Altova MapForce Server 2021 Advanced Edition

User & Reference Manual
# Table of Contents

1. **Introduction**  
2. **Installation**  
   2.1 System Requirements  
   2.2 Processor Cores and Licenses  
   2.3 Important Paths  
   2.4 Linux  
   2.4.1 Installing MapForce Server  
   2.4.2 Viewing Currently Installed Products  
   2.4.3 Uninstalling MapForce Server  
   2.5 macOS  
   2.5.1 Installing MapForce Server  
   2.5.2 Uninstalling MapForce Server  
   2.6 Windows  
   2.6.1 Installing MapForce Server  
   2.6.2 Uninstalling MapForce Server  
3. **Running Mappings**  
   3.1 Preparing Mappings for Server Execution  
   3.2 Global Resources  
   3.3 Join Optimization  
   3.4 Credentials  
   3.4.1 Example: OAuth 2.0 Authorization  
   3.5 Dynamic Authentication  
4. **MapForce Server Command Line**  
   4.1 assignlicense (Windows only)
7 XBRL Taxonomy Manager

7.1 Run XBRL Taxonomy Manager ................................................................. 145
7.2 Apply Patches .......................................................................................... 146
7.3 Install a Taxonomy .................................................................................... 148
7.4 View Installed Taxonomies .................................................................... 151
7.5 Uninstall a Taxonomy ............................................................................. 152
7.6 Command Line Interface ....................................................................... 153
  7.6.1 help ..................................................................................................... 154
  7.6.2 info ..................................................................................................... 155
  7.6.3 initialize ............................................................................................. 155
  7.6.4 install ................................................................................................. 156
  7.6.5 list ...................................................................................................... 157
  7.6.6 reset ................................................................................................... 158
  7.6.7 setdeflang ........................................................................................ 159
  7.6.8 uninstall ............................................................................................ 159
  7.6.9 update ............................................................................................... 160
  7.6.10 upgrade ........................................................................................... 161

8 Catalog Files ............................................................................................ 162

Index ............................................................................................................. 165
1 Introduction

MapForce Server is an enterprise software solution that runs data mapping transformations on Windows, Linux, and macOS operating systems. The data mappings themselves (or Mapping Design Files, *.mfd) are visually designed with Altova MapForce (https://www.altova.com/mapforce), where you define the inputs, outputs, and any intermediate processing steps that must be applied to your data. The role of MapForce Server is to run MapForce Server Execution (.mfx) files compiled with MapForce, and to produce the output files or data, or even update databases or call Web services, according to the design of the underlying mapping.

MapForce Server can run standalone as well as under the management of Altova FlowForce Server (https://www.altova.com/flowforceserver). When installed on the same machine as MapForce Server, FlowForce Server automates execution of mappings through scheduled or trigger-based jobs, which can also be exposed as Web services. In addition to this, FlowForce Server includes a built-in library of functions that enable you to take additional automated actions before or after mapping execution, such as sending email, copying files and directories, uploading files to FTP, running shell commands, and others.

Features

- Server-level performance when executing data mappings
- Cross-platform: MapForce Server runs on Windows, Linux, or macOS operating systems
- Command line interface
- An API that you can call from C++, C#, Java, VB.NET, VBScript, or VBA code
- Native integration with FlowForce Server
- Support for Altova Global Resources—a way of making file, folder, or database references configurable and portable across multiple environments and across multiple Altova applications, see Altova Global Resources
- Accelerates execution of mappings where join optimization is possible (see About Join Optimization)
- Runs mappings that apply functions and defaults to multiple items simultaneously. Such mappings make it possible, for example, to easily replace all encountered null values with empty strings or custom text
- Runs mappings that read data from and write data to Protocol Buffers binary format
- Runs mappings that perform bulk database inserts

Limitations

- XML digital signatures are not supported
- ADO, ADO.NET, and ODBC database connections are supported only on Windows. On Linux and macOS, native database connectivity is available for SQLite and PostgreSQL databases. For other databases running on Linux or macOS, JDBC should be used.

Last updated: 25 February 2021
2 Installation

2.1 System Requirements

- **Windows**
  Windows 7 SP1 with Platform Update, Windows 8, Windows 10

- **Windows Server**
  Windows Server 2008 R2 SP1 with Platform Update or newer

On Windows, MapForce Server is available both as a 32-bit and 64-bit package.

- **Linux**
  - CentOS 7 or newer
  - RedHat 7 or newer
  - Debian 8 or newer
  - Ubuntu 16.04 LTS or newer

The following libraries are required as a prerequisite to install and run the application. If the packages below are not already available on your Linux machine, run the command `yum` (or `apt-get` if applicable) to install them.

<table>
<thead>
<tr>
<th>Required by</th>
<th>CentOS, RedHat</th>
<th>Debian</th>
<th>Ubuntu</th>
</tr>
</thead>
<tbody>
<tr>
<td>LicenseServer</td>
<td>krb5-libs</td>
<td>libgssapi-krb5-2</td>
<td>libgssapi-krb5-2</td>
</tr>
<tr>
<td>MapForce Server Advanced Edition</td>
<td>libidn, krb5-libs</td>
<td>libidn11, libgssapi-krb5-2</td>
<td>libidn11, libgssapi-krb5-2</td>
</tr>
</tbody>
</table>

- **macOS**
  macOS 10.13 or newer
2.2 Processor Cores and Licenses

The licensing of Altova server products is based on the number of physical processor cores available on the product machine (as opposed to the number of logical cores). For example, a dual-core processor has two cores, a quad-core processor four cores, a hexa-core processor six cores, and so on. The number of cores licensed for a product must be greater than or equal to the number of cores available on that server machine, whether the server is a physical or virtual machine. For example, if a server has eight cores, you must purchase an 8-core license. You can also combine licenses to achieve the core count. So, two 4-core licenses can be used for an eight-core server instead of one 8-core license.

If you are using a computer server with a large number of CPU cores but only have a low volume to process, you may also create a virtual machine that is allocated a smaller number of cores, and purchase a license for that number. Such a deployment, of course, will have less processing speed than if all available cores of the computer were utilized.

**Note**: Each Altova server product license can be used for only one client machine at a time—the machine on which the Altova server product is installed—even if the license has unused licensing capacity. For example, if a 10-core license is used for a client machine that has 6 CPU cores, then the remaining 4 cores of licensing capacity cannot be used simultaneously for another client machine.

**Estimate of core requirements**

There are various external factors that influence the data volumes and processing times your server can handle (for example: the hardware, the current load on the CPU, and memory allocation of other applications running on the server). In order to measure performance as accurately as possible, test the applications in your environment with data volumes and in conditions that approximate as closely as possible to real business situations.
2.3 Important Paths

Installation directory

The application is installed in the following directory:

- **Linux** /opt/Altova/MapForceServer2021/bin
- **Mac** /usr/local/Altova/MapForceServer2021/bin
- **Windows** `<ProgramFilesFolder>\Altova\MapForceServer2021\bin`

Executable

After installation, the MapForce Server executable can be found at the following path:

- **Linux** /opt/Altova/MapForceServer2021/bin/mapforceserver
- **Mac** /usr/local/Altova/MapForceServer2021/bin/mapforceserver
- **Windows** `<ProgramFilesFolder>\Altova\MapForceServer2021\bin\MapForceServer.exe`

Application Configuration Data

The application's configuration data is stored in the following directory:

- **Linux** /var/opt/Altova/MapForceServer
- **Mac** /var/Altova/MapForceServer
- **Windows** C:\ProgramData\Altova\MapForceServer
2.4 Linux

2.4.1 Installing MapForce Server

System requirements

Linux

- CentOS 7 or newer
- RedHat 7 or newer
- Debian 8 or newer
- Ubuntu 16.04 LTS or newer

The following libraries are required as a prerequisite to install and run the application. If the packages below are not already available on your Linux machine, run the command `yum` (or `apt-get` if applicable) to install them.

<table>
<thead>
<tr>
<th>Required by</th>
<th>CentOS, RedHat</th>
<th>Debian</th>
<th>Ubuntu</th>
</tr>
</thead>
<tbody>
<tr>
<td>LicenseServer</td>
<td>krb5-libs</td>
<td>libgssapi-krb5-2</td>
<td>libgssapi-krb5-2</td>
</tr>
<tr>
<td>MapForce Server Advanced Edition</td>
<td>libidn, krb5-libs</td>
<td>libidn11, libgssapi-krb5-2</td>
<td>libidn11, libgssapi-krb5-2</td>
</tr>
</tbody>
</table>

Prerequisites

- Perform installation either as root user or as a user with sudo privileges.
- The previous version of MapForce Server must be uninstalled before a new one is installed.

Installing MapForce Server

1. Download the installation package from https://www.altova.com/download#server to a local directory.
2. Change to the directory where you downloaded the installation package, for example:

   ```
   cd /home/User/Downloads
   ```
3. Install the MapForce Server package.

