webMethods API Gateway Upgrade and Migration

Version 10.11

October 2021

WEBMETHODS

Document ID: YAI-UMG-1011-20231210

Table of Contents

About this Documentation **5**
- Document Conventions 6
- Online Information and Support 6
- Data Protection 7

1 Upgrading API Gateway **9**
- Introduction 10
- Upgrading standalone API Gateway 13
- Upgrading API Gateway cluster 18
- Upgrade configurations 24
- Upgrade recovery 25
- Troubleshooting Tips 25

2 Upgrading API Gateway in Zero Downtime **31**
- Introduction 32
- Upgrading Major Versions in Zero Downtime 39
- Upgrading Minor Versions in Zero Downtime 47
- Quiesce Mode 51

3 Upgrading API Gateway with external Elasticsearch and Kibana **55**

4 Migrating from Mediator to API Gateway **57**

About this Documentation

- Document Conventions 6
- Online Information and Support 6
- Data Protection 7

This documentation describes how you can upgrade, and migrate to API Gateway.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies service names and locations in the format *folder.subfolder.service*, APIs, Java classes, methods, properties.
Italic	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at https://documentation.softwareag.com.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at https://empower.softwareag.com/.

You can find product information on the Software AG Empower Product Support website at https://empower.softwareag.com.

To submit feature/enhancement requests, get information about product availability, and download products, go to Products.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the Knowledge Center.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at https://empower.softwareag.com/public_directory.aspx and give us a call.

Software AG Tech Community

You can find documentation and other technical information on the Software AG Tech Community website at https://techcommunity.softwareag.com. You can:

- Access product documentation, if you have Tech Community credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.
- Go to our public GitHub and Docker repositories at https://github.com/softwareag and https://hub.docker.com/u/softwareag and discover additional Software AG resources.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

1 Upgrading API Gateway

- Introduction 10
- Upgrading standalone API Gateway 13
- Upgrading API Gateway cluster 18
- Upgrade configurations 24
- Upgrade recovery 25
- Troubleshooting Tips 25

Introduction

This guide explains how to upgrade API Gateway from versions 10.1 and above to a latest version. Upgrading API Gateway involves migrating the configurations and data from one version to another.

API Gateway manages different types of data that are migrated during an upgrade. This includes the data that are stored in Elasticsearch and in the file system.

The following table lists the different types of data and the migration procedure that API Gateway supports:

Data Type	How is it stored?	Where is it stored?	Migration information
API Gateway assets	Documents	API Data Store or Elasticsearch	For more information on the procedure to migrate these data, see "Migrating the API Gateway assets and the analytics data" on page 11.
API Gateway analytics	Documents	API Data Store or Elasticsearch	For more information on the procedure to migrate these data, see "Migrating the API Gateway assets and the analytics data" on page 11.
Platform configurations	Files	On server nodes	For more information on the procedure to migrate these data, see "Migrating the platform configurations" on page 11.

Terminologies used

Terminology	Description
Source	The source API Gateway installation directory.
Target	The target API Gateway installation directory.
Quiesce	Quiescing API Gateway temporarily disables access to the server so you can perform the required tasks while the server is not connected to any external resources. In API Gateway, quiesce mode is used during the zero downtime upgrade wherein the access to the server is temporarily disabled, so you can perform the upgrade tasks. For more details, see "Quiesce Mode" on page 51.
Blue-Green	API Gateway follows Blue-Green deployment approach to upgrade to newer version in zero downtime. Blue-green deployment is a technique that reduces the downtime and risk by running two identical production environments called blue and green. In such a deployment scenario, the old instance of API Gateway is allowed to run and serve the transactions while the new instance of API Gateway is being prepared for data migration.

Migrating the API Gateway assets and the analytics data

When you upgrade API Gateway to a latest version, you have to migrate the API Gateway assets and the analytics data, that is, the Elasticsearch or API Data Store data. If the source Elasticsearch or API Data Store instance is accessible by the target, then you can use the `migrate.bat datastore` command to migrate the API Gateway assets and the analytics data.

Note:
If you do not have the connectivity between the source and the target Elasticsearch or API Data Store instances, contact the Software AG support team to set up the data migration.

When you use the `migrate.bat datastore` command, it migrates both the core and analytics data. When the analytics data is of high volume, alternatively, you can use the `migrate.bat datastore` command to migrate only the core data, and use the backup and restore command to migrate the analytics data.

When you migrate using `migrate.bat datastore` command, it also migrates ports, keystores, aliases, and so on, which is also stored in the API Data Store, and these values override the existing port configurations and keystores in target server. Hence, ensure that the unnecessary data such as migrated ports, keystores, and aliases are cleaned up post migration if they are not used by the target system.

For more information on the procedure to upgrade API Gateway and the migration utility command, see "Upgrading standalone API Gateway" on page 13 and "Upgrading API Gateway cluster" on page 18.

Migrating the platform configurations

The platform configurations are mostly one time configurations, that is, these configurations do not change often and these configurations are stored in the version control system or similar repositories. These configurations must be migrated from the source to the target API Gateway instance. You can use these stored configurations to set up the target API Gateway instance and restore the configurations.

There are three ways to migrate the platform configurations from the source to the target API Gateway instance:

1. Migrating the platform configurations manually, that is, copy the configuration files manually from the source repository to the target repository, to restore the target API Gateway instance, if you have stored these configurations in the repositories.
2. Configuring using the externalized configurations. If you have externalized the configurations of the source API Gateway instance, you can use the same externalized configuration file to restore the target platform configurations. For more information, see *webMethods API Gateway Administration*.

   Note:
   Externalized configurations does not support externalizing all the API Gateway configurations. In future releases, API Gateway will support externalizing all the configurations.

3. Using the `migrate.bat apigateway` command.

The following table shows the API Gateway configurations that are stored in the file system:

Configuration	File name	File location	Is this supported by externalization?	Is this data migrated by the migrate.bat utility command?
Elasticsearch configuration	elasticsearch.yml	`SAGInstallDir/InternalDataStore/config/`	No	No
Elasticsearch client configuration	config.properties	`SAGInstallDir/IntegrationServer/instances/instance_name/packages/WmAPIGateway/config/resources/elasticsearch/`	Yes	No
Kibana configuration	kibana.yml	`SAGInstallDir/profiles/instance_name/apigateway/dashboard/config/`	Yes	No
Master password	mpw.dat	`SAGInstallDir/profiles/instance_name/configuration/security/passman/`	Yes	Yes
UI configurations	uiconfiguration.properties	`SAGInstallDir/profiles/instance_name/apigateway/config/`	No	Yes
WebApp settings	com.softwareag.catalina.connector.http.pid-apigateway.properties com.softwareag.catalina.connector.https.pid-apigateway.properties You have to manually migrate the certificates used for the WebApp ports.	`SAGInstallDir/profiles/instance_name/configuration/com.softwareag.platform.config.propsloader/`	Yes	Yes
Custom wrapper settings	custom_wrapper.conf	`SAGInstallDir/profiles/instance_name/configuration/`	No	Yes
Server Ports Configuration	If the `portClusteringEnabled` extended setting is set		No	No

Configuration	File name	File location	Is this supported by externalization?	Is this data migrated by the migrate.bat utility command?
	to false, you must create the server ports in each instance.			
Custom ESB packages	File name specified by users. Make sure that all the custom packages are installed and ready in the new instances.	Location used by users to save the file. Generally, the customized packages are saved in the following location: `SAGInstallDir\IntegrationServer\tenant\packages\`	No	Yes

For more information on the procedure to upgrade API Gateway and the migration utility command, see "Upgrading standalone API Gateway" on page 13 and "Upgrading API Gateway cluster" on page 18.

Upgrading standalone API Gateway

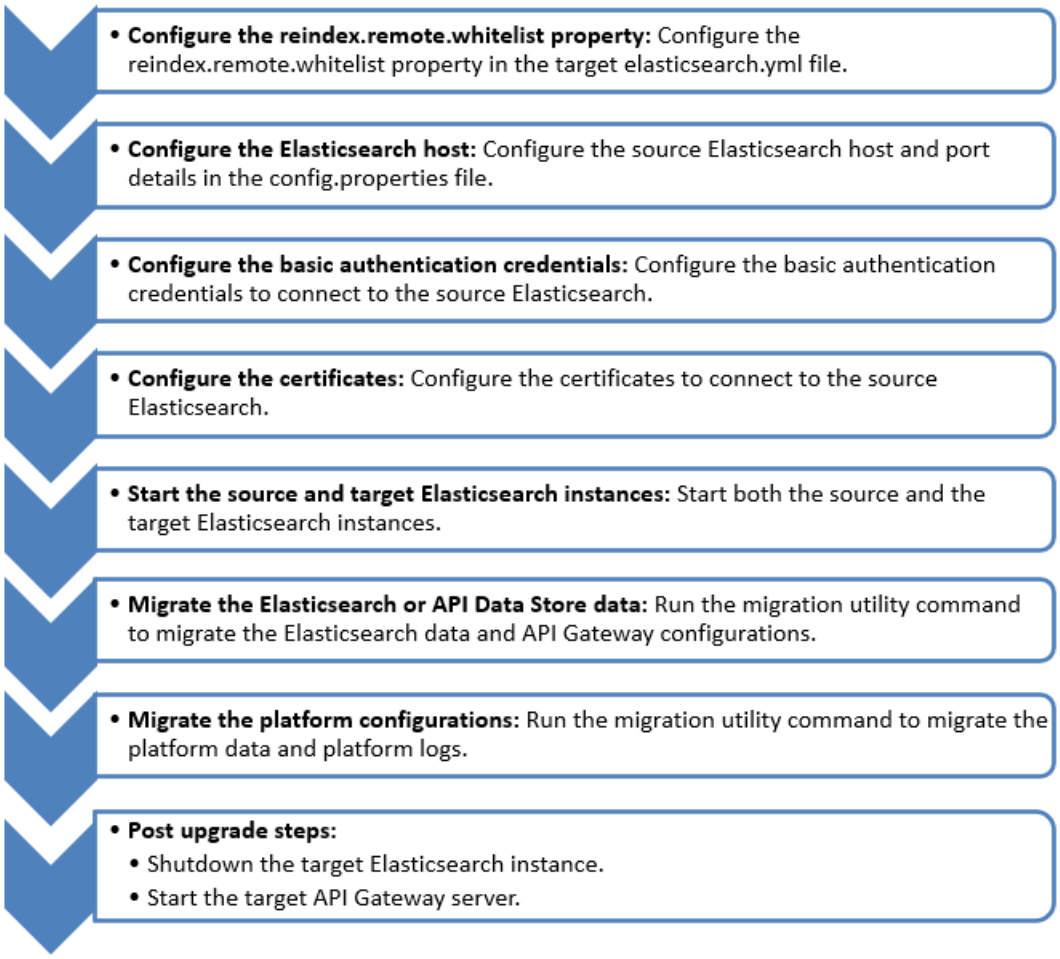
Prerequisites:

- Install latest fixes in both source and target versions.

  Note:
  The version of target API Gateway must be higher than the source API Gateway instance. Supported source API Gateway versions are 10.1 and above.

