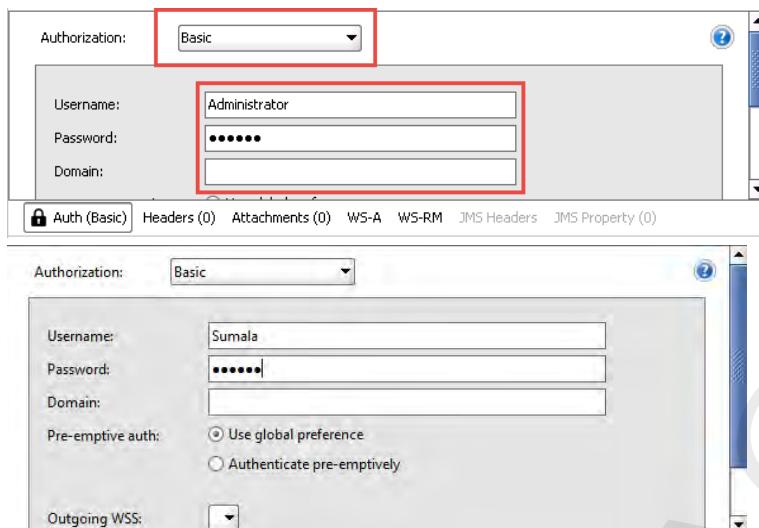


8. Within the SOAP envelope (soapenv) change the <index> value from ? to 5001. Add New Authorization.



- Type:** Basic
- Username:** Sumala
- Password:** manage

Exercise 7: Using a WSS Security Token

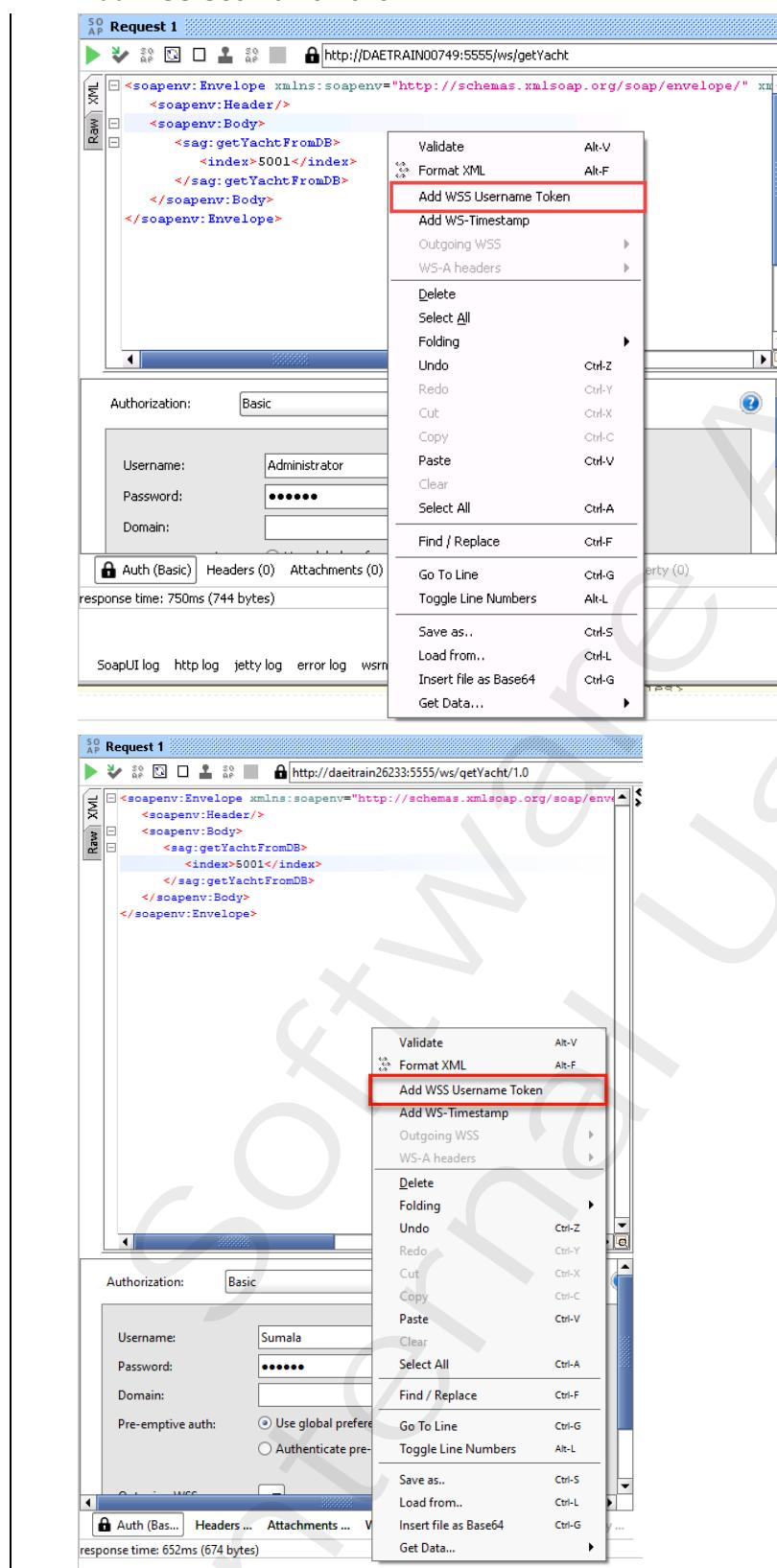


- Run the request by hitting the green triangle in the header lien of the Request. It will fail. Look at the error message in the response.

The screenshot shows two separate browser windows, each displaying a failed SOAP request to the same endpoint. The top window has a URL of `http://daetrain2633:5555/ws/getYacht/1.0`. The bottom window has a URL of `http://daetrain24496:5555/ws/getYacht/1.0`. Both requests are identical, containing a `<soapenv:Envelope>` block with a `<soapenv:Body>` section that includes a `<sag:getYachtFromDB>` operation. The responses from both servers are identical, showing a `<soapenv:Fault>` element with a `faultcode` of `wsse:MissingSecurityHeader` and a `faultstring` of `API Gateway encountered an error. Error Message: Missing wsse:Security header in request.`. A red box highlights this error message in both responses.

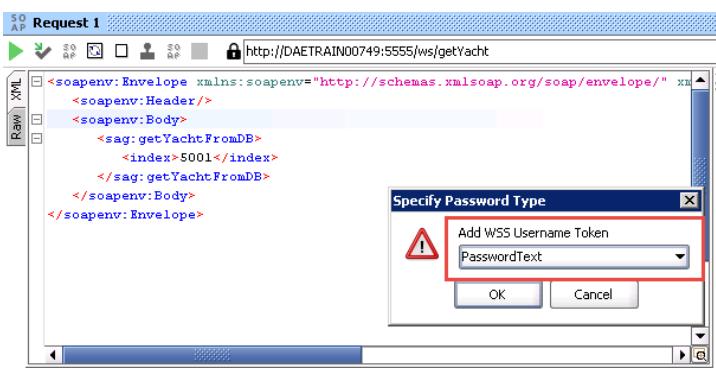
Exercise 7:
Using a WSS Security Token

10. To run a service that has a policy using WS token based identification, you have to add WSS Username Token. Select the Request Body section and press right mouse button. From the list select Add WSS Username Token.



11. Select Password Type: PasswordText.

Exercise 7:
Using a WSS Security Token



12. This will use credentials define in Basic Authentication to create a WSS Username Token. This wsse:security definition will be added to the Request Body in the soapenv:Header section.

13. Rerun the request. The SOAP response shows the result and is successful.

Exercise 7:
Using a WSS Security Token

The screenshot shows two separate requests in a SOAP UI tool, both targeting the endpoint `http://DAETRAIN0094:5555/ws/getYacht/1`.

Request 1 (Top):

- Authorization:** Basic
- Username:** Administrator
- Password:** *****

Request XML (Left):

```
<wsse:UsernameToken wsu:Id="UsernameToken-1">
    <wsse:Username>Administrator</wsse:Username>
    <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#PasswordText">*****</wsse:Password>
    <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#BinaryToken">2017-05-04T14:31:23.828Z</wsse:Nonce>
</wsse:UsernameToken>
```

Response XML (Right):

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ser-root="http://TrainHost.webmethods.com/TrainHost.YachtService.WEBMethods" xmlns:yacht="http://TrainHost.webmethods.com/TrainHost.YachtService.WEBMethods">
    <SOAP-ENV:Body>
        <ser-root:getYachtFromDBResponse xmlns:ser-root="http://TrainHost.webmethods.com/TrainHost.YachtService.WEBMethods">
            <yacht>
                <name>SANTA PAOLA</name>
                <ownerid>42</ownerid>
                <properties>
                    <description>Made in Britannia. Especially tailored for families. Many fridges for the kids.</description>
                    <type>JACK DANIELS</type>
                    <length>1300</length>
                    <width>300</width>
                    <draft>120</draft>
                    <sailsurface>114</sailsurface>
                    <motor>56</motor>
                    <headroom>200</headroom>
                    <bunks>5</bunks>
                </properties>
            </yacht>
        </ser-root:getYachtFromDBResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Request 2 (Bottom):

- Authorization:** Basic
- Username:** Sumala
- Password:** *****

Request XML (Left):

```
<wsse:UsernameToken wsu:Id="UsernameToken-2">
    <wsse:Username>Sumala</wsse:Username>
    <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#PasswordText">*****</wsse:Password>
    <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#BinaryToken">2018-03-08T11:37:00.012Z</wsse:Nonce>
    <wsu:Created>2018-03-08T11:37:00.012Z</wsu:Created>
</wsse:UsernameToken>
```

Response XML (Right):

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ser-root="http://TrainHost.webmethods.com/TrainHost.YachtService.WEBMethods" xmlns:yacht="http://TrainHost.webmethods.com/TrainHost.YachtService.WEBMethods">
    <SOAP-ENV:Body>
        <ser-root:getYachtFromDBResponse xmlns:ser-root="http://TrainHost.webmethods.com/TrainHost.YachtService.WEBMethods">
            <yacht>
                <name>SANTA PAOLA</name>
                <ownerid>42</ownerid>
                <properties>
                    <description>Made in Britannia. Especially tailored for families. Many fridges for the kids.</description>
                    <type>JACK DANIELS</type>
                    <length>1300</length>
                    <width>300</width>
                    <draft>120</draft>
                    <sailsurface>114</sailsurface>
                    <motor>56</motor>
                    <headroom>200</headroom>
                    <bunks>5</bunks>
                </properties>
            </yacht>
        </ser-root:getYachtFromDBResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

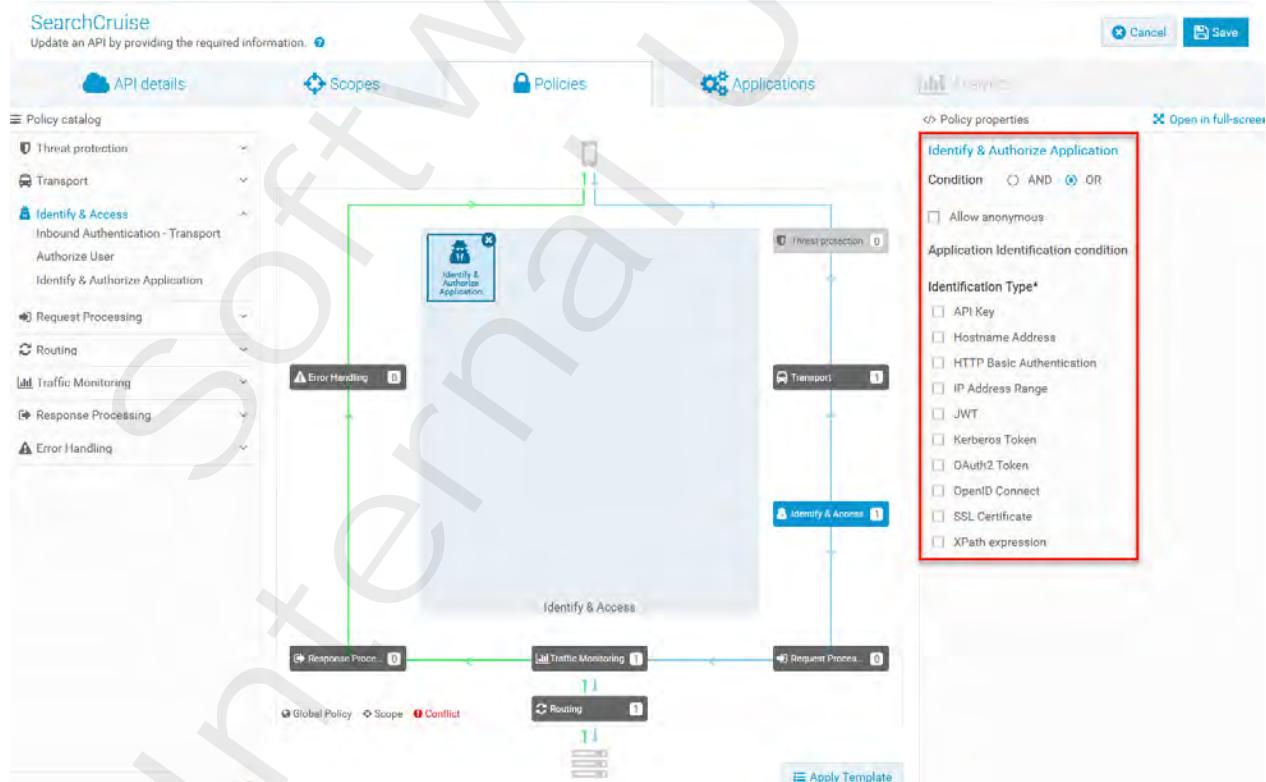
Exercise 8: Creating a Consumer Application

Objectives

In this exercise, you will learn how to automatically provision and enforce a consumer relationship in API Gateway.

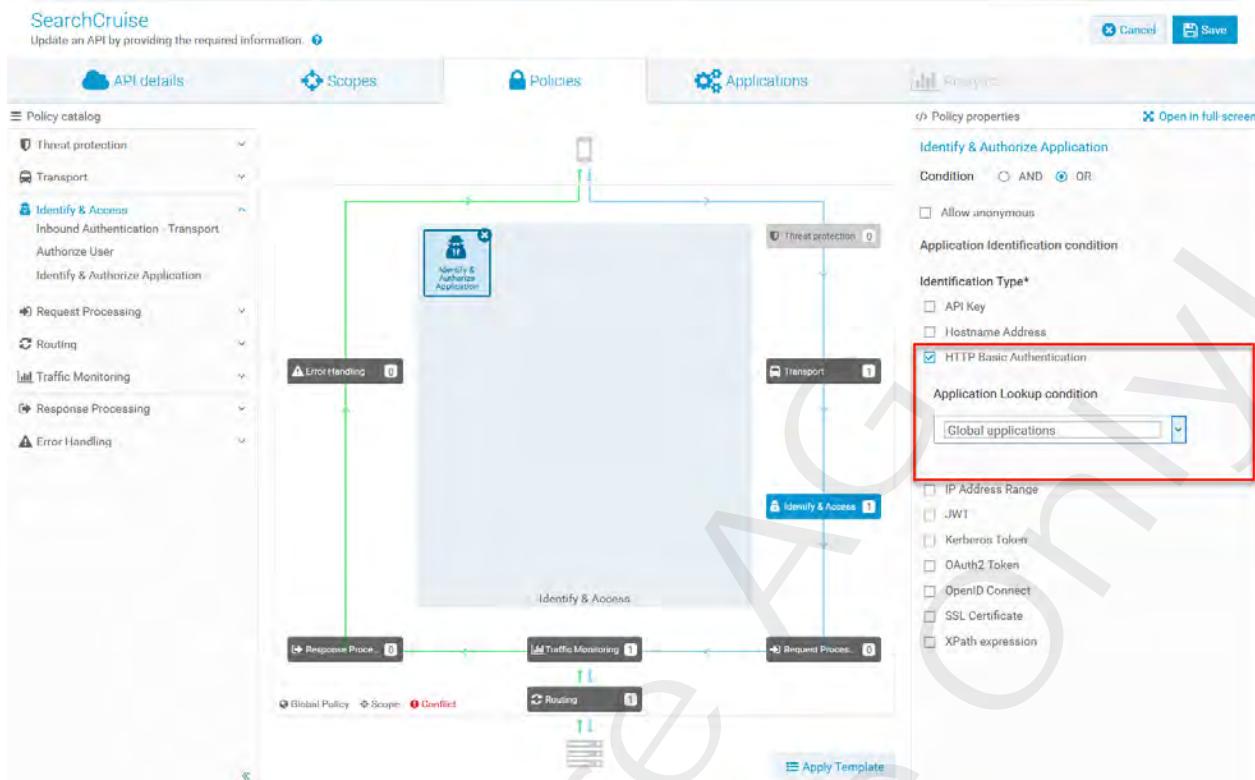
Steps

1. Open Windows Services UI to double check that the following services, needed for API Gateway are up and running. If service is not running, start the service.
 - a. Software AG Integration Server 10.1
 - b. Software AG Event Data store 10.1 – should be started together with IS
 - c. Software AG Runtime 10.1
2. Login to the **API Gateway UI** as user **Sumala/manage**. Search for the **SearchCruise** API. Deactivate the API in order to change the policy configuration. Click the **Deactivate** button.
Click **Yes**.
 - a. Click **Edit** to open API in Edit mode.
 - b. Navigate to **Policies > Identify & Access**.
 - c. Remove the **Inbound Authentication – Transport** Policy action.
3. Click on **Identity & Authorize Application** in **Policy catalog** section. This will move the policy action to the section **Policy properties**.



- d. In **Policy properties** select:
 - i. **Identification Type:** HTTP Basic Authentication
 - ii. **Application Lookup condition:** Global applications

Exercise 8:
Creating a Consumer Application



Click Save. Click Activate.

- In the API Gateway header select the **Applications** to create a consumer application.

The screenshot shows the "Applications" tab selected in the API Gateway header. The "Manage applications" section displays a table with columns for Name, Description, and Version. A "Create application" button is visible on the right.

- Click **Create Application**. Provide the following properties.

- d. **Name:** MyFirstApplication
- e. **Version:** 1.0
- f. **Description:** Global consumer application based on user identifiers

The screenshot shows the "Create application" form for "MyFirstApplication". The "Basic information" tab is selected, showing fields for Name (MyFirstApplication), Version (1.0), and Description (Global consumer application based on user identifiers). The "Identifiers" tab is also visible. A "Continue to Identifiers" link is at the bottom.

- Click **Continue to Identifiers**. Within **Identifiers** select **Other identifiers** and do the following configuration:

- g. **Username:** Sumala
- Click **Add**.

Exercise 8: Creating a Consumer Application

The screenshot shows the 'MyFirstApplication' consumer application configuration page. It includes sections for 'Basic information', 'Identifiers', 'APIs', 'OAuth2 Credentials', 'IP address range', 'Partner identifier', 'Client certificates', 'Claims' (with a red box highlighting the 'Add claims set' button), and 'Other identifiers'. A table under 'Other identifiers' shows a single entry: 'Username' (Sumala). At the bottom, there is a 'Continue to APIs' button.

Click **Save**.

7. Open **Rest Client** and run request against **SearchCruise** using **Basic Authentication** set to **Sumala/manage**

The screenshot shows the RESTClient interface with a 'Request' tab selected. The method is set to 'GET', the URL is 'http://daetrain25856:5555/gateway/SearchCruise/1.0/cruises', and the 'Basic authentication' field contains 'Sumala/manage'. The 'Response' tab shows a JSON response with one cruise entry:

```
1 <!
2 < "cruises": [
3   < "cruiseID": "100001",
4   < "title": "Alaska whale Watching Photo Expedition",
5   < "description": "Photographing the Whale in Alaska is a unique experience. Join us on this tour to this unique area. The cruise is a luxury tour for this that what to spend time with others with a like for see these unique animals of the sea. The tour will be conducted by a pro photography who will work with each of the members of the tour as needed.",
6   < "startPort": "Juneau, Alaska",
7   < "numDays": "8",
8   < "startDate": "20150820",
9   < "endDate": "20150827",
10  < "roomTypes": [
11    < "roomID": "100",
12    < "roomDetails": {
```

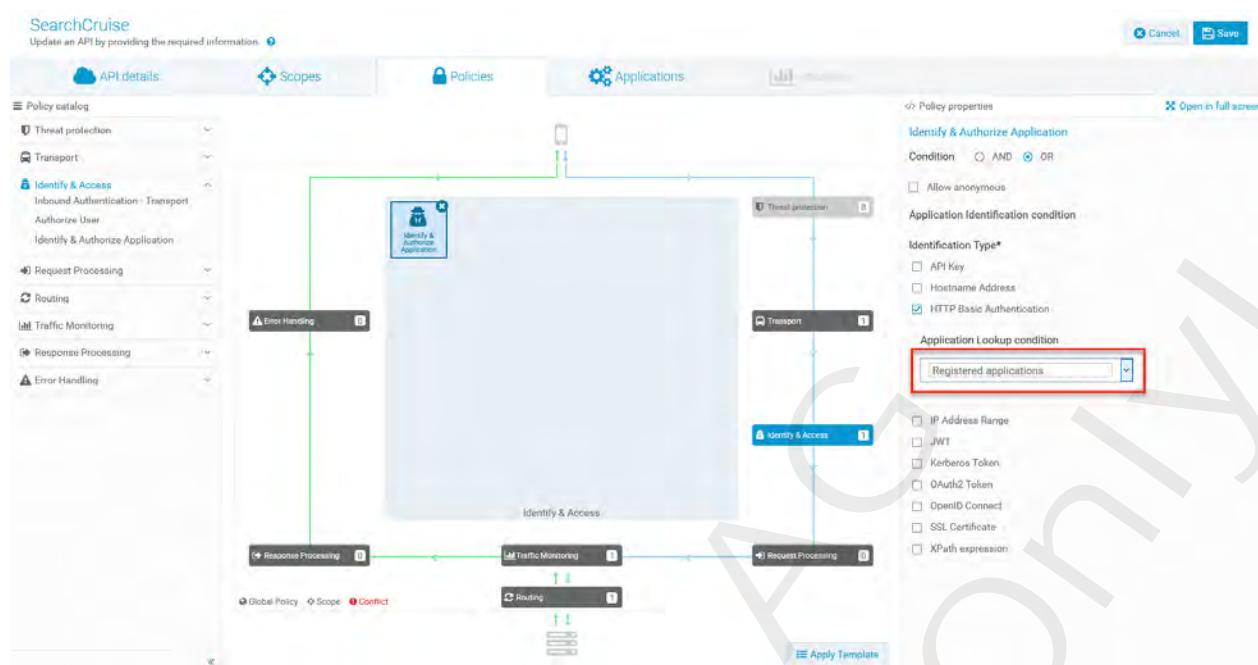
8. Change the user credentials to **Administrator/manage**. Administrator is a valid user in Integration Server, but he is not member of the application we have just created. Therefore the request fails.

The screenshot shows the RESTClient interface with a 'Request' tab selected. The method is set to 'GET', the URL is 'http://daetrain24496:5555/gateway/SearchCruise/1.0/cruises', and the 'Basic authentication' field contains 'Administrator/manage'. The 'Response' tab shows an error message:

```
{"exception":"API Gateway encountered an error. Error Message: Unauthorized to identify the application for the request. Request Details: Service - SearchCruise, Operation - /cruises, Invocation Time:17:02 PM, Date:Jan 12, 2015, Client IP = 0:0:0:0:0:0:0:1, User = Default and ApplicationAdminUser"}
```

9. Now we want to enforce a close relationship between the API and the Consumer. Therefor we change the Application Lookup condition to Registered Application. Go back to **API Gateway**. Deactivate the **SearchCruise API** and switch to **Edit mode**. Navigate to the **Policies > Identify & Authorize Application**. Within the **Policy properties** change the **Application Lookup condition** as following:
- Application Lookup condition: Registered applications

Exercise 8: Creating a Consumer Application



Save the changes and activate the API.

10. Go back to **REST Client** and try to rerun the **SearchCruise API** using credentials for **Sumala/manage**. The request will fail, because the application we have created is not registered or assigned to the **SearchCruise API**.

The REST Client interface shows a request to 'http://daethain24496:5555/gateway/SearchCruise/v1/cruises'. The response header indicates a 401 Unauthorized status. The response body contains an error message: 'Exception: API Gateway encountered an error. Error Message: Unable to identify the application for the request. Request Details: Service - SearchCruise, Operation - /cruises, Invocation Time: 4:33:25 PM, Date: Jan 12, 2018, Client IP - 310:0:0:0:0:0:0:1, User - Default and ApplicationName:'. A red box highlights the error message in the response body.

11. We change the global application to be an application for the **SearchCruise API**. Navigate to **Applications** in **API Gateway** header. Select the application **MyFirstApplication**. Switch to **Edit** mode.
12. From the navigation panel under **Application Details** select **APIs**

The screenshot shows the 'MyFirstApplication' details page. The 'APIs' tab is selected. The left sidebar has options: Basic Information, Identifiers, APIs (which is selected), and OAuth2 Credentials. The right side shows a 'Find APIs' search bar and a 'Selected APIs' section which currently says 'No APIs selected for registration.' There are 'Cancel' and 'Save' buttons at the top right.

13. Click the Search icon and select the **SearchCruise API**.

Exercise 8:
Creating a Consumer Application

The screenshot shows the 'MyFirstApplication' configuration interface. The 'APIs' tab is selected. In the 'Search results' section, the 'SearchCruise' API is listed with a red box around it. The 'Selected APIs' section below it also contains a red box around the 'SearchCruise' entry.

14. Now the **SearchCruise API** is added to the application.

The screenshot shows the 'MyFirstApplication' configuration interface. The 'APIs' tab is selected. A red box highlights the 'Selected APIs' section, which lists the 'SearchCruise' API.

15. Click **Save**. Go back to **SearchCruise API** and verify that the application is also seen from the **SearchCruise API**.

The screenshot shows the 'WEBMETHODS API Gateway' interface. The 'APIs' tab is selected. A red box highlights the 'Applications' tab in the top navigation bar. Below, a table titled 'Selected applications' shows one entry: 'MyFirstApplication' with a red box around it.

16. Rerun the request against **SearchCruise API** in **RESTClient**. Now the user **Sumala/manage** can run the request.

Exercise 8: Creating a Consumer Application

The screenshot shows a RESTClient interface with a "Request" tab selected. The URL is `http://dawtrain24496:5555/gateway/SearchCruise/1/cruises`. The response body contains the following JSON data:

```
[{"cruises": [{"cruiseID": "00001", "title": "Alaska Whale Watching Photo Expedition", "description": "Photographing the whale in Alaska is a unique experience. Join us on this tour to this unique area. The cruise is a luxury tour for those that want to spend time with others with a like for sea these unique animals of the sea. The tour will be conducted by a pro photography who will work with each of the members of the tour as needed.", "startPort": "Juneau, Alaska", "numDays": "5", "startDate": "20150820", "endDate": "20150827", "roomTypes": [{"roomID": "1FD", "roomDetails": {"roomType": "Inside Premier Double", "title": "Inside Premier Double cabin", "description": "An Eco-friendly double room with desk, TV and Shower", "maxOccupants": "2", "totalRoomsOfType": "3"}, {"roomID": "OPD", "roomDetails": {"roomType": "Seaview Premier Double", "title": "Premier Double cabin with seaview", "description": "An Eco-friendly double room with seaview, desk, TV and Shower", "maxOccupants": "2", "totalRoomsOfType": "3"}, {"roomID": "SDD", "roomDetails": {"roomType": "Seaview Deluxe Room", "title": "Deluxe cabin with seaview", "description": "For those wanting that extra bit of luxury. The Deluxe cabins come with a private balcony, desk, TV and Shower", "maxOccupants": "2", "totalRoomsOfType": "2"}, {"roomID": "SCDD", "roomDetails": {"roomType": "Seaview Double Deluxe Room", "title": "Deluxe cabin with 2 double beds and a seaview", "description": "For those wanting that extra bit of luxury. The Double Deluxe cabins come with 2 double beds a private balcony, desk, TV and Shower", "maxOccupants": "4", "totalRoomsOfType": "5"}, {"roomID": "SSZ", "roomDetails": {"roomType": "Seaview Junior Suite", "title": "2 bedroom suite with seaview", "description": "The Junior suite cabins come with 1 bedrooms, a separate sitting area, a private balcony, TVs in each room and Shower", "maxOccupants": "4", "totalRoomsOfType": "1"}, {"roomID": "S3Z", "roomDetails": {"roomType": "Seaview Junior Suite", "title": "3 bedroom suite with seaview", "description": "The Deluxe suite cabins come with 3 bedrooms, a spacious entertaining room, a private balcony with hot tub, TVs in each room and Bath with Jacuzzi", "maxOccupants": "4", "totalRoomsOfType": "1"}, {"roomID": "SDS", "roomDetails": {"roomType": "Seaview Double Room", "title": "Deluxe cabin with 2 double beds and a seaview", "description": "For those wanting that extra bit of luxury. The Double Deluxe cabins come with 2 double beds a private balcony, desk, TV and Shower", "maxOccupants": "4", "totalRoomsOfType": "5"}, {"roomID": "SSZ", "roomDetails": {"roomType": "Seaview Junior Suite", "title": "2 bedroom suite with seaview", "description": "The Junior suite cabins come with 1 bedrooms, a separate sitting area, a private balcony, TVs in each room and Shower", "maxOccupants": "4", "totalRoomsOfType": "1"}, {"roomID": "S3Z", "roomDetails": {"roomType": "Seaview Junior Suite", "title": "3 bedroom suite with seaview", "description": "The Deluxe suite cabins come with 3 bedrooms, a spacious entertaining room, a private balcony with hot tub, TVs in each room and Bath with Jacuzzi", "maxOccupants": "4", "totalRoomsOfType": "5"}], "cruiseID": "00001", "title": "Panama Canal", "description": "Cruise from San Diego to Mexico and on to the canal, then up to Miami. The canal is 77.1 Km (48 miles) long and connects the Atlantic Ocean with the Pacific Ocean. This cruise shows a side of the canal that is unique for those that want to see it in luxury.", "startPort": "San Diego", "endDate": "14", "startDate": "20150710", "roomTypes": [{"roomID": "1FD", "roomDetails": {"roomType": "Inside Premier Double", "title": "Inside Premier Double cabin", "description": "An Eco-friendly double room with desk, TV and Shower", "maxOccupants": "2", "totalRoomsOfType": "3"}, {"roomID": "OPD", "roomDetails": {"roomType": "Seaview Premier Double", "title": "Premier Double cabin with seaview", "description": "An Eco-friendly double room with seaview, desk, TV and Shower", "maxOccupants": "2", "totalRoomsOfType": "3"}, {"roomID": "SDD", "roomDetails": {"roomType": "Seaview Deluxe Room", "title": "Deluxe cabin with seaview", "description": "For those wanting that extra bit of luxury. The Deluxe cabins come with a private balcony, desk, TV and Shower", "maxOccupants": "2", "totalRoomsOfType": "2"}, {"roomID": "SCDD", "roomDetails": {"roomType": "Seaview Double Deluxe Room", "title": "Deluxe cabin with 2 double beds and a seaview", "description": "For those wanting that extra bit of luxury. The Double Deluxe cabins come with 2 double beds a private balcony, desk, TV and Shower", "maxOccupants": "4", "totalRoomsOfType": "5"}, {"roomID": "SSZ", "roomDetails": {"roomType": "Seaview Junior Suite", "title": "2 bedroom suite with seaview", "description": "The Junior suite cabins come with 1 bedrooms, a separate sitting area, a private balcony, TVs in each room and Shower", "maxOccupants": "4", "totalRoomsOfType": "1"}, {"roomID": "S3Z", "roomDetails": {"roomType": "Seaview Junior Suite", "title": "3 bedroom suite with seaview", "description": "The Deluxe suite cabins come with 3 bedrooms, a spacious entertaining room, a private balcony with hot tub, TVs in each room and Bath with Jacuzzi", "maxOccupants": "4", "totalRoomsOfType": "5"}, {"roomID": "SDS", "roomDetails": {"roomType": "Seaview Double Room", "title": "Deluxe cabin with 2 double beds and a seaview", "description": "For those wanting that extra bit of luxury. The Double Deluxe cabins come with 2 double beds a private balcony, desk, TV and Shower", "maxOccupants": "4", "totalRoomsOfType": "5"}, {"roomID": "SSZ", "roomDetails": {"roomType": "Seaview Junior Suite", "title": "2 bedroom suite with seaview", "description": "The Junior suite cabins come with 1 bedrooms, a separate sitting area, a private balcony, TVs in each room and Shower", "maxOccupants": "4", "totalRoomsOfType": "1"}, {"roomID": "S3Z", "roomDetails": {"roomType": "Seaview Junior Suite", "title": "3 bedroom suite with seaview", "description": "The Deluxe suite cabins come with 3 bedrooms, a spacious entertaining room, a private balcony with hot tub, TVs in each room and Bath with Jacuzzi", "maxOccupants": "4", "totalRoomsOfType": "5"}]}]
```

Exercise 9: Enforcing a Consumer Relationship using API Key

Objectives

In this exercise, you will learn how to enforce a valid APIKey for the SearchCruise API. For application creation and registering an application we will enforce the Approval process.