   ```
   [Debian, Ubuntu] sudo dpkg --install mapforceserver-2021-debian.deb
   [CentOS, RedHat] sudo rpm -ivh mapforceserver-2021-1.x86_64.rpm
   ```

Note: You may need to adjust the name of the package above to match the current release or service pack version.

Licensing MapForce Server

1. Ensure that the latest version of Altova LicenseServer (https://www.altova.com/licenseserver) is installed and running either on the local machine or network.
2. Register MapForce Server with Altova LicenseServer. For this, you need to know either the host name or the IP address of the computer where LicenseServer runs. For example, if LicenseServer runs on 127.0.0.1, you can register it as follows:

```
sudo ./mapforceserver licenseserver 127.0.0.1
```

**Note:** The command above assumes you are calling the executable from the program installation directory, see also Important Paths.

3. Log on to the LicenseServer administration interface and assign the license to the machine where MapForce Server runs.

For more information about licensing, refer to Altova LicenseServer documentation (https://www.altova.com/documentation).

### 2.4.2 Viewing Currently Installed Products

To check if an Altova server product is installed, run the following command:

- [Debian, Ubuntu] `dpkg --list | grep Altova`
- [CentOS, RedHat] `rpm -qa | grep server`

### 2.4.3 Uninstalling MapForce Server

To uninstall MapForce Server, run the following command:

- [Debian, Ubuntu] `sudo dpkg --purge mapforceserveradv`
- [CentOS, RedHat] `sudo rpm -e mapforceserver`

Note that the uninstallation of Debian and Ubuntu packages should be done with "--purge" instead of "--remove".
2.5 macOS

2.5.1 Installing MapForce Server

System requirements

macOS 10.13 or newer

Prerequisites

- Perform installation as a user with administrative (root) privileges.

FlowForce Server Integration


If you install MapForce Server together with FlowForce Server, it is recommended that you install FlowForce Server first. Otherwise, see the "Post-installation tasks" below.

Installing MapForce Server

1. Download the disk image (.dmg) file from the Altova Download Center (https://www.altova.com/download) and click to open it. Once you do this, the installer appears as a new drive on your computer.
2. Double-click the package (.pkg) file, and go through the successive steps of the installation wizard. These are self-explanatory and include one step in which you have to agree to the license agreement before being able to proceed. When you complete the wizard, the MapForce Server icon becomes available in Applications. The package is installed in the directory /usr/local/Altova/MapForceServer2021/.
3. If you have not installed Altova License Server already, or if you want to upgrade to the latest version, double-click the corresponding package (.pkg) file and follow the on-screen instructions.

Licensing MapForce Server

1. Ensure that the latest version of Altova License Server (https://www.altova.com/licenseserver) is installed and running either on the local machine or network.
2. Register MapForce Server with Altova License Server. For this, you need to know either the host name or the IP address of the computer where License Server runs. For example, if License Server runs on 127.0.0.1, you can register it as follows:

```
sudo mapforceserver licenseserver 127.0.0.1
```
3. Log on to the LicenseServer administration interface and assign the license to the machine where MapForce Server runs.

For more information, refer to Altova LicenseServer documentation (https://www.altova.com/documentation).

Post-installation tasks
If you installed MapForce Server before FlowForce Server, then, after having installed both MapForce Server and FlowForce Server, run the following command:

```
cp /usr/local/Altova/MapForceServer2019/etc/*.tool /usr/local/Altova/FlowForceServer2019/tools
```

The command above copies the .tool file from /etc directory of MapForce Server to the FlowForce Server /tools directory. The .tool file is required by FlowForce Server; it contains the path to the MapForce Server executable. You do not need to run this command if you install FlowForce Server before installing MapForce Server.

2.5.2 Uninstalling MapForce Server

To uninstall MapForce Server:

1. In Applications, right-click the MapForce Server icon and select Move to Trash. Note that your MapForce Server installation directory and configuration data are not removed by this step.
2. Optionally, if you want to remove the MapForce Server installation directory as well, run the following command:

```
sudo rm -rf /usr/local/Altova/MapForceServer2021/
```

**Note:** The steps above do not remove the configuration data from /var/Altova/MapForceServer2021. Be aware that once the configuration data is removed, it cannot be recovered later.
2.6 Windows

2.6.1 Installing MapForce Server

System requirements
- Windows 7 SP1 with Platform Update, Windows 8, Windows 10
- Windows Server 2008 R2 SP1 with Platform Update or newer

Prerequisites
- Perform installation as a user with administrative privileges.
- From version 2021 onwards, a 32-bit version of MapForce Server cannot be installed over a 64-bit version, or a 64-bit version over a 32-bit version. You must either (i) remove the older version before installing the newer version, or (ii) upgrade to a newer version that is the same bit version as your older installation.

Installing MapForce Server

To install MapForce Server standalone:

1. Download the installation file from the Altova Download Center (https://www.altova.com/download) and run it.
2. Optionally, select the installation language from the box in the lower left area of the wizard, and click Next. You can change the language later from the command line.
3. Do one of the following:
   a. If you haven't installed Altova LicenseServer yet, leave the default settings as is. The wizard will install the latest version of Altova LicenseServer on the computer where you are running the wizard.
   b. If you haven't installed Altova LicenseServer yet and want to install Altova LicenseServer on another computer, clear the Install Altova LicenseServer on this machine check box, and then choose Register Later. In this case, you will need to install Altova LicenseServer and register MapForce Server separately.
   c. If Altova LicenseServer has already been installed on your computer but it has a lower version than the one indicated by the installation wizard, leave the default settings as is. In this case, the installation wizard will automatically upgrade your LicenseServer version with the one indicated on the dialog box. Note that the existing registration and licensing information will be preserved after the upgrade.
   d. If Altova LicenseServer has already been installed on your computer or network, and if it has the same version as the one indicated by the wizard, do the following:
      i. Clear the Install Altova LicenseServer on this machine check box.
      ii. Under Register this product with, choose the Altova LicenseServer instance on which you want to register MapForce Server, or choose Register later. Note that you can always select Register Later if you want to ignore the LicenseServer associations and carry on with the
installation of MapForce Server.

4. Click **Next**.
5. Follow the wizard instructions to complete the installation.

**Installation on Windows Server Core**

Windows Server Core is a minimal Windows installation that does not use a number of GUI features. You can install MapForce Server on a Windows Server Core machine as follows:

1. Download the MapForce Server executable from the Altova Download Center ([https://www.altova.com/download](https://www.altova.com/download)). Make sure to choose the executable matching your server platform (32-bit or 64-bit).
2. On a Windows computer with a graphical user interface, open a command prompt window as Administrator and run the following command on the installer executable:

   ```
   MapForceServerAdv<version>.exe /u
   ```

3. This unpacks the .msi file(s) which you need for the next step. Copy the .msi file to the computer that runs on Windows Server Core.
4. If you are updating an earlier version of MapForce Server, shut down MapForce Server before carrying out the next step.
5. Install MapForce Server via the .msi file by using the following command:

   ```
   msiexec /i MapForceServerAdvanced.msi
   ```

To test the return value of the installation, you might want to run a script similar to the following:

```
start /wait msiexec /i MapForceServerAdvanced.msi /q
echo %errorlevel%
```

The return code of the install operation will be available in the `%errorlevel%` environment variable. The return code **0** indicates success.

For a silent installation with a return code and a log of the installation process, run:

```
start /wait msiexec /i MapForceServerAdvanced.msi /q /L*v! <pathToInstallLogFile>
```

**Note:** Keep the .msi files in a safe place. You will need them later to uninstall, repair or modify the installation.

To modify the installation, run:

```
msiexec /m MapForceServerAdvanced.msi
```

To repair the installation, run:

```
msiexec /r MapForceServerAdvanced.msi
```
To uninstall MapForce Server, run:

```
msiexec /x MapForceServerAdvanced.msi
```

Alternatively, the following command will uninstall MapForce Server silently and report the detailed outcome in a log file:

```
start /wait msiexec /x MapForceServerAdvanced.msi /q /L*v! <pathToUninstallLogFile>
```

To install taxonomies, use the Taxonomy Package Manager via the command line. See the MapForce Server manual for information about how to do this.

**Licensing MapForce Server**

1. Ensure that the latest version of Altova LicenseServer ([https://www.altova.com/licenseserver](https://www.altova.com/licenseserver)) is installed and running either on the local machine or network.
2. If you haven't done so already during installation, register MapForce Server with Altova LicenseServer. For this, you need to know either the host name or the IP address of the computer where LicenseServer runs. For example, if LicenseServer runs on 127.0.0.1, you can register it as follows:

```
MapForceServer.exe licenseserver 127.0.0.1
```

3. Log on to the LicenseServer administration interface and assign the license to the machine where MapForce Server runs.