- If custom keystore or truststore files are used in the source API Gateway installation, copy the files to the same location in the target installation.

The following image illustrates the different stages in upgrading a standalone API Gateway instance:



The API Gateway migration utility provides the support to:

- Separately migrate Elasticsearch and API Gateway file configurations.
- Migrate data from externally configured Elasticsearch.
- Log all the migration data in a single file, that is migrationLog.txt.
- Revert in case of failure in Elasticsearch data migration.

Note:
In the API Gateway versions 10.2 and above, the folder name EventDataStore is changed to InternalDataStore. Throughout this section, the source API Gateway installation directory is referred as <*SOURCE*> and the target API Gateway installation directory is referred as <*TARGET*>.

To upgrade a standalone API Gateway instance

1. Configure the reindex.remote.whitelist property in the target elasticsearch.yml file for re-indexing the data in the target Elasticsearch. This YAML file is located at `<TARGET>\InternalDataStore\config`. The reindex.remote.whitelist property copies the documents from the `<SOURCE>` to `<TARGET>` Elasticsearch instance.

Syntax for `reindex.remote.whitelist` property in the target elasticsearch.yml file is as follows:

```
reindex.remote.whitelist: <source_host>:<source_port>
```

Note:

- The value of `reindex.remote.whitelist` property in the target elasticsearch.yml file must match the value of the `pg.gateway.elasticsearch.hosts` property present in the config.properties file located at `<SOURCE>\IntegrationServer\instances\default\packages\WmAPIGateway\config\resource\elasticsearch`.
- For managed Elasticsearch instances, you do not have to configure the `reindex.remote.whitelist` property.

An example for the `reindex.remote.whitelist` property in the elasticsearch.yml file is as follows:

```
cluster.name: SAG_EventDataStore
node.name: SAG-1YVHZY21633236119876
path.logs: C:\SoftwareAG_1011\InternalDataStore/logs
network.host: 0.0.0.0
http.port: 9240
discovery.seed_hosts: ["localhost:9340"]
transport.tcp.port: 9340
path.repo: ['C:\SoftwareAG_1011\InternalDataStore/archives']
cluster.initial_master_nodes: ["node1"]
reindex.remote.whitelist: localhost:9240
```

2. *Optional.* Configure the source Elasticsearch host and port details in the config.properties file located at `<SOURCE>\IntegrationServer\instances\default\packages\WmAPIGateway\config\resources\elasticsearch`.

   Syntax for Elasticsearch host and port details in the source config.properties file is as follows:

   ```
   pg.gateway.elasticsearch.hosts=<source_host>:<source_port>
   ```

   Note:
   This step is applicable only if the source API Gateway version is 10.1. From versions 10.2 and above, the Elasticsearch host and port values are populated automatically.

3. Configure the basic authentication credentials to connect to the source Elasticsearch, if the source Elasticsearch is protected.

   To configure the basic authentication credentials, add the following properties to the config.properties file located at `<SOURCE>\IntegrationServer\instances\default\packages\WmAPIGateway\config\resources\elasticsearch`:

   ```
   pg.gateway.elasticsearch.http.username=<username>
   pg.gateway.elasticsearch.http.password=<password>
   ```

4. Configure the certificates to connect to the source Elasticsearch.

   If the source Elasticsearch is protected with HTTPS, add the source certificates (public key) into the target Elasticsearch JVM's truststore. For example, in case of API Data Store, add the public keys to the truststore cacerts file located at `<TARGET>\jvm\jvm\jre\lib\security`.

Note:
If, external Elasticsearch is used for the target API Gateway, then the certificates must be imported to its corresponding JVM.

Sample command to import the truststore of the source Elasticsearch into the target Elasticsearch JVM is as follows:

```
<TARGET>\jvm\jvm\bin\keytool -import -keystore
<TARGET>\jvm\jvm\jre\lib\security\cacerts -file <truststore.jks> -alias <alias>
```

Property	Description
truststore.jks	Truststore used in <*SOURCE*> Elasticsearch. Provide the full path of the truststore. For 10.1, it is available at `<SOURCE>\WmAPIGateway\config\resources\bean\gateway-es-store.xml` and the property is `<prop key="searchguard.ssl.http.truststore_filepath"\>sagconfig/root-ca.jks\</prop>`. For 10.2 and above, it is available at `<SOURCE>\WmAPIGateway\config\resources\elasticsearch\config.properties` and the property is `pg.gateway.elasticsearch.https.truststore.filepath`. Example: sagconfig/root-ca.jks
alias	Alias used in <*SOURCE*> Elasticsearch. Example: wm.sag.com

5. Start both the source and the target Elasticsearch instances and make sure that API Gateway (Integration Server) instances are not started.

Note:
Avoid port conflict. If the source and target API Gateway instances are running in the same machine, then you might not be able to start both the source and target Elasticsearch instances in parallel with the default port configurations. In this case, the target Elasticsearch instance port can be changed temporarily for performing the migration. Both, HTTP and TCP ports must be changed. To change the target Elasticsearch instance ports:

a. Open the elasticsearch.yml file located at `<TARGET>/InternalDataStore/config`. Change the value of the HTTP port in the `http.port` property, and TCP port in the `transport.tcp.port` property.
b. Open the config.properties file located at `<TARGET>/IntegrationServer/instances/default/packages/WmAPIGateway/config/resources/elasticsearch` and find the `pg.gateway.elasticsearch.hosts` property. If the property is set to `changeOnInstall`, then do not make further changes to property. If a port is configured, then update it to a new value.

Revert the temporary target Elasticsearch instance ports to the default port configurations, after completing the migration and the target Elasticsearch instance is shutdown.

If you are reverting the elasticsearch ports in the elasticsearch.yml file after the migration, then you need to specify the same ports in the config.properties file located at `<TARGET>/IntegrationServer/instances/default/packages/WmAPIGateway/config/resources/elasticsearch`.

6. Migrate the API Gateway assets and the analytics data. Go to `<TARGET>\IntegrationServer\instances\default\packages\WmAPIGateway\bin\migrate` and run the following migration utility command.

```
migrate.bat datastore -dstoreSrc <full path to source Elasticsearch
config.properties>
```

Property	Description

dstoreSrc	If the source and target API Gateway instances are running on the same machine, provide the location where the <*SOURCE*> config.properties file is located. Sample command is as follows:
	`migrate.bat datastore -dstoreSrc<SOURCE>\IntegrationServer\instances\default\packages\WmAPIGateway\config\resources\elasticsearch\config.properties`
	If the source and target instances are running on different machines, then the source installation directory or at least the Elasticsearch config.properties file must be shared in the network. Otherwise, copy and paste the source config.properties file to the shared location. Sample command is as follows:
	`migrate.bat datastore -dstoreSrc\chebackup01\installations\source\IntegrationServer\instances\default\packages\WmAPIGateway\config\resources\elasticsearch\config.properties`

7. Migrate the platform configurations.

To migrate the platform configurations using externalized configurations, you can still use the `migrate.bat apigateway` command to migrate the platform configurations. To use the `migrate.bat apigateway` command, go to `<TARGET>\IntegrationServer\instances\default\packages\WmAPIGateway\bin\migrate` and run the following command.

```
migrate.bat apigateway -srcDir <SOURCE> -instanceName <source instance name>
-newInstanceName <target instance name>
```

Property	Description
srcDir	If the source API Gateway instance is installed on the same machine, provide the source API Gateway installation directory. Sample command is as follows: `-srcDir C:\installations\source` If the source API Gateway instance is installed on a different machine, then share the entire installation folder. Sample command is as follows:

Property	Description
	`-srcDir \chebackup01\source`
instanceName	*Optional.* Provide the <*SOURCE*> instance name that you want to migrate. If you do not provide any name, then the default instance is used. Sample command is as follows: `-instanceName test`
newInstanceName	*Optional.* Provide the <*TARGET*> instance name that you want to migrate to. If you do not provide any name, then the default instance is used. If you have created a new instance other than the default in Integration Server and you want to migrate to the new instance then provide its name. Sample command is as follows: `-newInstanceName prod`

8. Perform these steps after migrating the platform data and platform logs.

   a. Shutdown the target Elasticsearch instance.

   b. Start the target API Gateway server.

   The API Gateway upgrade is complete.

   - You can find the logs at the target directory `<TARGET>/install/logs/migrationLog.txt`.
   - You can find the API Gateway migration reports at `<TARGET>\install\logs\APIGW_Migration_Reports_<date_time>`.

Upgrading API Gateway cluster

Note:
This procedure is applicable only for on-premise installation.

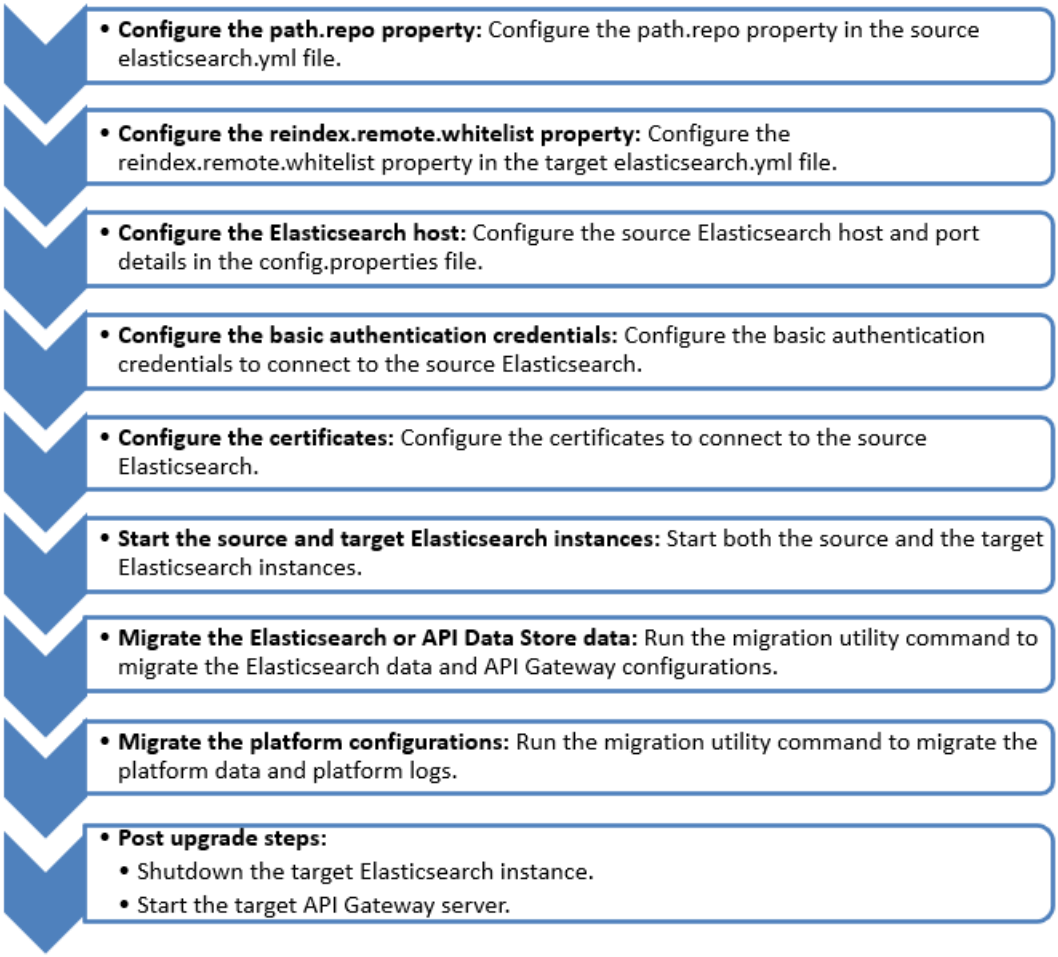
Prerequisites:

- Install latest fixes in both source and target versions.