Steps

1. Open Windows Services UI to double check that the following services, needed for API Gateway, are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1**
 - b. **Software AG Event Data store 10.1 – should be started together with IS**
 - c. **Software AG Runtime 10.1**
 - d. **hMail-Server**
2. Login to the **API Gateway UI** as user **Administrator/manage** and navigate to **Administration > Approval Configuration**.

The screenshot shows the API Gateway UI with the following details:

- Header:** WEBMETHODS API Gateway, APIs, Policies, Applications, Packages, Administrator
- Left sidebar:** Home, Administration, General, Load balancer, Extended settings, API fault, Approval configuration (highlighted with a red box), Create application, Register application, Update application, Subscribe package, Outbound proxy, URL aliases.
- Top navigation:** General, Security, Destinations, System settings.
- Approval configuration section:**
 - Create application:** Configure the approval workflow for creating a new application.
 - Enable:** Checked.
 - Approvers:** Administrators.
 - Approved by:** Anyone.
 - Configure approval initiate request mail template to be sent to approver:**
 - Send notification:** Selected.
 - Subject:** Approval request pending.
 - Content:** Hello @approver.name,
A request by @requestor.name to @event.type needs your review and approval.
Best Regards,
API Gateway Team
*** This notification was sent automatically. Do not reply to this email.***
 - Configure request approved mail template to be sent to requester:**
- Buttons:** Cancel, Save.

3. Within **Create application** provide the following properties
 - a. **Enable:** - turned on -
 - b. **Approvers:** Approvers
 - c. **Approved by:** Anyone
 - d. **Configure approval initiate request mail template to be sent to approver**
 - i. **Send notification:** - select -
 - ii. **Subject:** - keep the defaults -
 - iii. **Content:** - keep the defaults -
 - e. **Configure request approval mail template to be sent to requester**
 - i. **Send notification:** - select -
 - ii. **Subject:** - keep the defaults -
 - iii. **Content:** - keep the defaults -
 - f. **Configure rejection mail template to be sent to requester**
 - i. **Send notification:** - select -

Exercise 9: Enforcing a Consumer Relationship using API Key

- ii. **Subject:** - keep the defaults -
- iii. **Content:** - keep the defaults -

Administration
Implement and manage the general and security related configurations for API Gateway.

General Security Destinations System settings

Load balancer Extended settings API fault Approval configuration Create application Register application Update application Subscribe package Outbound proxy URL aliases

Create application

Configure the approval workflow for creating a new application.

Enable Approvers Approved by

Configure approval initiate request mail template to be sent to approver

Send notification Subject Approval request pending Content

Hello @approver.name,
A request by @requestor.name to @event.type needs your review and approval.
Best Regards,
API Gateway Team
*** This notification was sent automatically. Do not reply to this email.***

Configure request approved mail template to be sent to requester

Cancel Save

Click **Save**.

4. Do the same configuration for the workflows **Register application** and **Update application**.
5. Login to the **API Gateway UI** as user **Sumala/manage** and create a new application. Navigate to Applications, click **Create application**. Provide the following properties:

- a. **Name:** MyApplicationJustAPIKey
- b. **Version:** 1.0
- c. **Description:** Application following the Approval workflow and without identifiers. Just the API Key is identifying the consumer.
- d. **Requester comment:** Application requested for 3rd party access

Click **Continue to Identifiers**.

Create application
Create an application by providing the basic information, defining identifiers, and adding the required APIs.

Application details

Basic information Identifiers APIs

IP address range + Add

Partner identifier

Client certificates Browse + Add

Claims Add claims set

Other identifiers Hostname + Add

Continue to APIs >

Cancel Save

Click **Continue to APIs**. Provide the following properties:

- e. **Selected APIs:** SearchCruise

Exercise 9:
Enforcing a Consumer Relationship using API Key

The screenshot shows the 'Create application' interface. In the 'Selected APIs' table, the row for 'SearchCruise' is highlighted with a red border. The 'Save' button is located at the top right of the form.

Click **Save**.

6. Review the request for creating this application. Navigate to **Sumala > Pending Requests**.

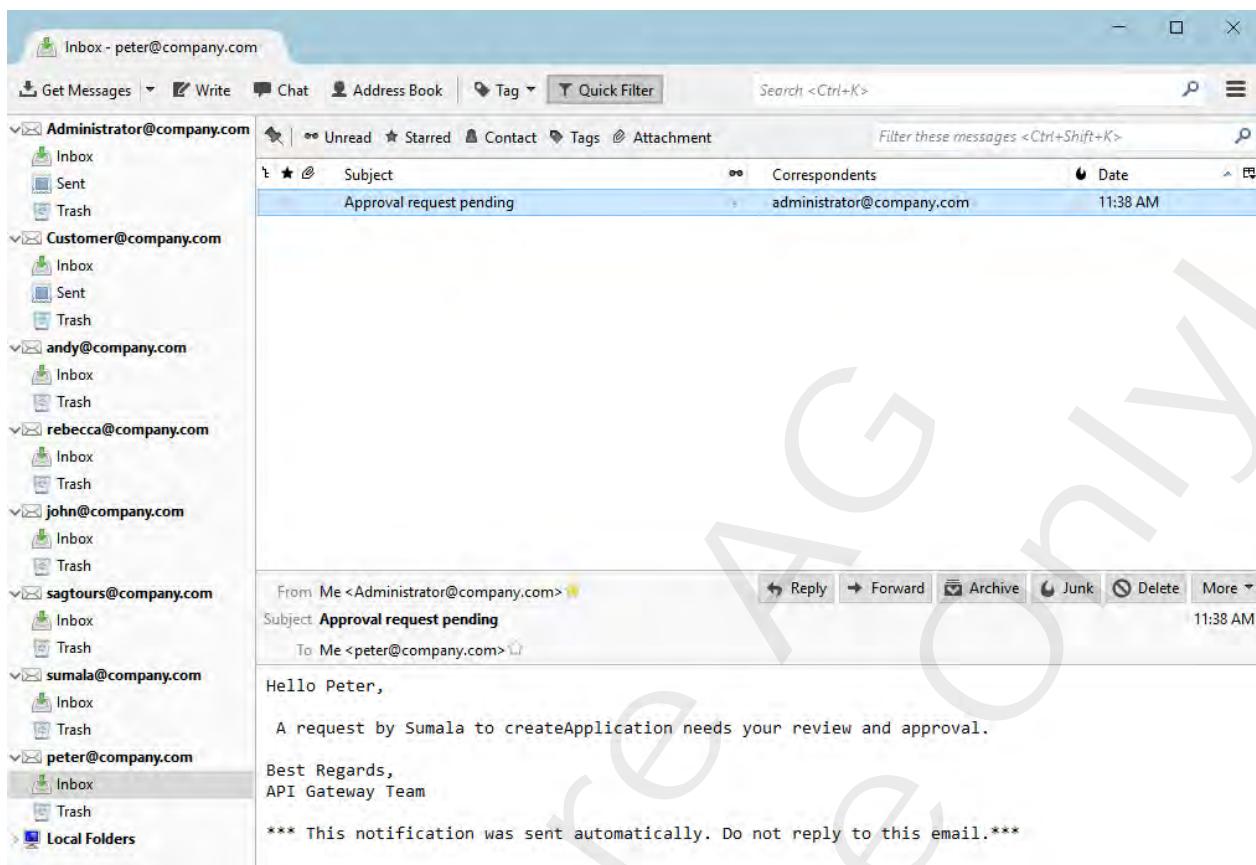
The screenshot shows the 'Manage applications' page. A new application named 'MyFirstApplication' is listed. On the right, the user 'Sumala' is logged in, and the 'Pending Requests' option in the dropdown menu is highlighted with a red border.

You will find **Sumala's** request within the tab **My requests**.

The screenshot shows the 'Pending Requests' page. It displays one pending request from 'Sumala' with a comment 'test'. The 'Create application' event is listed under Request details.

7. As configured in the approval workflow the user attached to the **Approver access profile** will get an email notification. In exercise 1 we have created the user **Peter** as user of the approver group with corresponding access profile. Open **Mozilla Thunderbird** and get messages for user **peter@company.com**.

Exercise 9:
Enforcing a Consumer Relationship using API Key



8. Logon to API Gateway as user Peter/manage.

The screenshot shows the 'Welcome to webMethods API Gateway' page. At the top, there's a navigation bar with links for WEBMETHODS API Gateway, APIs, Policies, Applications, and Packages. A user dropdown shows 'Peter'. The main area features three large cards: 'Analyze APIs' (with a bar chart icon), 'Manage APIs' (with a wrench icon), and 'Create API' (with a cloud icon). Below these cards, there's descriptive text about the API gateway's capabilities: 'Helps you leverage your internal assets by exposing them as simple APIs in a secure fashion, provides relief from threat protection and complex routing and traffic management problems, provides extensive mapping and transformation support, monitors performance metrics, and provides API analytics.'

9. Navigate to Peter > Pending requests. You will find Sumala's request to create the application within the tab Pending Requests. The requests are listed with the events definition.

The screenshot shows the 'Pending Requests' screen in the API Gateway administration interface. The top navigation bar includes links for Home, Administration, Pending Requests, APIs, Policies, Applications, and Packages. A red box highlights the 'Peter' user in the top right corner. The main area has tabs for 'My requests' and 'Pending Requests', with 'Pending Requests' being the active tab (also highlighted with a red box). Below the tabs, there's a table with columns: Requested by, Requestor comment, Event, and Request details. One row in the table shows a request from 'Sumala' with the event 'Create application' and a link to 'Create application request details'. At the bottom right of the table area, there are 'Approve' and 'Reject' buttons.

- a. Select the request **Create application** and click **Approve** to approve the create application request.

Exercise 9:
Enforcing a Consumer Relationship using API Key

Pending Requests
Manage your pending requests here. [?](#)

My requests Pending Requests

Requested by	Requestor comment	Event	Request details
Sumala	(test)	Create application	Create application request details

[Approve](#) [Reject](#)

- b. After the **Create application** request is approved, the **Register application** request is shown in the list of **Pending Requests**.

Pending Requests
Manage your pending requests here. [?](#)

My requests Pending Requests

Requested by	Requestor comment	Event	Request details
Sumala		Register application	Register application request details

[Approve](#) [Reject](#)

- c. Approve this request.

10. Navigate to **Applications**. The new created application is listed. Review the application **MyApplicationJustAPIKey**. Because Peter is not the owner of this application he doesn't see the access token definitions.

WEBMETHODS API Gateway

Home | Applications | MyApplicationJustAPIKey

MyApplicationJustAPIKey
View application details, identifiers, and access token information along with the APIs associated with the application. [?](#)

Application details

Basic information

Name	MyApplicationJustAPIKey
Version	1.0
Owner	
Created	2018-01-16 12:48:57 GMT

Description Application following the Approval workflow and without identifiers. Just the API Key is identifying the consumer

Identifiers

Access tokens

Scopes Name Description

APIs

Name	Description	Version
SearchCruise	Searches for all cruises, or searches for a cruise that matches...	1.0

11. Logon to **API Gateway** as user **Sumala/manage**. Navigate to the **Application Details** page of **MyApplicationJustAPIKey**. The **API Key** is created as part of the application.

Exercise 9: Enforcing a Consumer Relationship using API Key

MyApplicationJustAPKey
View application details, identifiers, and access token information along with the APIs associated with the application. [Edit](#)

Application details

Basic information

Name: MyApplicationJustAPKey
Version: 1.0
Owner:
Created: 2018-01-16 12:48:37 GMT
Description: Application following the Approval workflow and without identifiers. Just the API Key is identifying the consumer

Identifiers

Access tokens

API key: 1415d3f3-9a81-4bdd-a267-9ff8b40bc9eff

OAuth2 Credentials

Type: confidential
Client ID: f6e65217-2e51-4552-a148-d0c0de9217db
Client secret: df6b276d-7317-4869-a509-e0d00c16d3e0
Client name: MyApplicationJustAPKey-28c759e7-afac-40de-82ab-1e554af13fc4

5. Now we need to configure the security policy action for the service **SearchCruise**. Navigate to APIs, which lists the service **SearchCruise**. Click on **SearchCruise**.

WEBMETHODS API Gateway [Edit](#)

Home | Applications | MyApplicationJustAPKey

MyApplicationJustAPKey
View application details, identifiers, and access token information along with the APIs associated with the application. [Edit](#)

Application details

Basic information

API access key: 1415d3f3-9a81-4bdd-a267-9ff8b40bc9eff

OAuth2 Credentials

Type: confidential
Client ID: f6e65217-2e51-4552-a148-d0c0de9217db
Client secret: df6b276d-7317-4869-a509-e0d00c16d3e0
Client name: MyApplicationJustAPKey-28c759e7-afac-40de-82ab-1e554af13fc4

Scopes

Name	Description
0841fdff-0ec6-412c-a76f-ed2fe28f1a72	SearchCruise/1.0 / API Scope

Token lifetime: 3600
Token refresh limit: 0
Redirect URLs: https://placeholder_redirect_url

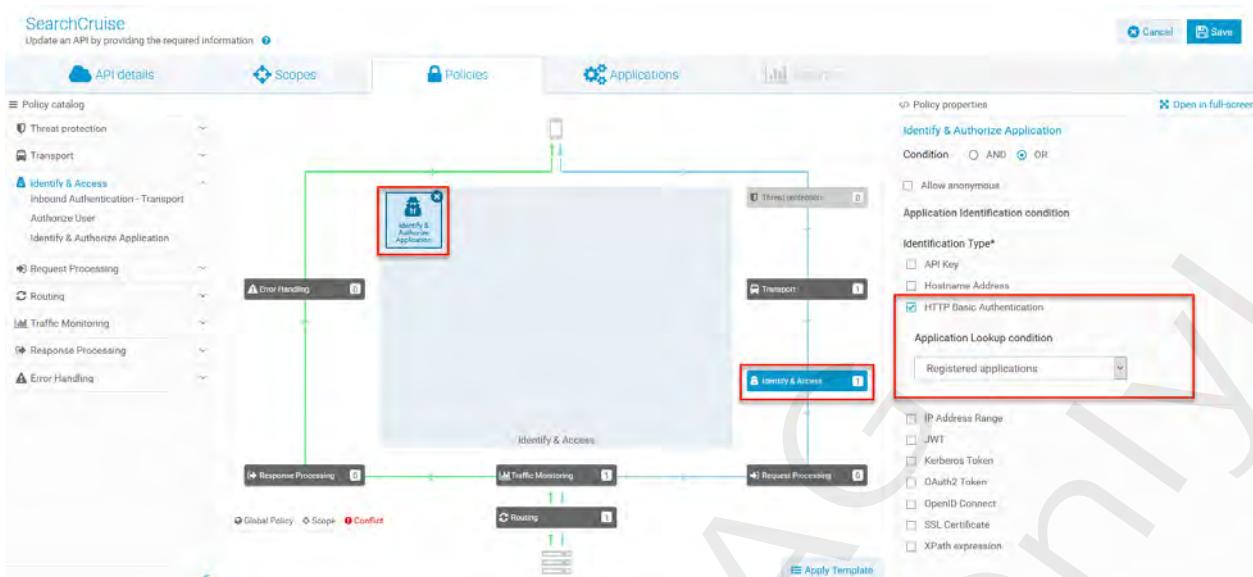
APIs

Name	Description	Version
SearchCruise	Searches for all cruises, or searches for a cruise that matches...	1.0

This will open the Detail View of the **SearchCruise** service.

6. Deactivate the API and switch to **Edit** mode. Navigate to **Policies -> Identify & Access > Identify & Authorize Application**

Exercise 9: Enforcing a Consumer Relationship using API Key

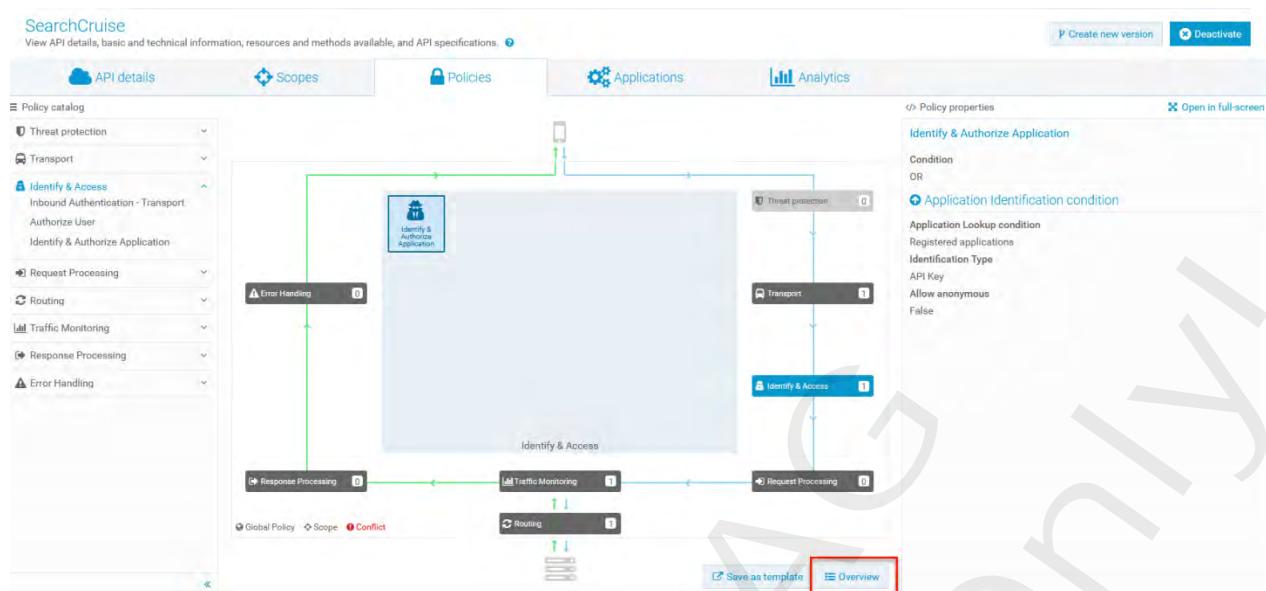


8. Within Policy properties > Identify & Authorize Application > Identification Type
 - a. Deselect HTTP Authentication
 - b. Select API Key



9. Save and activate the **SearchCruise** API.
10. Review all policy configurations by clicking the Overview icon.

Exercise 9: Enforcing a Consumer Relationship using API Key



11. The **Overview** page shows a list of policies for the overall API and for a single resource or method. Close the page by clicking the blue cross in the right hand upper corner.

The screenshot shows the 'Overview' page. At the top, there's a 'Close' button (blue cross). Below it, two sections are displayed: 'API level policies' and 'Resource or method level policies'.

API level policies:

Name	Description
SearchCruise	Searches for all cruises, or searches for a cruise that matches the specified criteria. The GET met..

Policies:

- Threat protection: 0 Policy(s) applied
- Transport: 0 Policy(s) applied
- Identify & Access: 1 Policy(s) applied
 - Identify & Authorize Application: Policy applied from API level policy
- Request Processing: 0 Policy(s) applied
- Traffic Monitoring: 1 Policy(s) applied
 - Log Invocation: Policy applied from API level policy
- Routing: 1 Policy(s) applied
 - Straight Through Routing

Resource or method level policies:

Select resource & method: cruises

cruises

Policies:

- Threat protection: 0 Policy(s) applied
- Transport: 1 Policy(s) applied
 - Require HTTP / HTTPS: Policy applied from API level policy
- Identify & Access: 1 Policy(s) applied
 - Identify & Authorize Application: Policy applied from API level policy
- Request Processing: 0 Policy(s) applied
- Traffic Monitoring: 1 Policy(s) applied
 - Log Invocation: Policy applied from API level policy
- Routing: 1 Policy(s) applied
 - Straight Through Routing

12. The API Key which need to be sent by the consumer when calling the **SearchCruise API** is part of the application registered to the **SearchCruise API**. Open **Application** tab to list all registered applications. Please open the application **MyFirstApplication** in Detail View. Hover your mouse over

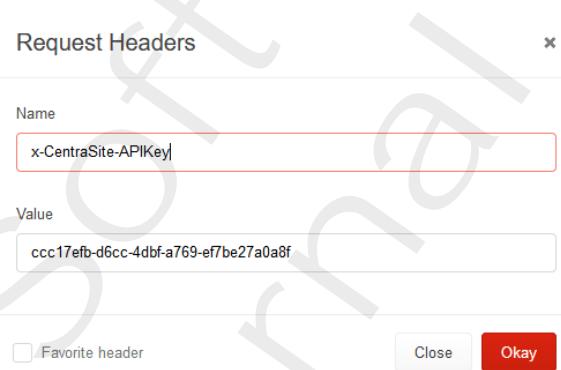
Exercise 9: Enforcing a Consumer Relationship using API Key

the little blue icon next to the API Key value. You will see a note how to pass along the API key.

The screenshot shows the 'Access tokens' section of the API Management interface. It displays the 'API access key' field with the value 'ccc17efb-d6cc-4dbf-a769-ef7be27a0a8f'. A tooltip above the field says 'Pass along this value as 'x-Gateway-APIKey' header.' Other fields shown include 'Client ID', 'Client secret', 'Client name', and 'Scopes'.

13. Open **REST Client**. Configure a **GET** request against our virtual service including the header parameter which provides the API Key. In addition to get the name of the header variable as mention in the previous step, you will find the name of the custom headers in the file C:\Training\APIGateway\Exercise9\SearchCruiseGET.txt.

- Methods:** GET
- URL:** Copy and paste the Access URL out of the **Consumer Overview** profile: <http://localhost:5555/gateway/SearchCruise/cruises>
- Custom Header:**
 - Accept:** application/json
 - x-Gateway-APIKey:** - value of API key as contained in the email or in API Consumer Profile, see above



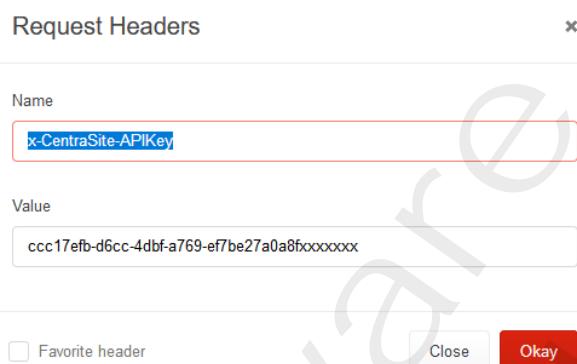
- d. Run the request using the Send button and verify the Response message. It should be the same result as before when running without **API Key** configuration.

Exercise 9:
Enforcing a Consumer Relationship using API Key

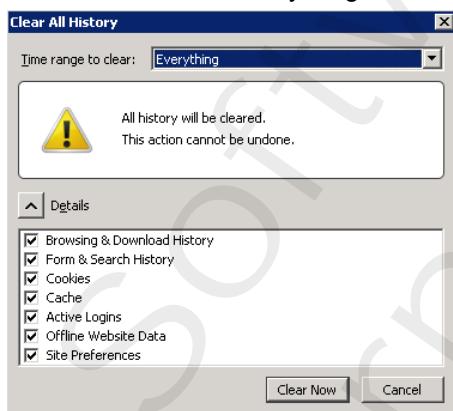
The screenshot shows a RESTClient interface. In the 'Request' tab, a GET request is made to the URL `http://daetrain24496:5555/gateway/SearchCruise/1/cruises`. The 'Headers' section includes an 'x-CentraSite-APIKey' header with the value `da01febf-d6cc-4dbf-a769-ef7be27a0a8fxxxxxx`. The 'Body' tab shows the JSON response:

```
[{"cruises": [{"cruiseID": "00001", "title": "Alaska Whale Watching Photo Expedition", "description": "Photographing the whale in Alaska is a unique experience. Join us on this tour to this unique area. The cruise is a luxury tour for those that want to spend time with others with a like for see these unique animals of the sea. The tour will be conducted by a pro photography who will work with each of the members of the tour as needed.", "startPort": "Juneau, Alaska", "numDays": "3", "startDate": "20150820", "endDate": "20150827", "roomTypes": [{"roomID": "100", "roomDetails": {"roomType": "Inside Premier Double", "title": "Inside Premier Double", "description": "An eco-friendly double room with desk, TV and Shower", "maxOccupants": "2", "totalRoomsOfType": "3"}, {"roomID": "100P", "roomDetails": {"roomType": "Seaview Premier Double", "title": "Premier Double cabin with seaview", "description": "An eco-friendly double room with seaview, desk, TV and Shower", "maxOccupants": "2", "totalRoomsOfType": "3"}, {"roomID": "200P", "roomDetails": {"roomType": "Seaview Deluxe Room", "title": "Deluxe cabin with seaview", "description": "For those wanting that extra bit of luxury. The Deluxe cabins come with a private balcony, desk, TV and Shower", "maxOccupants": "2", "totalRoomsOfType": "2"}], "numRoomsAvailable": "3"}]}]
```

14. Verify that a valid API key is needed to call the service. Open the HTTP header defined to pass on the API key and change the valid API key to an invalid one by adding a number of xxxxxxxx to it.



15. In case you don't get an error response, please clear the Firefox cache (press Ctrl + shift + delete). Set the time to clear to Everything.



16. Login to **API Gateway** as user **Administrator/manage**. Navigate to **Administration > Approval configuration**. Within Create application use the slider to disable the approval workflow.

Exercise 9: Enforcing a Consumer Relationship using API Key

The screenshot shows the SAP API Gateway Administration interface. The left sidebar lists various configuration options: General, Security, Destinations, System settings, Load balancer, Extended settings, API fault, Approval configuration (which is selected), Create application, Register application, Update application, Subscribe package, Outbound proxy, and URL aliases. The main content area is titled 'Update application' and 'Configure the approval workflow to modify an existing application.' A red box highlights the 'Enable' checkbox, which is checked. Below it, there are fields for 'Approvers' (set to 'Approvers') and 'Approved by' (set to 'Anyone'). There are sections for configuring approval initiate request mail template and request approved mail template, both with checkboxes for 'Send notification'. The 'Content' section contains a template message. At the bottom, there are 'Cancel' and 'Save' buttons, with 'Save' also highlighted by a red box.

Click **Save**.

17. Disable the workflow for the processes: **Register application**.

This page intentionally left blank

Software AG
Internal Use Only!

Exercise 10: Managing API Threat Protection

Objectives

In this exercise, you will define some threat protection rules which will be imposed by API Gateway before the request reaches the internal applications. The internal application on our Virtual Machine is inside another installation of Integration Server running on port 7777.

The internal Application is the echo service. The URL to the WSDL file of this API is:

localhost:7777/ws/EchoWS?wsdl

Steps

1. Open Windows Services UI to double check that the following services, needed API Gateway, are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1**
 - b. **Software AG Internal Integration Server 10.1**
 - c. **Software AG Event Data store 10.1 – should be started together with IS**
2. Logon to **API Gateway** as user **Administrator/manage**. Create a new API where the endpoint is on the preconfigured VM. Use the import function using a WSDL file.
 - a. **Select File:** C:\Training\456-71E\Exercise10\Echo.wsdl
 - a. **Name:** EchoOnInternalIS
 - b. **Type:** WSDL
 - c. **Version:**

Lets Get Started!

Import API from file
Create an API by importing API from a specified file.

Select file*
Echo.wsdl

Name
EchoOnInternalIS

Type
WSDL

Create

Import API from URL
Create an API by Importing it from an URL.

Create API from scratch
Create an API from scratch by providing the required information.

Click Create.

Navigate to Technical Information and verify that the input is on the specific Integration Server

EchoOnDMZ
View API details, basic and technical information, resources and methods available, and API specifications.

Enable mocking Update Create new version Edit Activate

API details Scopes Policies Applications Analytics

Basic information
Name: EchoOnDMZ
Version: 1
Active: No
Maturity state: Beta
Created: 2018-01-19 10:26:55 GMT

Technical information
Native endpoint(s)
http://localhost:7777/ws/EchoWS.EchoWSHttpSoap11Endpoint
http://localhost:7777/ws/EchoWS.EchoWSHttpSoap12Endpoint

Exercise 10:
Managing API Threat Protection

3. Navigate to the user menu (**Administrator** menu) and select **Administration > Security > Ports..**

Ports	Alias	Protocol	Type	Enabled	Description	Action
5555	DefaultPrimary	HTTP	Regular		Default Primary Port	
5483	APIGatewayHTTPS	HTTPS	Regular	✓	Integration Server HTTPS port:5483	

Click the **Add ports** button to create the External Port which is exposed to the consumer

4. Select type **API Gateway External**.

Add port

Type

HTTP
HTTPS
API Gateway external

Cancel Add

Click **Add**.

5. Provide the following properties

a. **API Gateway external listener configuration**

- i. **External port:** 8888
- ii. **Protocol:** HTTP
- iii. **Alias:** ExtPortAlias
- iv. **Bind Address:** - leave the defaults –
- v. **Description:** Integration Server HTTP Port 8888
- vi. **Backlog:** - leave the defaults -
- vii. **Keep alive timeout:** - leave the defaults –

b. **API Gateway registration listener configuration**

- i. **Registration port:** 8889
- ii. **Alias:** RegPortAlias

Click **Add**

API Gateway registration listener configuration

Registration port* Alias*

8889 RegPortAlias + Add

Protocol

HTTP

Description (optional)

Integration Server HTTP Port 8889

Bind address (optional)

- iii. **Protocol:** HTTP
- iv. **Bind Address:** - leave the defaults –
- v. **Description:** Integration Server HTTP Port 8889

Click **Add** to add the external listener configuration.