For more information, refer to Altova LicenseServer documentation ([https://www.altova.com/documentation](https://www.altova.com/documentation)).

**2.6.2 Uninstalling MapForce Server**

**Uninstalling MapForce Server**

1. Right-click the Windows "Start" button and select **Settings**.
2. Open the Control Panel (start typing "Control Panel" and click the suggested entry).
3. Under **Programs**, click **Uninstall a program**.
4. In Control Panel, select "Altova MapForce Server" and then click **Uninstall**.
Running Mappings

The role of MapForce Server is to execute data mappings created with Altova MapForce (https://www.altova.com/mapforce.html).

First, you design the data mappings (or Mapping Design Files, *.mfd) visually in MapForce, where you define the inputs, outputs, and any intermediate processing steps that must be applied to your data (including sorting, filtering, custom functions, and others). Once your mapping is ready, you can execute it with MapForce Server in one of the following ways:

- On the Windows machine where MapForce runs, compile the mapping to a MapForce Server Execution File (.mfx). The .mfx files are in fact data mappings packaged for execution in a server environment. You can copy such files to any of the supported operating systems where MapForce Server runs (including across different platforms, see System Requirements). On the server machine, you can execute the .mfx file using the command line interface of MapForce Server, or using the MapForce Server API.

- On the Windows machine where MapForce runs, deploy the mapping to a server machine where both MapForce Server and FlowForce Server are installed. The server machine can be a different operating system (see System Requirements). Mappings deployed in this way become FlowForce Server functions and you can create scheduled or trigger-based jobs from them. When mappings run as FlowForce Server jobs, they can also be exposed as Web services, chained as sub-steps of other jobs, or made part of workflows which include sending emails, verifying exit codes, running shell commands, and others.
For more information about this scenario, see the FlowForce Server documentation (https://www.altova.com/documentation).

How to execute mappings compiled as MapForce Server Execution files
1. Run MapForce Enterprise or Professional Edition.
2. Open the mapping to be compiled.
3. On the File menu, click Compile to MapForce Server Execution file, and select a destination directory.
4. Copy the .mfx file to the destination directory or server, along with any input files or dependencies. For further information, see Preparing Mappings for Server Execution.
5. Call the "run" command of the command line interface, or the equivalent method of the MapForce Server API.

How to execute mappings deployed to FlowForce Server
1. Open in MapForce Enterprise or Professional the mapping to be deployed.
2. Make sure that the transformation language (execution engine) of the mapping is set to Built-in. To change the execution engine to Built-in, select the menu command Output | Built-In Execution Engine, or click the Select Built-In Execution Engine toolbar button.
4. Enter the server connection details (host, port), the FlowForce credentials, and the destination FlowForce container. To proceed to creating the FlowForce job immediately in the browser, select the option Open web browser to create new job. You can also create the FlowForce job later (see next step).
5. Open a browser, log on to the FlowForce Server Web administration interface, and navigate to the container where you deployed the mapping (see previous step). This step is not required if you selected the option Open web browser to create new job in the previous step.
6. Define the FlowForce Server job, including its triggers, parameters, or additional execution steps (for examples, refer to the FlowForce Server documentation https://www.altova.com/documentation). Whenever the job is configured to run, the underlying mapping transformation will be executed, and the mapping output will be produced.

Note: If MapForce Server runs on a machine other than the one where the mapping was designed, make sure to adjust paths to input files or database connection details in such a way that they are meaningful in the new target execution environment. For example, if a mapping calls a database and requires a database driver, the driver must also be installed in the target environment in order for the mapping to
be executed successfully. To view or adjust the database connection details, right-click the database component in MapForce and select **Properties**. After making any changes to the mapping design in MapForce, remember to recompile it to a MapForce Server execution file (.mfx) or, depending on the case, redeploy it to FlowForce Server. For more information, see [Preparing Mappings for Server Execution](#).
3.1 Preparing Mappings for Server Execution

A mapping designed and previewed with MapForce may refer to resources which are outside of the current machine and operating system (such as databases). In addition to this, in MapForce, all mapping paths follow Windows-style conventions by default. Thirdly, the machine where MapForce Server runs might not support the same database connections as the machine where the mapping was designed. For this reason, running mappings in a server environment typically requires some preparation, especially if the target machine is not the same as the source machine.

**Note:** The term "source machine" refers to the computer where the MapForce is installed and the term "target machine" refers to the computer where MapForce Server or FlowForce Server is installed. In the most simple scenario, this is the same computer. In a more advanced scenario, MapForce runs on a Windows machine whereas MapForce Server or FlowForce Server runs on a Linux or macOS machine.

As best practice, always make sure that the mapping validates successfully in MapForce before deploying it to FlowForce Server or compiling it to a MapForce Server execution file.

If MapForce Server runs standalone (without FlowForce Server), the required licenses are as follows:

- On the source machine, MapForce Enterprise or Professional edition is required to design the mapping and compile it to a server execution file (.mfx).
- On the target machine, MapForce Server or MapForce Server Advanced Edition is required to run the mapping.

If MapForce Server runs under FlowForce Server management, the following requirements apply:

- On the source machine, MapForce Enterprise or Professional edition is required to design the mapping and deploy it to a target machine.
- Both MapForce Server and FlowForce Server must be licensed on the target machine. The role of MapForce Server is to run the mapping; the role of FlowForce is to make the mapping available as a job which benefits from features such as scheduled or on demand execution, execution as a Web service, error handling, conditional processing, email notifications, and others.
- FlowForce Server must be up and running at the configured network address and port. Namely, the “FlowForce Web Server” service must be started and configured to accept connections from HTTP clients (or HTTPS if configured) and must not be blocked by the firewall. The “FlowForce Server” service must also be started and running at the designated address and port.
- You have a FlowForce Server user account with permissions to one of the containers (by default, the /public container is accessible to any authenticated user).

**General considerations**

- If you intend to run the mapping on a target machine with standalone MapForce Server, all input files referenced by the mapping must be copied to the target machine as well. If MapForce Server runs under FlowForce Server management, there is no need to copy files manually. In this case, the instance and schema files are included in the package deployed to the target machine.
- If the mapping includes database components which require specific database drivers, such drivers must be installed on the target machine as well. For example, if your mapping reads data from a Microsoft Access database, then Microsoft Access or Microsoft Access Runtime (https://www.microsoft.com/en-us/download/details.aspx?id=50040) must be installed on the target machine as well.
When you deploy a mapping to non-Windows platforms, ADO, ADO.NET and ODBC database connections are automatically changed to JDBC. Native SQLite and native PostgreSQL connections are preserved as such and require no additional configuration. See also "Database connections" below.

If the mapping contains custom function calls (for example, to .dll or .class files), such dependencies are not deployed together with the mapping, since they are not known before runtime. In this case, copy them manually to the target machine. The path of the .dll or .class file on the server must be the same as in the "Manage Libraries" window in MapForce, for example:

Some mappings read multiple input files using a wildcard path. In this case, the input file names are not known before runtime and so they are not deployed. For the mapping to execute successfully, the input files must exist on the target machine.

If the mapping output path includes directories, those directories must exist on the target machine. Otherwise, an error will be generated when you execute the mapping. This behavior is unlike MapForce, where non-existing directories are generated automatically if the option Generate output to temporary files is enabled.

If the mapping calls a Web service that requires HTTPS authentication with a client certificate, the certificate must be transferred to the target machine as well, see Digital Certificate Management.

If the mapping connects to file-based databases such as Microsoft Access and SQLite, the database file must be manually transferred to the target machine or saved to a shared directory which is accessible to both the source and the target machine and referenced from there, see "File-based databases" below.

Making paths portable

If you intend to run the mapping on a server, ensure that the mapping follows the applicable path conventions and uses a supported database connection.

To make paths portable to non-Windows operating systems, use relative instead of absolute paths when designing the mapping in MapForce:

1. Open the desired mapping design file (.mfd) with MapForce on Windows.
2. On the File menu, select Mapping Settings, and clear the Make paths absolute in generated code check box if it is selected.
3. For each mapping component, open the Properties dialog box (by double-clicking the component's title bar, for example), and change all file paths from absolute to relative. Also, select the Save all file paths relative to MFD file check box. For convenience, you can copy all input files and schemas in the same folder as the mapping itself, and reference them just by the file name.
For more information about dealing with relative and absolute paths while designing mappings, refer to MapForce documentation.

Importantly, both MapForce Server and FlowForce Server support a so-called “working directory” against which all relative paths will be resolved. The working directory is specified at mapping runtime, as follows:

- In FlowForce Server, by editing the “Working-directory” parameter of any job.
- In MapForce Server API, through the WorkingDirectory property of the COM and .NET API, or through the setWorkingDirectory method of the Java API.
- In MapForce Server command line, the working directory is the current directory of the command shell.