  Note:
  The version of target API Gateway must be higher than the source API Gateway instance. Supported source API Gateway versions are 10.1 and above.

- If custom keystore or truststore files are used in the source API Gateway installation, copy the files to the same location in the target installation.

The following image illustrates the different stages in upgrading an API Gateway cluster:



The API Gateway migration utility provides the support to:

- Separately migrate Elasticsearch and API Gateway file configurations.
- Migrate data from externally configured Elasticsearch.
- Log all the migration data in a single file, that is migrationLog.txt.
- Revert in case of failure in Elasticsearch data migration.

Note:
In the API Gateway versions 10.2 and above, the folder name EventDataStore is changed to InternalDataStore. Throughout this section, the source API Gateway installation directory is referred as <*SOURCE*> and the target API Gateway installation directory is referred as <*TARGET*>.

To upgrade an API Gateway cluster

1. Configure the path.repo property in the source elasticsearch.yml file located at `<SOURCE>\InternalDataStore\config`. Make sure that the path.repo is a shared network folder and is accessible for all the Elasticsearch nodes in the cluster.

2. Configure the reindex.remote.whitelist property in the target elasticsearch.yml file for re-indexing the data in the target Elasticsearch. This YAML file is located at `<TARGET>\InternalDataStore\config`. The reindex.remote.whitelist property copies the documents from the `<SOURCE>` to `<TARGET>` Elasticsearch instances.

   Syntax for `reindex.remote.whitelist` property in the target elasticsearch.yml file is as follows:

   ```
   reindex.remote.whitelist: <source_host>:<source_port>
   ```

   Note:

   - The `reindex.remote.whitelist` value in the target elasticsearch.yml file must match the value of the `pg.gateway.elasticsearch.hosts` property present in the config.properties file located at `<SOURCE>\IntegrationServer\instances\default\packages\WmAPIGateway\config\resource\elasticsearch`.
   - For managed Elasticsearch instances, you do not have to configure the `reindex.remote.whitelist` property.

   An example for the `reindex.remote.whitelist` property in the elasticsearch.yml file is as follows:

   ```
   cluster.name: SAG_EventDataStore
   node.name: SAG-1YVHZY21633236119876
   path.logs: C:\SoftwareAG_1011\InternalDataStore/logs
   network.host: 0.0.0.0
   http.port: 9240
   discovery.seed_hosts: ["hostname1:9340","hostname2:9340","hostname3:9340"]
   transport.tcp.port: 9340
   path.repo: ['C:\SoftwareAG_1011\InternalDataStore/archives']
   cluster.initial_master_nodes: ["node1","node2","node3"]
   reindex.remote.whitelist: localhost:9240
   ```

3. *Optional.* Configure the source Elasticsearch host and port details in the config.properties file located at `<SOURCE>\IntegrationServer\instances\default\packages\WmAPIGateway\config\resources\elasticsearch`.

   Syntax for Elasticsearch host and port details in the source config.properties file is as follows:

   ```
   pg.gateway.elasticsearch.hosts=<source_host>:<source_port>
   ```

   Configure the source Elasticsearch host and port details for all the API Gateway nodes in the cluster.

   Note:
   This step is applicable only if the source API Gateway version is 10.1. From versions 10.2 and above, the Elasticsearch host and port values are populated automatically.

4. Configure the basic authentication credentials to connect to the source Elasticsearch, if the source Elasticsearch is protected.

To configure the basic authentication credentials, add the following properties to the config.properties file located at <*SOURCE*>\IntegrationServer\instances\default\packages\WmAPIGateway\config\resources\elasticsearch:

```
pg.gateway.elasticsearch.http.username=<username>
pg.gateway.elasticsearch.http.password=<password>
```

Configure the basic authentication credentials for all the API Gateway nodes in the cluster.

5. Configure the certificates to connect to the source Elasticsearch.

   If the source Elasticsearch is protected with HTTPS, add the source certificates (public key) into the target Elasticsearch JVM's truststore. For example, in case of API Data Store, add the public keys to the truststore cacerts file located at <*TARGET*>\jvm\jvm\jre\lib\security. Do for all the target API Gateway nodes.

   Note:
   If, external Elasticsearch is used for the target API Gateway, then the certificates must be imported to its corresponding JVM.

   Sample command to import the truststore of the source Elasticsearch into the target Elasticsearch JVM is as follows:

   ```
   <TARGET>\jvm\jvm\bin\keytool -import -keystore
   <TARGET>\jvm\jvm\jre\lib\security\cacerts -file <truststore.jks> -alias <alias>
   ```

Property	Description
truststore.jks	Truststore used in <*SOURCE*> Elasticsearch. Provide the full path of the truststore. For 10.1, it is be available at <*SOURCE*>\WmAPIGateway\config\resources\bean\gateway-es-store.xml and the property is <prop key="searchguard.ssl.http.truststore_filepath"\>sagconfig/root-ca.jks\</prop> For 10.2 and above, it is available at <*SOURCE*>\WmAPIGateway\config\resources\elasticsearch\config.properties and the property is pg.gateway.elasticsearch.https.truststore.filepath. Example: sagconfig/root-ca.jks
alias	Alias used in <*SOURCE*> Elasticsearch. Example: wm.sag.com

6. Start both the source and target Elasticsearch instances and make sure that API Gateway (Integration Server) instances are not started.

   Note:
   Avoid port conflict. If the source and target API Gateway instances are running in the same machine, then you might not be able to start both the source and target Elasticsearch instances in parallel with the default port configurations. In this case, the target Elasticsearch instance

port can be changed temporarily for performing the migration. Both, HTTP and TCP ports must be changed. To change the target Elasticsearch instance ports:

a. Open the elasticsearch.yml file located at `<TARGET>/InternalDataStore/config`. Change the value of the HTTP port in the `http.port` property, and TCP port in the `transport.tcp.port` property.
b. Open the config.properties file located at `<TARGET>/IntegrationServer/instances/default/packages/WmAPIGateway/config/resources/elasticsearch` and find the `pg.gateway.elasticsearch.hosts` property. If the property is set to `changeOnInstall`, then do not make further changes to property. If a port is configured, then update it to a new value.

Revert the temporary target Elasticsearch instance ports to the default port configurations, after completing the migration and the target Elasticsearch instance is shutdown.

If you are reverting the elasticsearch ports in the elasticsearch.yml file after the migration, then you need to specify the same ports in the config.properties file located at `<TARGET>/IntegrationServer/instances/default/packages/WmAPIGateway/config/resources/elasticsearch`.

7. Migrate the API Gateway assets and the analytics data. Go to `<TARGET>\IntegrationServer\instances\default\packages\WmAPIGateway\bin\migrate` and run the following migration utility command.

Note:
Before running the migration utility command, set up the target Elasticsearch cluster and make sure that all the source and target Elasticsearch instances in the clusters are up and running. Also, do not start the API Gateway instances on any of the nodes.

```
migrate.bat datastore -dstoreSrc <full path to source Elasticsearch
config.properties>
```

You should run the `migrate.bat datastore` command in only one target API Gateway node of the cluster.

Property	Description

dstoreSrc	If the source and target API Gateway instances are running on the same machine, provide the location where <*SOURCE*> config.properties file is located. Sample command is as follows:
	`migrate.bat datastore -dstoreSrc<SOURCE>\IntegrationServer\instances\default\packages\WmAPIGateway\config\resources\elasticsearch\config.properties`
	If the source and target instances are running on different machines, then the source installation directory or the Elasticsearch config.properties file must be shared in the network. Otherwise, copy and paste the source config.properties file to the shared location. Sample command is as follows:
	`migrate.bat datastore -dstoreSrc\chebackup01\installations\source\IntegrationServer\instances\default\`

Property	Description
	`packages\WmAPIGateway\config\resources\elasticsearch\config.properties`

8. Migrate the platform configurations.

   To migrate the platform configurations using externalized configurations, you can still use the `migrate.bat apigateway` command to migrate the platform configurations. To use the `migrate.bat apigateway` command, go to `<TARGET>\IntegrationServer\instances\default\packages\WmAPIGateway\bin\migrate` and run the following command.

   ```
   migrate.bat apigateway -srcDir <SOURCE> -instanceName <source instance name>
   -newInstanceName <target instance name>
   ```

   Note:
   You should run the `migrate.bat apigateway` command in all the target API Gateway nodes of the cluster.

Property	Description

srcDir	If the source API Gateway instance is installed on the same machine, provide the source API Gateway installation directory. Sample command is as follows:
	`-srcDir C:\installations\source`
	If the source API Gateway instance is installed on a different machine, then share the entire installation folder. Sample command is as follows:
	`-srcDir \chebackup01\source`

instanceName	*Optional.* Provide the *<SOURCE>* instance name that you want to migrate. If you do not provide any name, then the default instance is used. Sample command is as follows: `-instanceName test`
newInstanceName	*Optional.* Provide the *<TARGET>* instance name that you want to migrate to. If you do not provide any name, then the default instance is used. If you have created a new instance other than the default in Integration Server and you want to migrate to the new instance then provide its name. Sample command is as follows: `-newInstanceName prod`

9. Perform these steps after migrating the platform data and platform logs.

a. Shutdown the target Elasticsearch instance.

b. Start the target API Gateway server.

The API Gateway upgrade is complete.

- You can find the logs at the target directory `<TARGET>/install/logs/migrationLog.txt`.
- You can find the API Gateway migration reports at `<TARGET>\install\logs\APIGW_Migration_Reports_<date_time>`.

Upgrade configurations

You can modify certain migration parameters based on the requirements by modifying the migration.properties property file located at `<TARGET>\IntegrationServer\instances\default\packages\WmAPIGateway\bin\migrate`. This property file is specific to an instance and you can have different configurations while migrating different instances.

The following table describes the general migration configurations:

Property	Description	Default Value	Possible Values
`apigateway.migration.srcTenantName`	The source tenant name is assumed as default. But, if the source API Gateway has multiple tenants, this property can be used to specify the tenant name from which the data has to be migrated to the target tenant.	default	Any available tenant in Elasticsearch.
`apigateway.migration.batchSize`	The batch size with which the data is processed. For example, if the size is 100, then by default 100 documents are processed. You can increase or decrease this value based on the network availability.	100	Appropriate batch size. It depends on the number of documents and the size of the documents in the API Data Store.
`apigateway.migration.logLevel`	Log level for migration. You can change the log level to debug, error, etc.	info	info, debug, error, warn, trace
`apigateway.migration.reindex.status.check.sleep.interval`	Interval configuration in milliseconds. Once the re-indexing process has started from the source to target instances, migration process is triggered after every configured sleep interval to check whether the re-indexing is complete. It will check the status of the task id.	5000	Appropriate sleep interval

Upgrade recovery

During migration, if there is any problem in the execution, to make sure that the assets are migrated properly, you can clean the target instance and run the migration again. The clean command clears the target data store (the one configured in the config.properties of the target machine). During this procedure, all the indices are removed.