Exercise 10: Managing API Threat Protection

Integration Server HTTP Port 8889

Private threadpool configuration

Security configuration

API Gateway registration listener configuration

Registration port	Alias
8889	RegPortAlias

Protocol: HTTP

Description (optional): Integration Server HTTP Port 8889

Security configuration

Add

- Activate the ports you have created in the previous step. Click the cross in the enabled column of the list of ports

Ports	Alias	Protocol	Type	Enabled	Description	Action
5555	DefaultPrimary	HTTP	Regular	✓	Default Primary Port	
5843	APIGatewayHTTPS	HTTPS	Regular	✗	Integration Server HTTPS port: 5843	
8889	RegPortAlias	HTTP	API Gateway registration	✗	Integration Server HTTP Port 8889	
8888	ExtPortAlias	HTTP	API Gateway external	✗	Integration Server HTTP Port 8888	

Add ports

Click Yes to confirm to enable the port.

Ports	Alias	Protocol	Type	Enabled	Description	Action
5555	DefaultPrimary	HTTP	Regular	✓	Default Primary Port	
5843	APIGatewayHTTPS	HTTPS	Regular	✗	Integration Server HTTPS port: 5843	
8889	RegPortAlias	HTTP	API Gateway registration	✓	Integration Server HTTP Port 8889	
8888	ExtPortAlias	HTTP	API Gateway external	✓	Integration Server HTTP Port 8888	

Add ports

- Within Firefox open the **IS Internal Administrator** for the installation of the **internal IntegrationServer** where the **Echo API** is running which we have already added to **API Gateway**.
 - We need to configure the connection to **API Gateway**. Navigate to **Security > Ports > Add Port**.

Exercise 10: Managing API Threat Protection

The screenshot shows the WEBMETHODS Integration Server interface. The left sidebar has sections for Server, Logs, Packages, Solutions, Adapters, webMethods Cloud, and Security. Under Security, there are sub-options like ACLs, Certificates, CSRF Guard, JWT, Kerberos, Keystore, OAuth, OpenID, Outbound Passwords, and Ports. The Ports section is highlighted with a red box. The main content area shows a 'Port List' table with two rows. The first row is for port 7070 with alias 'DefaultPrimary', protocol 'HTTP', type 'Primary', package 'WMRoot', enabled status checked, access mode 'Allow', and IP access 'Allow'. The second row is for port 9997 with alias 'DefaultDiagnostic', protocol 'HTTP', type 'Diagnostic', package 'WMRoot', enabled status checked, access mode 'Allow', and IP access 'Allow'. A red box highlights the 'Add Port' link under the 'Security > Ports' menu.

Port	Alias	Protocol	Type	Package	Enabled	Access Mode	IP Access	Advanced	Delete	Description
7070	DefaultPrimary	HTTP	<input checked="" type="checkbox"/> Primary	WMRoot	<input checked="" type="checkbox"/> Yes	Allow	Allow		X	Default Primary Port
9997	DefaultDiagnostic	HTTP	<input type="checkbox"/> Diagnostic	WMRoot	<input checked="" type="checkbox"/> Yes	Allow	Allow		X	Default Diagnostic Port

b. Select Internal Server and click Submit.

The screenshot shows the WEBMETHODS Integration Server interface. The left sidebar includes the Ports section, which is highlighted with a red box. The main content area displays a 'Select Type of Port to Configure' dialog. It lists various port types with radio buttons: webMethods/HTTP, webMethods/HTTPS, webMethods/FilePolling, webMethods/FTPS, webMethods/FTP, webMethods/Email, HTTP Diagnostic, HTTPS Diagnostic, Enterprise Gateway Server, Internal Server (which is selected and highlighted with a red box), and webMethods/WebSocket. A red box highlights the 'Submit' button at the bottom of the dialog.

c. and configure the following properties:

- i. **Enable:** Yes
- ii. **Protocol:** HTTP
- iii. **Package Name:** WMRoot
- iv. **Alias:** InternalPortAlias
- v. **Description:**
- vi. **Max Connections:** 5
- vii. **Threadpool:** Enable

viii. **Enterprise Gateway Server:**

1. **Host:** localhost
2. **Port:** 8889

ix. **Registration Credentials (Optional)**

1. **Name:** Administrator
2. **Password:** manage

x. **External Client Security:**

1. **Client Authentication:** Username/Password

The screenshot shows the 'Edit Internal Server Configuration' page in the webMethods Integration Server administration interface. The left sidebar contains navigation links for Server, Logs, Packages, Solutions, Adapters, webMethods Cloud, Security, Ports, SAML, User Management, and Settings. The main content area has three tabs: 'Internal Server', 'Enterprise Gateway Server', and 'External Client Security'. Under 'Internal Server', the host is set to 'localhost' and port to '8889'. Under 'Enterprise Gateway Server', the host is 'localhost' and port is '8889'. Under 'External Client Security', the client authentication method is set to 'Username/Password'. Registration credentials are listed as 'User Name: Administrator' and 'Password: [REDACTED]'. A note states 'Must match Client Authentication setting on the Enterprise Gateway Server External Port.' A 'Save Changes' button is at the bottom.

Click **Save Changes**.

- d. Change the **Access Mode** of the internal port. Select the new created port and click on **Deny+**

Exercise 10: Managing API Threat Protection

WEBMETHODS
Integration Server default :: daeitrain24496.eur.ad.sag :: Administrator

Server

- Scheduler
- Service Usage
- Statistics

Logs

Packages

- Management
- Publishing
- Subscribing

Solutions

- VCS ...

Adapters

webMethods Cloud

Security

- ACLs
- Certificates
- CSRF Guard
- Enterprise Gateway Rules
- JWT
- Kerberos
- Keystore
- OAuth
- OpenID
- Outbound Passwords

Ports

- SAML
- User Management

Settings

Security > Ports > Edit Access Mode > HTTPListener@8889@localhost

- [Return to Port List](#)
- [Add Folders and Services to Allow List](#)
- [Set Access Mode to Allow by Default](#)**
- [Reset to default access settings](#)

Port Service Access Settings	
Access Mode	Deny by Default
Allow	Deny

Allow List	
Folders and Services	Remove
wm.server.csrfguard:getCSRFSecretToken	X
wm.server.csrfguard:isCSRGuardEnabled	X
wm.server.csrfguard:replaceSpecialCharacters	X
wm.server.tx:end	X
wm.server.tx:execute	X
wm.server.tx:restart	X
wm.server.tx:start	X
wm.server:connect	X
wm.server:disconnect	X
wm.server:getClusterNodes	X
wm.server:getServerNodes	X
wm.server noop	X
wm.server:ping	X

Click to Set Access Mode to Allow by Default. Return to port list.

WEBMETHODS
Integration Server default :: daeitrain24496.eur.ad.sag :: Administrator

Security > Ports

- [Add Port](#)
- [Create Primary Port](#)
- [Change Global IP Address Restrictions](#)

Port List

Port	Alias	Protocol	Type	Package	Enabled	Access Mode	IP Access	Advanced	Uptime	Description
8889	InternalPortAlias	HTTP	Registration Internal	WMRoot	<input checked="" type="checkbox"/> Yes	Allow	Not Applicable	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>	Integration Server HTTP port: 8889
7777	DefaultPrimary	HTTP	<input checked="" type="checkbox"/> Primary	WMRoot	<input checked="" type="checkbox"/> Yes	Allow	Not Applicable	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>	Default Primary Port
9997	DefaultDiagnostic	HTTP	Diagnostic	WMRoot	<input checked="" type="checkbox"/> Yes	Deny	Allow	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>	Default Diagnostic Port

- In order to use the **Reverse Invoke** functionality we have to modify the routing endpoint of the API we have created in one of the last steps. Logon to **API Gateway** as user **Administrator/manage**. Open the API **EchoOnInternalIS** in **Edit** mode. Navigate to **Policies > Straight Through Routing Endpoint URI**.

Exercise 10: Managing API Threat Protection



- Modify the first part of the routing endpoint from **http://localhost:7777** to **apigateway://RegPortAlias**

Straight Through Routing

Endpoint URI *

apigateway://RegPortAlias/ws/EchoWS.EchoWSHttpSoap11Endpoint

SOAP Optimization Method

None

Click **Save and Activate**.

- Open **SOAP UI** and create a new **SOAP UI** project using the endpoint URL pointing to the external port 8888 of the Echo API as shown in **API Gateway**.



Provide **EchoOnInternalS** as Project Name.

New SOAP Project

Creates a WSDL/SOAP based Project in this workspace

Project Name: EchoOnInternalS

Initial WSDL: http://daeittrain25856:8888/ws/EchoOnInternalS/1?wsdl

Create Requests: Create sample requests for all operations?

Create TestSuite: Creates a TestSuite for the imported WSDL

Relative Paths: Stores all file paths in project relatively to project file (requires save)

OK Cancel

- Within the new created **EchoOnInternalS** project select one of the listed requests. Change the port-number in the endpoint from **5555** to **8888**.

`http://daeitrain25856:8888/ws/EchoOnInternalIS/1`

11. Provide some text for the name element. And run the request.

The screenshot shows a SOAP UI interface with two panes. The left pane displays the raw XML request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope">
<soapenv:Header>
<soapenv:Body>
<axis:sayHello>
<!--Optional:-->
<axis:name>hello on internal IS</axis:name>
</axis:sayHello>
</soapenv:Body>
</soapenv:Envelope>
```

The right pane displays the raw XML response:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope">
<soapenv:Header>
<ns:sayHelloResponse xmlns:ns="http://ws.apache.org/axis2">
<ns:return>Hello hello on internal IS</ns:return>
</ns:sayHelloResponse>
</soapenv:Header>
</soapenv:Envelope>
```

Below the panes, there are tabs for Headers, Attachments, SSL Info, WSS, JMS, and JMS Property. The status bar at the bottom indicates a response time of 22ms (292 bytes).

12. We want to use the option that whenever a rule is violated in **API Gateway**, an alert will be generated by **API Gateway**. We want to configure the alert setting on the global level. Within **API Gateway** navigate to **Policies > Threat Protection > Alert Settings**.

The screenshot shows the AEM Admin Center Policies page. The top navigation bar has tabs for WEBMETHODS, APIs, Policies, Applications, and Packages. The Policies tab is selected. Below the tabs, there are three buttons: Threat protection (highlighted with a red box), Global policies, and Policy templates.

The main content area shows several policy categories: Global denial of service, Denial of service by IP, Denied IPs, Rules, Mobile devices and apps, and Alert settings (highlighted with a red box). Under Alert settings, there are fields for Alert destination (Email or Flow service), Send alert (On rule violation or Every), and a 'webMethods IS Service' field containing the value `pub.apigateway.threatProtection:violationListener`. At the bottom are Cancel and Save buttons.

13. Provide the following properties.

- Alert Destination:** Flow Service
- Send alert:** On rule violation
- webMethods Is Service:** `pub.apigateway.threatProtection:violationListener`
- Run as user:** Administrator

The screenshot shows the same Policies page as before, but with updated values in the 'Alert settings' section. The 'Flow service' checkbox is checked under 'Alert destination'. The 'Run as user' field now contains the value 'Administrator'.

Click **Save**.

14. Now we create a global denial of service policy. Navigate to **Policies > Threat protection > Global denial of service**. Provide the following properties:

- Maximum requests:** 2
- In (seconds):** 60
- Maximum requested inprogress:** 1

Exercise 10:
Managing API Threat Protection

- d. **Block interval (minutes):** 1
e. **Error Message:** Receiving too many requests. Rejecting all requests and entering passive mode!!!
f. **Trusted IP addresses:**

Policies
Implement and manage global policies.

Threat protection **Global policies** **Policy templates**

Global denial of service

Denial of service by IP

Denied IPs

Rules

Mobile devices and apps

Alert settings

Enable

Maximum requests * 2 **In (seconds) *** 60

Maximum requests in progress * 1

Block intervals (minutes) * 1

Error message *
Receiving too many requests. Rejecting all requests and entering passive mode!!!

Trusted IP addresses **Add**

Cancel **Save**

Click **Save**.

15. Enable the Policy

Policies
Implement and manage global policies.

Global threat protection

Global denial of service

Denial of service by IP

Denied IPs

Rules

Mobile devices and apps

Alert settings

Enable

Maximum requests * 2 **In (seconds) *** 60

Maximum requests in progress * 1

Block intervals (minutes) * 1

Click **Save**.

Policies
Implement and manage global policies.

Threat protection **Global policies** **Policy templates**

Global denial of service

Denial of service by IP

Denied IPs

Rules

Mobile devices and apps

Alert settings

Enable

Maximum requests * 2 **In (seconds) *** 60

Maximum requests in progress * 1

Block intervals (minutes) * 1

Error message *
Receiving too many requests. Rejecting all requests and entering passive mode!!!

Trusted IP addresses **Add**

Cancel **Save**

16. Go back to project **EchoOnInternalIS** in **SOAP UI**. Rerun the already configured request several times. After 2 times the request will respond the error message as defined in the global policy

Exercise 10:
Managing API Threat Protection

50 API Request 1 http://daetrain24496:8888/ws/EchoOnDMZ/1

Raw XML Headers (0) Attachments (0) WS-A WS-RM JMS Headers JMS Property (0)

response time: 17ms (80 bytes)

Raw XML Headers (6) Attachments (0) SSL Info WSS (0) JMS (0)

Receiving too many requests. Rejecting all requests and entering passive mode!!!

17. Now we want to use a rule to reject all incoming requests based on size of the request. Open **API Gateway** and navigate to **Policies**. Disable the global policy we have defined before.

18. Select Rules and click **Add rule** to create a new rule. Provide the following properties:

- a. **Rule name:** MessageSizeRule
- b. **Description:** This rule will restrict the incoming request by evaluating the message size from incoming request
- c. **Action:** Deny request and alert
- d. **Error message:** You have exceeded the message size limit.
- e. **Request type:** all
- f. **Alert Settings**
 - i. **Custom:**
 - ii. **Alert destination:** Flow service
 - iii. **Send Alert**
 - 1. **On rule violation**
 - 2. **webMethods IS Service:** pub.apigateway.threatProtection:violationListener
 - 3. **Run as user:** Administrator
- g. **Message size filter**
 - i. **Maximum message size (MB):** 1

Denied IPs

Rules

Mobile devices and apps

Alert settings

Rule name*
MessageSizeRule

Action
Deny request and alert

Description
This rule will restrict the incoming request by evaluating the message size from incoming request

Request type
ALL

Error message
You have exceeded the message size limit

Alert settings

Default Custom

Alert destination.
 Email Flow service

Send alert
 On rule violation Every

webMethods IS Service
pub.apigateway.threatProtection:violationListener

Run as user
Administrator

Message size filter

Enable

Maximum message size(MB)
1

Click **Save**.

19. Enable the rule.

Exercise 10: Managing API Threat Protection

The screenshot shows the SAP API Management Threat protection interface. In the top navigation bar, there are tabs for Threat protection, Global policies, and Policy templates. Under Threat protection, there are sections for Global denial of service, Denial of service by IP, Denied IPs, Rules (which is selected), Mobile devices and apps, and Alert settings. A 'Denial rules' table is displayed, showing a single entry: 'MessageSizeRule' with a 'Filter' of 'Message size filter', a 'Rule type' of 'DENY', and an 'Enabled' status. An 'Alert options' column shows a red-bordered 'x' icon, indicating no alerts are configured.

20. Open the file **OneMB.txt** in folder **C:\Training\APIGateway\Exercise10** in Notepad+.
21. Open **SOAP UI** and copy and paste the content of the **OneMB.txt** file as value to the **name** element.
22. Run the request, make sure that your endpoint points to port 8888. The response will be the error message you have just created.

The screenshot shows a SOAP UI session titled 'Request 1'. The request URL is 'http://daetrain24496:8888/ws/EchoOnDMZ/1'. The response body displays a Java stack trace followed by the error message 'You have exceeded the message size limit' in a red box. The response status code is 500 Internal Server Error. The bottom of the screen shows the SAP logo and the text 'SAP API Management'.

Run this request several time to produce some threat protection violations.

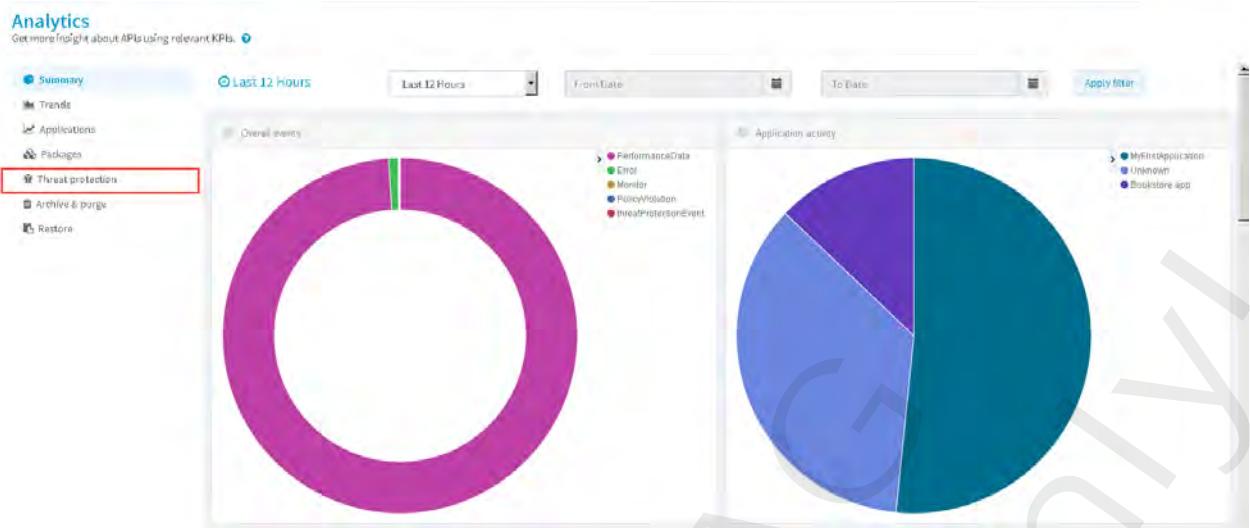
23. We want to review Dashboard information Threat Protection. Go back to **API Gateway** as user **Administrator/manage**.

The screenshot shows the SAP API Management dashboard. The top navigation bar includes 'WEBMETHODS', 'API Gateway', 'APIs', 'Policies', 'Applications', and 'Packages'. On the right, a user menu for 'Administrator' is open, showing options like Administration, Aliases, Analytics (which is selected and highlighted with a red border), Import, Pending Requests, User Management, About, Help, and Logout.

Open the User menu and select **Analytics**.

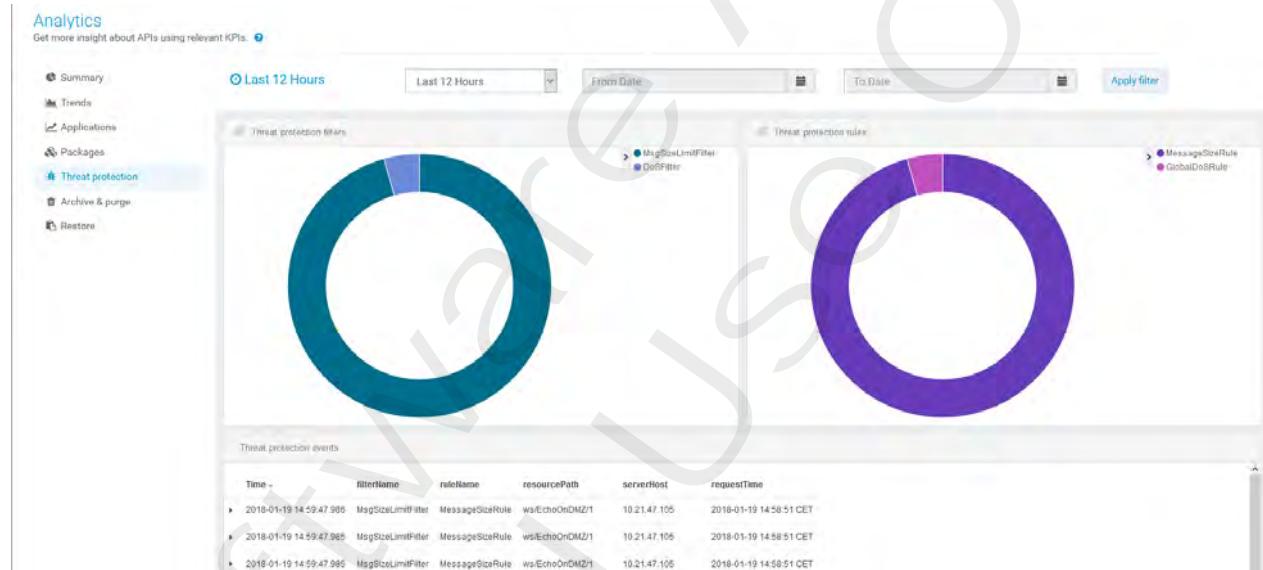
24. The **Summary** dashboards show **Overall events**,

Exercise 10: Managing API Threat Protection



Within the left hand menu select **Threat Protection**.

25. Review Threat protection filters and Threat protection rules.



26. Click the triangle next to a threat protection event and review details.

Threat protection events						
Time	filterName	ruleName	resourcePath	serverHost	requestTime	
2018-01-19 14:59:47.986	MsgSizeLimitFilter	MessageSizeRule	ws/EchoOnDMZ/1	10.21.47.105	2018-01-19 14:58:51 CET	
Link to /gateway_default_analytics/threatProtectionEvents/4b8d278d-d944-4918-808d-e1ee56140787						
_id	4b8d278d-d944-4918-808d-e1ee56140787					
_index		gateway_default_analytics				
_score						
_type		threatProtectionEvents				
alertAction		DENY				
creationDate		2018-01-19 14:59:47,986				
eventType		threatProtectionEvent				
filterName		MsgSizeLimitFilter				
id		4b8d278d-d944-4918-808d-e1ee56140787				
message		Message size exceeded the limit of 1 MB.				
requestHost		10.21.47.105				

27. Disable the Threat protection rule. Navigate to **Policies > Threat protection > Rules**. Within the list of rules disable the rule **MessageSizeRule** click the sign within the **Enable** column.

Exercise 10: Managing API Threat Protection

The screenshot shows the 'Threat protection' section of the 'Policies' page. It includes tabs for 'Global policies' and 'Policy templates'. A table lists 'Denial rules' with one entry: 'MessageSizeRule' (Message size filter) with a 'DENY' rule type. The 'Enabled' column for this rule has a red box around its checkmark.

28. Disable the API Gateway registration port and the **API Gateway external** port. Navigate to **Administration > Security > Ports**. Within the list of ports disable these ports. Click the sign within the **Enable** column for each of the ports.

The screenshot shows the 'Ports' configuration page under 'Security'. It lists listener ports with columns for Port, Alias, Protocol, Type, Enabled (checkbox), Description, and Action. Two specific ports are highlighted with a red box: port 8889 (RegPortAlias, API Gateway registration) and port 8888 (ExtPortAlias, API Gateway external). Both have their 'Enabled' checkboxes checked.

Ports	Configure listener ports in API Gateway					
Ports	Alias	Protocol	Type	Enabled	Description	Action
5555	DefaultPrimary	HTTP	Regular	<input checked="" type="checkbox"/>	Default Primary Port	
5843	APIGatewayHTTPS	HTTPS	Regular	<input checked="" type="checkbox"/>	Integration Server HTTPS port: 5843	
8889	RegPortAlias	HTTP	API Gateway registration	<input checked="" type="checkbox"/>	Integration Server HTTP Port 8889	
8888	ExtPortAlias	HTTP	API Gateway external	<input checked="" type="checkbox"/>	Integration Server HTTP Port 8888	

This page intentionally left blank

Software AG
Internal Use Only!

Exercise 11: Manage Fine Granular Exposure of APIs

Objectives

Define an APIs SAGTours application. An API for signing-up to SAGTours is available as well as an API for booking of a cruise. Before a cruise can be booked, the customer must sign-up to the SAGTours. The operations GET, PUT and DELETE of the Sign-up API are protected via Basic Authentication. The email you provide as user-email is the username you need for BASIC authentication. This fine granular security enforcement will be implemented using scope level policy actions.

All operations in the Bookings API are protected with Basic Authentication.

In addition to the enforced security layer for a set of operation we will enforce a general logging policy.

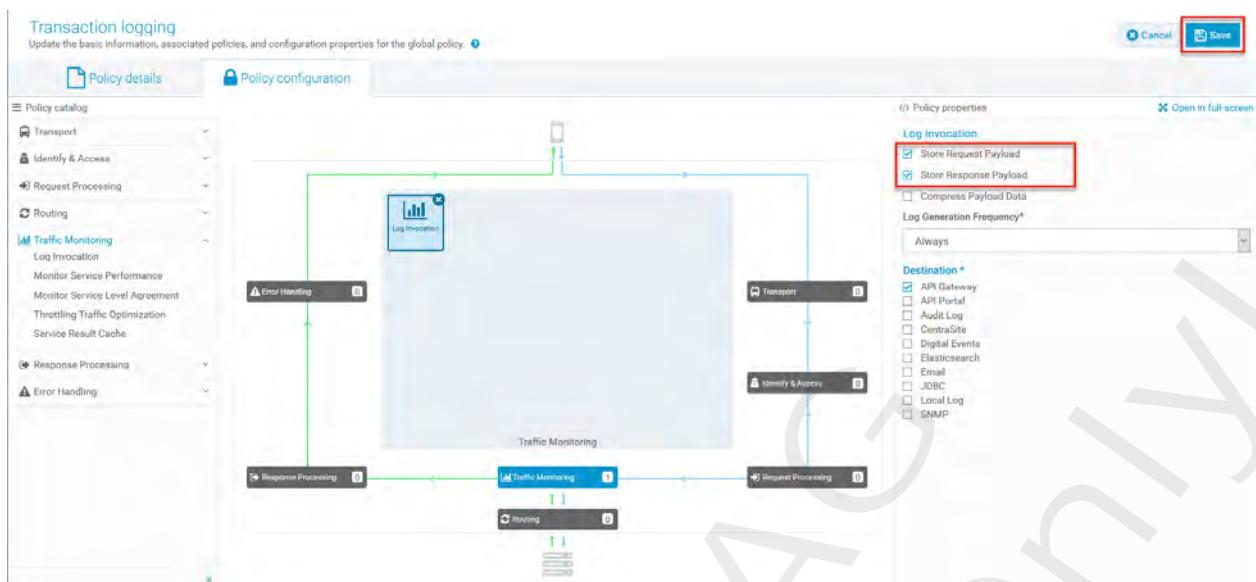
Steps

1. Open Windows Services UI to double check that the following services, needed for API Gateway are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1**
 - b. **Software AG Event Data store 10.1 – should be started together with IS**
 - c. **Software AG Runtime 10.1**
2. Login to the **API Gateway UI** as user **Administrator/manage**. We want to define a global policy for Logging. Naviagte to **Policies > Global policies**. A default **Transaction logging** policy is already predefined with the installation of **API Gateway**. Select the policy to open the detail view of this policy. Because it is a global policy all APIs you have created until now are listed within the **APIs** section.

The screenshot shows the API Gateway UI interface. At the top, there's a navigation bar with tabs for WEBMETHODS, API Gateway, APIs, Policies, Applications, and Packages. The Policies tab is selected. Below the navigation bar, the URL is shown as Home > Policies > Transaction logging. The main content area has a title 'Transaction logging' with a sub-instruction 'View the basic information, applicable APIs, associated policies, and configuration properties of the global policy.' There are two tabs: 'Policy details' (selected) and 'Policy configuration'. Under 'Policy details', there's a 'Basic information' section with fields for Name (Transaction logging), Description (This is a system policy, which has log invocation policy and filters associated to log request or response payloads to a specified destination. These transactions are monitored and logged across all APIs in API Gateway), and API Type (REST/SOAP). To the right of this section is a 'Filters' section. Below these sections is a table titled 'APIs' with columns for Name, Description, and Version. It lists three APIs: 'Bookstore' (REST, Version 1.0), 'getYacht!' (SOAP, Version 1.0), and 'SearchCruises' (REST, Version 1.0). On the far right of the table are 'Edit' and 'Activate' buttons. The bottom of the page has a footer with links to 'Software AG' and 'Help'.

3. Navigate to **Policy configuration** and switch over to **Edit** mode. For the policy action **Log invocation** provide the following properties:
 - a. Store Request Payload: - select -
 - b. Store Response Payload: - select -

Exercise 11: Manage Fine Granular Exposure of APIs



Click **Save**. Click **Activate**.

4. Login to the **API Gateway UI** as user **Sumala/manage**. Navigate to **APIs** and create a new **SignUp API** using the import functionality. The swagger file **Sign-upAPI.json** is available in the folder **C:\Training\456-71E\Exercise11**.

The screenshot shows the 'Create API' screen. It has three main options: 'Import API from file', 'Import API from URL', and 'Create API from scratch'. Under 'Import API from file', 'Sign-upAPI.json' is selected. A red box highlights the 'Create' button at the bottom of this section.

5. Reviewing the **Resource and methods** section you find the resource **/customer** with operation **POST** to add a new customer. The result of this operation will provide a **customerID** in the response body. The 2nd resource is **/customer/{customerID}** to access a specific customer account. These operations **GET**, **PUT** and **DELETE** enforce **HTTP Basic Authentication**. In order to define a corresponding security action for this combination of resource and operations we will define a scope. Navigate to **Scopes** and switch on to **Edit mode**.

The screenshot shows the 'Scopes' page for the 'SignupAPI'. It has tabs for 'API details', 'Scopes', 'Policies', and 'Applications'. The 'Scopes' tab is active. A red box highlights the 'Add scope' button in the 'List of scopes' section. Below the table, a message says 'No scopes are available for this API. Click Add Scope to create a scope.'

Click **Add scope**.

6. Within the **Scopes** page provide the following properties:

- a. **Name:** Customer Specific Data

Exercise 11:
Manage Fine Granular Exposure of APIs

b. **Description:** Accessing customer data in the SAGTours application, like GET, PUT, DELETE

c. **Resources and methods**

i. **/customer**

1. **Add resource to scope:** - leave unselected -

ii. **/customer/{customerId}:**

1. **Add resource to scope:** - select -

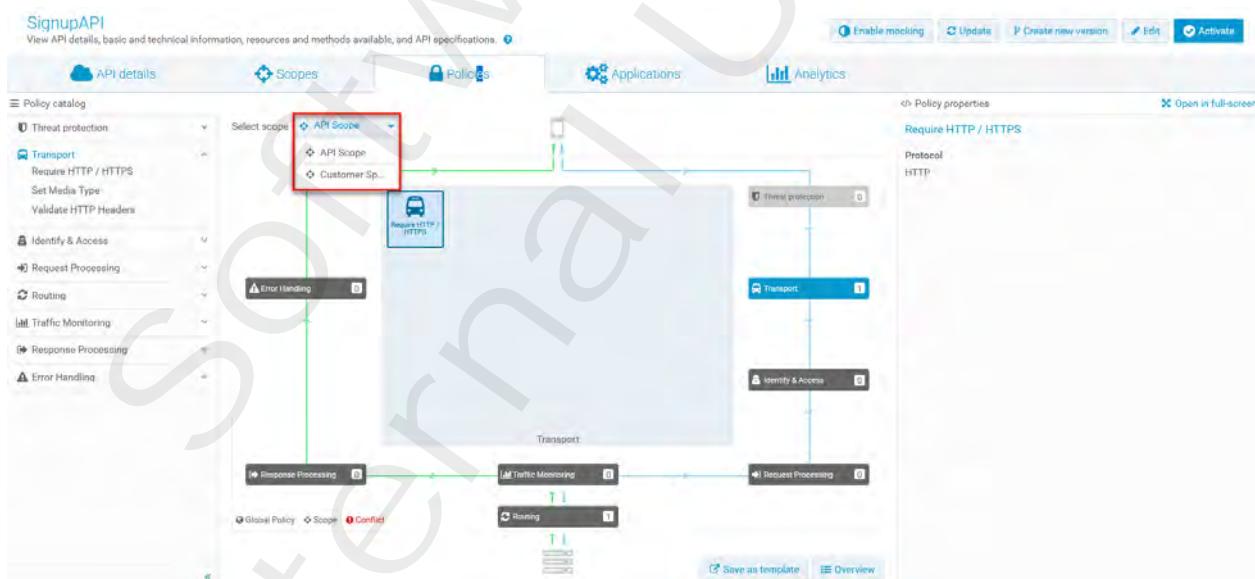
2. **Methods:**

- leave methods GET, PUT, DELETE selected -

The screenshot shows the 'SignupAPI' configuration page. In the 'Scopes' tab, there is a description: 'Accessing customer data in the SAGTours application, like GET, PUT, DELETE'. In the 'Resources and methods' section, under the '/customer' resource, there is a table for '/customer/{customerId}'. The 'Methods' column has three rows: 'GET' (selected), 'PUT' (selected), and 'DELETE'. A red box highlights this row.