Database connections

Be aware that ADO, ADO.NET, and ODBC connections are not supported on Linux and macOS machines. Therefore, if the target machine is Linux or macOS, such connections are converted to JDBC when you deploy the mapping to FlowForce or when you compile the mapping to a MapForce Server execution file. In this case, you have the following options before deploying the mapping or compiling it to a server execution file:

- In MapForce, create a JDBC connection to the database
- In MapForce, fill the JDBC database connection details in the "JDBC-specific Settings" section of the database component.

If the mapping uses a native connection to a PostgreSQL or SQLite database, the native connection is preserved and no JDBC conversion takes place. If the mapping connects to a file-based database, such as Microsoft Access and SQLite, additional configuration is required, see "File-based databases" below.

Running mappings with JDBC connections requires that the Java Runtime Environment or Java Development Kit be installed on the server machine. This may be either Oracle JDK or an open source build such as Oracle OpenJDK.

- The JAVA_HOME environment variable must point to the JDK installation directory.
- On Windows, a Java Virtual Machine path found in the Windows registry will take priority over the JAVA_HOME variable.
- The JDK platform (64-bit, 32-bit) must be the same as that of MapForce Server. Otherwise, you may get an error with the reason: "JVM is inaccessible".

To set up a JDBC connection on Linux or macOS:

1. Download the JDBC driver supplied by the database vendor and install it on the operating system. Make sure to select the 32-bit version if your operating system runs on 32-bit, and the 64-bit version if your operating system runs on 64-bit.
2. Set the environment variables to the location where the JDBC driver is installed. Typically, you will need to set the CLASSPATH variable, and possibly a few others. To find out which specific environment variables must be configured, check the documentation supplied with the JDBC driver.

Note: On macOS, the system expects any installed JDBC libraries to be in the /Library/Java/Extensions directory. Therefore, it is recommended that you unpack the JDBC driver to this location; otherwise, you will need to configure the system to look for the JDBC library at the path where you installed the JDBC driver.
Oracle Instant Client connections on macOS

These instructions are applicable if you connect to an Oracle database through the **Oracle Database Instant Client**, on macOS. Prerequisites:

- Java 8.0 or later must be installed. If the Mac machine runs a Java version prior to Java 8, you can also connect through the **JDBC Thin for All Platforms** library, and disregard the instructions below.
- Oracle Instant Client must be installed. You can download the Oracle Instant Client from the Oracle official download page. Note that there are several Instant Client packages available on the Oracle download page. Make sure to select a package with Oracle Call Interface (OCI) support, (for example, Instant Client Basic). Also, make sure to select the 32-bit version if your operating system runs on 32-bit, and the 64-bit version if your operating system runs on 64-bit.

Once you have downloaded and unpacked the Oracle Instant Client, edit the property list (.plist) file shipped with the installer so that the following environment variables point to the location of the corresponding driver paths, for example:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASSPATH</td>
<td>/opt/oracle/instantclient_11_2/ojdbc6.jar:/opt/oracle/instantclient_11_2/ojdbc5.jar</td>
</tr>
<tr>
<td>TNS_ADMIN</td>
<td>/opt/oracle/NETWORK_ADMIN</td>
</tr>
<tr>
<td>ORACLE_HOME</td>
<td>/opt/oracle/instantclient_11_2</td>
</tr>
<tr>
<td>DYLD_LIBRARY_PATH</td>
<td>/opt/oracle/instantclient_11_2</td>
</tr>
<tr>
<td>PATH</td>
<td>$PATH:/opt/oracle/instantclient_11_2</td>
</tr>
</tbody>
</table>

**Note:** Edit the sample values above to fit the paths where Oracle Instant Client files are installed on your operating system.

File-based databases

File-based databases such as Microsoft Access and SQLite are not included in the package deployed to FlowForce Server or in the compiled MapForce Server execution file. Therefore, if the source and target machine are not the same, take the following steps:

1. In MapForce, right-click the mapping and clear the check box **Make paths absolute in generated code**.
2. Right-click the database component on the mapping and add a connection to the database file using a relative path. A simple way to avoid path-related issues is to save the mapping design (.mfd file) in the same directory as the database file and to refer to the latter from the mapping just by file name (thus using a relative path).
3. Copy the database file to a directory on the target machine (let's call it "working directory"). Keep this directory in mind since it will be required to run the mapping on the server, as shown below.

To run such mappings on the server, do one of the following:

- If the mapping will be run by MapForce Server under FlowForce Server control, configure the FlowForce Server job to point to the working directory created previously. The database file must reside in the

- If the mapping will be run by standalone MapForce Server at the command line, change the current directory to the working directory (for example, `cd path\to\working\directory`) before calling the run command of MapForce Server.
- If the mapping will be run by the MapForce Server API, set the working directory programmatically before running the mapping. To facilitate this, the property `WorkingDirectory` is available for the MapForce Server object in the COM and .NET API. In the Java API, the method `setWorkingDirectory` is available.

If both the source and the target machines are Windows machines running on the local network, an alternative approach is to configure the mapping to read the database file from a common shared directory, as follows:

1. Store the database file in a common shared directory which is accessible by both the source and the target machine.
2. Right-click the database component on the mapping and add a connection to the database file using an absolute path.

Global Resources

If a mapping includes references to Global Resources instead of direct paths or database connections, you will be able to use Global Resources on the server side as well. When you compile a mapping to a MapForce Server execution file (.mfx), the references to Global Resources will be kept intact, so that you can provide these on the server side, at mapping runtime. When deploying a mapping to FlowForce Server, you can optionally choose whether it should use resources on the server.

For mappings (or mapping functions, in case of FlowForce Server) to run successfully, the actual file, folder, or database connection details that you supply as Global Resources must be compatible with the server environment. For example, files and folders paths must use the Linux convention for paths if the mapping will run on a Linux server. Likewise, Global Resources defined as database connections must be possible on the server machine.


XBRL Taxonomy Packages

When you deploy a mapping that references XBRL Taxonomy Packages to FlowForce Server, MapForce collects all external references from the mapping and then resolves them using the current configuration and currently installed taxonomy packages. If there are resolved external references that point to a taxonomy package, then the taxonomy package is deployed together with the mapping. FlowForce Server will use that package—as it was during deployment—to execute the mapping. To refresh the taxonomy package used by FlowForce Server, you will need to change it in MapForce and redeploy the mapping.

Note that the root catalog of MapForce Server influences the way taxonomies are resolved on the target machine. The root catalog is found at the following path relative to the MapForce Server installation directory: `etc/RootCatalog.xml`.

Taxonomy packages that were deployed with a mapping will be used if the root catalog of MapForce Server does not already contain such package or does not contain a package that is defined for the same URL prefix. The root catalog of MapForce Server has priority over the deployed taxonomy.

If MapForce Server runs standalone (without FlowForce Server), it is possible to specify the root catalog that
Preparing Mappings for Server Execution

should be used by the mapping as follows:

- At the command line, this is possible by adding the option `-catalog` to the `run` command.
- In the MapForce Server API, call the method `SetOption`, and supply the string `"catalog"` as first argument, and the path to the root catalog as second argument.

If a mapping uses XBRL components with table linkbases, the taxonomy package or the taxonomy package configuration file must be supplied to the mapping at runtime, as follows:

- At the MapForce Server command line, add the option `--taxonomy-package` or `--taxonomy-packages-config-file` to the `run` command.
- In the MapForce Server API, call the method `SetOption`. The first argument must be either `"taxonomy-package"` or `"taxonomy-packages-config-file"`. The second argument must be the actual path to the taxonomy package (or taxonomy package configuration) file.
3.2 Global Resources

Altova Global Resources are portable references to files, folders, or databases. When stored as Global Resources, paths and database connection details become reusable and available across multiple Altova applications. For example, if you frequently need to open the same file in multiple Altova desktop applications, you may find it convenient to define it as a Global Resource. This way, you don't even need to remember the file path because you can select the respective Global Resource from the "Open File" dialog box instead. This also has the advantage that, if the file path ever changes, you will change it in one place only.

A typical usage of Global Resources is to define a database connection once and reuse it across all Altova applications that support Global Resources. For example, you can create a database connection on the machine where a MapForce mapping was designed and then reuse the same connection on the machine where MapForce Server runs the mapping (this may require, in some cases, that both machines have the same database client software installed).

Optionally, you can create multiple variations of the same Global Resource (known as "configurations"). This lets you easily switch file or folder paths (or even databases) depending on your needs. For example, you could create a "database" resource with two configurations: "development" and "production". Taking MapForce Server as example, you could then easily retrieve data from either the development or production database, by indicating the desired configuration as command line parameter at mapping runtime.