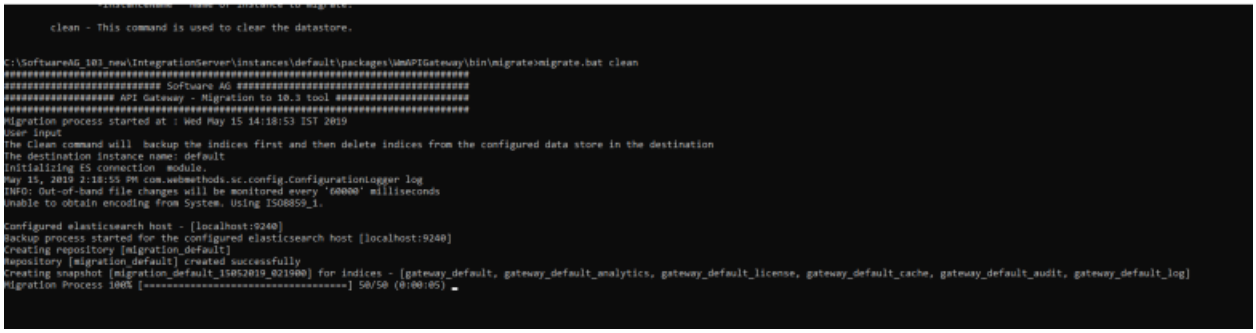
Note:
Running the clean command and removal of indices is a non-reversible action.

Before clearing the data, back up the existing data (you can also restore it). Once the data is cleared, run the migration again. When you run the clean command, the process waits for 5 seconds before starting the cleanup. If you start the command unintentionally and want to kill the process, you can do so within this 5-second interval.

Go to `<TARGET>\IntegrationServer\instances\default\packages\WmAPIGateway\bin\migrate` and run the clean command:

```
migrate.bat clean
```

Sample clean command is as follows:



For managed Elasticsearch, run the clean command with backup as false. Sample clean command with backup as false is as follows:

```
migrate clean -backup false
```

Note:
Since this command removes all the assets from the API Data Store, make sure that the target API Data Store is properly configured in the config.properties file located at `<TARGET>\IntegrationServer\instances\default\packages\WmAPIGateway\config\resources\elasticsearch`.

Troubleshooting Tips

I see that a scheduled API Gateway upgrade fails due to the re-indexing issue.

Upgrading API Gateway fails due to the re-indexing issue in the migration process.

Sample error is as follows:

Data store migration failed.

Reason - Error while reindexing APIs. Error type - illegal_argument_exception, reason Remote responded with a chunk that was too large. Use a smaller batch size.

Resolution:

To resolve this issue, modify the value of the `apigateway.migration.batchSize` property in the migration.properties file located at `SAG_Installdir\IntegrationServer\instances\default\packages\WmAPIGateway\bin\migrate`.

The default value of the `apigateway.migration.batchSize` property is 100.

Modify the value of the `apigateway.migration.batchSize` property based on the size of the payload. If you have a huge payload, reduce the `apigateway.migration.batchSize` property value to a smaller number. For example, `apigateway.migration.batchSize=25`.

The migration time increases, if you reduce the `apigateway.migration.batchSize` property value.

I see that the data store migration fails when I provide hostname instead of localhost in the elasticsearch.yml file.

My data store migration fails when I provide hostname instead of localhost in elasticsearch.yml.

Basically, I am adding the following line in the file `<installDir>\InternalDataStore\config\elasticsearch.yml` before migration.

```
reindex.remote.whitelist:hostName:9240
```

It works fine with localhost, but with actual hostName it fails with the following error,

Backup process started for the configured elasticsearch host [localhost:8240] Creating repository [migration_default] 2019-06-13 17:00:41 ERROR AbstractElasticsearchClient:98 - APIGW Migration Exception:Error while creating repository - migration_default. Message - listener timeout after waiting for [100000] ms java.io.IOException: listener timeout after waiting for [100000] ms at org.elasticsearch.client.RestClient$SyncResponseListener.get(RestClient.java:660) ~ [elasticsearch-rest-client-5.6.4.jar:5.6.4]

Resolution:

You can use regular expressions instead of using the exact hostname. To learn more about how to set regex values for the reindex.remote.whitelist in the elasticsearch.yml file, see Elasticsearch documentation.

I see an Elasticsearch exception while migrating an API Gateway instance using direct standalone mode.

Upgrading API Gateway using direct standalone mode fails with an Elasticsearch exception.

The Elasticsearch exception is as follows:

ElasticsearchException[Error while reindexing APIs. Error type - illegal_argument_exception,reason [localhost:1240] not whitelisted in reindex.remote.whitelist]

Reason: Remote re-indexing property is missing in the target elasticsearch.yml file.

Resolution:

Add the `reindex.remote.whitelist` property in the target elasticsearch.yml file located at `<TARGET>\InternalDataStore\config`.

Syntax for `reindex.remote.whitelist` property in the target elasticsearch.yml file is as follows:

```
reindex.remote.whitelist: <source_host>:<source_port>
```

I see a No such command exception while migrating an API Gateway instance using direct standalone mode.

Upgrading API Gateway using direct standalone mode fails with a No such command exception.

The exception is as follows:

Exception thrown during migration operation. Exiting the operation. No Such command - -srcDir

Resolution:

The API Data Store and API Gateway argument must be passed in the following migration utility command:

```
migrate.bat apigateway -srcDir <SOURCE> -instanceName <source instance name>
-newInstanceName <target instance name>
```

Sample migration utility command is as follows:

```
migration.bat apigateway -srcDir C:\SoftwareAG_10.1 -instanceName default
-newInstanceName default
```

I see java.io.FileNotFoundException exception while migrating an API Gateway instance using backup standalone mode.

Upgrading API Gateway using backup standalone mode fails with a java.io.FileNotFoundException exception.

The exception is as follows:

Deleting the backup folder before migration... Migration Process 0% [>] 0/100 (0:00:00)
java.io.FileNotFoundException: 15:41:13.580 [main] ERROR com.softwareag.apigateway.utility.command.backup.instance.BackupApiGatewayInstance - Error occurred while trying to parse the Manifest file to obtain the version information.
java.io.FileNotFoundException:(The system cannot find the file specified)

Reason: Existence of `isExtract` file due to migration failure.

Resolution:

Delete the `isExtract` folder in the directory `C:\<SAGInstallDir>\`.

I see an Elasticsearchclient bean creation exception while migrating an API Gateway instance.

Upgrading API Gateway fails with an Elasticsearchclient bean creation exception.

Resolution:

Make sure that all the respective Elasticsearch nodes are up and running.

I see a RepositoryVerificationException while migrating API Gateway cluster using backup mode.

Upgrading API Gateway fails with a RepositoryVerificationException exception.

The exception is as follows:

[2018-12-19T12:34:03,892][WARN][o.e.r.VerifyNodeRepositoryAction][SAG-2KXGBH2.eur.ad.sag1544697848038] [default] failed to verify repositoryorg.elasticsearch.repositories.RepositoryVerificationException: [default] a file written by master to the store [C:\SoftwareAG_10.3\InternalDataStore\dasoSnap\default] cannot be accessed on the node [{SAG-2KXGBH2.eur.ad.sag1544697848038} {yL84xqhZQSuFfUmj0GrFbQ}{uGv-BmTBT6e8LCpOsxpGOg} {10.60.37.18}{10.60.37.18:9340}]. This might indicate that the store [C:\SoftwareAG_10.3\InternalDataStore\dasoSnap\default] is not shared between this node and the master node or that permissions on the store don't allow reading files written by the master node

Resolution:

Make sure that the `path.repo` property is a single common location reachable and accessible by all other nodes.

Configure a single common shared path for the `path.repo` property in the elasticsearch.yml file located at `<SAGInstallDir>\InternalDataStore\config`.

I see a system cannot find the path specified error while migrating API Gateway.

Upgrading API Gateway fails with a system cannot find the path specified error.

Resolution:

Make sure that you are running the .bat file from a proper location.

I see a content too long exception while upgrading API Gateway.

Upgrading API Gateway fails with a content too long exception.

The exception is as follows:

2019-10-16 11:59:36 ERROR ESDataStoreHandler:327 - {"type":"illegal_argument_exception","reason":"Remote responded with a chunk that was too large.

Use a smaller batch size.","caused_by":{"type":"content_too_long_exception","reason":"entity content is too long [185463385] for the configured buffer limit [104857600]"}}

Reason: The size of the documents selected for re-indexing as per the batch size configuration is large.

Resolution:

Try with a smaller document batch size number.

I see an error while getting task details when upgrading API Gateway.

Upgrading API Gateway fails with an error while getting task details.

Sample error is as follows:

Error while getting task details for task id - 6L6RMNEOQF64lQbNeI7J_g:683354. Message - task [6L6RMNEOQF64lQbNeI7J_g:683354] isn't running and hasn't stored its results

Reason: When the disk space in the machine is less than 10%, Elasticsearch marks the index as read-only and the re-index task fails abruptly.

Resolution:

Increase the memory size to resolve this issue.

I see an error while restoring the index when upgrading API Gateway.

Upgrading API Gateway fails with an error while restoring the index, when migrating a standalone API Gateway instance or an API Gateway cluster using backup mode.

Sample error is as follows:

ERROR ESDataStoreHandler:432 - Error while restoring index - gateway_default_analytics java.io.IOException: listener timeout after waiting for [100000] ms

Reason: While restoring the Elasticsearch data from backup snapshot, the elasticsearch client is timed out due to the time taken to restore the large data in Elasticsearch.

Resolution:

To resolve this issue, increase the value of the socket timeout and maximum retry timeout properties accordingly.

socketTimeout
Read timeout in milli seconds between API Gateway and API Data Store.

Property: `pg.gateway.elasticsearch.http.socketTimeout`

maxRetryTimeout
Time out in milli seconds to wait for retries.

Property: `pg.gateway.elasticsearch.http.maxRetryTimeout`

It is advisable to set the maximum retry time for a request to (number of nodes * socketTimeOut)+ connectionTimeout.

2 Upgrading API Gateway in Zero Downtime

- Introduction 32
- Upgrading Major Versions in Zero Downtime 39
- Upgrading Minor Versions in Zero Downtime 47
- Quiesce Mode 51

Introduction

Data migration is one of the most important operations to be performed during upgrade. Data migration is performed in two phases. In the first phase, the old instance of API Gateway is blocked for core design time data updates and all the core design time data is migrated to the new API Data Store, while the new API Gateway instance is running. After this, the new instance is restarted with the migrated design time data and the endpoint is added to the load balancer. At this time, the transactions are served by both the old and new instances. In the second phase, all the transactions to the old API Gateway instance is blocked and the logs and events data are migrated to the new API Gateway instance while it is serving the transactions.