Click **Save**.

7. Navigate to **Policies**. Click Select Scope > API Scope.



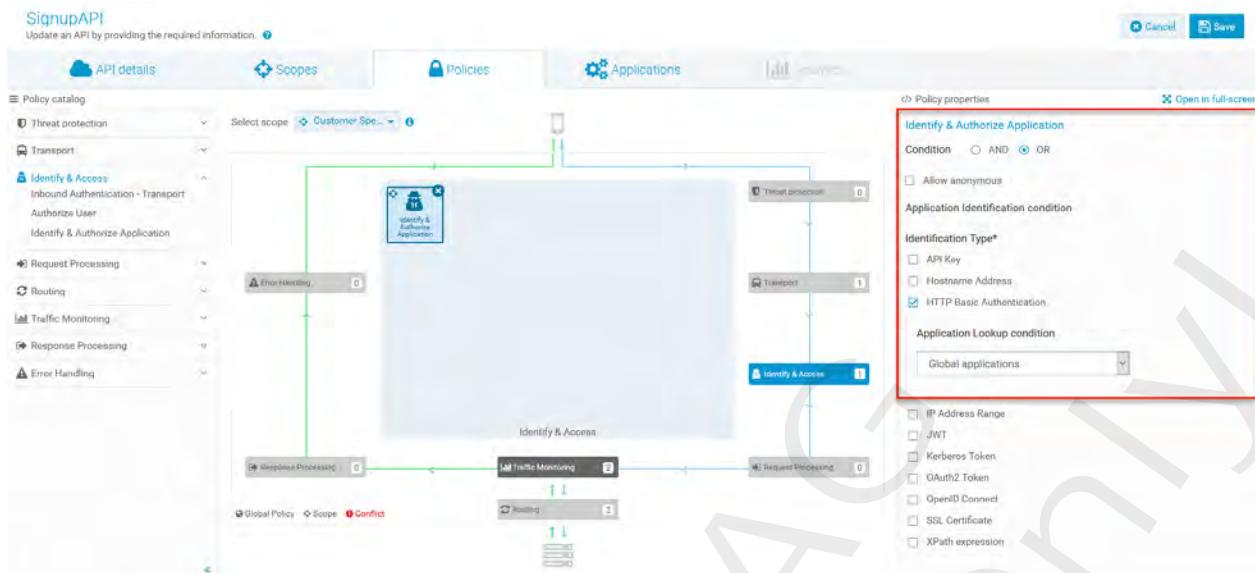
From the list select the just created scope **Customer Specific Data**.

8. Switch to **Edit** mode. Add the polciy action **Identify & Access > Identify & Authorize Application**. Provide the following properties

i. **HTTP Basic Authentication**

ii. **Application Lookup Condition:** Global applications

Exercise 11: Manage Fine Granular Exposure of APIs



- Click **Overview** on the bottom of the page to see the policy definitions structured by API level, Scope-level and Resource or method level.

The screenshot shows the SAP API Management Overview page. It has three main sections: 'API level policies', 'Scope-level policies', and 'Resource or method level policies'. The 'API level policies' section shows a single entry for 'SignupAPI'. The 'Scope-level policies' section shows one entry for 'Customer Specific DataNew Scope'. The 'Resource or method level policies' section has a search bar with '/customer' typed into it, which is highlighted with a red box. Below the search bar, there are two entries: '/customer' and '/customer'. The entire 'Resource or method level policies' section is also highlighted with a red box.

Within the section **Select resource or method** scroll down to the resource & method **PUT - /customer/{customerId}**

Exercise 11: Manage Fine Granular Exposure of APIs

The screenshot shows the SAP API Management Overview page with three main sections:

- API level policies:** Shows one policy named "SignupAPI" with the description: "API for signing up to SAGTours. The signing-up resources creates/updates/deletes a customer from SAG..."
- Scope-level policies:** Shows one policy named "Customer Specific DataNew Scope" with the description: "Accessing customer data in the SAGTours application, like GET, PUT, DELETE".
- Resource or method level policies:** Shows a list of policies applied to the resource "/customer/{customerId}" via the method "PUT". A red box highlights the "Select resource & method" input field.

Close the **Overview** page.

10. Activate the **SignupAPI**.
11. Import the **BookingsAPI** using the swagger file **BookingsAPI.json** available in the folder **C:\Training\456-71E\Exercise11**. Navigate to **Policies** and switch to **Edit** mode. Add the the policy **Identify & Access > Identify & Authorize Application**. Set the property **Identification Type** to to **HTTP Basic Authentication** with **Lookup Condition** set to **Global applications** as you have done for the **SignupAPI**. **Save the API**.
12. Open **Firefox**. Open a **Private Window**. Open **RESTClient** and create a new account in the SAGTours application. Use the API **SignupAPI** selecting the resource **/customer** and the operation **POST**. You will find a corresponding request body in the file **SignUp API.txt** in folder **C:\Training\456-71E\Exercise11**. You need to provide the following
 - a. **Headers:**
 - i. **Accept:** application/json
 - ii. **Content-Type:** application/json
 - b. **Body**
 - i. Copy and paste corresponding request body for method **GET** out of the file **SignUp API.txt** in folder **C:\Training\456-71E\Exercise11**.

The screenshot shows the RESTClient interface with a POST request to the endpoint `http://daetran24496:5555/gateway/SignupAPI/1.0/customer`. The request body is a JSON object representing a customer profile:

```

{
    "customer": {
        "firstname": "Somala",
        "surname": "Somala",
        "title": "Mr.",
        "emailAddress": "somala@company.com",
        "passwordID": "1220123456789",
        "passportExpiration": "2022-01-01",
        "passportOrgin": "USA",
        "sex": "M",
        "dob": "1982-09-01"
    },
    "address": {
        "street": "221B Baker St",
        "city": "London",
        "state": "null",
        "zipcode": "M1M 1EX",
        "country": "UK"
    }
}

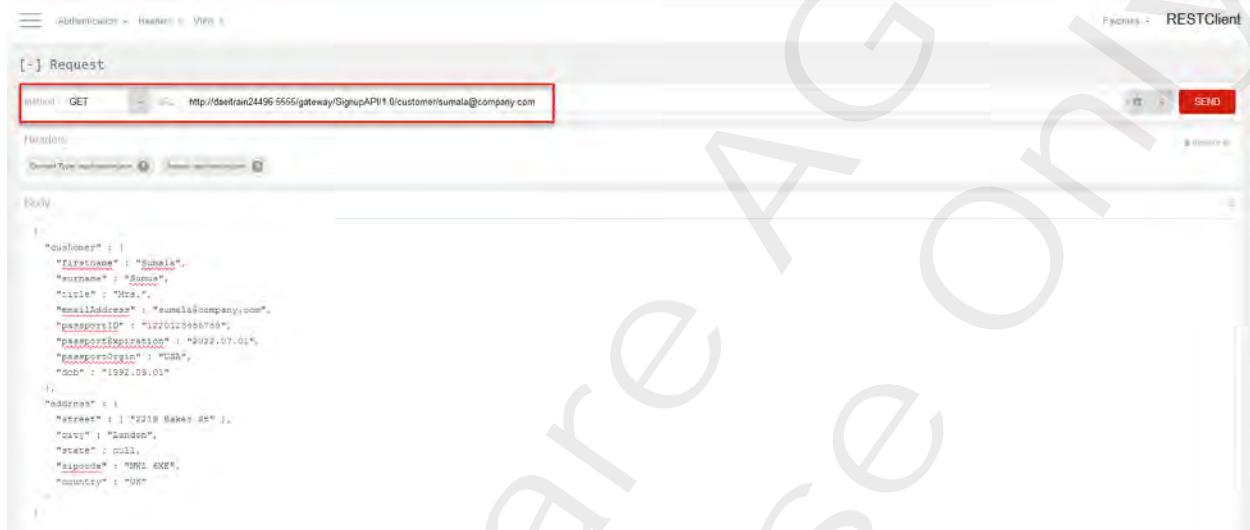
```

Exercise 11:
Manage Fine Granular Exposure of APIs

Click **Send**.

13. A new account with **customerID** set to **sumala@company.com** is created in the SAGTours application. To verify this we would like to review the customer detail using the **GET** operation on the resource **/customer/{customerID}** within the API **SignupAPI**. Open **Firefox**. Open a **Private Window**. Open **RESTClient** and provide the following properties

- URL: - copy and paste the Gateway endpoint. Append the the following -
`/customer/sumala@companay.com`
- Headers:**
 - Accept:** application/json

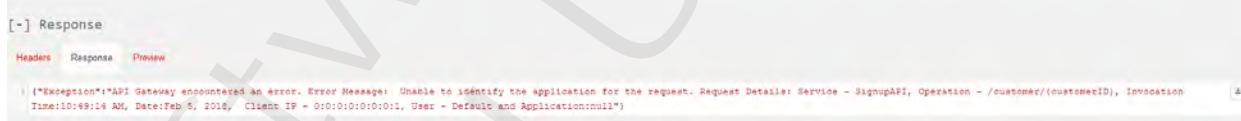


The screenshot shows the RESTClient interface. In the Request tab, a GET request is made to the URL `http://daetrain24496:5555/gateway/SignupAPI/1.0/customer/sumala@company.com`. The request body contains the following JSON data:

```
{
  "customer": {
    "firstName": "Sumala",
    "lastName": "Sumala",
    "title": "Mrs.",
    "emailAddress": "sumala@company.com",
    "customerID": "1234567890",
    "passportExpiry": "2022-07-01",
    "passportOrigin": "USA",
    "dob": "1992-09-01"
  },
  "address": {
    "street": "221B Baker St",
    "city": "London",
    "state": null,
    "zipcode": "W1E 6XX",
    "country": "UK"
  }
}
```

Click **Send**.

This leads to an error, because Basic Authentication is missing.

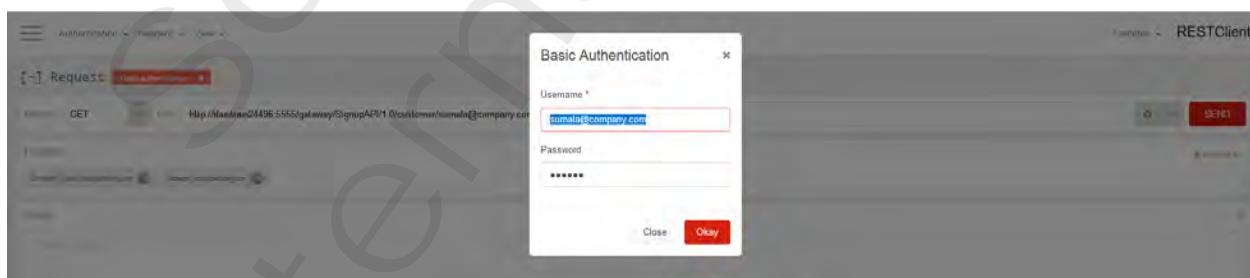


The screenshot shows the RESTClient interface with a response tab. The response body displays an error message:

```
{"Exception": "API Gateway encountered an error. Error Message: Unable to identify the application for the request. Request Details: Service - SignupAPI, Operation - /customer/{customerID}, Invocation Time:10:49:14 AM, Date:Feb 5, 2018, Client IP - 0:0:0:0:0:0:0:1, User - Default and Applicationnull"}
```

14. Provide **Basic Authentication** in RESTClient:

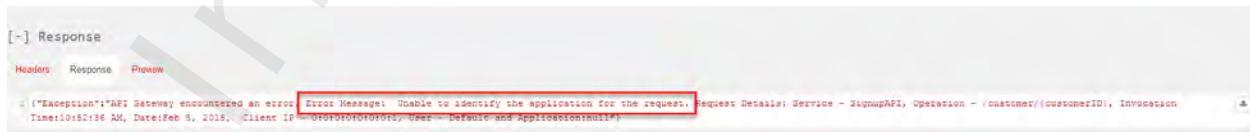
- Username:** sumala@company.com
- Password:** manage



The screenshot shows the RESTClient interface with a basic authentication dialog box overlaid. The dialog box has two fields: "Username" containing `sumala@company.com` and "Password" containing `manage`. The "Okay" button is highlighted in red.

Click **Send**.

This leads to an error, because API Gateway is unable to identify the user in any of the applications.



The screenshot shows the RESTClient interface with a response tab. The response body displays an error message:

```
{"Exception": "API Gateway encountered an error. Error Message: Unable to identify the application for the request. Request Details: Service - SignupAPI, Operation - /customer/{customerID}, Invocation Time:10:52:36 AM, Date:Feb 5, 2018, Client IP - 0:0:0:0:0:0:0:1, User - Default and Applicationnull"}
```

15. Within **API Gateway** navigate to **Applications**. Open the existing application **MyFirstApplication**. Switch to **Edit mode** and add another identifier. Provide the following properties:

Exercise 11:
Manage Fine Granular Exposure of APIs

- a. Identifier: Username
- b. Token: sumala@company.com

Click **Add**.

MyFirstApplication
Update an application by providing required information.

Cancel Save

Application details

Basic information IP address range

Identifiers Partner identifier

APIs Client certificates

OAuth2 Credentials Claims

Add claims set

Other identifiers Username

Name Value

Username	Sumala
	sumala@company.com

Continue to APIs >

Click **Save**.

16. Within RESTClient run the request again. This time the request runs successfully.

[-] Request

Methode: GET URL: http://daeftrain24196:5555/gateway/SignupAPI/1.0/customer/sumala@company.com

Headers:

Content-Type: application/json Accept: application/json

Body:

[-] Response

Headers Response Preview

```
1: {
2:   "customer": {
3:     "emailAddress": "sumala@company.com",
4:     "title": "Mrs.",
5:     "firstName": "Sumala",
6:     "surname": "Sumala",
7:     "dob": "1992-09-01",
8:     "passportID": "1234567899",
9:     "passportExpirationDate": "2022-07-01",
10:    "passwordOrigin": "MSA"
11:  },
12:  "address": {
13:    "street": "201B Baker St",
14:    "city": "London",
15:    "zipcode": "W1S 8XE",
16:    "country": "UK"
17:  }
}
```

17. Now we want to change the details of the customer sumala@company.com. Use the same setting as in step 16 and 17. Change the following properties:

- a. Method: PUT
- b. Body:

- i. Copy and paste corresponding request body for method **PUT** out of the file **SignUp API.txt** in folder **C:\Training\456-71E\Exercise11**. This will change the attribute **title** from **Mrs.** To **Dr...**

Exercise 11:
Manage Fine Granular Exposure of APIs

The screenshot shows a RESTClient interface with a PUT request to `http://daetran24496:5555/gateway/SignupAPI/1.0/customer/sumala@company.com`. The request body contains the following JSON:

```
{
  "customer": {
    "firstname": "Sumala",
    "surname": "Sumala",
    "title": "Dr.",
    "emailAddress": "sumala@company.com",
    "passportID": "1234567890123456789",
    "passportExpiration": "2022-07-01",
    "passportOrigin": "USA",
    "dob": "1992-09-01"
  },
  "address": {
    "street": "221B Baker St",
    "city": "London",
    "state": null,
    "zipcode": "W1N 6XE",
    "country": "UK"
  }
}
```

Click **Send**.

18. To verify the changes change the **Method** back to **GET**. Click **Send**.

The screenshot shows a RESTClient interface with a GET request to `http://daetran24496:5555/gateway/SignupAPI/1.0/customer/sumala@company.com`. The response body contains the same JSON as the PUT request, but the `title` field is now highlighted with a red box.

```
{
  "customer": {
    "firstname": "Sumala",
    "surname": "Sumala",
    "title": "Dr."
  },
  "address": {
    "street": "221B Baker St",
    "city": "London",
    "state": null,
    "zipcode": "W1N 6XE",
    "country": "UK"
  }
}
```

19. Login as user Administrator/manage and disable the Global Policy **Transaction logging**.

The screenshot shows the Global Policies interface. A policy named "Transaction logging" is selected and has a red box around its disable icon.

Exercise 12: Monitoring the Virtual API

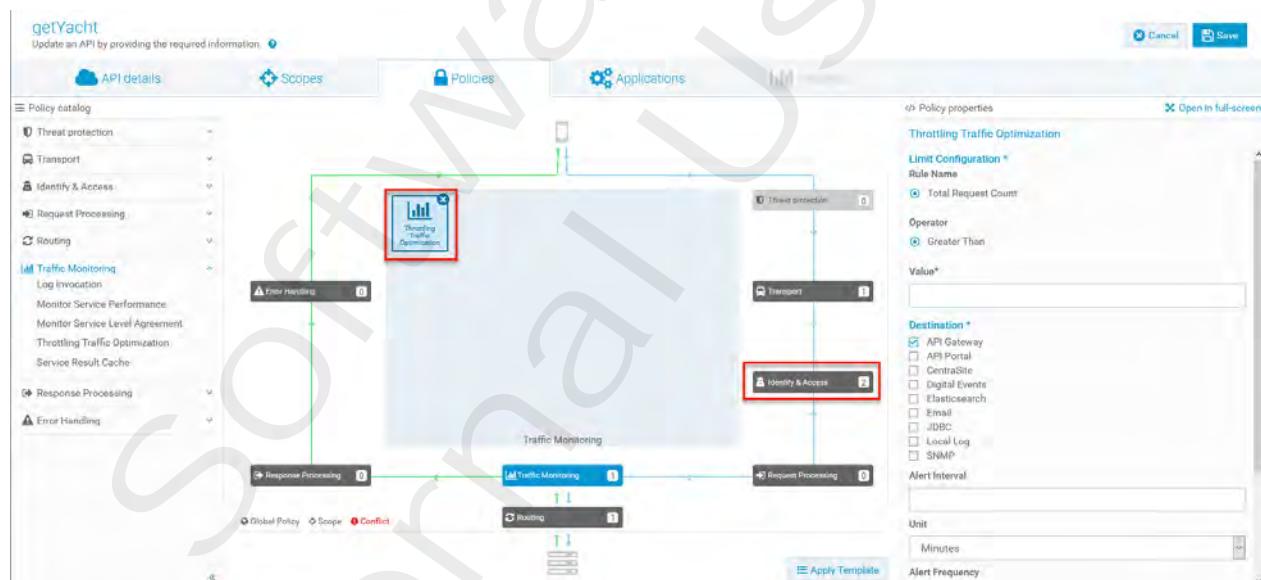
Objectives

Scenario:

Create and use new Policy Action: Throttling Traffic Optimization and Logging

Steps

1. Open Windows Services UI to double check that the following services, needed for API Gateway, are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1**
 - b. **Software AG Event Data store – should be started together with IS 10.1**
 - c. **Software AG Runtime 10.1**
2. Logon to **API Gateway UI** as user **Sumala/manage**. Navigate to the **APIs**. Select the API **getYacht**. Deactivate the API and switch do Edit mode.
3. Navigate to **Policies > Traffic Monitoring**.
4. Switch to **Edit mode**. Within the section **Traffic Monitoring** select the action **Throttling Traffic Optimization**. A dialog box will ask you whether you want to add the depending policy action **Identify & Authorize Application**. Click **Yes**.



5. Provide the following parameters for the selected policy action. Use the button **Open in full-screen** in order to switch to full screen mode.
 - a. **Limit Configuration**
 - i. **Total Request:** - selected -
 - ii. **Greater Than:** - selected -
 - iii. **Value:** 2
 - b. **Destination:**
 - i. **API Gateway:** - select -

Exercise 12:
Monitoring the Virtual API

- ii. **Alert Interval:** 1
- iii. **Unit:** Minutes
- iv. **Alert Frequency:** Every Time
- v. **Alert Message:** Too many calls for getYacht API
- vi. **Consumer Applications:** MyFirstApplication

The screenshot shows the 'Throttling Traffic Optimization' configuration interface. It includes fields for 'Rule Name' (Total Request Count), 'Operator' (Greater Than), 'Value' (2), 'Alert Interval' (1 minute), 'Alert Frequency' (Every Time), and an 'Alert Message' (Too many calls for getYacht API). A table lists consumer applications, with 'MyFirstApplication' selected.

Click OK.

6. Navigate to **Policies > Identify & Access > Identify & Authorize Application**. Provide the following properties:
 - a. **Application Identification condition**
 - i. **Identification Type:** WS Security Username token
 - ii. **Application Lookup condition:** Registered applications



7. Click **Save and Activate**.
8. We need to add the **getYacht** API to the application **MyFirstApplication**. Navigate to **Applications**. From the list of applications select **MyFirstApplication**. Switch to **Edit** mode. Navigate to **Application Details > APIs**. Add the API **getYacht**.

Exercise 12: Monitoring the Virtual API

The screenshot shows the 'MyFirstApplication' interface with the 'APIs' tab selected. A search bar at the top contains the text 'get'. Below the search bar, a table lists selected APIs. The first API is 'SearchCruise' with a description 'Searches for all cruises, or searches for a cruise that matches the specified criteria. The GET met...'. The second API is 'getYacht' with a description 'Searches for all yachts, or searches for a yacht that matches the specified criteria. The GET met...'. The 'getYacht' row is highlighted with a red border.

Click **Save**.

9. Open **SOAP UPI** and open the existing project **getYacht** which you have created in exercise 7 or create a new SOAP project. Provide **Basic Authentication** with
 - a. **Username:** Sumala
 - b. **Password:** manage
 Add **WSS Username token**.

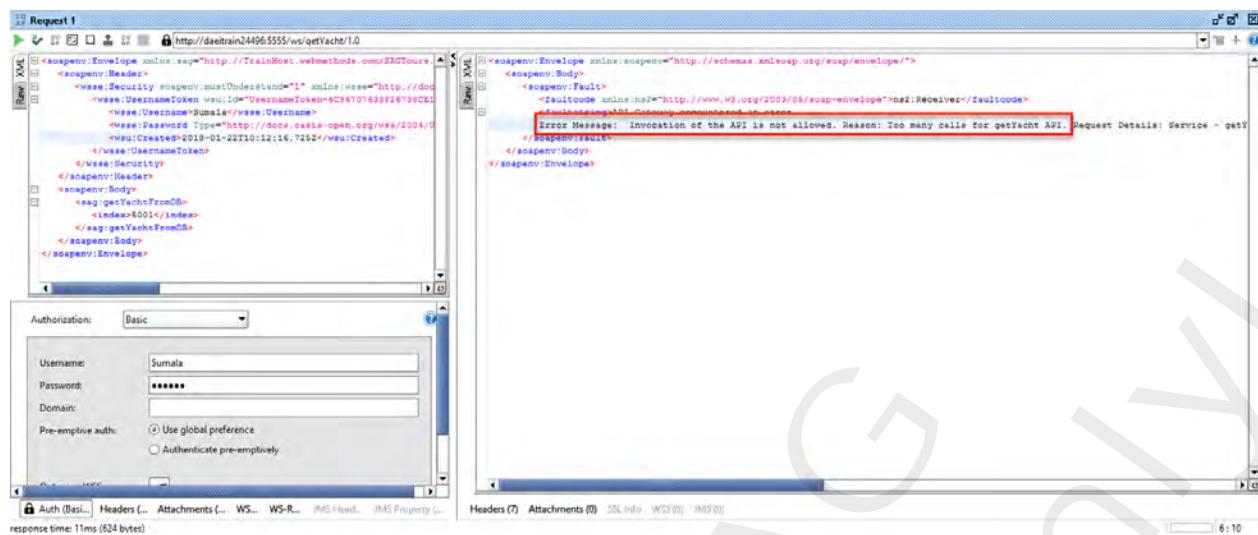
10. Run the request.

11. Within the request body delete the line beginning with **<wsse:Nonce**

The screenshot shows the SoapUI interface with a request labeled 'Request 2'. The left pane shows the raw XML of the SOAP message. The right pane shows the raw XML of the response. The bottom left pane shows the 'Authorization' settings for 'Basic' authentication with 'Username: Sumala' and 'Password: manage'. A red box highlights the 'Wsse:Nonce' element in the request XML.

12. Run the request several time. An error will be returned because of the throttling policy mentioning of too many calls within the defined time limit.

Exercise 12: Monitoring the Virtual API



13. Now we try to find these error messages and also the warnings within **API Gateway UI**. Go back to **API Gateway UI** and select the service **getYacht**. Open the tab **Analytics**.

14. Set the filter to **Last 1 Hour**. Click **Apply filter**



15. Within **Runtime events** open the **PolicyViolation** event. Search for the corresponding alert description.

Runtime events					
Time	apiName	applicationName	eventType	alertDesc	eventSource
2018-01-22 11:17:32.280	getYacht	-	PerformanceData	-	-
2018-01-22 11:18:32.283	getYacht	-	PerformanceData	-	-
2018-01-22 11:18:32.280	getYacht	-	PerformanceData	-	-
2018-01-22 11:15:22.000	getYacht	MyFirstApplication	PolicyViolation	A violation of policy was detected - Invocation of the API is not allowed. Reason: Too many calls for getYacht API	-

Below the table, a JSON table view is shown, with the alert description from the previous row highlighted in a red box.

Exercise 13: Using a Routing Policy Action

Objectives

In this exercise, you will define a routing policy action using context based routing.

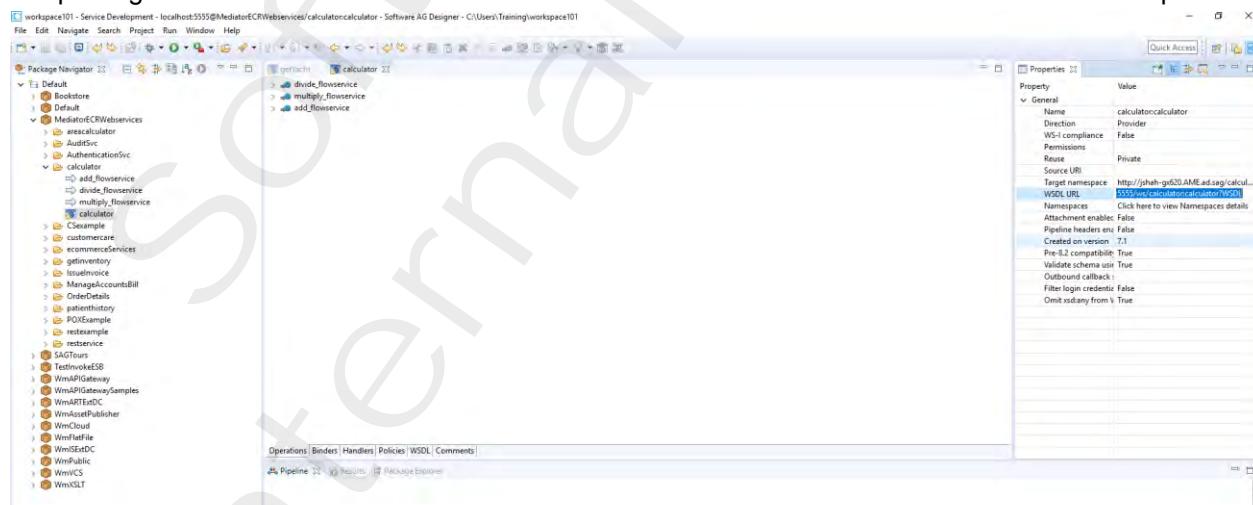
Steps

1. Open **Windows Services UI** to double check that the following services, needed for API Gateway, are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1**
 - b. **Software AG Event Data store 10.1 – should be started together with IS**
2. Open **Software AG Designer 10.1** from the Windows Start Menu > Software AG > Tools. In case you are asked for a **secure storage password**, provide the password **manage**.



Accept the default workspace .

3. Select Window -> Open Perspective -> Service Development. Within Package Navigator open the package **MediatorECRWebservices** and edit the **calculator:calculator** Web Service Descriptor.



In the Properties, copy the Value of WSDL URL and paste it into the Import URL field in **API Gateway**. The value pasted should be similar to the following:

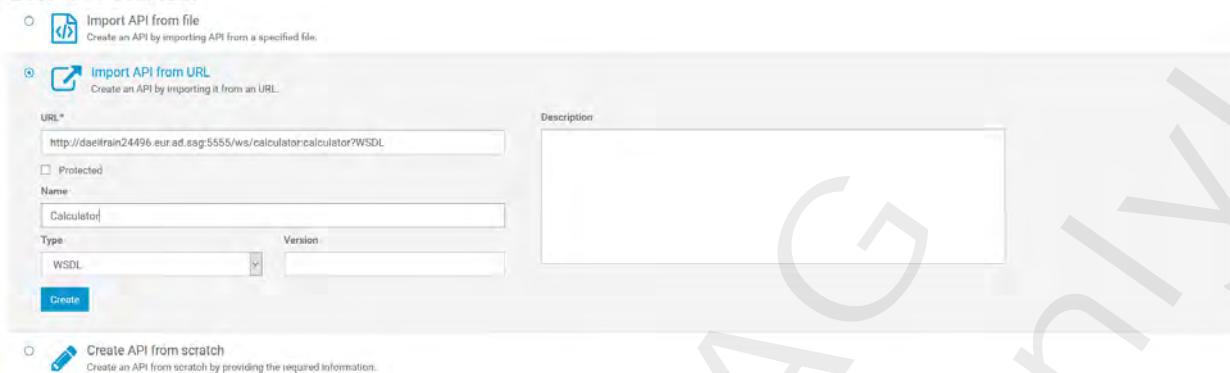
http://DAETRAIN24496.eur.ad.sag:5555/ws/calculator:calculator?WSDL

4. Login to the **API Gateway UI** as user **Sumala/manage** and create a new SOAP based API. Use the option **Import API from URL**.

Exercise 13: Using a Routing Policy Action

- a. URL: - as mentioned above –
- b. Name: Calculator
- c. Type: WSDL

Lets Get Started!



Click **Create**.