You can create Global Resources from the following Altova desktop applications: Altova Authentic, DatabaseSpy, MobileTogether Designer, MapForce, StyleVision, and XMLSpy. On the server side, Global Resources can be consumed by the following Altova server applications: FlowForce Server, MapForce Server, RaptorXML Server, RaptorXML+XBRL Server.

For more information about creating Global Resources, refer to the "Altova Global Resources" chapter of MapForce documentation.

Resources in MapForce Server

When you compile a mapping to a MapForce Server execution file (.mfx), any Global Resource references used by the mapping are preserved, not resolved. This means that you will need to provide these references on the server side in order to run the mapping successfully. More specifically, in MapForce Server, the following is required to run an .mfx file which uses Global Resources:

1. The Global Resource XML file. On the machine where MapForce is installed, the file is called GlobalResources.xml, and you can find it in the Documents\Altova folder. You can copy this file to the machine where MapForce Server runs, and you can also create multiple such files if necessary.

2. The Global Resource configuration name. Each Global Resource has a default configuration called "Default", but you can create additional configurations if so required. If you created additional configurations, then you must specify the desired configuration name instead of "Default".

In MapForce, the Global Resource file path and the Global Resource configuration name are set or changed from the graphical user interface. In MapForce Server, these are specified at mapping runtime.

- If you run the mapping through the command line interface, set the options --globalresourceconfig and --globalresourcefile after the run command, for example:

```bash
c:\Program Files (x86)\Altova\MapForceServer2021\bin\MapForceServer.exe run --globalresourceconfig --globalresourcefile
```
If you run the mapping through the MapForce Server API, call the method `SetOptions` two times before calling the `Run` method. The first call is required to supply the Global Resource XML file path as option, and the second one is required to supply the Global Resource configuration name. For further information, see the MapForce Server API.
3.3 Join Optimization

Join optimization accelerates execution of data mappings in which large sets of data are being filtered or joined.

Join optimization works by eliminating nested loops that occur internally as a mapping is being executed. A nested loop occurs when the mapping iterates each item of a set as many times as there are items in a second set. Note that it is normal for the mapping execution engine* to perform loops (iterations) over various sequences of items, by virtue of its design. When nested independent loops occur (that is, loops which iterate over other loops), the mapping can benefit from join optimization, which would significantly reduce the time required to execute the mapping. Nested loops are hardly noticeable when running mappings where the input data is not significantly large; however, this can become a challenge in case of mappings that process files or databases that consist of a very large number of records.

* The execution engine of a mapping can be MapForce, MapForce Server, or a C#, C++, or Java program generated by MapForce. Join optimization is available exclusively in the MapForce Server Advanced Edition.

To designate MapForce Server as target execution engine, click the BUILT-IN toolbar button in MapForce. This will also ensure your mapping benefits from most available features. If you select another transformation language, certain MapForce features might not be supported in that language.

As mentioned above, the primary concern of join optimization is to address nested loops in an efficient way. Let's now have a closer look at how nested loops occur in first place.

The typical case when nested loops occur is when the mapping contains at least one Join component, and SQL JOIN mode** is not possible.

** When certain conditions are met in MapForce, mappings could allow for a special execution mode called "SQL Join mode". SQL Join mode is possible only if the mapping reads data from a database. When data is joined this way, the join operation is undertaken by the database (that is, an SQL JOIN takes place), and this eliminates the need for nested loops in the mapping execution engine. For more information about SQL Join mode, refer to the MapForce documentation (https://www.altova.com/documentation.html).

For example, the image below illustrates a mapping (designed with Altova MapForce) which combines data from two XML files using a Join component. On the computer where MapForce is installed, this mapping is available at the following path: ..

Documents\Altova\MapForce2021\MapForceExamples\Tutorial\JoinPeopleInfo.mfd. Some people data is available only in the first XML file (Email, Phone), while some other data is available only in the second XML file (City, Street, Number). The goal of the mapping is to write to the target XML file the merged data of all people where FirstName and LastName correspond in both source structures.
In MapForce, a Join component pairs items in two sets according to some custom condition, which implies comparing each item in set 1 with each item in set 2. The total number of comparisons represents the cross-join (Cartesian product) of both sets. For example, if the first set contains 50 items, and if the second set contains 100 items, then a total of 5000 (50 x 100) comparisons will occur. In the mapping above, the sets that are being compared correspond to all instance items of the two XML structures connected to the Join component.

Note: Join optimization (a feature of MapForce Server Advanced Edition) should not be confused with Join components (a feature of MapForce). For more information about Join components, refer to the MapForce documentation (https://www.altova.com/documentation.html).

As expected, from a performance perspective, mappings that contain nested loops would need more time to run. Imagine a situation where both joined sets contain millions of records. This can easily affect performance, and this is where join optimization is useful. In very broad lines, join optimization behaves like a database engine that is optimized to look up (index) extremely large sets of data. Except that, as illustrated by the mapping above, join optimization deals not only with data originating from databases. Join optimization eliminates nested loops regardless of the data kind, by building, where possible, internal lookup tables which are queried at mapping runtime. This significantly improves the mapping performance and ultimately reduces the time required to execute the mapping.

Note: When join optimization occurs, running the mapping will take less time but typically require more memory as well. Be aware that memory usage patterns depend on various complex factors; therefore, observed behaviour may differ depending on the case.

Join optimization can accelerate not only mappings with joins, but also those which use filter components. In MapForce, a filter processes a sequence of items (that is, it checks a given Boolean condition for each instance of the item connected to the node/row input). If the Boolean condition is connected to a function
which, in its turn, must iterate over another sequence of items, and if the mapping context demands it, then a situation similar to a join happens. If the filter must perform a cross-comparison of each item in two sets, then it qualifies for join optimization.

In order for the mapping to benefit from join optimization, it must be run by MapForce Server Advanced Edition. To execute a mapping with MapForce Server Advanced Edition, open it in MapForce, and compile it to a mapping execution (.mfx) file using the menu command File | Compile to MapForce Server Execution File. Then run the .mfx file by using an API method in your language of choice, or the run command of the command line interface (see also How It Works).
3.4 Credentials

Credential objects provide a way to make authentication data (such as usernames, passwords, and OAuth authentication details) portable across various mapping execution environments, in a secure way. Credentials are useful in mappings that require basic HTTP authentication or OAuth 2.0 authorization. You can define credentials in MapForce and also in FlowForce Server. If credentials were defined in MapForce, you can optionally deploy them to FlowForce Server, similar to how mappings are deployed.

After you compile the mapping to a MapForce Server execution file (.mfx), MapForce Server will run the .mfx file depending on your choices at mapping design time.

If you selected the Include in MapForce Server Execution File and Mapping Deployment check box when creating the credential in MapForce, MapForce Server will use at mapping runtime any credentials that were stored in the .mfx file. This means that you can run the mapping with a command such as:

<exec> run mapping.mfx

Where <exec> is the path to the MapForce Server executable. This path can be either absolute or, if the current directory is the same as the executable, you can enter just the executable name.

If you entered only the credential name (without username and password) in MapForce, then you must explicitly provide these details at mapping runtime, with the help of the --credential command line option available for the run command. This way, you can use, for example, a different set of credentials in production, as opposed to those used when you designed the mapping. The --credential option has the form --credential=KEY:VALUE. where

- KEY is the name of the credential as it was defined in MapForce.
- VALUE is a credential property, or a list of properties separated by ampersand (&). For credentials of type "password", the possible properties are username and password. For credentials of type OAuth 2.0, the only supported property is oauth:token.
- The actual property values are supplied just like query parameters in a URL, using the "=" sign.

For example:

<exec> run mapping.mfx --credential="mycredential:username=admin&password=4xJ38dnx7"

In the code listing above, the value of the --credential option was enclosed within quotes in order to treat the value literally, since the username and password are separated by an ampersand character.

If your mapping needs multiple sets of named credentials to run, you can specify the --credential option multiple times.

The credentials supplied as command line options take precedence over stored credentials.

If you did not select the Include in MapForce Server Execution File and Mapping Deployment check box, the sensitive fields are missing. This means that you must supply the password at the command line while still referring the credential by its name, for example:

<exec> run mapping.mfx --credential="mycredential:username administer"
The following fields are considered sensitive data:

- **Password** (for credentials of type "Password")
- **Client Secret**, **Access Token**, and **Refresh Token** (for credentials of type "OAuth 2.0")

For mappings that require OAuth 2.0 authorization, the MapForce Server command line accepts an OAuth 2.0 access token as input at the mapping runtime. Note that the MapForce Server command line does not provide an interactive GUI by design, so you will need to obtain the OAuth 2.0 access token by external means (for example, by requesting it with MapForce) when using the command line specifically. This is, however, not necessary if MapForce Server runs under FlowForce Server management, since the latter is capable of acquiring a new OAuth 2.0 access token at runtime by itself.