In zero downtime upgrade, API Gateway only migrates the API Data Store data. The migration of the file system configurations and custom Integration Server packages must be done manually.

Upgrade Capabilities

To upgrade API Gateway in zero downtime, the following capabilities are required:

- API Gateway Quiesce mode
- Migration REST API
- Webhook Notifications
- Shutdown REST API

API Gateway Quiesce Mode

You must perform this operation in the old API Gateway instance.

Quiescing API Gateway temporarily disables access to the API Gateway server. When the API Gateway server access is disabled, you can perform the required upgrade tasks while the API Gateway server is not connected to any external resources. For more information on quiesce mode, see "Quiesce Mode" on page 51.

While flushing the performance, license, and quota metrics data, API Gateway performs the following operations:

- Resets performance and license metrics intervals for all the APIs and stores the data in the Elasticsearch or API Data Store.
- Stores the subscription quota in the Elasticsearch or API Data Store.

When all the performance metrics and other data are flushed completely, quiesce mode is enabled for all the API Gateway nodes, and a notification is sent through a webhook event.

To operate API Gateway in quiesce mode, execute the following steps:

1. Create a quiesce port.

Before you invoke the quiesce mode, you must create a quiesce port, and enable it in the old API Gateway instance.

You can use the following request to create and enable the quiesce port.

```
POST /rest/apigateway/ports
{
    "factoryKey": "webMethods/HTTP",
    "pkg": "WmRoot",
    "port": "4444",
    "portAlias": "QuiescePort"
}
```

In the example above, the 4444 port is used.

Once you create the port, use the following request to enable the quiesce port.

```
PUT rest/apigateway/ports/enable
{
    "listenerKey": "HTTPListener@4444",
    "pkg": "WmRoot",
    "requestServiceStatus": ""
}
```

After the quiesce port is enabled, use the following request to set the quiesce port. You must use the same `portAlias` value that you used while creating the quiesce port. In this example, the `portAlias` used while creating the quiesce port is `QuiescePort`.

```
PUT invoke/wm.API Gateway server.quiesce/setQuiescePort
{
    "portAlias": "QuiescePort"
}
```

2. Enable or disable the quiesce mode.

You can use the following request to enable or disable the quiesce mode.

```
PUT /rest/apigateway/quiescemode
{
    "enable": "true/false",
    "block": "designtime/all",
    "flush": [
        "license_metrics",
        "performance_metrics",
        "subscription_quota"
    ]
}
```

The parameters used in the above request are described in the following table:

Parameter	Description
`enable`	Set the enable parameter as `true` to enable quiesce mode and `false` to disable the quiesce mode. You can disable the quiesce mode only when the block parameter is set as `designtime`. When the block parameter is

Parameter	Description
	set as `all`, API Gateway is quiesced and is unreachable. Hence, you cannot disable the API Gateway quiesce mode by invoking this request.
`block`	Set the block parameter as `designtime` to block only design time requests and `all` to block all the requests.
`flush`	This parameter is not applicable when you set the block parameter as `designtime`. The flush parameter is applicable only when you set the block parameter as `all`. In the above example, when you set the block parameter as `all`, API Gateway blocks all the requests and flushes the license metrics, performance metrics, and the subscription quota data. If you do not specify any value in this parameter, all the data is flushed.

3. Retrieving the quiesce mode status.

   Apart from the webhook notification of the quiesce mode status, you can use the following request to retrieve the quiesce mode status.

```
GET /rest/apigateway/quiescemode
```

Response

```
{
    "enable": "true/false",
    "block": "designtime/all",
    "flush": [
        "license_metrics",
        "performance_metrics",
        "subscription_quota"
    ],
    "status": "success",
    "failureReason": null
}
```

Note:
This request works only for quiesce mode for `designtime`. This request will not work for `all`, because the API Gateway package is disabled when you invoke quiesce mode for `all`. In such cases, you must invoke the API Gateway health check API through the quiesce port. If it returns a status code other than 200 or no response, then the quiesce mode for `all` is completed in that node successfully.

Migration REST API

You must perform the migration in the new API Gateway instance using the migration REST API.

The migration REST API triggers the migration action and returns a 202 status code, indicating that the migration request is accepted. When the action is completed, a webhook event with the migration status is sent to the registered webhook clients.

1. Clean the Elasticsearch or API Data Store.

This action deletes the data from the Elasticsearch or API Data Store. You must run this request before invoking the re-index request for core indices.

```
POST /rest/apigateway/migration
{
    "action": "clean"
}
```

Note:
As a safety measure, by default this action takes a backup, before deleting the data. As a prerequisite to backup data, you must configure the `path.repo` property in the elasticsearch.yml file for the Elasticsearch or API Data Store, to take a snapshot in the location specified in the `path.repo` property. If the administrator deliberately wants to exclude the backup of the data, then add `"backup": false` in the payload.

2. Re-index the Elasticsearch or API Data Store.

   The re-index action re-indexes the data from the old Elasticsearch or API Data Store to new Elasticsearch or API Data Store.

   Note:
   The remote data store host must be explicitly allowed in elasticsearch.yml file using the `reindex.remote.whitelist` property. Configure the `reindex.remote.whitelist` in the elasticsearch.yml file located at `SAGInstallDIr/InternalDataStore/config/`.

   An example to set the value of old Elasticsearch or API Data Store host and port is as follows:

   ```
   reindex.remote.whitelist: localhost:9200
   ```

   The request to re-index the Elasticsearch or API Data Store is as follows:

   ```
   POST /rest/apigateway/migration
   {
       "action": "reindex",
       "indicesType": "core/analyticsandlogs",
       "sourceElasticsearch": {
           "url": "http://remotehost:9240",
           "username": "username",
           "password": "password"
       },
       "properties": {
           "apigateway.migration.srcTenantName": "default",
           "apigateway.migration.batchSize": 100,
           "apigateway.migration.logLevel": "info",
           "apigateway.migration.reindex.status.check.sleep.interval": 5000,
           "apigateway.migration.batchSize.gateway_{0}_apis": 50,
           "apigateway.migration.batchSize.gateway_{0}_log": 100,
           "apigateway.migration.batchSize.gateway_{0}_audit_auditlogs": 50,
          "apigateway.migration.batchSize.gateway_{0}_analytics_transactionalevents":
    50
       }
   }
   ```

   The parameters used in the above request are described in the following table:

Parameter	Description
`indicesType`	Specifies the type of indices to be re-indexed from the old Elasticsearch or API Data Store to new Elasticsearch or API Data Store. The possible values are `core` and `analyticsandlogs`. Core indices are used to store the design time assets like APIs, alias, configurations, and so on. Analyticsandlogs indices are used to store the logs and events like performance metrics, license metrics, and logs.
`sourceElasticsearch`	The old Elasticsearch or API Data Store details.
`properties`	These migration properties can be used to specify the old API Gateway instance's tenant name and optimize the performance of the re-index action. If not specified, the values from `migration.properties` file located at `SAG\IntegrationAPI Gateway server\instances\default\packages\WmAPIGateway\bin\migrate` are taken.

3. Transform the Elasticsearch or API Data Store data.

   The transform action transforms the re-indexed assets in the new Elasticsearch or API Data Store to be compatible with the new API Gateway version. The request to perform the transformation is as follows:

```
POST /rest/apigateway/migration
{
    "action": "transform"
}
```

4. *Optional.* Use the following request to check the migration action status.

```
GET /rest/apigateway/migration/status?action=[reindex | transform | clean]
```

Response

```
{
    "status": "success",
    "failureReason": null
}
```

Webhook Notifications

When quiesce mode or a migration action is completed, a webhook event with the migration status is sent to the registered webhook clients. The following webhook events are supported.

Webhook Event	Action
apigateway:API Gateway server:started	This event is started after API Gateway server is up.

Webhook Event	Action
migration:clean:API Datastore:completed	This event is started after the clean action is completed.
migration:quiesce:all:completed	This event is started after the quiesce mode for `all` the requests are completed.
migration:quiesce: designtime:completed	This event is started after the quiesce mode for `designtime` requests are completed.
migration:reindex: core:completed	This event is started after re-indexing the core Elasticsearch indices.
migration:reindex: analyticsandlogs:completed	This event is started after re-indexing the logs and events Elasticsearch indices.
migration:transform: assets:competed	This event is started after all the migration handlers for asset transformation is finished running.

The create, read, update, and delete (CRUD) operations on webhooks, can be performed using the API Gateway webhook API. The operations of the webhook APIs are as follows:

Create Webhook

You can use the following request to create a webhook.

```
POST /rest/apigateway/wehooks
{
    "config": {
        "url": "http://apigatewaymig.free.beeceptor.com/my/api/path/test",
        "headers": {"type": "core"},
        "username": null,
        "password": null,
        "truststoreAlias": null
    },
    "events": [
        "migration:reindex:core:completed"
    ],
    "active": true
}
```

The parameters used in the request are described in the following table:

Parameter	Description
`config`	Specifies webhook client endpoint configuration details like URL, headers, and so on.
`events`	Specifies the list of interested events the webhook is registered for.
`active`	Specifies whether this webhook is active or not. The default value is false.

Update Webhook

You can use the following request to update a webhook.

```
PUT /rest/apigateway/wehooks/{id}
{
    "config": {
        "url": "http://apigatewaymig.free.beeceptor.com/my/api/path/test",
        "headers": {"type": "core"},
        "username": null,
        "password": null,
        "truststoreAlias": null
    },
    "events": [
        "migration:reindex:core:completed"
    ],
    "active": true
}
```

Retrieve Webhook

You can use the following request to retrieve the webhook with a specific id.

```
GET /rest/apigateway/wehooks/{id}
```

You can use the following request to retrieve all the defined webhooks from API Gateway.

```
GET /rest/apigateway/wehooks
```

Delete Webhook

You can use the following request to delete the webhook with a specific id.

```
DELETE /rest/apigateway/wehooks/{id}
```

Shutdown REST API

You can use the following request to shutdown the API Gateway server.

```
POST /rest/apigateway/shutdown
{
    "bounce": "true/false",
    "option": "force/drain",
    "timeout": 10,
    "quiesce": "true/false"
}
```

The parameters used in the request are described in the following table:

Parameter	Description
`bounce`	Specifies whether to restart the API Gateway server after shutdown. If you specify the value as `true`, the API Gateway server restarts. The default value is `false`.

Parameter	Description
`option`	Specifies whether to shutdown the API Gateway server immediately or after all client sessions are ended. If you specify the value as `force`, the API Gateway server is shutdown immediately. If you specify the value as `drain`, API Gateway waits for a maximum period of time for all the client sessions to end before shutdown.
`timeout`	Specifies the maximum wait time in minutes before the API Gateway server is shutdown when `drain` option is specified.
`quiesce`	If you specify value as `true`, API Gateway flushes the in-memory data like performance metrics, license metrics, and subscription quota to Elasticsearch or API Data Store before the API Gateway server is shutdown. After this when API Gateway is restarted either manually or using bounce parameter, the Integration Server is started in quiesce mode. **Note:** In a cluster, the flushing of in-memory data is performed only on one of the nodes and on other nodes the API call returns immediately by eliminating the flush time. The default value is `false`.