5. Navigate to Policies > Routing. Switch to Edit mode.



6. Switch to Edit mode. Remove the default configured Policy action Straight Through Routing.



7. Select Content-based Routing in the section Policy catalog.

Exercise 13:
Using a Routing Policy Action

8. To provide the routing definitions, open the text-file **CalculatorRouting.txt** which is located in folder **C:\Training\APIGateway\Exercise13**
9. Open the Content-based Routing policy action in **full-screen** mode. Provide the following properties:
 - a. **Default Route To.**
 - i. **Endpoint URI:** - copy and paste Native endpoint value –
 - ii. **SOAP Optimization Method:** None
 - iii. **HTTP Connection Timeout(seconds)** 30
 - iv. **Read Timeout (Seconds)** 30
 - v. **Pass WS-Security Headers:** - do not check this -
 - b. **SSL Configuration**
 - i. **Keystore Alias:** - leave empty -
 - ii. **Key Alias:** - leave empty –

The screenshot shows the 'Content-based Routing' configuration screen. Under 'Default Route To', the 'Endpoint URI' is set to 'http://localhost:5555/ws/calculator:calculator_calculator_Port'. The 'SOAP Optimization Method' is 'None', 'HTTP Connection Timeout (seconds)' is 30, and 'Read Timeout (seconds)' is 30. The 'Pass WS-Security Headers' checkbox is unchecked. Under 'SSL Configuration', both 'Keystore Alias' and 'Key Alias' fields are empty. A 'Rules' section is visible at the bottom.

10. Click the blue icon next to **Rules** and create the following Rules

- a. Rule to follow when calling the Add operation
 - i. **Name:** MyContentRoutingForAdd
 - ii. **XPathExpression:** - copy X-Path expression for Add –
 - iii. **Route To**
 1. **Endpoint URI:** - copy changed endpoint for Add operation -

Leave all other default values.

The screenshot shows the 'Content-based Routing' configuration screen with a new rule being added. In the 'Rules' section, a new rule is selected. The 'Name' field contains 'MyContentRoutingForAdd'. The 'XPath Expression' field contains '/soapenv:Envelope/soapenv:Body/cal:add_rewservice'. The 'Namespace' section shows 'Namespace Prefix' as 'calculator' and 'Namespace URI' as 'http://calculator/calculator'. The 'Route To' section has the 'Endpoint URI' set to 'http://localhost:5555/ws/calculator:calculator_calculator_Port', 'SOAP Optimization Method' as 'None', 'HTTP Connection Timeout (seconds)' as 30, and 'Read Timeout (seconds)' as 30. The 'Pass WS-Security Headers' checkbox is unchecked. Under 'SSL Configuration', both 'Keystore Alias' and 'Key Alias' fields are empty. At the bottom, there are 'Cancel' and 'Add' buttons, with 'Add' being highlighted.

Click Add.

11. Add a second rule for operation **multiply**. Provide the following properties:

Exercise 13:
Using a Routing Policy Action

- a. Rule to follow when calling the **Multiply** operation

- i. **Name:** MyContentRoutingForMultiply
ii. **XPathExpression:** - copy X-Path expression for Multiply –
iii. **Route To**
1. **Endpoint URI:** - copy changed endpoint for Multiply operation –
Leave all default values.

The screenshot shows the 'Content-based Routing' configuration dialog. A new rule is being added with the following details:

Name	XPath Expression	Endpoint URI
MyContentRoutingForAdd	/soapenv:Envelope/soapenv:Body/cal:adder_flowservice	http://Addint5555/ws/calculatorcalculator/calculator_calculator_Port

Add rule

Name: MyContentRoutingForMultiply

XPath Expression *: /soapenv:Envelope/soapenv:Body/cal:multiply_flowservice

Namespace

Namespace Prefix	Namespace URI
55ws/calculatorcalculator/calculator_calculator_Port	

Route To *

Endpoint URI *	SOAP Optimization Method	HTTP Connection Timeout (seconds)	Read Timeout (seconds)
Enter search terms to see suggestions	None	30	30

Pass WS-Security Headers

SSL Configuration

Keystore Alias: Enter search terms to see suggestions

Key Alias: Enter search terms to see suggestions

Rules *

Name	XPath Expression	Endpoint URI	Policy
MyContentRou...	/soapenv:Envelope...	http://Addint5...	
MyContentRou...	/soapenv:Envelope...	http://Multipl...	

Buttons: Cancel, Add, OK

Click **Add**. Click **OK**.

The screenshot shows the 'Content-based Routing' configuration dialog with the newly added rules listed:

Name	XPath Expression	Endpoint URI	Policy
MyContentRou...	/soapenv:Envelope...	http://Addint5...	
MyContentRou...	/soapenv:Envelope...	http://Multipl...	

Buttons: + Add rule

12. Save and activate the **CalculatorAPI**. Navigate to **API details > Technical information**.

Exercise 13: Using a Routing Policy Action

13. Open **SOAP UI** and create a new SOAP project with name **Calculator** using the Gateway endpoint value and adding .?wsdl.

http://DAETRAIN00749:5555/ws/Calculator/1?wsdl

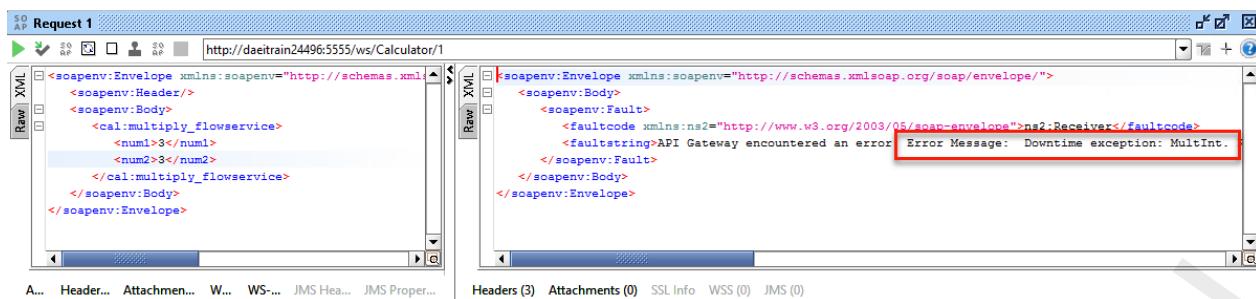
14. Within Projects navigate to **Calculator > calculator_calculator_Binder > add_flowservice > Request 1. Request 1** will be opened in a new window. Provide the following values:

- Num1:** 3
- Num2:** 3

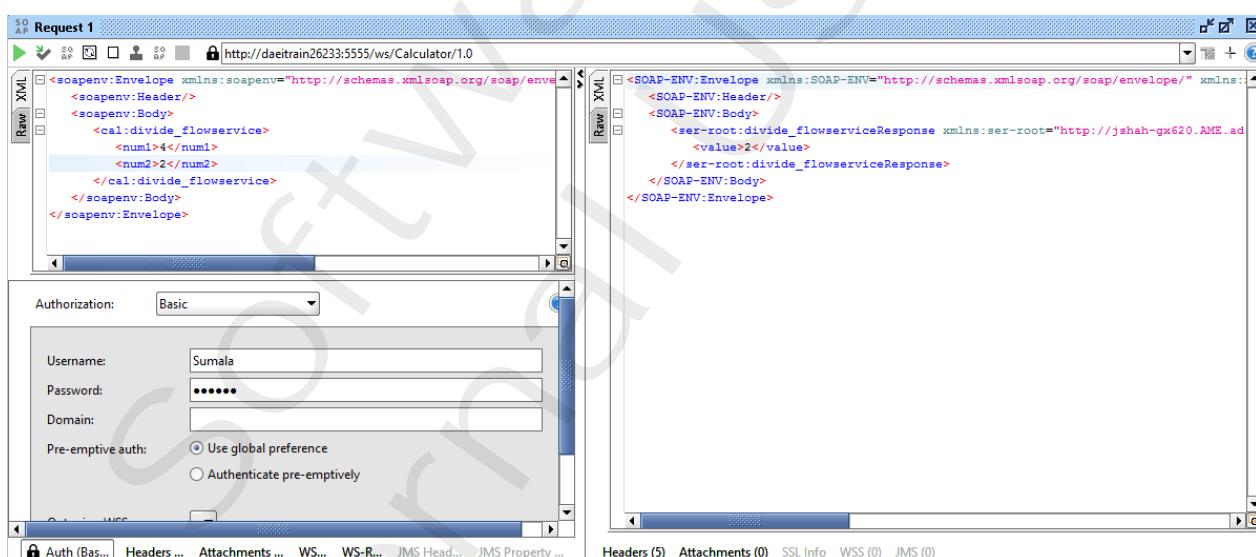
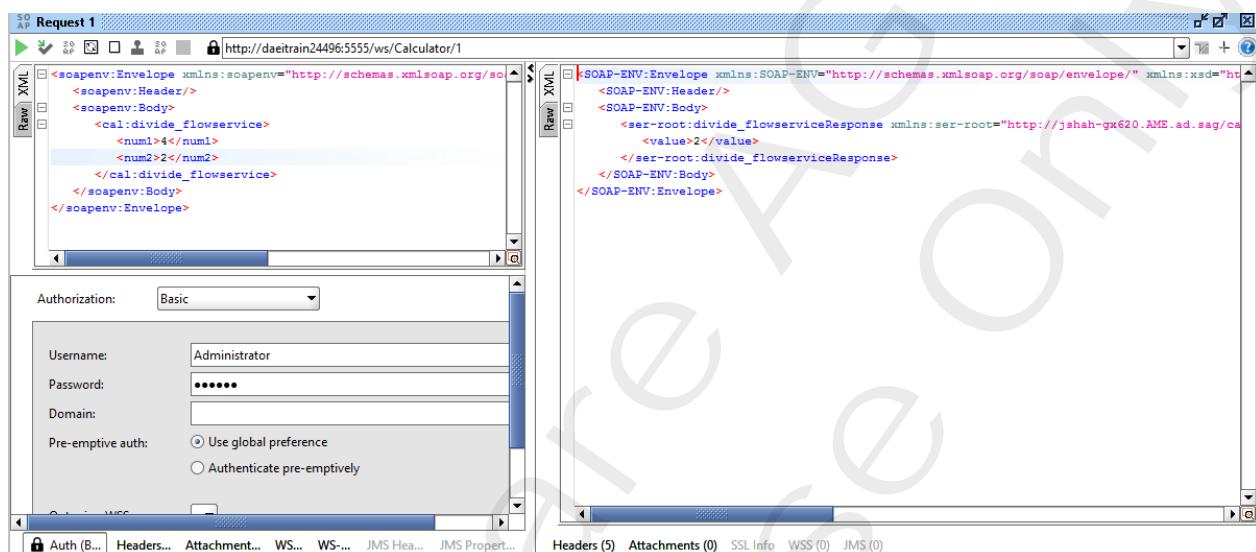
Hit the run button. SOAP UI will return an error, because host **AddInt** does not exist.

15. Try to run operation **multiply_flowservice**. This will also fail as SOAP UI tries to run service on host **MultInt**.

Exercise 13:
Using a Routing Policy Action



16. Try to run **divide_flowservice**. This will fail with a different error message, because of missing credentials. Add **Basic Authentication** for user **Sumala/manage**.



NOTE: You can enable proper execution of the add_flowservice and multiply_flowservice by adding appropriate entries for the hosts "AddInt" and "MultInt" to your hosts file.

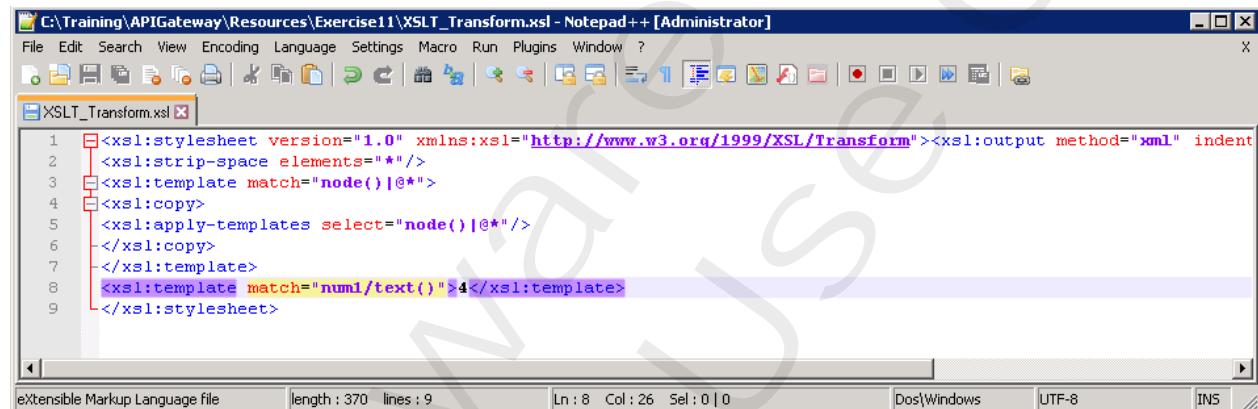
Exercise 14: Updating Content Using XSLT Stylesheet

Objectives

In this exercise, you will update the SOAP request content by concatenating a string to a particular XML element value, before sending request to the native Service.

Steps

2. Open Windows Services UI to double check that the following services, needed for API Gateway, are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1**
 - b. **Software AG Event Data store 10.1 – should be started together with IS**
3. Navigate to folder **C:\Training\APIGateway\Resources\Exercise14** and open the file **XSLT_Transform.xsl** in **Notepad++**.



```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:output method="xml" indent="yes"/><xsl:strip-space elements="*"/><xsl:template match="node()|@*"/><xsl:copy><xsl:apply-templates select="node()|@*"/></xsl:copy></xsl:template><xsl:template match="num1/text()">4</xsl:template></xsl:stylesheet>
```

This XML stylesheet will replace whatever is in element “num1” with the value “4”.

4. Login to the **API Gateway UI** as user **Sumala/manage** and reconfigure the SOAP based API **Calculator**.
5. From the list of APIs select the **API Calculator**. On the APIs Detail Page, click **Deactivate**. Switch to **Edit** mode.
6. Navigate to **Policies > Routing**. Remove the rules you have defined in **Content-based Routing**.

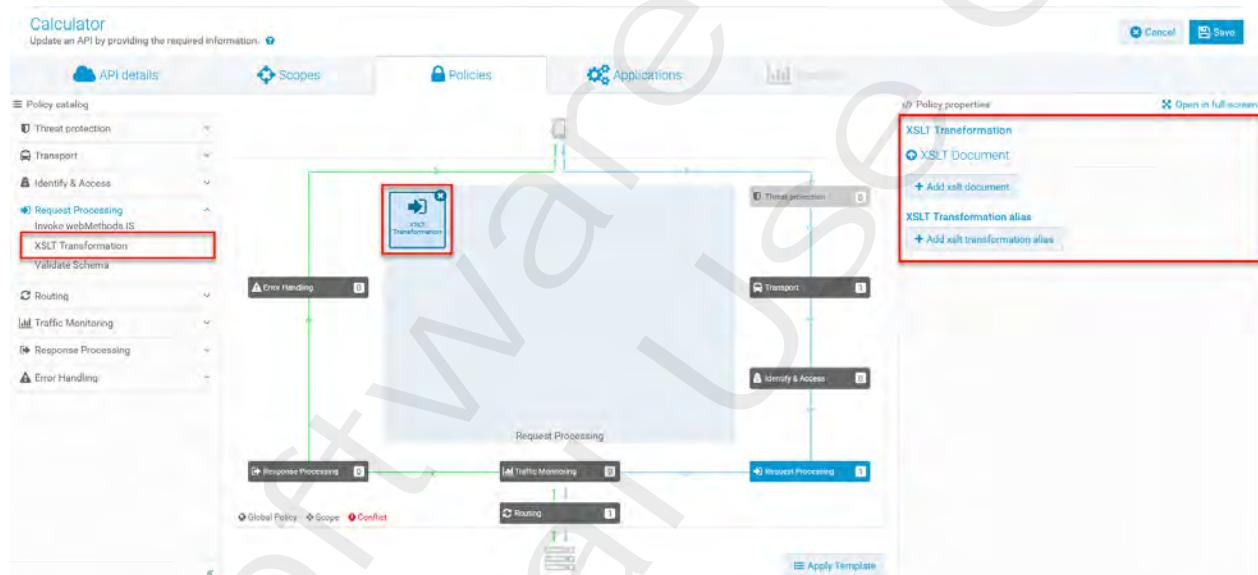


Exercise 14:
Updating Content Using XSLT Stylesheet

7. Navigate to Policies > Request Processing.



8. Select XSLT Transformation in the Policy catalog.



9. Within the Policy Properties click Add xsit document and provide the following values:

- Click Browse and search for the file **XSLT_Transform.xsl** in folder **C:\Training\APIGateway\Exercise14**. Do not provide properties for **XSLT Features**.

This is a screenshot of the 'Add xsit document' dialog box. It has two tabs: 'XSLT Transformation' (selected) and 'XSLT Document'. Under 'XSLT Transformation', there is a 'Browse' button to select the XSLT file 'XSLT_Transform.xsl'. Under 'XSLT Document', there is a 'Feature Name' input field and a 'Feature value' input field, both currently empty. At the bottom are 'Cancel' and 'Add' buttons, with 'Add' being highlighted.

Click **Add**.

10. Click **Save** and click **Activate**.

11. Open **SOAP UI** and navigate to the project **Calculator** you have configured in the last exercise.

Exercise 14:
Updating Content Using XSLT Stylesheet

12. Open the request **Request 1** for the operation **add:flowservice**. Run the service with the following properties:
- num1:** 5
 - num2** 5
 - Basic Authentication:** Sumala/manage

13. The SOAP response shows the result. During execution the XSLT Transformation changed `<num1>5</num1>` to `<num1>5</num1>`, making the result to `<ans>9</ans>`.

The screenshot shows a SOAP request and response in a web-based interface. The request (left pane) shows a SOAP envelope with a body containing a call to 'add_flowservice' with two parameters: 'num1' and 'num2', both set to 5. The response (right pane) shows a SOAP envelope with a body containing a 'ser-root:add_flowserviceResponse' element, which contains an 'ans' element with the value 9. A red box highlights the 'ans' element in the response. The interface includes an 'Authorization' dropdown set to 'Basic' and a form for entering credentials: Username (Administrator), Password (*****), Domain (empty), and Pre-emptive auth (radio button for 'Use global preference' selected). Below the interface, it says 'response time: 21ms (489 bytes)' and has tabs for Auth (Basic), Headers ..., Attachments ..., WS..., WS-R..., JMS Head..., and JMS Property ...

This page intentionally left blank

Software AG
Internal Use Only!

Exercise 15:

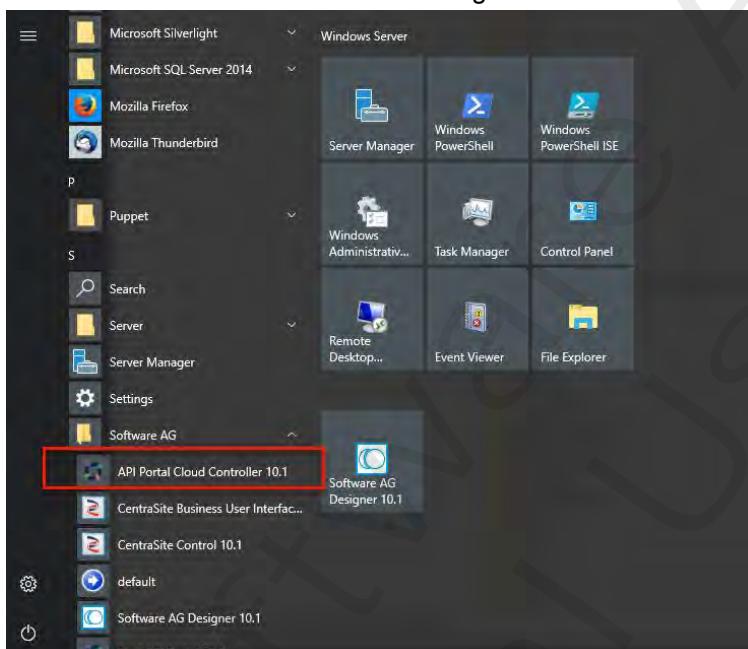
Setting up Integration with API Portal

Objectives

In this exercise, you will create a Package and a Plan to allow a set of Quality definitions like costs, availability of the service for a set of APIs.

Steps

1. Open Windows Services UI to double check that the following services, needed API Gateway, are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1**
 - b. **Software AG Event Data store 10.1 – should be started together with IS**
 - c. **Software AG API Portal 10.1**
2. Start **API Portal Cloud Controller** using Windows Start Menu



3. Provide the command **list** to verify the state of all runnables. Wait until all runnables are started. In case of problems use the command **stopall** and then the command **startall**.

NOTE: IT MAY TAKE A FEW MINUTES FOR ALL RUNNABLES TO START.

```
ACC+ localhost>list
Node localhost - 7 installed runnables.
ID          State    Version   Type
zoo_s       STARTED  10.0.2.1 com.aris.runnables.zookeeper-run-prod
postgres_s  STARTED  10.0.2.1 com.aris.runnables.PostgreSQL-run-prod
cloudsearch_s STARTED  10.0.2.1 com.aris.cip.y-cloudsearch-run-prod
elastic_s   STARTED  10.0.2.1 com.aris.runnables.elasticsearch-run-prod
kibana_s    STARTED  10.0.2.1 com.aris.runnables.kibana-run-prod
apiportalbundle_s STARTED 10.0.2.2 com.aris.bundles.apiportalbundle-run-prod
loadbalancer_s STARTED 10.0.2.1 com.aris.runnables.httpd.httpd-run-prod
```

- 3.4. Login to **API Gateway** as user **Administrator/manage**. Select **Administration**.

Exercise 15:
Setting up Integration with API Portal
Objectives

4.5. Navigate to General > Extended settings. Adapt the following setting by adding following values to the comma separated list:

a. **apiGroupingPossibleValues:**

Customer Management, Product Catalog

Click Save.

5.6. Navigate to Destinations > API Portal > Configuration

6.7. Provide the following properties:

- Name:** MyPortal
- Version**: 1.0
- Portal Configuration**
 - Base URL:** http://<your hostname as predefined in the Base URL of the Gateway configuration> :18101
 - Tenant:** default
 - Username:** Andy

Exercise 15:
Setting up Integration with API Portal
Objectives

- iv. **Password:** manage
- d. **Gateway Configuration**
- i. **Base URL:** - leave defaults -
 - ii. **Username:** Andy
 - iii. **Password:** manage

Administration

Implement and manage the general and security related configurations for API Gateway.

General Security Destinations System settings

API Gateway API Portal Configuration Events Audit Log CentraSite Configuration Events Database Digital Events Elasticsearch Configuration Events Email Configuration Templates SNMP Configuration

Basic information

Name: MyPortal

Version: 1.0

Portal configuration

Base URL: http://localhost:18101

Tenant: default

Username: Andy

Password: *****

Gateway configuration

Base URL: http://daeltrain24496.eur.ad.sag:5555

Username: Andy

Password: *****

Publish

Click **Publish**. You will get a success message as response.

10. Navigate to **Destinations > API Portal > Events**. We want to send **Performance Metrics** and **Events** to **API-Portal**. Set the following properties

- a. **Event Types:**
- i. **Error:** - select -
 - ii. **Lifecycle:** - select -
 - iii. **Policy violation:** - select -
- b. **Performance Metrics Data:**
- i. **Report performance data:** - select -
 - ii. **Publish Interval (minutes):** 2

Administration

Implement and manage the general and security related configurations for API Gateway.

General Security Destinations System settings

API Gateway API Portal Configuration Events Audit Log CentraSite Configuration Events Database Digital Events Elasticsearch Configuration Events Email Configuration Templates SNMP Configuration

Event types

Configure the event types to publish to the destination.

Error

Lifecycle

Policy violation

Performance metrics data

Configure the performance metric settings.

Report performance data

Publish interval (minutes): 2

Save

Click **Save**.

This page intentionally left blank

Software AG
Internal Use Only!

Exercise 16: Publishing API to API-Portal

Objectives

In this exercise, you will learn how to publish the virtual REST APIs to the API Portal. Andy as API Provider will publish the APIs belonging to the SAGTours application to API Portal in order to expose them to external consumers.

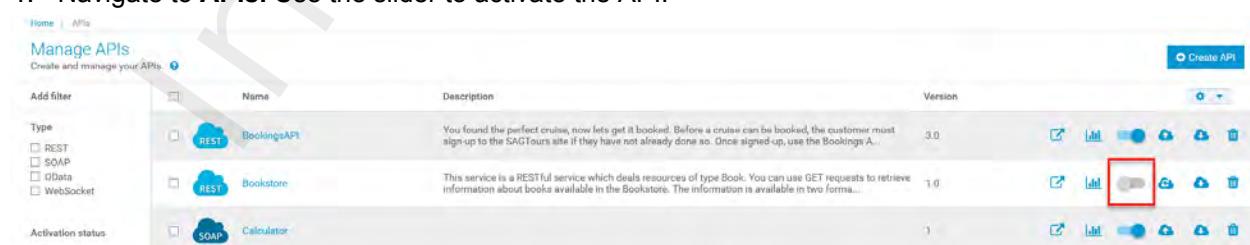
Steps

1. Open Windows Services UI to double check that the following services are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1**
 - b. **Software AG Event Data store 10.1 – should be started together with IS**
 - c. **Software AG API Portal**
2. Use the **API-Portal Cloud Controller** to verify that **Software AG API Portal** is running.
3. Login to **API Gateway** as user **Andy/manage**. Navigate to **APIs** and select the **API Bookstore**. Deactivate the API and switch over to **Edit** mode. Navigate to **Policies > Routing**. Add the **Outbound Authentication – Transport** policy. Set the following properties:
 - a. **Authentication Scheme:** Basic
 - b. **Authenticate using:** Custom Credentials
 - i. **Custom Credentials**
 1. **Username:** Andy
 2. **Password:** manage
 3. **Domain:** - leave empty –



Click **Save**.

4. Navigate to **APIs**. Use the slider to activate the API.



Exercise 16:
Publishing API to API-Portal

5. Click the cloud icon to publish the API.

The screenshot shows the 'Manage APIs' interface with three listed services: 'BookingsAPI', 'Bookstore', and 'Calculator'. The 'Bookstore' service is selected. Its toolbar includes several icons, one of which is a blue cloud icon, highlighted with a red box. This indicates the step to click to publish the API.

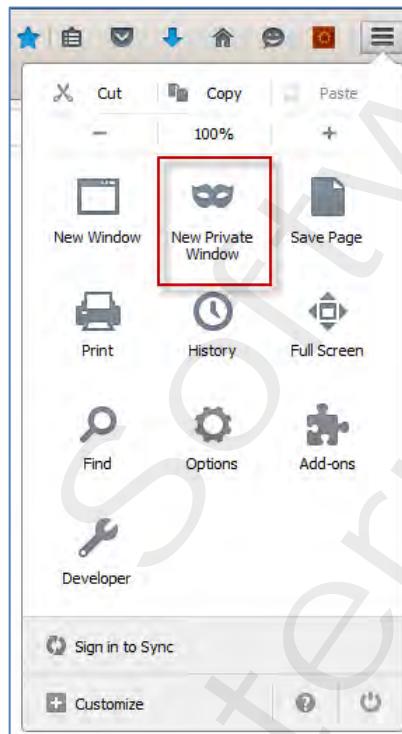
- a. Within the next dialog keep the default settings and click **Publish**.



You will get a success message and the publish icon is changed to a republish icon.

This screenshot shows the 'Manage APIs' interface again. The 'Bookstore' service is selected. In its toolbar, the blue cloud icon has been replaced by a white cloud icon with a blue border, indicating that the API is now published. This is highlighted with a red box.

6. In your **Firefox Browser** create a new Windows as **New Private Window**



This helps to avoid caching problems.

7. Open **API Portal** using the URL <http://localhost:18101/#default/home>. A shortcut **API-Portal** is configured for the Browser Mozilla Firefox.

Exercise 16:
Publishing API to API-Portal

8. Click the **API Gallery** button in the header to see the registered APIs.

9. Select the **Bookstore** API to look for the detailed information. Click **View details**. On the left hand side you find an overview listing the different sections of the API definition/description.

10. Navigate to **Try API>>**

Exercise 16: Publishing API to API-Portal

The screenshot shows the API Portal interface with the following details:

- REST RESOURCES:** GET /books, PUT /books.
- Bookstore:** A card with the URL `GET http://daetrain24496:5555/gateway/Bookstore/1.0/books`.
- Query parameters:** An input field for parameters.
- Header parameters:** Headers tab selected. Headers listed: Accept (application/json, text/xml) and Content-Type (text/xml, application/json).

11. Click **Test** to run the API with the default settings. Remember the Basic authentication requested by the native Bookstore API is provided with the **Outbound Authentication – Transport** policy we have defined on the API on API Gateway.

The screenshot shows the API Portal interface with the **Test** button highlighted in red.

12. Scroll down to the section **Response**. You will find the response code **200 OK** and within the **Payload** section you will find the list of book.

The screenshot shows the API Portal interface with the following sections visible:

- REST RESOURCES:** GET /books, PUT /books.
- Bookstore:** A card with the URL `GET http://daetrain24496:5555/gateway/Bookstore/1.0/books`.
- Query parameters:**
- Header parameters:**
- Request:**
- Response:** The **200 OK** status code is highlighted in red. The **Payload** section contains the XML response:

```
<?xml version='1.0'?>
<booklist>
  <book isbn='9781591488347'>
```

Exercise 17: Publishing SAGTours APIs to API-Portal

Objectives

In this exercise, you will learn how to publish the virtual REST APIs to the API Portal. Andy as API Provider will publish the APIs belonging to the SAGTours application to API Portal in order to expose them to external consumers.

Steps

13. Open Windows Services UI to double check that the following services are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1**
 - b. **Software AG Event Data store 10.1 – should be started together with IS**
 - c. **Software AG API Portal 10.1**
14. Use the **API Portal Cloud Controller** to verify that **Software AG API Portal** is running.
15. Login to **API Gateway** as user **Andy/manage**. Navigate to **APIs** and select the **API BookingsAPI**. Deactivate the API and switch over to **Edit mode**. Navigate to **API Details > Basic information**.

BookingsAPI
Update an API by providing the required information.

API details **Scopes** **Policies** **Applications** **Cancel** **Save**

Basic information

Name* BookingsAPI

Version 3.0

Maturity state Beta

Description You found the perfect cruise, now let's get it booked! Before a cruise can be booked, the customer must sign-up to the SAGTours site if they have not already done so. Once signed-up, use the Bookings API to book a cruise. You need the credentials you gave during sign-up to book the cruise. The credentials should be sent to the

Continue to provide Technical information for this API ▶

16. Provide the following properties and scroll down to the profile **API Portal Information**.

- a. **API Maturity Status:** Production
- b. **API Grouping:** Product Catalog

BookingsAPI
Update an API by providing the required information.

API details **Scopes** **Policies** **Applications** **Cancel** **Save**

Basic information

Name* BookingsAPI

Version 3.0

Maturity state **Production**

API grouping

Search

Transportation and Warehousing

Customer Management

Product Catalog

From the list of **API grouping** values select **Product Catalog** and click the + sign.

Exercise 17:
Publishing SAGTours APIs to API-Portal

The screenshot shows the 'BookingsAPI' configuration page. The 'Basic information' tab is selected. The 'Name' field contains 'BookingsAPI', 'Version' is '3.0', 'Maturity state' is 'Production', and 'API grouping' is set to 'Product Catalog'. The 'Description' field contains a note about booking a cruise. The 'Save' button at the top right is highlighted with a red box.

17. Provide a sample request body. Navigate to **API Details > Resources and methods > /bookings > POST**. Click **Add Request**. Provide the following properties

- Content type:** application/json
- Sample:** copy and paste the request sample out of the file **BookingAPI.txt** available in folder **C:\Training\456-71E\Exercise16**

The screenshot shows the 'Add Request' dialog for the '/bookings' endpoint. The 'Content type' dropdown is set to 'application/json' and the 'Add' button is highlighted with a red box. The 'Schema' tab is selected, showing a JSON sample code block.