At the command line, running the mapping with stored credentials is possible as long as the stored OAuth 2.0 token has not expired or has not been revoked by the Web service provider. To address this, supply a new OAuth 2.0 access token (obtained by some external means) by using the `--credential` option, for example:

```bash
<exec> run mapping.mfx --credential=my_oauth_credential:oauth:token=jdsaflkajlkewsaiurthczv904215-jhd
```

Where:

- `my_oauth_credential` is the name of the OAuth 2.0 credential created from MapForce.
- `oauth:token` is the way to indicate to MapForce Server that a new OAuth 2.0 access token is being supplied at runtime.

**MapForce Server API**

The MapForce Server API provides methods to create credentials, add properties to credentials, and close credentials after you finished declaring them. The following code listing illustrates the typical way of declaring password credentials in a C# program that runs a mapping:

```csharp
//Create a MapForce Server object
// Set the credential name as it was defined in MapForce
objMFS.BeginCredential("mycredential");
// Add the credential properties
objMFS.AddCredentialProperty("username", "altova");
objMFS.AddCredentialProperty("password", "b45ax78!");
// Close the credential
objMFS.EndCredential();
```

To perform OAuth 2.0 authorizations from a program that runs a mapping, the credential property name must be set to `oauth:token`, as illustrated below:

```csharp
//Create a MapForce Server object
```
If the mapping needs multiple credential sets, use the methods above to add as many sets of credentials as required. Once you have declared all the required credentials, you can run the mapping execution file in a standard way, by calling the Run() method. For more information, see the API Reference.

### 3.4.1 Example: OAuth 2.0 Authorization

This example shows you how to call a REST-style Web service that requires OAuth 2.0 Authorization. The client application is a MapForce Server execution file (.mfx) that will retrieve calendar events using the Google Calendar API (https://developers.google.com/calendar/). To keep things simple, the .mfx file will retrieve the calendar information "as is" and will just output the raw JSON result without any other processing.

Prerequisites:

- MapForce Enterprise Edition
- MapForce Server Advanced Edition
- To follow this example step-by-step, you must have a Google account. If you would like to call another Web service, obtain OAuth 2.0 credentials from your Web service provider and use them in the instructions below instead.

**Obtain the OAuth 2.0 credentials**

If you already have the OAuth 2.0 credentials required to access the Web service, you can skip this step. Otherwise, the exact instructions to obtain them depend on the provider of the Web service that your mapping will call. To call the Google Calendar API like in this example, follow these steps:

1. Login to the Google API Console (https://console.developers.google.com/).
2. Create a new project.
3. Click **OAuth consent screen**.

4. Select **External** as user type, unless you have a G Suite account which would enable you to grant API access only to users in your organization.
5. Enter "mapforce-demo" as application name and save the settings.

6. Click **Create credentials** and then select **OAuth Client ID**.

7. Enter **Desktop app** as application type and "MapForce Client" as the client name.
8. Click **Create**. The client ID is created and becomes available in the **Credentials** page.

9. Click **Download** to download the OAuth 2.0 authorization details as a JSON file.
You have now obtained the OAuth 2.0 authorization details from Google Console API, namely:

1. Authorization Endpoint
2. Token Endpoint
3. Client ID
4. Client Secret

Enable the Google Calendar API

To accept calls from clients, the Google Calendar API used in this example must be enabled. In the Google API Console, click Library, search for the Google Calendar API and enable it:

In this example, we are going to call the list method of the Events entity. You can find detailed reference to this API method at https://developers.google.com/calendar/v3/reference/events/list. For now, note the following important points:

1. As pointed out in documentation, the method must be called by sending a GET request to https://www.googleapis.com/calendar/v3/calendars/calendarId/events, where calendarId is the identifier of a Google Calendar. The calendarId request parameter will be configured from MapForce in a subsequent step.
2. Calling the API method requires at least one of the following scopes:
   - https://www.googleapis.com/auth/calendar.readonly
   - https://www.googleapis.com/auth/calendar
   - https://www.googleapis.com/auth/calendar.events.readonly
   - https://www.googleapis.com/auth/calendar.events

   During the OAuth 2 authorization process, your mapping will have to provide one of the scopes above—
this will also be configured in a subsequent step. For the purpose of this example, the first "read-only" scope is sufficient.

**Request an authorization token**

In order to preview the mapping in MapForce, you will need to add the OAuth 2.0 authorization details to the mapping and request an authorization token, as illustrated below.

1. In MapForce, right-click an empty area on the mapping, and select **Open Credentials Manager** from the context menu.
2. Click **Add Credential**.
3. Enter a name ("my.oauth", in this example), and select **OAuth 2** as type.
4. Fill in the **Authorization Endpoint**, **Token Endpoint**, **Client ID**, **Client Secret** text boxes with the corresponding values from the JSON file downloaded previously.
5. Enter https://www.googleapis.com/auth/calendar.readonly in the **Scope** text box.
6. Leave all other settings as is.
7. Click **Request Access Token** to obtain the token from the authorization server (in this example, Google). A browser window opens asking you to connect to your Google account.
8. Login to your Google account. Since you haven't submitted any app verification requests to Google yet, the following page appears.
9. Click **Advanced**, and then click **Go to mapforce-demo (unsafe)**.
10. Click **Allow**. A confirmation is now displayed in the browser.

**OAuth 2.0 authorization code retrieved.**

Return back to Altova MapForce.

MapForce also notifies you that the OAuth 2.0 authorization code has been retrieved successfully.
11. Click OK. Notice that the **Access Token** and **Refresh Token** fields have now been populated with data.

![MapForce successfully received new OAuth 2.0 credentials.]

12. Save the mapping as **GetCalendarEvents.mfd**.

In this tutorial, the **Save encrypted in MFD file** check box is selected on the Edit Credentials dialog box. Therefore, the sensitive fields **Client Secret**, **Authorization Token**, and **Refresh Token** will be saved in encrypted form in the mapping design file (.mfd) when you save the mapping.

Be aware that the authorization token will eventually expire after a period. When that happens, you will no longer be able to run the mapping (at this stage, no mapping has been designed, but this will happen in a subsequent step). Whenever you need to obtain a new authorization code manually, click **Request Access Token**, and follows the steps described above.

**Design the Web service call**

The mapping **GetCalendarEvents.mfd** created so far does not do anything yet. The only thing it contains are OAuth 2.0 credentials that enable access to the Google Calendar API.

Let's now design the Web service call in MapForce, as follows:

1. Open the **GetCalendarEvents.mfd** mapping.
2. On the **Insert** menu, click **Web Service Function**. The “Web Service Call Settings” dialog box appears.
3. Click **Manual**.
4. Select **GET** as request method and enter the URL to the Web service mentioned in a previous step:
5. Because `calendarId` is a placeholder that must be provided as a parameter, enclose it within curly braces as shown below.

![Web Service Call Settings](https://www.googleapis.com/calendar/v3/calendars/calendarId/events)

6. Click the **Add Parameter** button and define the parameter details as follows:

   ![Parameters](https://www.googleapis.com/calendar/v3/calendars/calendarId/events)

   In the configuration above, the "Template" style makes it possible to replace the URL part enclosed within curly braces with the parameter value at runtime. "Mappable" means that you can supply the value from the mapping (for example, from a constant, or perhaps from an input parameter). Finally, the parameter has been marked as "Required" because the API call cannot take place without it.

7. Click the **Edit** button adjacent to **HTTP Security Settings**.
8. On the "HTTP Security Settings" dialog box, select **Use Credential** and choose the "my.oauth" credential record configured previously.
The Web service configured so far has the following appearance on the mapping:

You can now complete the design by taking the following steps:
1. On the **Insert** menu, click **Insert Input**, and configure the component as follows:

As illustrated above, the input component has the design-time value "primary". According to the API's documentation, the value "primary" instructs the API server to access the primary Google calendar of the currently logged in user. Note that this value is a design-time value and is applicable only when you preview the mapping in MapForce. When the mapping runs in a server environment, you will need to provide the desired value at runtime.

2. Drag the **decode-mime-entity** function from the Libraries window into the mapping area. This function converts the raw MIME body received from the server into a string.

3. On the **Insert** menu, click **Insert Output**, and add a simple output component whose role is to output the result as a plain string.

4. Make the connections between components as illustrated below.

This concludes the design part in MapForce.

**Test the mapping execution**

To test the mapping execution in MapForce, click the **Output** tab and observe the result displayed in the Messages window.

If you get an authorization error such as "Unauthorized (401)“, note the following troubleshooting tips:

1. Make sure that the Google Calendar API is enabled, see **Enabling the Google Calendar API**.
2. **Request a new authorization token**, in the event that the access token obtained previously has already expired.
3. Double-check that all OAuth 2.0 details were entered correctly in MapForce.