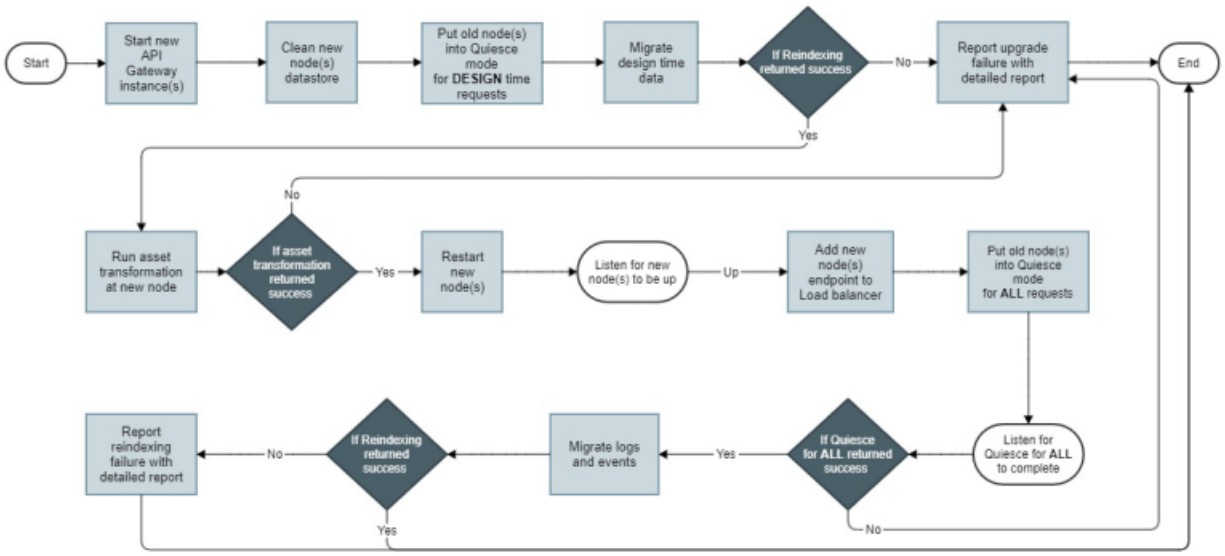
Upgrading Major Versions in Zero Downtime

This section explains the steps to upgrade API Gateway in zero downtime for major versions like 10.5 to 10.7, 10.7 to 10.11, and so on.

Prerequisites

- The minimum version of the API Gateway instance that is being upgraded to a higher version (10.7, 10.11, and so on) must be version 10.5. The target API Gateway instance version can be 10.7 and above.
- Ensure that the old API Gateway instance is running.
- Create a quiesce port in old API Gateway instance. If you are using a cluster setup, you must create a quiesce port for each API Gateway instance in the old instance. To learn more about quiesce port, see "API Gateway Quiesce Mode" on page 32.

The following image represents the steps to perform a major upgrade:



Each of the above step is explained in the following sections:

Start the new API Gateway instance

You must install the new API Gateway instance and perform the following steps:

1. As zero downtime upgrade deals only with the migration of Elasticsearch or API Data Store data, the platform configurations must be migrated to the new API Gateway instance either manually or using the externalized configurations..

   For more information on the platform configurations that must be migrated to the target API Gateway instance, see "Migrating the platform configurations" on page 11.

2. Configure `reindex.remote.whitelist` in the elasticsearch.yml file.

   You must explicitly allow the remote Elasticsearch or API Data Store host in elasticsearch.yml using the **reindex.remote.whitelist** property.

   Configure `reindex.remote.whitelist` in elasticsearch.yml file located at `SAG/InternalDataStore/config`. Set the value of old Elasticsearch or API Data Store host and port .

   An example to set the Elasticsearch or API Data Store host and port using the as `reindex.remote.whitelist` is as follows:

   ```
   reindex.remote.whitelist: localhost:9200
   ```

3. If the old Elasticsearch or API Data Store instance is secured with SSL, you must perform one of the following steps to configure the new Elasticsearch or API Data Store to securely connect to the old Elasticsearch or API Data Store.

   - **Configuring SSL parameters in elasticsearch.yml**

Configure the SSL related re-index settings in the new Elasticsearch or API Data Store's elasticsearch.yml file. For more information, see Elasticsearch documentation for re-index SSL settings. If you are having a cluster setup, this must be performed in each node.

- **Importing old Elasticsearch or API Data Store certificates into new Elasticsearch or API Data Store JVM**

  You must add the old instance certificates (public key) into the new Elasticsearch or API Data Store JVM's truststore. For example, in case of internal Elasticsearch or API Data Store you must add the public keys to the truststore cacerts file located at `\jvm\jvm\jre\lib\security`.

  Note:
  If external Elasticsearch is used for the new API Gateway instance, you must import the certificates to their corresponding JVM.

  The following sample command imports the truststore of old Elasticsearch or API Data Store to the new Elasticsearch or API Data Store JVM.

```
$>\jvm\jvm\bin\keytool -import -keystore \jvm\jvm\jre\lib\security\cacerts
-file <truststore.jks> -alias <alias>
```

Property	Detail	Example
truststore.jks	Truststore used in the Elasticsearch. Provide the full path of the truststore. It is available at `\WmAPIGateway\config\resources\elasticsearch\config.properties` file with property **pg.gateway.elasticsearch.https.truststore.filepath**.	`sagconfig/root-ca.jks`
alias	The alias used in the Elasticsearch.	wm.sag.com

You can now start the new API Gateway instance with the new Elasticsearch or API Data Store (cluster and Terracotta cluster).

Clean the data in the new API Data Store

You must perform this operation on the new API Gateway instance. For a cluster setup, you must do this in only one of the nodes.

Once you start the new API Gateway instance, you must clean the data in the Elasticsearch or API Data Store. You can run the following request to clean the data in the Elasticsearch or API Data Store.

```
POST /rest/apigateway/migration
{
    "action": "clean"
}
```

Note:

As a safety measure, by default this action takes a data backup, before cleaning the data. As a prerequisite to backup data, you must configure the `path.repo` property in the elasticsearch.yml file for the Elasticsearch or API Data Store, to take a snapshot in the location specified in the `path.repo` property. If the administrator deliberately wants to exclude the backup of data, then add `"backup": false` in the payload.

When the clean operation is completed, API Gateway sends out a notification with the result details through the registered webhooks. You can run the following request to check the status of the operation.

```
GET /rest/apigateway/migration/status?action=clean
```

If the above request returns success as the status, the data is cleaned successfully.

If the clean request fails with an error or the status returned with a failure, stop proceeding with the next steps. If the error persists, contact Software AG support team for help with all the relevant logs for further analysis.

Put the old API Gateway instances in quiesce mode for design time

You must perform this operation in the old API Gateway instances. For a cluster, you must do this at each node as quiesce mode is enabled at the instance level.

API Gateway data in the API Data Store is migrated in two phases. In the first phase, only the design time data like APIs, applications, and policies are migrated. The runtime data like audit logs, transaction logs, performance metrics, and so on are migrated in the next phase, that is, after the runtime transactions to the old instances are stopped. This is performed after the new instance is up and running with the new data and runtime traffic is allowed to it.

Before migrating design time data in the first phase, all the updates to them through UI and REST APIs must be blocked. This can be achieved by using the quiesce mode capability available in API Gateway.

You can use the following request to enable quiesce mode for design time in the old API Gateway instance.

```
PUT /rest/apigateway/quiescemode
{
    "enable": true,
    "block": "designtime"
}
```

If the invocation is successful, the request returns a 200 OK message.

Now, all the design time invocations to the old instance are blocked and API Gateway returns 503 message to the client. Only the following requests are allowed:

- All GET requests
- /rest/apigateway/quiescemode
- /rest/apigateway/search

- /rest/facade/apigateway/searchApis

When the quiesce mode for design time is complete, API Gateway sends out a notification with the result details through registered webhooks. You can also run the following request to check the status of the operation.

```
GET rest/apigateway/quiescemode
```

A success status message implies that quiesce mode for design time is completed.

If the quiesce mode for design time API call fails with an error or the status returned with a failure, you must stop execution of the next steps. Disable the quiesce mode for design time in all the nodes and bring them back to normal state. If the error persists, contact Software AG support team for help with all the relevant logs for further analysis.

Migrate the design time data

You must perform this operation in the new API Gateway instance. For a cluster setup, you must do this in only one of the nodes.

When the old API Gateway instance is put into quiesce mode for design time, it is safe to migrate the design time data to the new API Gateway instance's Elasticsearch or API Data Store.

You can run the following request in the new API Gateway instance to migrate the design time data. For a cluster, you must do this only on one of the instances. If the old Elasticsearch or API Data Store is protected with basic authentication, send the username and password in the request payload.

The source tenant name is assumed as *default*. But, if the source API Gateway has multiple tenants, this property can be used to specify the tenant name from which the data has to be migrated to the target tenant. For more details, see "Upgrade configurations" on page 24.

```
POST /rest/apigateway/migration
{
    "action": "reindex",
    "indicesType": "core",
    "sourceElasticsearch": {
        "url": "http://localhost:9200",
        "username": "username",
        "password": "password"
    },
    "properties": {
        "apigateway.migration.srcTenantName": "default",
        "apigateway.migration.batchSize": 100,
        "apigateway.migration.logLevel": "info",
        "apigateway.migration.reindex.status.check.sleep.interval": 5000,
        "apigateway.migration.batchSize.gateway_{0}_apis": 50,
        "apigateway.migration.batchSize.gateway_{0}_log": 100,
        "apigateway.migration.batchSize.gateway_{0}_audit_auditlogs": 50,
        "apigateway.migration.batchSize.gateway_{0}_analytics_transactionalevents":
50
    }
}
```

Since this is a long running operation, the request initiates the migration and returns 201 Accepted message, if the invocation is successful.

When the reindex operation is completed, API Gateway sends out a notification with the result details through registered webhooks. You can run the following request to check the status of the operation. .

```
GET /rest/apigateway/migration/status?action=reindex
```

A success status message implies that the re-indexing is completed.

If the re-index request invocation for design time data fails with an error or the status returned with a failure, stop proceeding with the next steps. Disable the quiesce mode for design time in all the nodes and bring them back to normal state. Clean the data and retry. If the error persists, contact Software AG support team for help with all the relevant logs for further analysis.

Transform the design time data

You must perform this operation in the new API Gateway instance. For a cluster setup, you must do this in only one of the nodes.

After the design time data is migrated to new API Data Store, you must transform the data to be compatible with the new API Gateway instance. You can run the following request to transform the data to be compatible with the new API Gateway instance.

```
POST /rest/apigateway/migration
{
    "action": "transform"
}
```

Transform action status

Since transformation is a long running operation, the request initiates the action and returns 201 Accepted status, if the invocation is successful. When the re-index operation is complete, API Gateway sends out a notification with the result details through registered webhooks. You can use the following request to check the status of the operation.

```
GET /rest/apigateway/migration/status?action=transform
```

A success status message implies that transformation is completed.