Click **Add**. Click **Save**.

18. Navigate to **APIs**. Use the slider to activate the API.

The screenshot shows the 'Manage APIs' page. The 'BookingsAPI' service is listed under the REST category. The cloud icon for publishing is highlighted with a red box.

19. Click the cloud icon to publish the API. Within the next dialog keep the default settings and click **Publish**.

The screenshot shows the 'Publish API' dialog. It lists the API endpoint 'http://daeltrain24496:555/gateway/bookingsAPI/3.0' and the community 'Public Community'. The 'Publish' button at the bottom is highlighted with a red box.

You will get a success message and the publish icon is changed to a republish icon.

Exercise 17:
Publishing SAGTours APIs to API-Portal

The screenshot shows the 'Manage APIs' interface with two listed APIs:

- BookingsAPI**: REST type, version 3.0. Description: You found the perfect cruise, now lets get it booked. Before a cruise can be booked, the customer must sign up to the SAGTours site if they have not already done so. Once signed up, use the Bookings API to book the cruise. Activation status: Active.
- Bookstore**: REST type, version 1.0. Description: This service is a RESTful service which deals resources of type Book. You can use GET requests to retrieve information about books available in the Bookstore. The information is available in two forms... Activation status: Inactive.

A red box highlights the 'Create API' button in the top right corner.

20. Open **SearchCruise** and provide the following properties:

a. **API Maturity Status:** Production

b. **API Grouping:** Search

Save your changes. Activate and publish the API to **API-Portal**.

21. Open **Signu-UpAPI** and provide the following properties:

a. **API Maturity Status:** Production

b. **API Grouping:** Customer Management

c. **API Details > Resources and methods > /customer > POST:** copy and paste the request sample for POST out of the file **SignUpAPI.txt** available in folder **C:\Training\456-71E\Exercise16**

i. Click **Add**.

d. **API Details > Resources and methods > /customer/{customerId} > PUT:** copy and paste the request sample for PUT out of the file **SignUpAPI.txt** available in folder **C:\Training\456-71E\Exercise16**

i. Click **Add**.

Click **Save**.

Activate and publish the API to **API-Portal**.

The screenshot shows the 'Manage APIs' interface with the following updated list of APIs:

- BookingsAPI**: REST type, version 3.0. Description: You found the perfect cruise, now lets get it booked. Before a cruise can be booked, the customer must sign up to the SAGTours site if they have not already done so. Once signed up, use the Bookings API to book the cruise. Activation status: Active.
- Bookstore**: REST type, version 1.0. Description: This service is a RESTful service which deals resources of type Book. You can use GET requests to retrieve information about books available in the Bookstore. The information is available in two forms... Activation status: Inactive.
- Calculator**: SOAP type, version 1. Activation status: Inactive.
- EchoOnIDMZ**: SOAP type, version 1. Activation status: Inactive.
- getYacht**: SOAP type, version 1.0. Activation status: Inactive.
- SearchCruise**: REST type, version 1.0. Description: Searches for all cruises, or searches for a cruise that matches the specified criteria. The GET method has two forms: GET /cruises returns all cruises that are currently available. GET /cruises?... Activation status: Active.
- SignUpAPI**: REST type, version 1.0. Description: API for signing-up to SAGTours. The signing-up resources creates/updates/deletes a customer from SAGTours. Sign-up and Booking are protected via Basic Authentication. If an error occurs, the Sign-up ... Activation status: Inactive.

Red boxes highlight the 'Create API' button in the top right corner and the 'Publish' button for each active API entry.

The screenshot shows the 'Manage APIs' interface with the following updated list of APIs:

- BookingsAPI**: REST type, version 3.0. Description: You found the perfect cruise, now lets get it booked. Before a cruise can be booked, the customer must sign up to the SAGTours site if they have not already done so. Once signed up, use the Bookings API to book the cruise. Activation status: Active.
- Bookstore**: REST type, version 1.0. Description: This service is a RESTful service which deals resources of type Book. You can use GET requests to retrieve information about books available in the Bookstore. The information is available in two forms... Activation status: Inactive.
- Calculator**: SOAP type, version 1. Activation status: Inactive.
- EchoOnIDMZ**: SOAP type, version 1. Activation status: Inactive.
- getYacht**: SOAP type, version 1.0. Activation status: Inactive.
- SearchCruise**: REST type, version 1.0. Description: Searches for all cruises, or searches for a cruise that matches the specified criteria. The GET method has two forms: GET /cruises returns all cruises that are currently available. GET /cruises?... Activation status: Active.
- SignUpAPI**: REST type, version 1.0. Description: API for signing-up to SAGTours. The signing-up resources creates/updates/deletes a customer from SAGTours. Sign-up and Booking are protected via Basic Authentication. If an error occurs, the Sign-up ... Activation status: Inactive.

Red boxes highlight the 'Create API' button in the top right corner and the 'Publish' button for each active API entry.

22. In your **Firefox Browser** create a new Windows as **New Private Window** and open **API Portal**.

This helps to avoid caching problems.

23. Open the **API Portal** using the URL <http://localhost:18101/#default/home>. A shortcut **API-Portal** is configured for the Browser Mozilla Firefox.

24. Click the **API Gallery** button in the header to see the registered APIs.

Exercise 17:
Publishing SAGTours APIs to API-Portal

The screenshot shows the WEBMETHODS API Portal interface. At the top, there's a navigation bar with links for Home, API Gallery, API Packages, App Gallery, and Support. On the right side of the header, there's a 'My account' link. Below the header, the page title is 'API Gallery'. Underneath, there are two main sections: 'Customer Management' and 'Product Catalog'. Each section contains a card for an API. The first card, 'SignupAPI', is under 'Customer Management' and is described as a REST API for signing up to SAGTours. The second card, 'BookingsAPI', is under 'Product Catalog' and is described as a REST API for booking cruises. Both cards have a 'View details >>' button.

25. The default grouping is **API group** like **Customer Management, Product Catalog**,
26. We would like to review the APIs of the **SAGTours** application as API Provider. Login to **API Portal**. Navigate to **My account > Log in**. Login as user **Andy/manage**

The screenshot shows the login page of the WEBMETHODS API Portal. The header has the same navigation links as before. Below the header, there's a 'Log in' button which is highlighted with a red box. There's also a 'Register' button below it.

27. Try to find the APIs you have deployed to the **API Portal**. Use the **Search** location in the header bar and provide the following characters
a. replace **Search** with **Search** and wait or click the **Search** icon

The screenshot shows the search results for 'SearchCruise' in the API Portal. The search bar at the top has 'SearchCruise' typed into it. The results table shows one entry: 'SearchCruise' (REST API). The entire row for this API is highlighted with a red box. The table also shows other columns for 'Collaboration' and 'No results found'.

28. From the list of APIs select the API **SearchCruise**. Hover the mouse over the API and click. This will open the **SearchCruise** in Detail view. On the left hand side you find an overview listing the different sections of the API definition/description.

Exercise 17:
Publishing SAGTours APIs to API-Portal

The screenshot shows the API Gallery interface for the SearchCruise API. The left sidebar contains links for 'About SearchCruise', 'API DETAILS', 'API resources', 'API documents', 'API policies', and 'FURTHER INFORMATION'. A red box highlights the 'About SearchCruise' link. The main content area displays the 'SearchCruise' logo and the title 'About SearchCruise'. Below the title is a section titled 'Click here to edit.' followed by a detailed description of the API's search functionality. To the right, there are social sharing icons (Facebook, Google+, Email) and a sidebar with sections for 'What's next?', 'Rate this API' (No ratings), and 'List of followers' (Andy Roth).

29. Search and open **BookingsAPI**. Use the **API Resources** link to list the resources available within the **BookingsAPI**. Within **API Resources** you find a list of the Resources and Methods. Click on the paths to show details about a particular resource invocation. The combination of Methods and Resources as defined in **API Gateway**.

The screenshot shows the API Gallery interface for the BookingsAPI. The left sidebar contains links for 'About BookingsAPI', 'API DETAILS', 'API resources' (which is highlighted with a red box), 'API documents', 'API policies', and 'FURTHER INFORMATION'. The main content area displays the 'About BookingsAPI' page with a section titled 'Click here to edit.' and a detailed description of booking a cruise. Below this is a section titled 'API resources' with two resource entries: '/bookings' and '/bookings/{BookingID}'. Each entry has a 'Click here to edit.' link and a table with 'GET' and 'POST' methods. A red box highlights the '/bookings' resource entry. To the right, there are social sharing icons and a sidebar with sections for 'Rate this API' (No ratings), 'List of followers' (Andy Roth), and 'Edit'.

30. Click on the Method to show details about a particular resource invocation. The combination of Methods and Resources as defined in **API Gateway** including

- Parameters
- Request

Exercise 17: Publishing SAGTours APIs to API-Portal

The screenshot shows the API Portal interface. On the left, a sidebar lists sections like 'GETTING STARTED', 'API DETAILS', 'API resources', 'API documents', 'Access API', and 'FURTHER INFORMATION'. The main area displays the 'BookingsAPI' resource. It includes a 'Schema definitions' section with a 'null' value, a 'Sample request' section with a JSON object, and a 'Try API' button.

31. In order to use the Test functionality of the API Portal navigate back to the section **About the BookingsAPI**. The content of this section mentions that the customer needs to sign-up to the SAGtours before he can book a cruise.

Before a cruise can be booked, the customer must sign-up to the SAGTours site if they have not already done so.

32. Use the **Search** functionality again and search for the **SignupAPI**.

The screenshot shows two separate instances of the API Portal search results for 'SignupAPI'. Both results are highlighted with red boxes. Each result shows the 'SignupAPI' REST API, GET method, URL (http://daetrain24496:5555/gateway/SignupAPI), and a 'Collaboration' section with 'No results found'.

33. Navigate to the section **Try API >>** and select the REST Resource **POST/customer**.

Exercise 17: Publishing SAGTours APIs to API-Portal

The screenshot shows the WEBMETHODS API-Portal interface. The top navigation bar includes links for API Gallery, API Packages, App Gallery, and Support. Below the navigation, the REST Resources section lists several endpoints: POST /customer, DELETE /customer/{customerId}, GET /customer/{customerId}, and PUT /customer/{customerId}. The main focus is on the 'SignupAPI' configuration. It shows a POST method with the URL `http://daetrain24496.5555/gateway/SignupAPI/1.0/customer`. Under 'Header parameters', there are two entries: 'Accept' with value 'application/json' and 'Content-Type' with value 'application/json'. The 'Request payload' section contains a JSON object representing a customer:

```
{ "Customer": { "firstname": "Sumala", "surname": "Sumala", "title": "Mrs.", "emailAddress": "sumala@company.com", "customerAction": "CreateCustomer" } }
```

34. Provide header parameters. Open the section **Header Parameters** and make sure that values for Accept and for Content-Type are set

- Accept:** application/json
- Content-Type:** application/json

This screenshot shows the same configuration page as above, but with the 'Header parameters' section highlighted. The 'Accept' and 'Content-Type' fields are both enclosed in a red rectangular box, indicating they are the values being referred to in the exercise instructions.

This screenshot shows the configuration page again, but with the 'Request payload' section highlighted. This section contains the JSON object for creating a customer, which is identical to the one shown in the first screenshot.

35. Replace the values in the **Request Payload** with dummy-data:

- customer**
 - firstname:** Dev
 - surname:** Developer

Exercise 17:
Publishing SAGTours APIs to API-Portal

- iii. **title:** Mr
iv. **emailAddress:** developer@company.com

b. leave all the other values

</> Request payload

```
{  
  "customer": {  
    "firstname": "Dev",  
    "surname": "Developer",  
    "title": "Mr.",  
    "emailAddress": "developer@company.com",  
    "passportID": "12202123456789",  
    "passportExpiration": "2022.07.01",  
    "passportOrigin": "USA",  
    "dob": "1992.09.01"  
  },  
  "address": {  
    "street": "221B Baker St",  
    "city": "London",  
    "state": null,  
    "zipcode": "NW1 6XE",  
    "country": "UK"  
  }  
}
```

Browse... No file selected.

36. Click the **Test** button.

The screenshot shows the API-Portal interface for testing the SignupAPI. The 'Test' button is highlighted with a red box. The request payload is identical to the one shown in the previous step.

37. Review the **Response** and verify that the Response Code is 200.

The screenshot shows the API-Portal interface displaying the response from the API call. The Headers section is highlighted with a red box. The response code is 200, and the Headers include standard HTTP headers like Date, Server, and Content-Type, along with specific API-related headers such as com.eis.per.requestnumber, Cache-Control, and Content-Length.

38. We skip the step to verify that the customer has been added using the REST Resource **GET /customer/{customerId}** from the navigation section. You have done this in one of the former exercises using RESTClient.

Software AG
Internal Use Only!

This page intentionally left blank

Software AG
Internal Use Only!

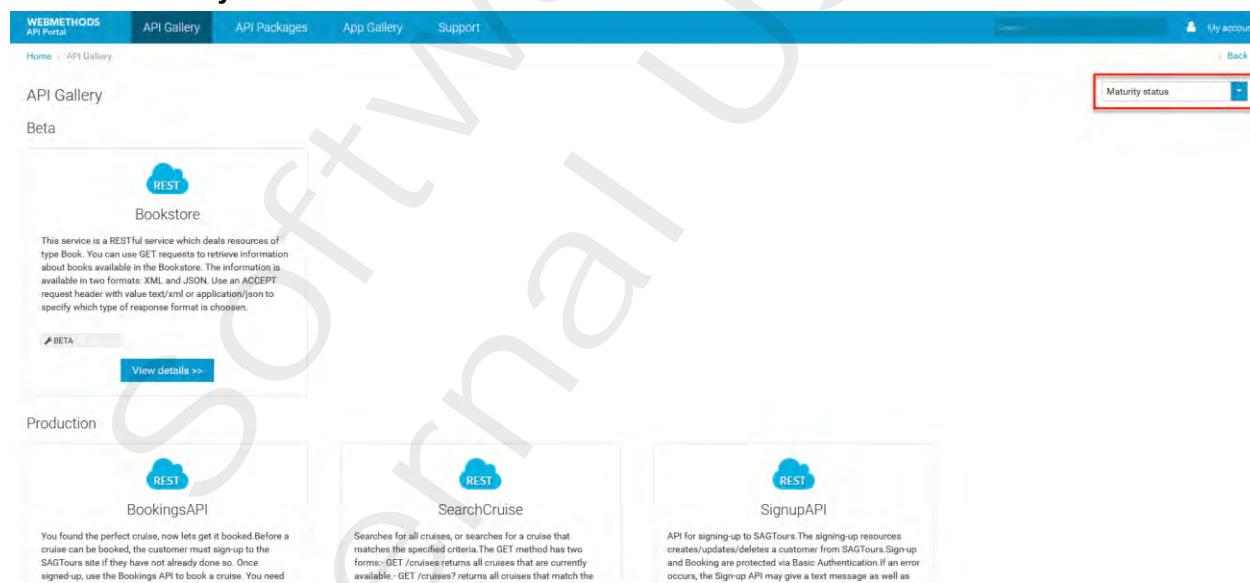
Exercise 18: Using API Portal as Consumer

Objectives

In this exercise, you will learn how to use API Portal as consumer.

Steps

1. Open Windows Services UI to double check that the following services are up and running.
If service is not running, start the service.
 - a. **hMailServer**
 - b. **Software AG Runtime 10.1**
 - c. **Software AG Integration Server 10.1**
 - d. **Software AG Event Data store 10.1 – should be started together with IS**
 - e. **Software AG API-Portal 10.1**
2. Use the **API Portal Cloud Controller** to verify that **API Portal** is running. Use the **list** command to verify that all services are up and running.
3. In your **Firefox Browser** create a new Windows as **New Private Window**
4. Open the **API Portal** and use the **API Gallery** to see all APIs provided by API Gateway. In case you are still logged as user **Andy**, logout.
5. Change the listing of the APIs by changing the ordering which is per default ordering by **API Group**. Click the **Grouped by** link and you will get a list of predefined grouping criteria. From the list select the value **Maturity status**.



The screenshot shows the Software AG API Portal interface. At the top, there's a navigation bar with links for WEBMETHODS API Portal, API Gallery, API Packages, App Gallery, and Support. On the right side of the header, there are 'My account' and 'Back' buttons. Below the header, the page title is 'Home : API Gallery'. The main content area is titled 'API Gallery' and 'Bookstore'. It features a 'REST' icon and a brief description of the Bookstore service. Below this, there are three service cards: 'Production' (BookingsAPI), 'SearchCruise', and 'SignupAPI'. Each card has a 'REST' icon and a brief description. The 'SearchCruise' card is currently selected, and its details are shown in a larger box. At the top right of the main content area, there's a dropdown menu labeled 'Grouped by' with a red box around it, containing options like 'Maturity status', 'Last updated', and 'Alphabetical'.

6. Select **SearchCruise**, click **View Details**. You will find the same set of information as you already have seen in the last exercise. The **About SearchCruise** informs about the procedure how to use the **SearchCruise API** and which search options are provided.

Exercise 18: Using API Portal as Consumer

The screenshot shows the 'About SearchCruise' section of the API documentation. It includes a brief description of the API, search parameters like 'startPort', 'endDate', and 'numDays', and a note about date formats. A sidebar on the right provides links to 'What's next?', 'Support forum', 'Get access token', 'Export API as', and 'Download Client SDK'.

- Within the **SearchCruise API** page click **API policies**. You will find out, that the API requires the usage of API Keys for Authentication and Authorization. We have done this configuration in Exercise 9.

The screenshot shows the 'API policies' section of the API documentation. It includes a note about requiring API keys for authentication and authorization, with two variants: passing the key as a header or as a query parameter. A red box highlights the 'API policies' link in the sidebar and the 'Identify & Authorize Application' section in the main content area.

- Within the **SearchCruise API** page click **Try API>** available within **Further Information**. A **Log in** page will pop up. In order to use APIs which are API Key protected you must have your own account. Right now you don't have an account. Therefore, click **Close**.

The screenshot shows a standard log in form with fields for 'Email' and 'Password'. Below the form is a 'Forgot password?' link. At the bottom is a blue 'Log in' button, with a red box highlighting the 'Close' button below it.

- In order to request an **API Key**, the guest user needs to be registered to the **API Portal**. Navigate to **My account** in the header line.

The screenshot shows the 'My account' section of the API portal. It includes a 'Log in' button and a 'Register' button. A red box highlights the 'Register' button.

Click the **Register** button.

- Make sure that **hMailServer** is running. If not running start **hMailServer**.
- The popup will ask you for user details in order to create a new account for the **API-Portal**. Provide the following information:
 - First Name:** Carl
 - Last Name:** Customer
 - Email:** customer@company.com

- d. **Password:** manage
- e. **Confirm Password:** manage
- f. **Company:** - leave empty-
- g. **Comment:** - leave empty-
- h. **Accept terms and conditions of site:** - select -

Register

The screenshot shows a registration form with the following fields:

- Email: customer@company.com
- Password:
- Confirm Password:
- Company: Company
- Comment:
- Accept terms of use

Below the form are two buttons: "Register" (highlighted in blue) and "Close".

Click **Register**. Click **Close**.

12. Open **Mozilla Thunderbird** and verify that an email has been sent by **API-Portal** to user **customer@company.com**, which says that the user account in **API-Portal** has been activated.

The screenshot shows an email in the Mozilla Thunderbird inbox with the following details:

- From: Me <sagtours@company.com>
- Subject: Activate your account
- To: Me <Customer@company.com>
- Date: 1:38 PM

The message body contains:

Hello Carl Customer,

Congratulations! You have successfully registered in API Portal.

You can activate your account by clicking this link : <https://daeitrain24496.eur.ad.sag:18102/#default/activateuser/email.verification.06e9a7a7-c5f6-4f9b-9347-43b3dcf1f0b7>

The above link will expire in 30 minutes.

Best Regards,
API Portal Team

*** This notification was sent automatically. Do not reply to this email.***

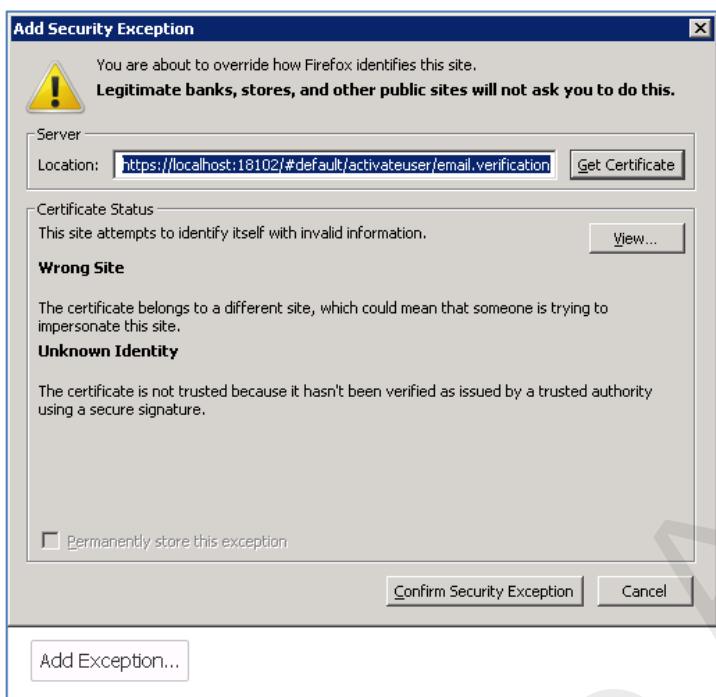
13. Copy the link in your email notification, open a new Browser Window as **New Private Window** and paste the URL. Agree that you understand the Risks.

--
Click **Advanced**.

Click **Add exception**

14. Click the button to **Confirm Security Exception**.

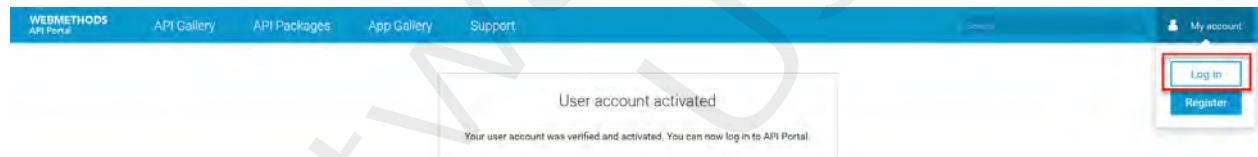
Exercise 18:
Using API Portal as Consumer



15. Now your user account is activated.



16. Use the **Log in** button to log in to **API-Portal** with your user credentials.



Use:

a. **Email:**

customer@company.com

b. **Password:**

manage

Log in

Login using your own account - [Create account](#)

[Forgot password?](#)

Log in

Close

17. After successful login the API-Portal header line looks different than it does for the unknown user. You can review your profile etc. clicking the icon on the very right hand side in the header.

Exercise 18:
Using API Portal as Consumer

The screenshot shows the WEBMETHODS API Portal homepage. At the top, there are links for API Gallery, API Packages, App Gallery, and Support. On the right side, there are icons for Applications, Dashboard, Manage Apps, Collaboration, and Communities. A user profile for 'Carl Customer' is displayed, with a red box highlighting the name.

18. You can review your profile by clicking the user name link.

The screenshot shows the user profile page for 'Carl Customer'. It includes a sidebar with links for User Settings (Account settings, Change password, Display settings), Applications (My applications), Activities (My activities stream), Reports (My scheduled reports), and Communities (My communities). The main area displays the user's profile picture, name 'Carl Customer', and a link to 'Manage account settings'. Below this, there is an 'Account settings' section with fields for First name ('Carl') and Last name ('Customer').

19. Search and open the **SearchCruise API**. Navigate to **What's next? > Get access token**.

The screenshot shows the 'About SearchCruise' page of the API details. On the right, there is a sidebar titled 'What's next?' which contains a 'Support forum' section with a red box around the 'Get access token' link. Other options in this section include 'Logout this token', 'Export API as', and 'Download Client SDK'. Below this are sections for 'Rate this API' (with a rating of 5 stars) and 'List of followers' (listing 'Andy Reilly').

20. In the Request API access token dialog provide the following details.

- Application Name:** Customer Application
- Application Description:** Customer portal to offer worldwide cruises
- Application redirect URI:** - leave empty -

The screenshot shows the 'Request API access token' dialog. It has fields for 'Application name' (set to 'Customer Application'), 'Application description' (set to 'Customer portal to offer worldwide cruises'), and 'Application redirect URI' (left empty). At the bottom are 'Close' and 'Request token' buttons.

Click the **Request Token** button.

21. You will get a Response Message that the Access Token has been requested successfully.

22. Use **Mozilla Thunderbird** and verify that an email has been sent by **API Gateway** which references the generated API token.

Exercise 18:
Using API Portal as Consumer

From: Me <sagtours@company.com>
Subject: Access token 'Customer Application' generated
To: Me <Customer@company.com>

Hello Carl Customer,

Your request for an access token has been processed. The details of the token is as follows:

Application name: Customer Application
Application description: Customer portal to offer worldwide cruises
APIs: SearchCruise

API Key
API access key: 6624d207-f411-441b-8176-c2a1e580feeb
Expiry date: Unlimited

OAuth2 Credentials
Client ID: 2e103f4d-dadb-4d18-93a1-7942bdde163c
Client secret: 173a240b-09d7-4f21-9629-2cf14deb5498

Best Regards,
API Portal Team

*** This notification was sent automatically. Do not reply to this email.***

23. The generated access token is also available in your **API Portal** user account. Open **Carl Customer -> Application > My applications**.

WEBMETHODS API Portal

Home User profile

USER SETTINGS Account settings Change password Display settings

APPLICATIONS My applications (highlighted with a red box)

ACTIVITIES My activities stream

REPORTS My scheduled reports

COMMUNITIES My communities

Carl Customer Manage account settings

Account settings

- From the list of applications select the new created application **Customer Application**. Review the generated **Access Token**.

Home Applications Customer Application Back

APP DETAILS Basic information Access tokens Associated APIs

Customer Application

Use the access token and build your application.

Basic information

Description: Customer portal to offer worldwide cruises
Purpose: API subscription

Access tokens

API key

API access key	6624d207-f411-441b-8176-c2a1e580feeb
Expiry date	Unlimited

OAuth2 credentials

Client ID: 2e103f4d-dadb-4d18-93a1-7942bdde163c
Client secret: 173a240b-09d7-4f21-9629-2cf14deb5498

- Go back to **API Gateway**. Login as User **Administrator/manage**. Verify that a new application has been added to **SearchCruise**. Navigate to **APIs > SearchCruise > Applications**.

Exercise 18:
Using API Portal as Consumer

Name	Description	Version
MyFirstApplication	Global consumer application based on user identifiers.	1.0
MyApplication.JustAPICKey	Application following the Approval workflow and without identifiers. Just the API Key is identifying...	1.0
Customer Application	Customer portal to offer worldwide cruises	1.0

25. Select the application **Customer Application**. Verify that the API Key is listed and **SearchCruise** is attached.

Customer Application
View application details, identifiers, and access token information along with the APIs associated with the application. [Edit](#)

Basic information

Name	Customer Application
Version	1.0
Owner	[redacted]
Created	2018-02-05 12:45:51 GMT
Description	Customer portal to offer worldwide cruises

Identifiers

Access tokens

API key
API access key 6624d207-f411-441b-8176-c2a1e580f0eb [Edit](#)

OAuth2 Credentials [Edit](#)

26. Go back to **API Portal** with credentials **customer@company.com/manage**. Search for **SearchCruise** API and navigate to **Further Information > Try API>>**. The new created Application is listed for the **SearchCruise**.

Home / API Gallery / SearchCruise / Try API [Back](#)

APPLICATION Customer Application

SearchCruise

GET http://daelittrain24496.5555/gateway/SearchCruise/1.0/cruises

cruise information like destination, ships, ports, ...

Query parameters

Name	Value	Description
endDate		Cruise end date, in the format YYYYMMDD. Selects cruises that end on this date. Combined with startDate, this parameter filters ...
startDate		Cruise start date, in the format YYYYMMDD. Selects cruises that start on this date. Combined with endDate, this parameter filters ...

Header parameters

Headers Security

Name	Value	Description
Accept		
Content-Type	x-Gateway-APIKey 6624d207-f411-441b-8176-c2a1e580f0eb	

27. Select the resource **GET /cruises**. Run API with the default Application named **Customer Application**.

- Verify that the Header Variable is set :
 - Accept application/json

Exercise 18: Using API Portal as Consumer

The screenshot shows the API Portal interface for a 'Customer Application'. On the left, there's a sidebar with 'APPLICATION' set to 'Customer Application' and 'REST RESOURCES' showing 'GET /cruises' and 'GET /cruises/{cruiseID}'. The main area is titled 'SearchCruise' with a 'GET' method and URL 'http://daetrain24496:5555/gateway/SearchCruise/1.0/cruises'. Below it says 'cruise information like destination, ships, ports, ...'. There are two buttons at the top right: 'Clean' and 'Test' (which is highlighted with a red box). Underneath, there are sections for 'Query parameters', 'Header parameters', and 'Headers'. The 'Headers' section has a table with one row where 'Accept' is set to 'application/json' (also highlighted with a red box). Other header fields shown are 'Content-Type' and 'x-Gateway APIKey'.

Click **Test**.

- b. Review the **Response**.

The screenshot shows the API Portal interface after clicking 'Test'. It displays the 'Response' section. The status code is '200'. Under 'Headers', it lists: Date : Mon, 05 Feb 2018 12:57:50 GMT, Server : Apache, com.aras.prf.requestNumber : 3, Cache-Control : no-store, Content-Length : 4530, x-xss-protection : 1; mode=block, X-Content-Type-Options : nosniff, Keep-Alive : timeout=5, max=100, Connection : Keep-Alive, Content-Type : application/json. Below that is a 'Payload' section with a single brace character '{'.

- c. Rerun the request several times in order to get **API Gateway** to collect some performance data.
d. Mark in the payload section one of the listed cruiseIDs for copy and paste.

Exercise 18: Using API Portal as Consumer

Response

200 OK

Headers

Date : Fri, 23 Dec 2016 14:33:04 GMT
Server : Apache
Content-Length : 4530
Keep-Alive : timeout=5, max=93
Connection : Keep-Alive
Content-Type : application/json; charset=UTF-8

Payload

```
{  
  "cruises": [  
    {  
      "cruiseID": "00001",  
      "title": "Alaska Whale Watching Photo Expedition",  
      "description": "Photographing the Whale in Alaska is a unique experience. Join us on this tour to this unique area. The cruise is a luxury tour for this that what to spend time with others with a like for see these unique animals of the sea. The tour will be conducted by a pro photography who will work with each of the members of the tour as needed.",  
      "startPort": "Juneau, Alaska",  
      "numDays": "8",  
      "startDate": "20150820",  
      "endDate": "20150827",  
      "roomTypes": [  
        {  
          "roomID": "IPD",  
          "roomDetails": {  
            "roomType": "Inside Premier Double",  
            "title": "Inside Premier Double cabin",  
            "description": "An Eco-friendly double room with desk, Tv and Shower",  
          }  
        }  
      ]  
    }  
  ]  
}
```

28. Navigate to test the resource **GET /cruises/{cruiseID}**. Provide all values and run the request to get 1 specific cruise

The screenshot shows the API Portal interface. In the top navigation bar, 'SearchCruise' is selected. Under 'REST RESOURCES', the URL 'GET /cruises/{cruiseID}' is highlighted with a red box. Below it, the 'Test' button is also highlighted with a red box. The 'Path parameters' section shows a 'cruiseID' field with the value '00001' entered, also highlighted with a red box.