On successful execution and OAuth 2.0 authorization from MapForce, the mapping output is expected to look
similar to the one below:

```json
{
  "kind": "calendar#events",
  "etag": "\"p32gbjmnvo63ek9q\"
  "summary": "2019-06-16T4:10:43.876Z",
  "timeZone": "Europe/Vienna",
  "accessRole": "owner",
  "defaultReminders": [ ]
  "method": "email",
  "minutes": 10
},

  "method": "popup",
  "minutes": 30
},

  "nextSyncToken": "CKC5tt_BhuoeCEKC5tt_BhuocGAU=",
  "items": []
}
```

If you used a Google account that does not have any calendar events like in this example, the "items" array is empty in the response. However, if you add an event to your Google Calendar and run the mapping again, the output will reflect that. As a side note, you could also retrieve events from a calendar other than the default one. For example, you could retrieve data from a public calendar like "Holidays in United States". To do this, set the value of `calendarId` parameter to `en.usa#holiday@group.v.calendar.google.com` instead of `primary`.

For information about other parameters that you can add to the API call, refer to the API method's documentation at https://developers.google.com/calendar/v3/reference/events/list.

Run the mapping with MapForce Server (standalone)

This section specifically deals with running the demo OAuth 2.0 mapping with MapForce Server installed as a standalone product, not under FlowForce Server management. For information about running such mappings with MapForce Server under FlowForce Server management, refer to the FlowForce Server documentation, where this example is continued.

To run an OAuth 2.0 mapping with MapForce Server standalone, there are two ways to deal with OAuth 2.0 credentials:

- Include the OAuth 2.0 token (in encrypted form) in the compiled .mfx file. With this approach, you will not need to supply any OAuth 2.0 credentials at the command line (or in the MapForce Server API call) because the embedded credential will be used. However, this also means that anyone with access to the .mfx file will be able to run it without providing the authorization token—until it expires or the authorization server revokes it. Importantly, you can always override the authorization token from the command line without having to recompile the .mfx file (see the next bullet).
Do not include the OAuth 2.0 token in the compiled .mfx file. With this approach, you (or another user who runs the .mfx file) will need to supply the OAuth 2.0 authorization token at the command line or in the MapForce Server API call. The authorization token itself must be obtained outside of MapForce Server, for example with MapForce, as already described previously.

In this example, the authorization token will not be included in the compiled .mfx file. Instead, it will be provided at runtime.

1. In MapForce, right-click an empty area on the mapping and select Open Credentials Manager.
2. Double-click the credential record ("my.oauth", in this example) and clear the Include in MapForce Server Execution File and Mapping Deployment check box.
3. Save the mapping design file (.mfd).

Let's now compile the mapping to a MapForce Server Execution file (.mfx):

2. Select a destination directory and save the file as GetCalendarEvents.mfx.

You can now open a Command Prompt window and run the .mfx file with a command like:

```
mapforceserver-exec run GetCalendarEvents.mfx --p=calendarId:"primary" --
credential=my.oauth:oauth:token=mytoken
```

Where:

- **mapforceserver-exec** is the path to the MapForce Server executable, typically C:\Program Files\Altova\MapForceServer2021\bin\MapForceServer.exe.
- **GetCalendarEvents.mfx** is the path to the .mfx file relative to the current directory of the command line. Adjust the path if applicable, or use an absolute path.
- **calendarId** is the name of the input parameter as it was created in MapForce.
- **my.oauth** is the name of the credential as it was created in MapForce in a previous step.
- **mytoken** is the value of the authorization token obtained externally (in this case, with MapForce).

On successful execution and OAuth 2.0 authorization, the command line output displays the response returned by the Google Calendar API, for example:
Be aware that the authorization token expires very quickly (the interval depends on the authorization server, which is Google in this case) and you may need to request a new one if you get "Unauthorized" errors, see Request an authorization token.
3.5 Dynamic Authentication

In MapForce, it is possible to configure mappings that call Web services for basic HTTP authentication. Dynamic authentication is one of the ways to achieve this; it is an alternative to using credentials. Dynamic authentication means designing the mapping so that it accepts the username and password as input parameters. For details about configuring dynamic authentication, refer to MapForce documentation (https://www.altova.com/documentation).

If you configured the mapping for dynamic authentication, then the respective username and password must be supplied as parameters at mapping runtime. This is not different from supplying any other parameter kinds to the mapping. For example, when calling MapForce Server at the command line, the syntax for a mapping like the one above is:

```
<exec> run mapping.mfx --p=username:admin --p=password:dj9JaVax
```

Where:

- `<exec>` is the path to the MapForce Server executable. This path can be either absolute or, if the current directory is the same as the executable, you can enter just the executable name.
- `username` and `password` are the names of the respective input parameters on the MapForce mapping.

When calling the MapForce Server API, you can authenticate the mapping by calling the `AddParameter` method before calling the `Run` method. For example, in C#, the code to achieve this could look as follows:

```csharp
try
{
    mfs.AddParameter("username", "admin");
    mfs.AddParameter("password", "dj9JaVax");
    mfs.WorkingDirectory = "C:\Work";
    if (mfs.Run("C:\Work\mapping.mfx"))
    {
        Console.WriteLine("Success");
    }
    else
    {
        Console.WriteLine(mfs.LastExecutionMessage);
    }
    Console.ReadLine();
}
catch(Exception ex)
{
    Console.WriteLine(ex);
}
```

Again, the `username` and `password` (first argument to the `AddParameter` method) must be the same as the names of the respective input parameters on the MapForce mapping.
4 MapForce Server Command Line

MapForce Server provides a command line interface that you can use for administrative tasks such as licensing, and also to run mapping execution files (.mfx). The available commands are listed below.

- **assignlicense**: (Windows only) Uploads a license to LicenseServer and assigns this license to MapForce Server.
- **exportresourcestrings**: Exports all application resource strings to an XML file.
- **help**: Displays information about the command that is submitted as argument (or about all commands if no argument is submitted).
- **licenseserver**: Registers MapForce Server with a LicenseServer on the local network.
- **run**: Runs a Mapping Execution File (.mfx) compiled with MapForce.
- **setdeflang**: Sets the default language of MapForce Server.
- **verifylicense**: (Windows only) Checks if current MapForce Server is licensed and, optionally, whether it is licensed with the given license key.
- **version**: Displays the version number of MapForce Server.

To call MapForce Server at the command line, you need to know the path of the executable as applicable to your operating system. By default, the MapForce Server executable is installed at the following path:

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td><code>/opt/Altova/MapForceServer2021/bin/mapforceserver</code></td>
</tr>
<tr>
<td>macOS</td>
<td><code>/usr/local/Altova/MapForceServer2021/bin/mapforceserver</code></td>
</tr>
<tr>
<td>Windows</td>
<td><code>C:\Program Files\Altova\MapForceServer2021\bin\MapForceServer.exe</code></td>
</tr>
</tbody>
</table>

**Note:** If MapForce Server 32-bit is installed on Windows 64-bit, change `C:\Program Files` to `C:\Program Files (x86)`.

By convention, this documentation omits the full path of the executable when describing a given command, and uses `mapforceserver` instead of the executable name, for example:

```
mapforceserver help
```

Where `mapforceserver` is the path or name of the executable. Note that, if you use an absolute path, you will be able to run commands regardless of the current directory that your command prompt window (terminal) is in. However, if you would like to call the executable just by typing its name, make sure to do one of the following first:

- Change the terminal's current directory to the MapForce Server installation directory
- Add the directory where the executable is to the PATH environment variable.

Both of these scenarios are described in more detail below.

**Tips and tricks**

If you are new to command line, be aware of the following tips and tricks.

- To find out the current directory where you command line window is, enter `pwd` on Linux and macOS. On Windows, enter `echo %CD%`. 

- Make use of the **Tab** key to quickly enter various file or directory paths without having to type them in full. For example, if you type `cd c:\prog` at the command line, and then press **Tab**, you will get `C:\Program Files` automatically pre-filled (or perhaps some other directory under C:\ whose name begins with "Prog").
- When entering paths that contain white space, such as `C:\Program Files` on Windows, enclose them within quotes.
- If you see a message similar to "This command is not recognized as an internal or external command, operable program or batch file", you have most likely mistyped a path or command.
- On Linux, make sure that you use the correct case for file or directory names. For example, typing a path such as `/home/nikita/downloads` will return an error if the directory name is actually `/home/nikita/Downloads`.
- When typing a path on Linux or macOS, use forward slashes, as opposed to back slashes on Windows.

**How to run a command**

1. Open a command prompt window.
   a. To open a command prompt on Windows, press the **Windows** key and then start typing **cmd**. Click the **Command Prompt** suggestion that appears.
   b. To open a terminal on Mac, click the **Finder** icon, and then select **Go > Utilities** from the menu. Double-click the **Terminal** icon in the Utilities window.
   c. If you run Linux from a graphical user interface, locate and run the **Terminal** command as applicable to your Linux distribution. If you run Linux from a command line interface, ignore this step.