If the transform request invocation for design time data fails with an error or the status returned with a failure, you must stop the execution of the next steps. Disable the quiesce mode for design time in all the nodes and bring them back to normal state. You can clean the data and retry from re-indexing of design time data. If the error persists, contact Software AG support team for help with all the relevant logs for further analysis.

Restart new API Gateway instances

The design time data is migrated to the new Elasticsearch or API Data Store and transformed to be compatible with the new API Gateway instance. For the new instances to take the new data,

you must restart the instances. You must run the following request to restart the API Gateway instances.

```
POST /rest/apigateway/shutdown
{
    "bounce": true,
    "option": "force"
}
```

For a cluster setup, you must run the request on all the nodes.

If the restart of new instances fails, you must stop the execution of next steps. Disable the quiesce mode for design time in all the nodes and bring them back to normal state. If the error persists, contact Software AG support team for help with all the relevant logs for further analysis.

Add traffic to the new API Gateway instance

After the new API Gateway instance is restarted, the runtime traffic can be allowed to flow to the new API Gateway instance. Adding traffic to the new instance depends on the deployment model.

For example, for on-premise and Docker deployment, it would be an update to the load balancer for adding the new endpoints. For Kubernetes deployment, it would be a label change in the service resources.

Put the old API Gateway instances into quiesce mode for all

You must perform this operation in the old API Gateway instance. For a cluster setup, you must do this on each node as Quiesce mode is performed at instance level.

At this time, the runtime traffic is flowing to both old and new API Gateway instances. You must stop the traffic to old instances and allow the traffic to be sent only to the new instances. But while stopping the old instances, there may be ongoing requests in the old instances that are in progress and cannot be lost. Moreover, all the in-memory data like performance metrics, license metrics, and subscription quota must be stored in the Elasticsearch or API Data Store before runtime data is migrated. This can be achieved by putting the old instances into quiesce mode with status as `all`.

For more information on the process of enabling and disabling quiesce mode, see "API Gateway Quiesce Mode" on page 32.

If the quiesce mode for all request invocation fails with an error or the status is returned with a failure, you must stop the execution of the next steps. Disable the quiesce mode for design time in all the nodes and bring them back to normal state. If the instances have already entered the quiesce mode, restart all the instances and update the load balancer to route the traffic only to it to go back to the original state before migration. If the error persists, contact Software AG support team for help with all the relevant logs for further analysis.

Migrate the analytics data

You must perform this operation on the new API Gateway instance. For a cluster setup, you must do this on each node.

Note:
If the analytics data volume is high, migrate only the core data using the APIs provided earlier, and migrate the analytics data using the backup and restore command.

The steps to migrate the analytics data using backup and restore command is as follows:

- Take a backup of the analytics data from the source API Gateway instance using the following command:

  ```
  <SOURCE>\IntegrationServer\instances\default\packages\WmAPIGateway\
  cli\bin\apigatewayUtil
  create backup -include analytics -name <backup_file_name>
  ```

  For more information on taking backup, see *webMethods API Gateway Administration*.

- Copy the backup file to the target API Gateway instance `path.repo` folder configured under `<TARGET>\IntegrationServer\InternalDataStore\config\elasticsearch.yml`.

- Restore the backup data in the target API Gateway instance using the following command:

  ```
  <TARGET>\IntegrationServer\instances\default\packages\WmAPIGateway\
  cli\bin\apigatewayUtil
  restore backup -include analytics -name <backup_file_name>.
  ```

  For more information on restoring backup, see *webMethods API Gateway Administration*.

Ignore the following step if you have chosen to migrate the analytics data using the backup and restore command.

Once you complete the quiesce mode for `all` in the old API Gateway instance, the analytics data can be migrated to the new instance. You can invoke the following request in the new API Gateway instance to migrate the data.

The source tenant name is assumed as *default*. But, if the source API Gateway has multiple tenants, this property can be used to specify the tenant name from which the data has to be migrated to the target tenant. For more details, see "Upgrade configurations" on page 24.

```
POST /rest/apigateway/migration
{
    "action": "reindex",
    "indicesType": "analyticsandlogs",
    "sourceElasticsearch": {
        "url": "http://localhost:9200",
        "username": "username",
        "password": "password"
    },
    "properties": {
     "apigateway.migration.srcTenantName": "default",
        "apigateway.migration.batchSize": 100,
        "apigateway.migration.logLevel": "info",
```

```
        "apigateway.migration.reindex.status.check.sleep.interval": 5000,
        "apigateway.migration.batchSize.gateway_{0}_apis": 50,
        "apigateway.migration.batchSize.gateway_{0}_log": 100,
        "apigateway.migration.batchSize.gateway_{0}_audit_auditlogs": 50,
        "apigateway.migration.batchSize.gateway_{0}_analytics_transactionalevents":
50
    }
}
```

After this, the runtime data is added to the existing runtime data. The upgrade is complete with this step.

If the migration of the analytics data fails with an error or the status returned with a failure, then troubleshoot the problem by looking at the logs. You can find the logs at the target directory `<TARGET>/profiles/IS_profile name/logs/wrapper.log`. If the error persists, contact Software AG support team for help with all the relevant logs for further analysis.

Note:
These are the logs and events indices in the Elasticsearch or API Data Store.

```
gateway_tenantId_quotaAccumulator, gateway_tenantId_analytics_performanceMetrics,
gateway_tenantId_analytics_threatProtectionEvents,
gateway_tenantId_analytics_lifecycleEvents,
gateway_tenantId_analytics_errorEvents, gateway_tenantId_analytics_transactionalEvents,

gateway_tenantId_analytics_policyViolationEvents,
gateway_tenantId_analytics_monitorEvents,
gateway_tenantId_license_licenseMetrics, gateway_tenantId_license_notifications,
gateway_tenantId_log and gateway_tenantId_audit_auditlogs.
```

Shutdown the old API Gateway instance

If migration is successful and the new API Gateway nodes are stable, shutdown the old nodes so that they do not send any metrics to any configured destinations like API Portal, external Elasticsearch, and so on. The new instance now receives the runtime transactions. You can remove the old instance's endpoints from the load balancer.

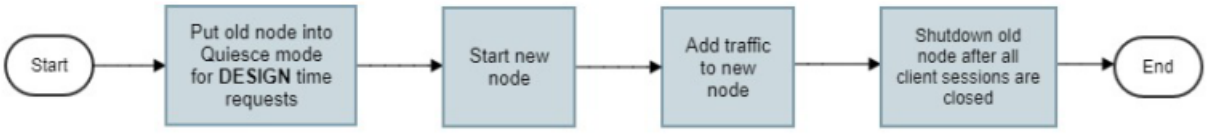
Upgrading Minor Versions in Zero Downtime

This section explains the steps to upgrade API Gateway in zero downtime for minor versions like fix versions of 10.11 fix 1 to 10.11 fix 2.

Upgrade Standalone Deployment

API Gateway follows the blue-green approach to upgrade a standalone deployment. In such deployment scenario, old instance are allowed to run and serve the transactions while the new API Gateway version is being prepared. There is no data migration as both the old and new versions would be pointing to the same Elasticsearch or API Data Store. The new API Gateway version is started and the old instance is blocked for core design time data updates. After that the new node's endpoint is added to the load balancer. At this time the transactions are served by both the old and new versions. Next, all the transactions to the old version is blocked and the logs and events data are migrated to the new version while it is serving the transactions.

The following image shows the workflow for a minor upgrade of a standalone API Gateway instance.



To upgrade API Gateway minor versions in zero downtime

1. Put the old API Gateway instance in quiesce mode for design time.

   You must perform this operation on the old API Gateway instance.

   API Gateway data in the Elasticsearch or API Data Store is migrated in two phases. In the first phase, only the design time data like APIs, applications, and policies are migrated. The runtime data like audit logs, transaction logs, performance metrics, and so on are migrated in the next phase, that is, after the runtime transactions to the old instances are stopped. This is performed after the new instance is up and running with the new data and runtime traffic is allowed to it.

   Before migrating design time data in the first phase, all the updates to them through UI and REST APIs must be blocked. This can be achieved by using the quiesce mode capability available in API Gateway.

   Run the following request to enable quiesce mode for design time in the old API Gateway instance.

   ```
   PUT /rest/apigateway/quiescemode
   {
       "enable": true,
       "block": "designtime"
   }
   ```

   If the invocation is successful, API Gateway returns 200 OK message.

   Now, all the design time invocations to the old instance are blocked and API Gateway returns 503 message to the client. Only the following requests are allowed:

   - All GET requests
   - /rest/apigateway/quiescemode
   - /rest/apigateway/search
   - /rest/facade/apigateway/searchApis

   When the quiesce mode for design time is complete, API Gateway sends out a notification with the result details through registered webhooks. You can also run the following request to check the status of the operation.

   ```
   GET rest/apigateway/quiescemode
   ```

A success status message implies that quiesce mode for design time is complete.

If the quiesce mode for design time request invocation fails with an error or the status returned with a failure, you must stop the execution of the next steps. Disable the quiesce mode for design time in all the nodes and bring them back to normal state. If the error persists, contact Software AG support team for help with all the relevant logs for further analysis.

2. Start the new API Gateway instance.

   You must install the new instance of API Gateway and perform the following steps:

   a. As zero downtime upgrade deals only with the migration of Elasticsearch or API Data Store data, the platform configurations and must be migrated to the new API Gateway instance either manually or using the externalized configurations.

      For more information on the platform configurations that must be migrated to the target API Gateway instance, see "Migrating the platform configurations" on page 11.

      You can now start the new API Gateway instance by connecting it to the old Elasticsearch or API Data Store as an external.

3. Add traffic to the new API Gateway instance.

   After the new API Gateway instance is restarted, the runtime traffic can be allowed to flow to it. Adding traffic to the new instance depends on the deployment model.

   For example, for on-premise and Docker deployment, it would be an update to load balancer for adding the new endpoints. For Kubernetes deployment, it would be a label change in service resources.

4. Shutdown the old API Gateway instance.

   When the new instance is added to the load balancer, you can shutdown the old instance after all the active client sessions in that node are closed. This ensures that no metrics are sent to any configured destinations like API Portal, external Elasticsearch, and so on. The new instance is now receiving the runtime transactions. You can remove the old instance endpoint from the load balancer.

   You can run the following request to shutdown the old API Gateway instance.

```
POST /rest/apigateway/shutdown
{
    "bounce": "true/false",
    "option": "force/drain",
    "timeout": 10,
    "quiesce": "true/false"
}
```

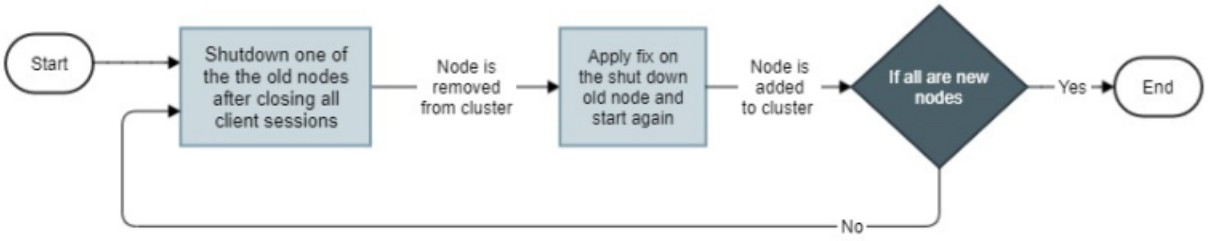
   The timeout value depends on the number of active client sessions.