Run the request using the **Test** button and verify the result.

29. Use the **Back** link to go back to the landing page for the **SearchCruise API**.

The screenshot shows the API Portal interface. In the top navigation bar, 'SearchCruise' is selected. Under 'REST RESOURCES', the URL 'GET /cruises/{cruiseID}' is shown. To the right, there is a 'Back' link highlighted with a red box.

30. Rate the API by selecting as many of the stars as you think it is.

The screenshot shows the API Portal interface. On the right side, there is a 'Rate this API' section. It includes a 5-star rating scale with 4 stars filled, an 'Average rating' of 5.0, and a 'Your rating' of 5.0. This entire section is highlighted with a red box.

31. Click the **Collaboration** link in the user menu to communicate with others.

Exercise 18: Using API Portal as Consumer

The screenshot shows the API Portal interface for the SearchCruise API. The top navigation bar includes links for API Portal, API Gallery, API Packages, App Gallery, and Support. On the left, there's a sidebar with sections like Getting Started, API Details, API resources, API documents, Access API, and Further Information. The main content area displays the API details for 'SearchCruise' with a REST logo. A search bar at the top of the content area says 'Search for all cruises or search for a cruise that matches the specified criteria'. Below the search bar, there's a link to 'About SearchCruise'. The top right corner features a user profile for 'Carl Customer' and a menu with icons for Applications, Dashboard, Manage Apps, Support, Follow, Export, Download, and a power icon. The 'Collaboration' icon, which looks like two people shaking hands, is highlighted with a red box.

Review the functionality of the collaboration inside the **API-Portal**.

This screenshot shows the user feed for 'Carl Customer' on the API Portal. The left sidebar shows 'My feed' with options like All company feed, My portal feeds, Groups, Find groups, Create group, Filters, My bookmarks, My liked items, and Create filter. The main feed area has tabs for Recent, Popular, and Active. It shows two recent posts: one from 'Carl Customer' stating they are now following 'SearchCruise', and another from 'Andy Roth' publishing 'SearchCruise' with the note 'The API SearchCruise is created.' Below each post are like, dislike, share, and more options buttons. To the right of the feed, there's a sidebar with a 'Rate this API' button and a note that says 'No ratings'.

32. Logout and review the ratings of the **SearchCruise API** as guest user. Open the **Menu** using the icon on the very right hand side. Within the menu, select the **Logout** icon in the lower left hand corner.

Exercise 19: Send Events to API Portal

Objectives

Scenario:

Login to API-Portal as API Administrator and try the API-Portal dashboard functionality to monitor the usage of API Portal. To monitor the runtime behavior of the API use API Gateway to review the performance profile of the API.

Steps

1. Open Windows Services UI to double check that the following services are up and running.
If service is not running, start the service.
 - a. **Software AG Runtime 10.1**
 - b. **Software AG Integration Server 10.1**
 - c. **Software AG Event Data store 10.1 – should be started together with IS**
 - d. **Software AG API-Portal 10.1**
 - e. **hMail-Server**
2. Use **API Cloud Controller** to verify that **API Portal** is up and running.
3. Logon to **API Gateway** as user **Administrator/manage**. In case the global policy **Transaction logging** is active, use the slider to deactivate the policy.

The screenshot shows the 'Policies' section of the API Cloud Controller. It lists a single policy named 'Transaction logging'. A red box highlights the 'Deactivate' button (a switch icon) for this policy. The policy description states: 'This is a system policy, which has log invocation policy and filters associated to log request or response payloads to a specified destination. These transactions are monitored and logged across all A...'. There are tabs for Threat protection, Global policies, and Policy templates.

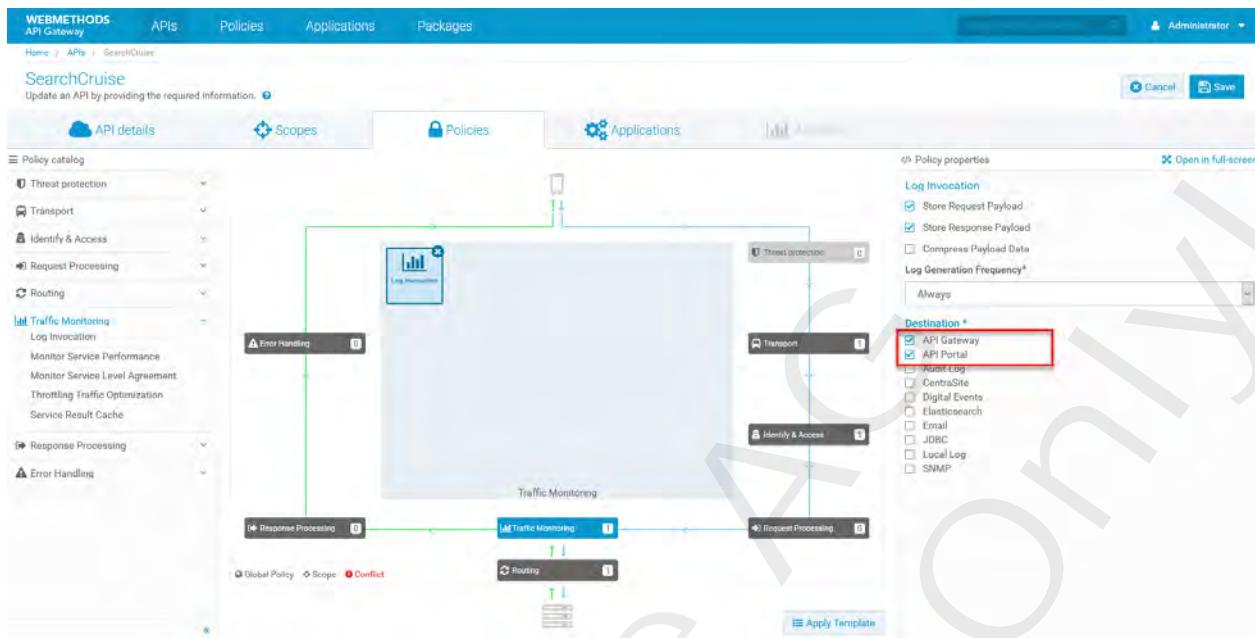
4. Navigate to APIs. Unpublish and deactivate the **SearchCruise API**.

The screenshot shows the 'Manage APIs' section of the API Gateway. It lists several APIs, including 'BookingsAPI', 'Bookstore', 'Calculator', 'EchoOnDMZ', 'getYacht', 'SearchCruise', and 'SignupAPI'. The 'SearchCruise' API is highlighted with a red box around its row. For each API, there are columns for Name, Description, Version, and various management icons. The 'SearchCruise' API is described as: 'Searches for all cruises, or searches for a cruise that matches the specified criteria. The GET method has two forms: - GET /cruises returns all cruises that are currently available. - GET /cruises?c...'. The 'Version' column shows '1.0' for most APIs and '3.0' for 'BookingsAPI'.

5. Switch to Edit mode. Navigate to **Policy Enforcement > Logging and Monitoring**. In case not already added to the API, select the policy **Log Invocation**.
 - a. Provide the following details:
 - i. **Payloads**
 1. **Request and Response**
 - ii. **Send Log Data**
 1. **API Gateway**

Exercise 19: Send Events to API Portal

2. API Portal



Click Save. Activate and publish SearchCruise to API Portal.

6. Login to **API Portal** as user **customer@company.com/manage** and run **SearchCruise** several times, with and without search or path parameter.

The screenshot shows the WebMethods API-Portal interface. On the left, there's a sidebar with sections for 'SANDBOX', 'APPLICATION' (set to 'Customer Application'), and 'REST RESOURCES' (listing 'GET /cruises'). The main area is titled 'SearchAPI' and shows a 'Path parameters' section with a table:

Name	Value	Description
cruiseID	00001	Cruise identifier. Given back as unique identifier when searching across all cruises.

Below this are sections for 'Query parameters', 'Header parameters', and a 'Request' preview. The 'Request' preview shows a GET request to 'http://sagbase:5555/ws/SearchAPI/1.0/cruises/00001' with 'Path parameters' 'cruiseID: 00001' and 'Request headers' 'Accept: application/json'.

Exercise 19: Send Events to API Portal

APPLICATION
Customer Application

REST RESOURCES
GET /cruises
GET /cruisesWithCruiseID

Query parameters

Header parameters

Request

Response

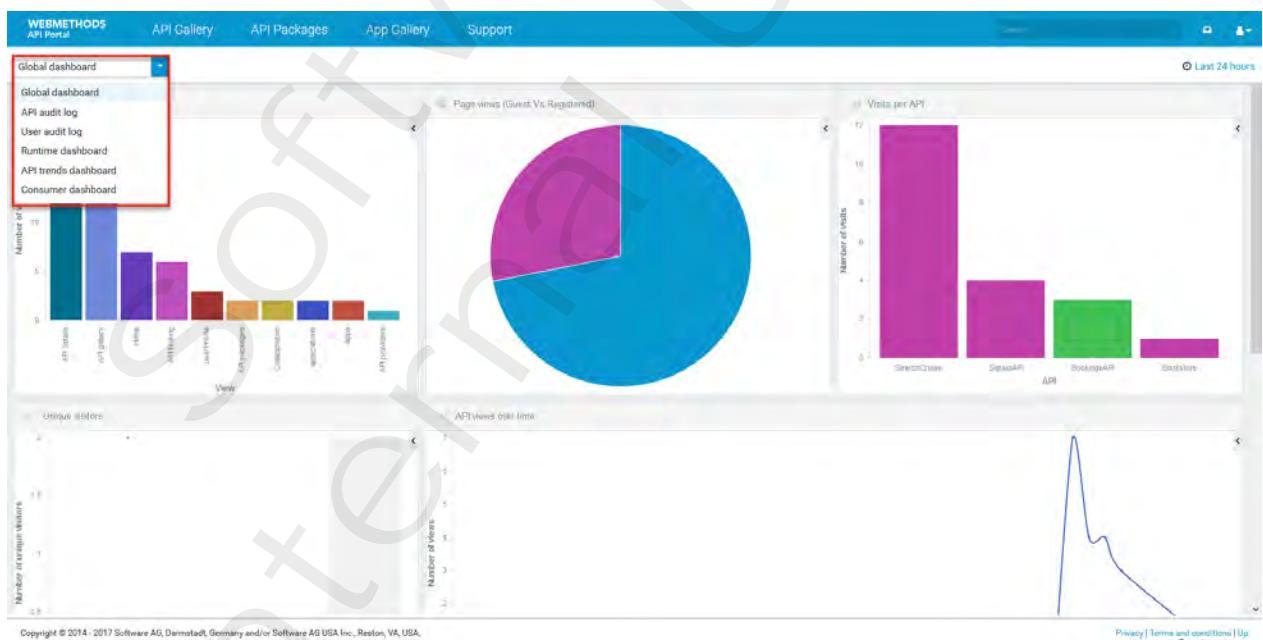
200 OK

Logout from API-Portal.

7. Login to **API-Portal** as user **john / manage**. Navigate to the User menu. **John** has additional entries in the user menu.

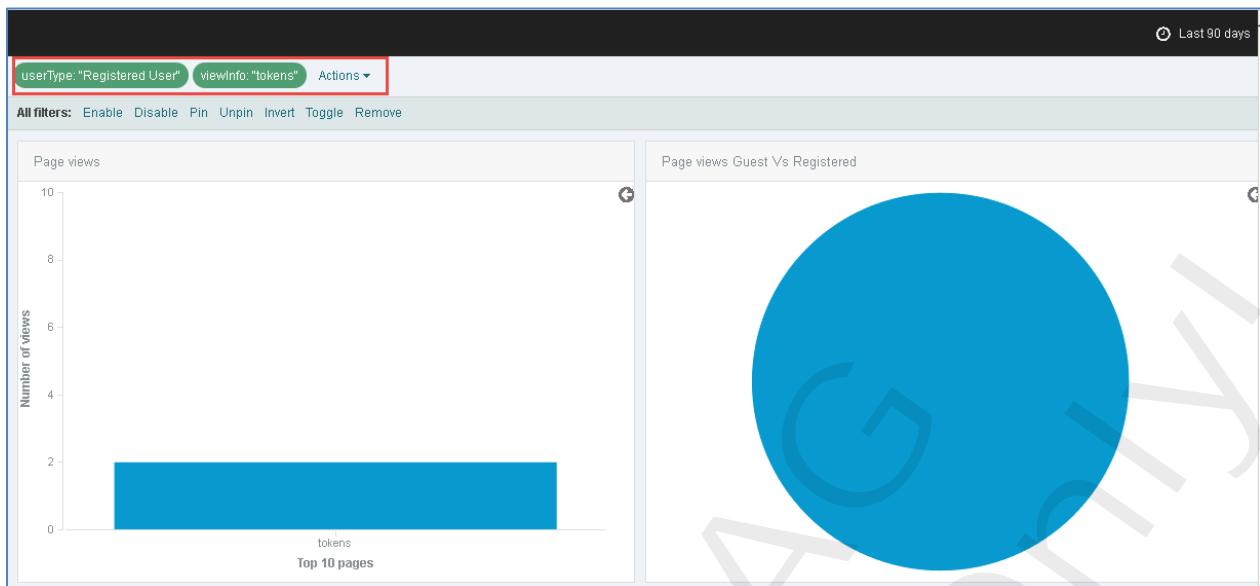
Click **Dashboards**.

8. On the **Dashboards** page you can use different groups of dashboards. The default dashboard **Global dashboard** is a collection of 6 different dashboards.

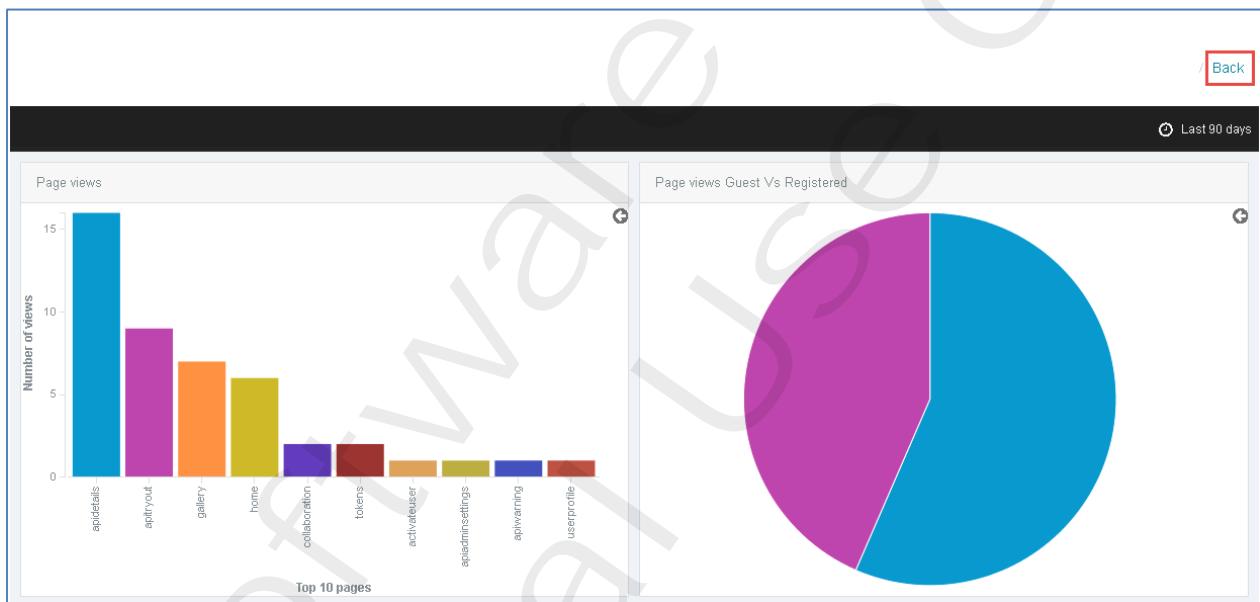


9. Hover the mouse over one of dashboards.
 - Within Page views this will show how often this page is viewed.
 - Click the page **API details**. This will drill down to just this page.

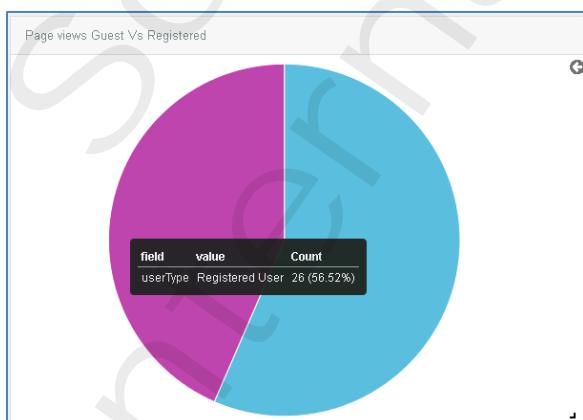
Exercise 19:
Send Events to API Portal



- c. Navigate to initial Configuration of **Global dashboard**. Now select **Page views (Guest vs. Registered)**

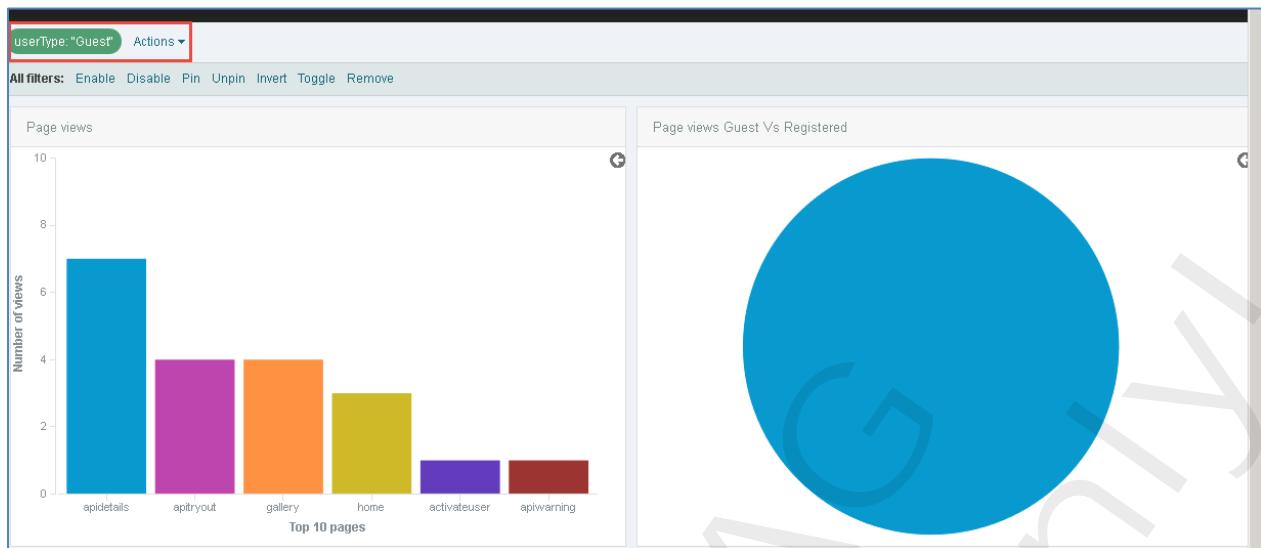


- d. Hover the mouse over one of the sections and review the details.



- e. Click one of the sections. Only information on views is displayed, see header line navigation.

Exercise 19:
Send Events to API Portal



10. Navigate to **Runtime dashboard**. Scroll down to the **Runtime events** dashboard. You will find **PerformanceMetrics** information.



11. Logout and login to **API Portal** as user API Consumer **customer@company.com/manage** and review metrics. Navigate to **User menu > Dashboards**.

Exercise 19:

Send Events to API Portal

The screenshot shows the WEBMETHODS API Portal Dashboard. At the top, there are navigation links: WEBMETHODS API Portal, API Gallery, API Packages, App Gallery, and Support. Below the navigation bar, the dashboard has a header with 'Dashboard' and dropdown menus for 'All Applications' and 'All APIs'. A timestamp 'Last 24 hours' is shown. The main area features four large numerical displays: 'Total requests' (10), 'Total requests(last 7 days)' (10), 'Total requests(last 6 months)' (10), and 'Total requests(last 1 hour)' (10). Below these are two charts: 'Requests per API' (a bar chart with one visible bar at 10) and 'Requests over time by API' (a line chart showing a sharp increase from 0 to 10 over a short period). A large watermark 'AG Only' is diagonally across the bottom right.

Select Customer Application.

12. Scroll down to **API request log**.

WEBMETHODS
API Portal API Gallery API Packages App Gallery Support

Dashboard Customer Application SearchCruise (1.0)

Last 24 hours

Time	apiname	apiversion	applicationname	applicationlp	request	response
05-Feb-2018 17:58:28	SearchCruise	1.0	Customer Application	-1	API required log	<pre>[{"cruiseID": "00001", "title": "Alaska Whale Watching Photo Expedition", "description": "Photographing the whale in Alaska is a unique experience. Join us on this tour to this unique area. The cruise is a luxury tour for this that what to spend time with others with a like for see these unique animals of the sea. The tour will be conducted by a pro photography who will work with each of the members of the tour as needed.", "startPort": "Juneau, Alaska", "numDays": "8", "startDate": "20150820", "endDate": "20150827", "roomTypes": [{"roomID": "P0", "roomDetail": {"roomType": "Inside Premier Double", "bed": "Inside Premier Double cabin", "description": "An Eco-friendly double room with desk, Tv and Shower", "maxOccupants": "2", "staRoomsOffered": "3", "numRoomsAvailable": "3"}, {"roomID": "OHD", "roomDetail": {"roomType": "Seawalk Premier Double", "bed": "Premier Double cabin with seawalk", "description": "An Eco-friendly double room with"}]}</pre>
05-Feb-2018 17:58:16	SearchCruise	1.0	Customer Application	-1		<pre>[{"cruiseID": "00001", "title": "Alaska Whale Watching Photo Expedition", "description": "Photographing the whale in Alaska is a unique experience. Join us on this tour to this unique area. The cruise is a luxury tour for this that what to spend time with others with a like for see these unique animals of the sea. The tour will be conducted by a pro photography who will work with each of the members of the tour as needed.", "startPort": "Juneau, Alaska", "numDays": "8", "startDate": "20150820", "endDate": "20150827", "roomTypes": [{"roomID": "P0", "roomDetail": {"roomType": "Inside Premier Double", "bed": "Inside Premier Double cabin", "description": "An Eco-friendly double room with desk, Tv and Shower", "maxOccupants": "2", "staRoomsOffered": "3", "numRoomsAvailable": "3"}, {"roomID": "OHD", "roomDetail": {"roomType": "Seawalk Premier Double", "bed": "Premier Double cabin with seawalk", "description": "An Eco-friendly double room with"}]}</pre>

13. You can use the **Last 24 hours** link to change the time interval.

The screenshot shows the API Portal interface with the 'Customer Application' selected in the top navigation bar. Below it, a search bar contains 'SearchCruise (1.0)'. On the right, there's a dropdown menu for selecting a time range, with 'Last 24 hours' highlighted by a red box. The main area displays a table of data with columns for 'Time' and 'Count'.

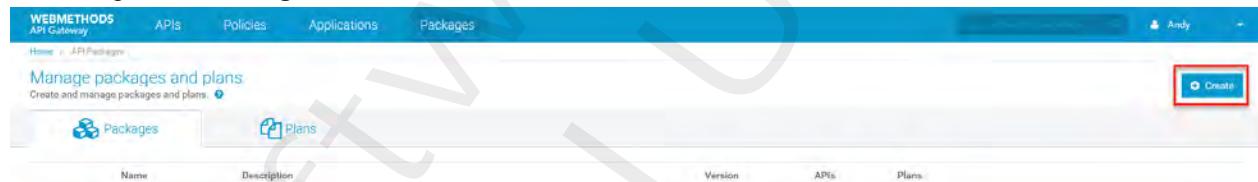
Exercise 20: Managing API Packages and Plans

Objectives

In this exercise, you will create a Package and a Plan to allow a set of Quality definitions like costs, availability of the service for a set of APIs. We would like to offer all APIs around the SAGTours application as a package.

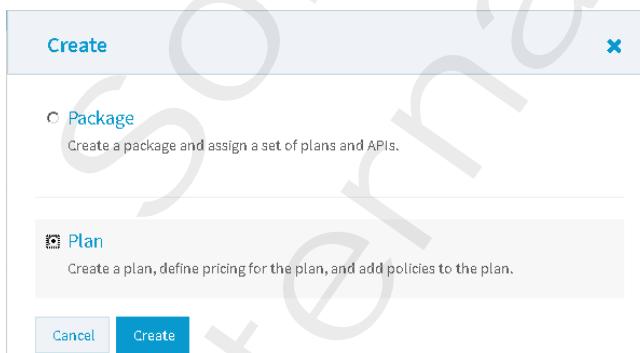
Steps

1. Open Windows Services UI to double check that the following services, needed API Gateway, are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1**
 - b. **Software AG Event Data store 10.1 – should be started together with IS**
 - c. **Software AG API Portal**
2. Login to **API Gateway** as user **Andy/manage**.
3. When subscribing to a package an application is generated and access to the APIs is controlled via API Key. Therefore we can remove the existing **Identify & Access** policies we have defined on the **SAGTours APIs**.
 - a. Deactivate and edit **SearchCruise**. Remove the policy **Identify & Access > Identify & Authorize Application**. Save the changes. Edit BookingsAPI and SignUpAPI as well and remove any Identify & Authorize policies that you may have.
4. Navigate to **APIs**. Activate and publish/republish **SearchCruise, BookingsAPI, SignUpAPI**.
5. Navigate to **Packages**. Click **Create**.



The screenshot shows the WEBMETHODS API Gateway interface. At the top, there's a navigation bar with tabs for 'APIs', 'Policies', 'Applications', and 'Packages'. Below the navigation bar, the page title is 'Manage packages and plans' with a subtitle 'Creates and manages packages and plans.' To the right, there's a user profile for 'Andy' and a '+' icon. In the center, there are two buttons: 'Packages' and 'Plans'. Below these buttons is a table with columns for 'Name', 'Description', 'Version', 'APIs', and 'Plans'. At the bottom right of the table area, there is a red box highlighting the 'Create' button.

6. Select **Plan** and click **Create**.



The screenshot shows a 'Create' dialog box. It has a 'Create' button at the top left and an 'X' button at the top right. There are two radio button options: 'Package' and 'Plan'. The 'Plan' option is selected. Below the radio buttons, there is a brief description: 'Create a plan, define pricing for the plan, and add policies to the plan.' At the bottom of the dialog box are two buttons: 'Cancel' and 'Create', with 'Create' being highlighted by a red box.

7. Create a new Plan
 - a. **Name:** Free
 - b. **Version:** 1.0
 - c. **Description:** Free plan

Exercise 20:
Managing API Packages and Plans

Create plan
Create a new plan with defined pricing and policies. [?](#)

Basic information
 Pricing
 Quality of service
Throttling traffic optimization

Name: Free
Version: 1.0
Description: Free plan
Icon: Browse [Change](#)

[Continue to add pricing >](#)

Cancel Save

Click Continue to add pricing

- d. Pricing
- i. Costs: 0
 - ii. Terms: Terms
 - iii. License: License

Create plan
Create a new plan with defined pricing and policies. [?](#)

Basic information
 Pricing
 Quality of service
Throttling traffic optimization

Cost: 0
Terms: Terms
License: Licensed

[Continue to Quality of Service >](#)

Cancel Save

Click Continue to Quality of Service

- e. Quality of Service -Throttling Traffic optimization
- i. Maximum request quota: 2
 - ii. Interval: 1
 - iii. Interval unit: Minutes
 - iv. Violation message: You have reached your Free plan quota

Create rule

Maximum request quota *
2

Interval: 2 Interval unit: Minutes

Violation message *
You have reached your Free plan quota

Cancel OK

Click OK. Click Save.

8. Navigate back to **Packages** and click **Create** to create a new **Package**
- a. Name: SAGTours
 - b. Version: 1.0
 - c. Description: SAGTours APIs

Exercise 20: Managing API Packages and Plans

Create package
Create a package and assign a set of plans and APIs. [?](#)

Basic information
 Plans
 APIs

Name: SAGTours
Version: 1.0
Description: SAGTours APIs
Icon: Browse

[Continue to add plans >](#)

Click Continue to add plans:

- d. Select plans: Free

Create package
Create a package and assign a set of plans and APIs. [?](#)

Basic information
 Plans
 APIs

Select plan: Free
[Continue to add APIs >](#)

Click Continue to add APIs>

- e. Search for **SearchCruise**. Click the API or the + sign to add the API.

Create package
Create a package and assign a set of plans and APIs. [?](#)

Basic information
 Plans
 APIs

Find APIs
Selected APIs
No APIs selected for registration

Name	Description	Version
SearchCruise	Searches for all cruises, or searches for a cruise that matches the specified criteria. The GET method returns a list of cruises, and the POST method creates a new cruise.	1.0

[+ Add API](#)

Add the APIs **SignupAPI** and **BookingsAPI**.

Create package
Create a package and assign a set of plans and APIs. [?](#)

Basic information
 Plans
 APIs

Find APIs
Selected APIs
Name: SearchCruise
Description: Searches for all cruises, or searches for a cruise that matches the specified criteria. The GET method returns a list of cruises, and the POST method creates a new cruise.
Version: 1.0
Name: SignupAPI
Description: API for signing up to SAGTours. The signing-up resources creates/updates/deletes a customer from SAGTours.
Version: 1.0
Name: BookingsAPI
Description: You found the perfect cruise, now lets get it booked. Before a cruise can be booked, the customer must be signed up.
Version: 3.0

Click Save.

9. Click **Activate**. Click **Yes**.

Exercise 20: Managing API Packages and Plans

The screenshot shows the 'Basic information' section for the package 'SAGTours'. The 'Activate' button at the top right is highlighted with a red box. Below it, there are sections for 'Plans' and 'APIs'. The 'Plans' section shows a single plan named 'Free'. The 'APIs' section lists two APIs: 'SignupAPI' and 'BookingAPI', both marked as 'REST'.

Review the just created Package.

The screenshot shows the 'Manage packages and plans' page. The 'Plans' tab is selected. A table lists the package 'SAGTours' with details: Name: SAGTours, Description: SAGTours APIs, Version: 1.0, APIs: 3, Plans: 1. The 'Activate' button at the top right is visible.

10. In order to publish the new created Package we need to first publish the allocated APIs. This we have already done in one of the former exercises. Select the new created package **SAGTours** and publish the package.

The screenshot shows the 'Manage packages and plans' page. The 'Plans' tab is selected. A table lists the package 'SAGTours' with details: Name: SAGTours, Description: SAGTours APIs, Version: 1.0, APIs: 3, Plans: 1. The 'Publish' button at the bottom right is highlighted with a red box.

Click **Yes**.

11. Open the **API Portal** as New Private Window

12. Navigate to **API Packages**

The screenshot shows the 'API Packages' section of the API Portal. The package 'SAGTours' is selected, indicated by a blue border. The 'Show details' button at the bottom left is highlighted with a blue box.

13. Select the package **SAGTours** and click **Show details**.

Review. The listed **APIs** and **Plans** to be the same as configured in API Gateway.