   The API Gateway upgrade is complete. You can find the logs at the target directory `<TARGET>/profiles/IS_profile name/logs/wrapper.log`.

Upgrade Cluster Deployment

Zero downtime upgrade of minor versions in a cluster setup is a rolling upgrade. Old nodes are brought down and taken out of the cluster one by one while new nodes are brought up and added to the cluster one by one. There is no data migration as both the old and new instances point to the same Elasticsearch or API Data Store.

The following image shows the workflow for a minor upgrade of a API Gateway cluster.



Prerequisites

The old API Gateway instance must be up and running.

To upgrade API Gateway minor versions in zero downtime

1. Shutdown one of the old API Gateway nodes.

   Note:
   If all the nodes in the cluster are new nodes, then skip this step.

   You must shutdown the old nodes after all the active client sessions in that node are closed. You can run the following request to shutdown the old instance.

   ```
   POST /rest/apigateway/shutdown
   {
       "bounce": "true/false",
       "option": "force/drain",
       "timeout": 10,
       "quiesce": "true/false"
   }
   ```

   The timeout value depends on the number of active client sessions.

   If the shutdown fails with error, restart all the instances and update the load balancer to route the traffic only to the instances, to go back to the original state before migration. If the error persists, contact Software AG support team for help with all the relevant logs for further analysis.

2. Apply the 10.11 fix and start.

   When the node in Step 1 is shutdown, apply the fix and start again.

Note:
For Docker case the fix can't be applied on the old container. Instead a new container from an image already built with the fix is started. Make sure the Integration Server configurations which are not stored in the Elasticsearch or API Data Store and custom Integration Server packages are pre-configured in the new containers when starting.

Repeat the steps 1 and 2, until all the nodes in the cluster become new nodes.

The API Gateway upgrade is complete. You can find the logs at the target directory `<TARGET>/profiles/IS_profile name/logs/wrapper.log`.

Quiesce Mode

Quiescing API Gateway temporarily disables access to the server so you can perform the required tasks while the server is not connected to any external resources. In API Gateway the quiesce mode is used during the zero downtime upgrade wherein the access to the server is temporarily disabled, so you can perform the upgrade tasks.

API Gateway supports the following two types of quiesce mode:

- Type 1: Blocks only design time requests.

  In this mode API Gateway server blocks all design time requests such as all the CRUD operations for API Gateway assets like APIs, policies, applications, and so on, and returns an appropriate status code (503) to the client. The following requests are an exception and are still allowed by API Gateway:

  - All GET requests
  - /rest/apigateway/quiescemode
  - /rest/apigateway/search
  - /rest/facade/apigateway/searchApis

- Type 2: Blocks all requests

  In this mode API Gateway becomes unreachable for all the requests, design time, and runtime. Any requests that are already in progress are permitted to finish, but any new inbound requests to the server are blocked. This mode is an extension of the Integration Server quiesce mode, where API Gateway flushes the Quota, performance, and transaction license metrics data to the data store before the API Gateway package is disabled. While flushing the performance, license, and quota metrics data, API Gateway performs the following operations:

  - Resets performance and license metrics intervals of the corresponding scheduler for all the APIs to be run immediately and flushes the data to be persisted into the database. For example:

    - API Gateway publishes performance metrics as per the `Publish interval` configured. That is, if the `Publish interval` is set as 60 seconds, then API Gateway publishes the performance metrics to the configured destination after 60 seconds. But, in quiesce mode API Gateway immediately publishes the performance metrics to the database.

- API Gateway publishes transaction license metrics after 30 minutes, which is the default interval scheduled to publish the license metrics, to the data store. But in quiesce mode API Gateway immediately publishes the license metrics to the data store.

In a cluster, only the controlling node performs this operation.

- Starts the subscription quota intervals and counters persisting scheduler immediately to store the data in the database. For example, API Gateway runs the subscription quota counters for an interval configured as per the `pgmen.quotaSurvival.interval` extended setting and publishes the data after the specified interval to the data store. But in quiesce mode API Gateway immediately publishes the quota metrics to the data store.

When all the performance metrics and other data are flushed completely, a notification mentioning that quiesce mode for all is enabled is sent through a webhook. The CRUD operations of webhook objects are supported in the Administration REST API of API Gateway. For more details, see API Gateway Administration API in the *webMethods API Gateway User's Guide*.

For details about quiesce mode in Integration Server, see *webMethods Integration Server Administrator's Guide*.

Enabling and Disabling Quiesce Mode

API Gateway provides a REST API to enable quiesce mode.

Enable or disable quiesce mode with the following REST call:

PUT /rest/apigateway/quiescemode

```
{
    "enable": true/false,
    "block": "designtime/all"
    "flush": [
        "license_metrics",
        "performance_metrics",
        "subscription_quota"
    ]
}
```

Set the block parameter as `designtime` to block only design time requests. The flush parameter is not applicable when you set the block parameter as `designtime`.

Set the block parameter as `all` to block all requests. The flush parameter is applicable when you set the block parameter as `all`. In the above example, when you set the block parameter as `all`, API Gateway blocks all the requests and flushes the license metrics, performance metrics, and subscription quota data. If you do not specify anything for the flush parameter, then all the data is flushed.

Set the enable parameter as `true` to enable quiesce mode.

Set the enable parameter as `false` to disable the quiesce mode. This is supported only when the block parameter is set as `designtime`. When the block parameter is set as `all`, API Gateway is quiesced and is unreachable. Hence, you cannot disable the API Gateway quiesce mode by invoking this API.

Use the Integration Server API `POST /invoke/wm.server.quiesce/setActiveModeinvoke/wm.server.quiesce/setActiveMode`, which is accessed through the quiesce port, to disable the Integration Server quiesce mode when the block parameter is set as `all`. This, in turn, disables the quiesce mode for all requests in API Gateway.

Note:
For invoking the quiesce mode for all requests, you must specify the port through which the server enters and exits quiesce mode. For details about specifying the quiesce port, see *webMethods Integration Server Administrator's Guide*.

Retrieve the quiesce mode settings as well as the current status with the following REST call.

GET /rest/apigateway/quiescemode

```
{
    "enable": true/false,
    "block": "designtime/all"
    "flush": [
        "license_metrics",
        "performance_metrics",
        "subscription_quota"
    ],
    "status": "success",
    "failureReason": null
}
```

3 Upgrading API Gateway with external Elasticsearch and Kibana

You can upgrade API Gateway when it uses external components like Elasticsearch and Kibana are configured with API Gateway.

External component flavors and upgrade procedures

External component	Upgrade procedure
External Elasticsearch for core data and analytics	API Gateway uses external Elasticsearch to store the core data and analytics. To upgrade a standalone API Gateway in such scenario, see "Upgrading standalone API Gateway" on page 13. To upgrade a API Gateway cluster in such scenario, see "Upgrading API Gateway cluster" on page 18.
External Elasticsearch for core data	API Gateway uses external Elasticsearch to store the core data. To upgrade a standalone API Gateway in such scenario, see "Upgrading standalone API Gateway" on page 13. To upgrade a API Gateway cluster in such scenario, see "Upgrading API Gateway cluster" on page 18.
External Elasticsearch for analytics through destination	API Gateway uses external Elasticsearch for analytics through destination. In this case, no migration is needed.
External Kibana	API Gateway uses external Kibana for search and data visualization capabilities. In this case, no migration is needed.

4 Migrating from Mediator to API Gateway

API Gateway supports the migration of Mediator 9.7 and later; earlier versions of Mediator should be migrated first.

Migrating Mediator Deployments to API Gateway

Existing Mediator deployments can be migrated to API Gateway by publishing the virtual service, applications, and runtime aliases to API Gateway. This lets you build an API Gateway runtime enforcement landscape in parallel to the existing Mediator landscape.

To migrate the existing Mediator deployments, perform the following procedure:

- For all installed Mediators:
  1. Stop Mediator.
  2. Install corresponding API Gateway.
  3. Migrate Mediator configuration to API Gateway.
- For all Mediator targets configured in CentraSite:
  1. Configure a corresponding API Gateway in CentraSite.
  2. Deploy all virtual services from the Mediator target to the corresponding API Gateway.
  3. *Optional.* Undeploy all virtual services from the Mediator target.

Note:
The procedure assumes that the Mediators and the corresponding API Gateway provide the same endpoints. Therefore either the Mediator or its corresponding API Gateway can be up and running. If the endpoint compatibility is not required it is not necessary to stop the Mediators. Also undeploying the Mediator deployments is optional. This means Mediator and API Gateway can be driven by CentraSite in parallel.

Migrating Mediator Configurations to API Gateway

As the publishing of virtual services, applications, and runtime aliases to API Gateway is done through CentraSite, CentraSite is required for migrating Mediator to API Gateway.

To migrate existing Mediator configurations to API Gateway, perform the following procedure:

1. Run IS migration using the IS migration tool.

   For details of the IS migration tool, see *Upgrading Software AG Products On Premises*.

2. Run Mediator migration using the API Gateway migration tool.

   The API Gateway migration tool is available within the IS instance running the API Gateway. If API Gateway is running in the default IS instance the tool is available in the folder: `Install_Dir/IntegrationServer/instances/default/packages/WmAPIGateway/bin/migrate`.

   The script migrateFromMediator.sh has two parameters:

   - Full path to Integration service installation running the Mediator to be migrated. (for example, E:/SoftwareAG/IntegrationServer)
   - Name of the instance that is running the Mediator (for example, default)

   On Unix the script can be invoked as follows:

   ```
   ./migrateFromMediator.sh /opt/softwareag/IntegrationServer default
   ```

   On Windows the script can be invoked as follows:

   ```
   migrateFromMediator.bat C:\SoftwareAG\IntegrationServer default
   ```

3. Start API Gateway.

The Mediator configuration migration covers the following configuration items:

- Elasticsearch
- SNMP
- Email
- HTTP Configuration
- Keystore Configuration
- Ports Configuration
- Service Fault
- Extended Settings

The following configuration items are not automatically migrated. The configuration of these items have to be done manually in API Gateway.

- Security Token Service (STS) Configuration
- `apig_rest_service_redirect` parameter: When you set this to `true`, the apig_rest_service_redirect in the extended Administration setting in API Gateway REST requests against the /mediator directive will be redirected to the /gateway directive. This means that REST requests can be sent to /mediator and to /gateway.

Note:

- The Mediator configuration migration can only be applied to a fresh API Gateway installation once.
- On migrating from Mediator to API Gateway, API Gateway does not modify or change anything that is part of the incoming request. The incoming request along with the query parameters or headers is forwarded to the native service as it is without any modification. If you require API Gateway to remove any invalid query parameters, in API Gateway UI, add webMethods IS service under **Request transformation policy** > **Advanced Transformation**, configure any flow service and select **Comply to IS spec**.