Exercise 20: Managing API Packages and Plans

The screenshot shows the SAGTours API package management interface. At the top, there's a navigation bar with links to Home, Packages, and SAGTours. Below that is a sidebar with PACKAGE, About SAGTours, APIs, and Plans. The main content area features a logo for SAGTours and sections for APIs and Plans.

APIs: Shows three APIs: BookingsAPI (REST, Untitled), SearchCruise (REST, Untitled), and SignupAPI (REST, Untitled).

Plans: Shows one plan: Free (Untitled). It indicates a cost of 0 and mentions reaching the free plan quota of 2.

Within the plan **Free** scroll down click **Subscribe**.

This screenshot shows the details of the Free plan. It includes fields for Cost (0), Throttling (Validation Interval: 1 minute, Violation Message: You have reached your free plan quota, Maximum Request Quota: 2), and License (2 licenses available). At the bottom, there are Terms and conditions and Terms links, and a prominent blue "Subscribe" button which is highlighted with a red rectangle.

14. Subscribing to a plan creates an **API Access token** which requires to have a valid user account in **API-Portal**. A user account for user **customer@comapny.com** you have already created in **API-Portal**. On the screen **Log in** provider the following credentials

- E-mail:** customer@company.com
- Password:** manage

This screenshot shows a log in form. It has fields for Email (customer@company.com) and Password (*****). Below the password field is a "Forgot password?" link. At the bottom are two buttons: a large blue "Log in" button and a smaller "Close" button.

Click **Log in**.

15. Click again **Subscribe**. On the screen **Request API access token**, provide the following Parameters.
- Application name:** SAGTours app

Exercise 20: Managing API Packages and Plans

b. Application description: application using SAGTours APIs

Request API access token

Please provide the required information below to receive a personalized access token via e-mail.

Application name
SAGTours app

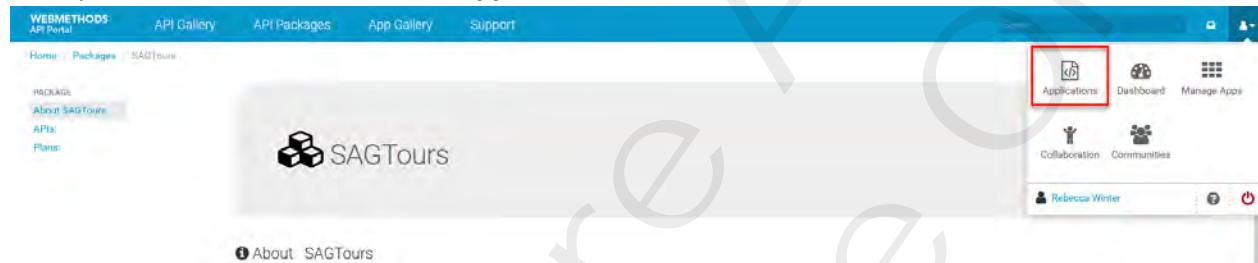
Application description
application using SAGTours APIs

Application redirect URI
http://sagtours.com/correctURI

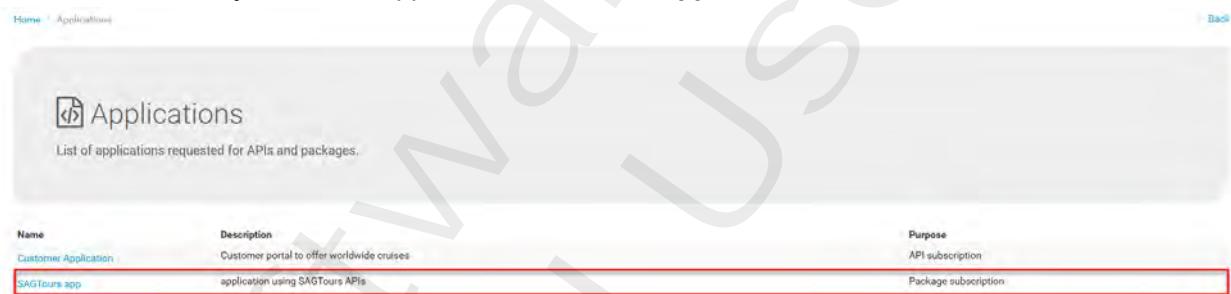
Close **Request token**

Click Request token. Click Close.

16. Open the user's menu and select Applications.



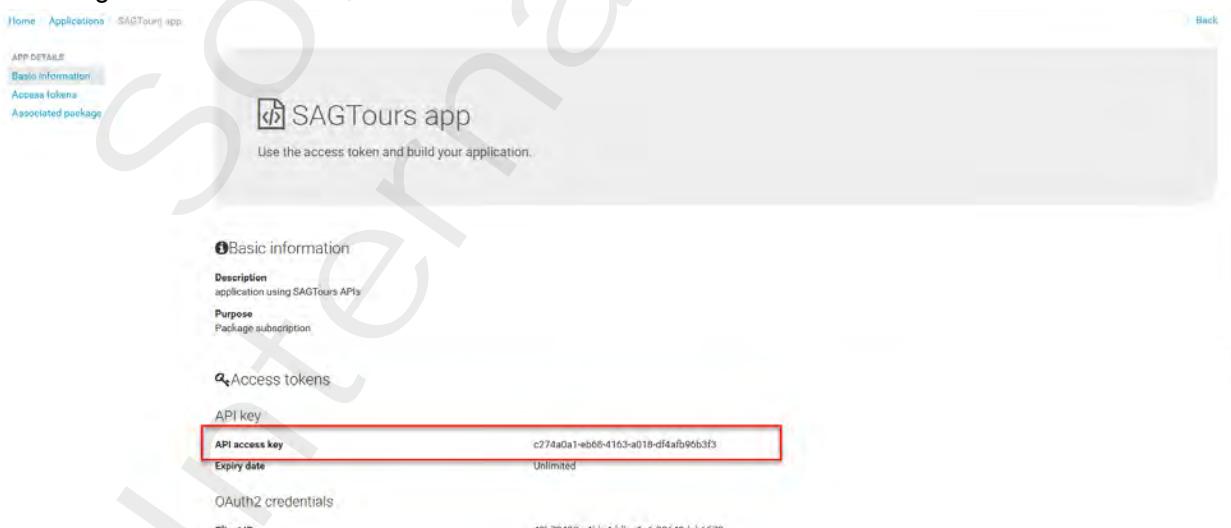
17. You will find the just created application in the list of Applications.



Name	Description	Purpose
Customer Application	Customer portal to offer worldwide cruises	API subscription
SAGTours app	application using SAGTours APIs	Package subscription

Select the application SAGTours app.

18. The generated Access tokens is listed.



API key	Expiry date
API access key	c274a0a1-eb6b-4163-a018-df4af90b3f3
Expiry date	Unlimited

19. Navigate to Associated packages. Select the package SAGTours. Select the API SearchCruise.

This will open the API in Detail view. Navigate to Try API>>. Within Application you find the API Keys and OAuth2 tokens for SAGTours app and Customer Application, which we have created already.

Exercise 20: Managing API Packages and Plans

Select SAGTours app (Subscription) with the list of API keys.

The screenshot shows the API Gallery interface for the SearchCruise API. The application dropdown is set to "SAGTours app(Subscription)". In the "Header parameters" section, the "x-Gateway-APIKey" header is listed with a value of "c274a0a1-eb58-4163-a016-df4aef5d4312". This header is highlighted with a red box.

20. Run the request clicking the **Test** button. Run the request several times until you see the expected error message.

The screenshot shows the API Gallery interface for the SearchCruise API. The application dropdown is set to "SAGTours app(Subscription)". In the "Response" section, the following error message is displayed: "429 Service invocation for this API was rejected based on Throttling policy applied, response code: 429".

21. Navigate to **Packages > APIs**. Select the **SignupAPI**. Navigate to **Try API>>**. You will find the **SAGTours app** listed, as well as the **API key** added to request as header parameter. We want to add a new customer to the SAGTours application. From the list or resources select the resource **POST /customer**.

- Provide following Header parameters
 - Accept:** application/json
 - Content-Type:** application/json

- b. Scroll down to the **Request payload** section. Within the payload provide the following properties
- Firstname: Carl
 - Surname: Customer
 - Title: Mr.
 - emailAddress: customer@company.com

```
{ "customer": { "firstname": "Carl", "surname": "Customer", "title": "Mr.", "emailAddress": "customer@company.com", "passportID": "1220123456789", "passportExpiration": "2022.07.01", "passportOrigin": "USA", "dob": "1992.09.01" }, "address": { "street": "221B Baker St", "city": "London", "state": null, "zipcode": "NW1 6XE", "country": "UK" } }
```

Click Test.

The response should show response code 200.

200 Headers

```
Date : Tue, 06 Feb 2018 12:03:07 GMT
Server : Apache
com.aris.per.requestnumber : 3
Cache-Control : no-store
Content-Length : 298
x-xss-protection : 1; mode=block
X-Content-Type-Options : nosniff
Keep-Alive : timeout=5, max=100
Connection : Keep-Alive
Content-Type : application/json
```

Payload

```
{ "customer": { "emailAddress": "customer@company.com", "title": "Mr.", "firstname": "Carl", "surname": "Customer", "dob": "1992.09.01", "passportID": "1220123456789", "passportExpiration": "2022.07.01", "passportOrigin": "USA" }, "address": { "street": "221B Baker St", "city": "London", "state": null, "zipcode": "NW1 6XE", "country": "UK" } }
```

22. We want to review the detailed information for this account. From the list or resources select the resource **GET /customer/{customerID}**.

- Provide the **Path parameter**
 - customerID:** customer@company.com
- Remember accessing an account needs Basic authentication. Within the section **Header Parameter**, select the **Security tab**. Provide the following properties:
 - Type:** Basic Authentication
 - User name:** customer@company.com
 - Password:** manage

Exercise 20: Managing API Packages and Plans

The screenshot shows the SAP API Management interface for the SignupAPI. In the 'Path parameters' section, the 'customerID' parameter is set to 'customer@company.com'. In the 'Header parameters' section, the 'Security' tab is selected, showing 'Basic Authentication' with 'User name' and 'Password' fields filled. A red box highlights the 'Security' tab.

Click Update.

23. Within **Header parameter** change back to **Headers**. A new parameter **Authorization** has been added

The screenshot shows the SAP API Management interface for the SignupAPI. In the 'Header parameters' section, the 'Headers' tab is selected, showing the 'Authorization' header with the value 'Basic Y3VzdG9tZXJAY29tcGFueS'. A red box highlights the 'Headers' tab and another red box highlights the 'Authorization' header value.

24. Click **Test** to run the request. The response code is **200** and the payload shows the new created account.

The screenshot shows the SAP API Management interface for the SignupAPI. The 'Test' button is highlighted with a red box. Below it, the 'Request' and 'Response' sections are shown. The 'Request' section displays the API endpoint 'GET http://daetrain24496:5555/gateway/SignupAPI/1.0/customer/{customerId}' and the 'Authorization' header with value 'Basic Y3VzdG9tZXJAY29tcGFueS'. The 'Response' section shows a status code of 200 and the response headers 'Date: Tue, 06 Feb 2018 12:06:16 GMT' and 'Server: Apache'.

This page intentionally left blank

Software AG
Internal Use Only!

Exercise 21:

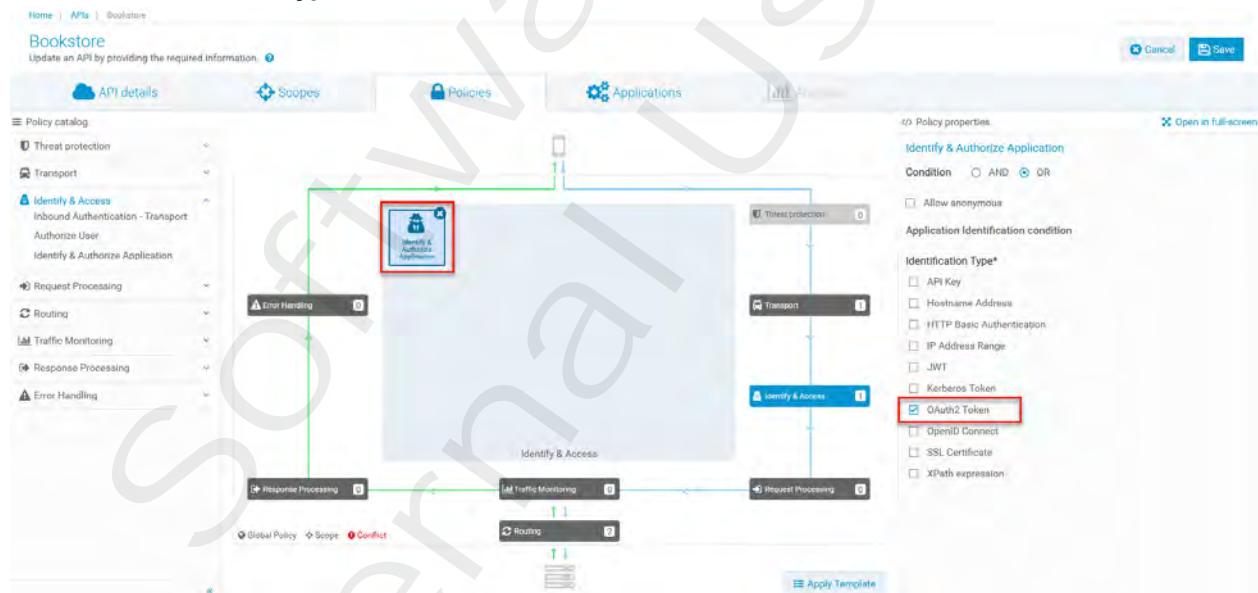
Try out API secured with OAuth2 token

Objectives

In this exercise, you will use API Portal Tryout function to access an **OAuth2** protected APIs published from API Gateway. We will enforce API Gateway to create a corresponding bearer token

Steps

1. Open Windows Services UI to double check that the following services, needed API Gateway, are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1**
 - b. **Software AG Event Data store 10.1 – should be started together with IS**
 - c. **Software AG API Portal 10.1**
2. In order to have API Gateway validating an **OAuth2 token**, we will need to enforce this in a corresponding **Identify & Access** policy. We will use the **Bookstore** API and enhance the API with adding enforcing a valid OAuth2 token. Logon to **API Gateway** as user **Andy/manage**. Navigate to **APIs**. From the list of APIs select **Bookstore**. Unpublish and deactivate the API and switch to **Edit** mode.
3. Navigate to **Policies > Identify & Access**. Select **Identify & Authorize Application** to be added to the API. Provide the following properties:
 - a. **Identification Type:** OAuth2 Token



Click **Save**.

4. Activate and republish **Bookstore** API to **API Portal**.
5. Login to **API Portal** as consumer **customer@company.com** with password **manage**. Use the Search functionality to get the detail view of the **Bookstore** API. Within in the left hand part of the page navigate to **API Details > API policies**. This shows that the API requires **OAuth2 token** authentication and explains how to obtain access to **Bookstore** API.

Exercise 21:
Try out API secured with OAuth2 token

The screenshot shows the 'About Bookstore' page of the Bookstore API documentation. On the left sidebar, 'API policies' is highlighted with a red box. In the main content area, there is a section titled 'Evaluate OAuth2 Token' which contains instructions for obtaining an access token. A red box highlights this section.

6. The properties you need in order to generate the **OAuth2 token** are part of an application. Click **What's next > Get access token** to request for an application.

The screenshot shows the same 'About Bookstore' page as before, but now the 'Get access token' link in the 'What's next?' sidebar is highlighted with a red box.

7. To request for an API access token provide the following properties
 - Application name:** Bookstore App
 - Application description:** Try Out application to test evaluation of OAuth2 access token
 - Application redirect URI:** - leave defaults –Click **Request token**.

8. Navigate to **Further information > Try API>>**. Click **BookstoreApp** within Application.

The screenshot shows the 'Try API' interface. In the 'APPLICATION' dropdown menu, 'Bookstore App' is selected, and 'OAuth2 tokens' is highlighted with a red box. Below the dropdown, there are sections for 'Query parameters', 'Header parameters', and a 'Tool' button. At the bottom right, there is a 'Get new token' button.

9. Click **Get new token** in the right hand bottom corner on the page.

Exercise 21:
Try out API secured with OAuth2 token

The screenshot shows the SAP API Management interface. On the left, there's a sidebar with 'APPLICATION' set to 'Bookstore App' and 'REST RESOURCES' showing 'GET /books' and 'PUT /books'. The main area is titled 'Bookstore' with a URL 'GET http://daeitrain24496.5555/gateway/Bookstore/1.0/books'. Below it, 'Query parameters' and 'Header parameters' sections are visible. Under 'Header parameters', 'Headers' is selected, showing 'Accept: application/json, text/xml' and 'Content-Type: text/xml, application/json'. A red box highlights the 'Get new token' button at the bottom right of the page.

10. To request for an OAuth token provide the following properties

a. Token name: test_authcode_token

Get OAuth token

Token name	test_authcode_token
Grant type	Authorization Code
Authorization URL	http://daeitrain24496.eur.ad.sag:5555/invoke/pub.apigateway.oauth2/authorize
Access token URL	http://daeitrain24496.eur.ad.sag:5555/invoke/pub.apigateway.oauth2/getAccessToken
Client ID	a4518b69-b97e-46b8-839e-7873f30f22cc
Client secret	9f5ea6c2-6f27-4d13-93b6-260ccbba43d6f
Scope(s)	

Close Get token

Click **Get token**.

11. A new Browser window will be opened, asking for **webMethods API Gateway Resource access approval**.

The screenshot shows a browser window titled 'webMethods API Gateway Resource access approval'. It displays a message: 'The application Bookstore App-729edd67-1da7-432d-9e6b-84716426736a (1.0) is requesting access to the following scopes. If you trust this application, select the scopes to which you want to grant access, and then click Approve.' There are two checkboxes: 'All' and 'Bookstore/1.0 - API Scope'. At the bottom, there are 'Deny' and 'Approve' buttons, with 'Approve' being highlighted by a red box.

Click **Approve**.

In case Integration Server asks for authentication, provide **Administrator/manage**.

12. On the next screens, if needed, click appropriate button to allow an unsecure connection.

13. The generated token will be given back from **API Gateway** to **API Portal**. It is listed on the **Try API** page of the **Boystore** API. A corresponding Header variable **Authorization** is added to the section of Header parameters.

Exercise 21:
Try out API secured with OAuth2 token

Header parameters

Headers Security

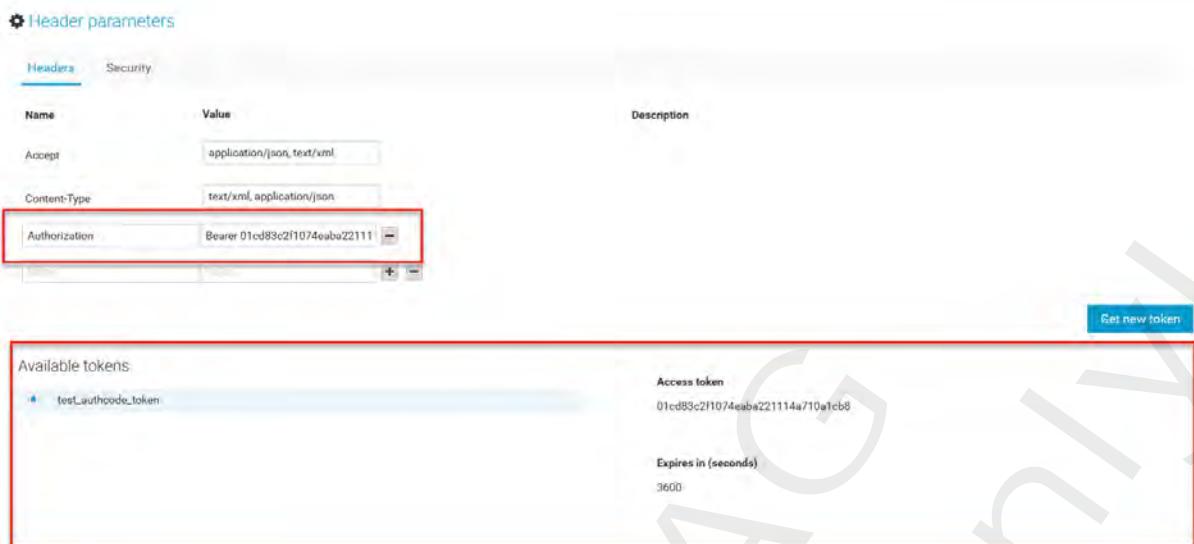
Name	Value	Description
Accept	application/json, text/xml	
Content-Type	text/xml, application/json	
Authorization	Bearer 01cd83c2f1074eabs22114a710a1cb8	

Available tokens:

- test_authcode_token

Access token: 01cd83c2f1074eabs22114a710a1cb8
Expires in (seconds): 3600

Get new token



14. Click Test to run the request with the OAuth token. The response with response code 200 shows the booklist.

REST RESOURCES

GET /books PUT /books

GET http://daetrain24496:5555/gateway/Bookstore/1.0/books

Query parameters Header parameters

Available tokens:

- test_authcode_token

Access token: 01cd83c2f1074eabs22114a710a1cb8
Expires in (seconds): 3600

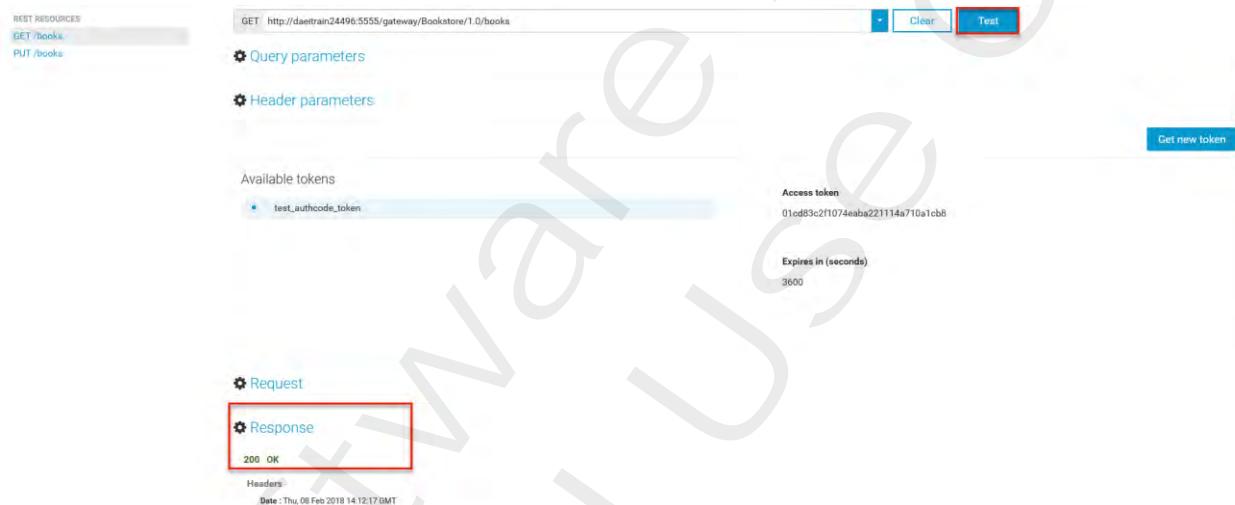
Get new token

Request Response

200 OK

Headers:

Date : Thu, 08 Feb 2018 14:12:17 GMT
Server : Apache
com.aris.per.requestnumber : 3



Exercise 22:

Editing APIs in API Portal

Objectives

In this exercise, you will customize/edit an API in API-Portal.

Steps

1. Open **Windows Services UI** to double check that the following services, needed for **API Gateway** and **API Portal** are up and running. If service is not running, start the service.
 - a. **Software AG Integration Server 10.1**
 - b. **Software AG Event Data store 10.1 – should be started together with IS**
 - c. **Software AG Runtime 10.1**
 - d. **Software AG API Portal 10.1**
 - e. **hMail-Server**
2. Verify that all runnables of **API Portal** are up and running using the **Start Menu -> Software AG -> Administration -> Start API Portal Cloud Controller**
3. Login to **API-Portal** in a private Firefox Browser window using the URL <http://sagbase:18101/#default/home> or use the shortcut defined when starting **Mozilla Firefox**.
4. Login as **API Provider** with credentials **andy/manage**. Navigate to the **API Gallery** menu and select **SearchCruise API**.
5. Navigate to **API Details > API resources**. Open the resource **/cruises/{cruiseID}**.
6. We would like to update the description of the resource. Hover your mouse over the description of the resource. An edit icon will pop up.

The screenshot shows the API Gallery interface for the SearchCruise API. Under the 'API resources' section, the '/cruises/{cruiseID}' resource is selected. The description field contains the placeholder 'Returns a specific cruise with information like destination, ships, ports, ...'. To the right of the description, there is an edit icon (pencil symbol) enclosed in a red box, indicating it can be clicked to edit the description.

- a. Only the cruiseIDs 00001 and 00002 are available cruiseIDs. Switch to edit mode of the description and add the following text to the existing description:
Available cruises are 00001, 00002
- b. Save your changes and click the corresponding icon

Exercise 22: Editing APIs in API Portal

API resources

Below is a list of resources available in the API. Click each resource to view more details.

/cruises

/cruises/{cruiseID}

ts, ... Available cruises are 00001, 00002

Parameters

cruises

cruisesWithCruiseID

7. We would like to update the description of the **GET** method. Click **Click here to edit**.

- a. Provide the following content:

Get a specific cruise

- b. Save your changes and click the corresponding icon

/cruises/{cruiseID}

Returns a specific cruise with information like destination, ships, ports, ... Available cruises are 00001, 00002

cruisesWithCruiseID

Parameters

GET

Get a specific cruise



Returns a specific cruise based on the specified cruise ID

Request

8. Verify your changes.

API resources

Below is a list of resources available in the API. Click each resource to view more details.

/cruises

cruises

/cruises/{cruiseID}

cruisesWithCruiseID

Returns a specific cruise with information like destination, ships, ports, ... Available cruises are 00001, 00002

Parameters

GET

Get a specific cruise

9. We would like to do more changes on the description of the **SearchCruise API**. Hover over the Description of the API and click the Edit icon.

Home API Gallery SearchCruise

Back

GETTING STARTED

About SearchCruise

API DETAILS

API resources

API documents

Access API

FURTHER INFORMATION

Latest posts

Try API+

REST SearchCruise

About SearchCruise

Click here to edit.

Searches for all cruises, or searches for a cruise that matches the specified criteria.

The GET method has two forms:
- GET /cruises returns all cruises that are currently available.
- GET /cruises?criteria= returns all cruises that match the specified criteria.
In both cases, the returned result is a list of cruises returned in JSON format.

The following criteria can be specified to filter the list of returned cruises:
- startdate - Start date of cruise. Cruises prior to this date will not be shown.
- enddate - End date of cruise. Limits cruises returned to those completing on or before this date. For example: Seattle: 8:11 am Denver: 9:11 am Reister: 11:11 am UK: 4:11 pm
- startPort - Port where the cruise begins.
- numDays - Maximum cruise duration, in days.
All dates are in the format YYYYMMDD.

Edit

What's next?

Support forum

Unfollow this API

Export API as

Download Client SDK

Rate this API

No ratings

List of followers

Andy Poffi

Exercise 22:
Editing APIs in API Portal

- a. Open the file **SearchCruise.md** available in folder C:\Training\456-71E\Exercise22 with **Notepad+**.
- b. Copy the content of this file and add the content at the end of the current description in **SearchCruise**. Save your changes and click the corresponding icon.

The screenshot shows the 'About SearchCruise' page in the API Portal. At the top right, there is an 'Edit' button. Below it, a code editor window contains Java code:

```
    ClientResponse response = service.path(restrain).path(resourcePath);
    type(MediaType.APPLICATION_FORM_URLENCODED).post(ClientResponse.class, form);
    System.out.println('Response' + response.getEntity(String.class));
}
```

A red box highlights the code editor area, and a red checkmark is visible in the bottom right corner of the editor window.

10. Verify the changes.

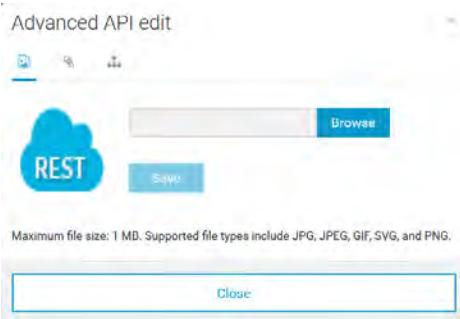
The screenshot shows the 'About SearchCruise' page in the API Portal. A large red box highlights the entire content area of the page, including the sidebar and the main content block.

11. We would like to add some additional attributes to **SearchCruises**. Switch over to Edit mode by clicking the **Edit**.

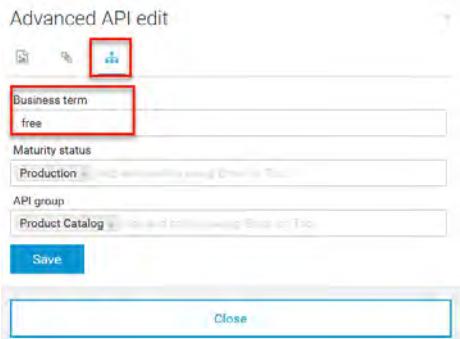
The screenshot shows the 'About SearchCruise' page in the API Portal. On the right side, there is a sidebar with various options. The 'Edit' button is highlighted with a red box.

- a. Click **What's next > Advanced edit**.

Exercise 22:
Editing APIs in API Portal

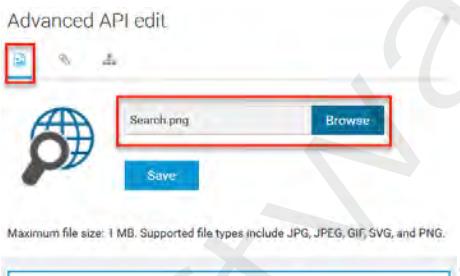


- b. Within the header select the icon to add a classification attribute. Provide the following value
i. **Business term:** Free



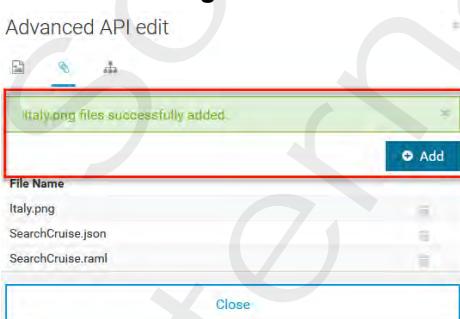
Confirm using **Enter**. Click **Save**.

- c. Within the header select the icon to add an icon. Browse for **Search.png** available in folder **C:\Training\456-71E\Exercise22**.



Click **Save**.

- d. Within the header select the icon to add a document. Click **Add**. Browse for **Italy.png** available in folder **C:\Training\456-71E\Exercise22**.



Click **Close**.

6. Use the **Search** functionality again and search for the **BookingsAPI**.
7. Use the Edit API functionality to add the following properties to the API. The icon is available in folder **C:\Training\456-71E\Exercise22**.
 - e. **API categories:**
 - i. **Business term:** Free
 - f. **Icon:** Booking.png

Click **Save**. Click **Close**.

8. Use the **Search** functionality again and search for the **SignupAPI**.
 9. Use the Edit API functionality to add the following properties to the API. The icon is available in folder **C:\Training\456-71E\Exercise22**
 - g. **API categories:**
 - i. **Business term:** Free
 - h. **Icon:** Customers.png
- Click **Close**.

Software AG
Internal Use Only!