2. Enter the full path to the executable, followed by the command you want to run. For example, the command below provides help at the command line.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td><code>/opt/Altova/MapForceServer2021/bin/mapforceserver help</code></td>
</tr>
<tr>
<td>macOS</td>
<td><code>/usr/local/Altova/MapForceServer2021/bin/mapforceserver help</code></td>
</tr>
<tr>
<td>Windows</td>
<td><code>C:\Program Files (x86)\Altova\MapForceServer2021\bin\MapForceServer.exe help</code></td>
</tr>
</tbody>
</table>

In the example above, the command **help** was run without any options or arguments. Other commands may have arguments and options, and those arguments and options could be mandatory or optional. For example, the **run** command has a mandatory argument that lets you supply the path or file name of the .mfx file to be run. Check the reference section for details about each command.

**Calling MapForce Server in the installation directory**

To call the executable without having to type the full path, change the current directory to the directory where the MapForce Server executable was installed, for example:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td><code>cd /opt/Altova/MapForceServer2021/bin</code></td>
</tr>
<tr>
<td>macOS</td>
<td><code>cd /usr/local/Altova/MapForceServer2021/bin</code></td>
</tr>
<tr>
<td>Windows</td>
<td><code>cd &quot;C:\Program Files (x86)\Altova\MapForceServer2021\bin\MapForceServer.exe&quot;</code></td>
</tr>
</tbody>
</table>
You can now run any command by typing just the executable name, for example:

<table>
<thead>
<tr>
<th></th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>./mapforceserver help</td>
</tr>
<tr>
<td>macOS</td>
<td>./mapforceserver help</td>
</tr>
<tr>
<td>Windows</td>
<td>MapForceServer.exe help</td>
</tr>
</tbody>
</table>

**Note:** On Linux and macOS systems, the prefix `./` indicates that the executable is in the current directory.

**Calling MapForce Server from any directory**

To call the executable from any directory, refer to it using the absolute path. Alternatively, if you want to call the program by typing just the executable name, you can edit the PATH environment variable of your operating system so that it includes the full path to the MapForce Server installation directory. For ways to change the PATH environment variable, refer to the documentation of your operating system.

**Note:** After changing the PATH environment variable, you may need to close the terminal window and open a new one, in order for the changes to take effect.
4.1 assignlicense (Windows only)

Syntax and description

The assignlicense command is available on Windows only (not on Linux or Mac systems). It uploads a license file to the Altova LicenseServer with which MapForce Server is registered (see the licenseserver command), and assigns the license to MapForce Server. It takes the path of a license file as its argument. The command also allows you to test the validity of a license.

mapforceserver assignlicense [options] FILE

- The FILE argument takes the path of the license file.
- The --test-only option uploads the license file to LicenseServer and validates the license, but does not assign the license to MapForce Server.

For details about licensing, see the LicenseServer documentation (https://www.altova.com/manual/AltovaLicenseServer/).

Examples

Examples of the assignlicense command:

mapforceserver assignlicense C:\licensepool\mylicensekey.altova_licenses
mapforceserver assignlicense --test-only=true C:\licensepool\mylicensekey.altova_licenses

- The first command above uploads the specified license to LicenseServer and assigns it to MapForce Server.
- The last command uploads the specified license to LicenseServer and validates it, without assigning it to MapForce Server.

Options

Options are listed in short form (if available) and long form. You can use one or two dashes for both short and long forms. An option may or may not take a value. If it takes a value, it is written like this: --option=value. Values can be specified without quotes except in two cases: (i) when the value string contains spaces, or (ii) when explicitly stated in the description of the option that quotes are required. If an option takes a Boolean value and no value is specified, then the option's default value is TRUE. Use the --h, --help option to display information about the command.

- test-only [t]
  --t, --test-only = true|false
  Values are true|false. If true, then the license file is uploaded to LicenseServer and validated, but not assigned.
4.2 exportresourcestrings

Syntax and description

The `exportresourcestrings` command outputs an XML file containing the resource strings of the MapForce Server application in the specified language. Available export languages are English (`en`), German (`de`), Spanish (`es`), French (`fr`), and Japanese (`ja`).

```
mapforceserver exportresourcestrings [options] LanguageCode XMLOutputFile
```

- The `LanguageCode` argument gives the language of the resource strings in the output XML file; this is the export language. Allowed export languages (with their language codes in parentheses) are: English (`en`), German (`de`), Spanish (`es`), French (`fr`), and Japanese (`ja`).
- The `XMLOutputFile` argument specifies the path and name of the output XML file.

How to create localizations is described below.

Examples

Examples of the `exportresourcestrings` command:

```
mapforceserver exportresourcestrings de c:\Strings.xml
```

- The command above creates a file called `Strings.xml` at `c:\` that contains the resource strings of MapForce Server in German.

Creating localized versions of MapForce Server

You can create a localized version of MapForce Server for any language of your choice. Five localized versions (English, German, Spanish, French, and Japanese) are already available in the `C:\Program Files (x86)\Altova\MapForceServer2021\bin` folder, and therefore do not need to be created.

Create a localized version as follows:

1. Generate an XML file containing the resource strings by using the `exportresourcestrings` command (see command syntax above). The resource strings in this XML file will be one of the five supported languages: English (`en`), German (`de`), Spanish (`es`), French (`fr`), or Japanese (`ja`), according to the `LanguageCode` argument used with the command.
2. Translate the resource strings from one of the five supported languages into the target language. The resource strings are the contents of the `<string>` elements in the XML file. Do not translate variables in curly brackets, such as `{option}` or `{product}`.
3. Contact Altova Support to generate a localized MapForce Server DLL file from your translated XML file.
4. After you receive your localized DLL file from Altova Support, save the DLL in the `C:\Program Files (x86)\Altova\MapForceServer2021\bin` folder. Your DLL file will have a name of the form `MapForceServer2021_lc.dll`. The `_lc` part of the name contains the language code. For example, in `MapForceServer2021_de.dll`, the `de` part is the language code for German (Deutsch).
5. Run the `setdeflang` command to set your localized DLL file as the MapForce Server application to use. For the argument of the `setdeflang` command, use the language code that is part of the DLL name.

**Note:** Altova MapForce Server is delivered with support for five languages: English, German, Spanish, French,
and Japanese. So you do not need to create a localized version of these languages. To set any of these languages as the default language, use MapForce Server’s `setdeflang` command.
4.3 help

Syntax and description
The `help` command takes a single argument (Command), which is the name of the command for which help is required. It displays the command's syntax, its options, and other relevant information. If the Command argument is not specified, then all commands of the executable are listed, with each having a brief text description.

```
  mapforceserver help Command
```

Example
Example of the `help` command to display information about the `licenser server` command:

```
  mapforceserver help licenser server
```

The --help option
Help information about a command is also available by using the `--help` option of the command for which help information is required. The two commands below produce the same results:

```
  mapforceserver licenser server --help
```

The command above uses the `--help` option of the `licenser server` command.

```
  mapforceserver help licenser server
```

The `help` command takes `licenser server` as its argument.

Both commands display help information about the `licenser server` command.
4.4 **licenseserver**

Syntax and description

The `licenseserver` command registers MapForce Server with the Altova LicenseServer specified by the `Server-Or-IP-Address` argument. For the `licenseserver` command to be executed successfully, the two servers (MapForce Server and LicenseServer) must be on the same network and LicenseServer must be running. You must also have administrator privileges in order to register MapForce Server with LicenseServer.

```
mapforceserver licenseserver [options] Server-Or-IP-Address
```

- The `Server-Or-IP-Address` argument takes the name or IP address of the LicenseServer machine.

Once MapForce Server has been successfully registered with LicenseServer, you will receive a message to this effect. The message will also display the URL of the LicenseServer. You can now go to LicenseServer to assign MapForce Server a license. For details about licensing, see the LicenseServer documentation (https://www.altova.com/manual/AltovaLicenseServer/).

**Examples**

Examples of the `licenseserver` command:

```
mapforceserver licenseserver DOC.altova.com
mapforceserver licenseserver localhost
mapforceserver licenseserver 127.0.0.1
```

The commands above specify, respectively, the machine named `DOC.altova.com`, and the user's machine (`localhost` and `127.0.0.1`) as the machine running Altova LicenseServer. In each case, the command registers MapForce Server with the LicenseServer on the machine specified. The last command calls the server-executable to execute the command.

**Options**

Options are listed in short form (if available) and long form. You can use one or two dashes for both short and long forms. An option may or may not take a value. If it takes a value, it is written like this: `--option=value`. Values can be specified without quotes except in two cases: (i) when the value string contains spaces, or (ii) when explicitly stated in the description of the option that quotes are required. If an option takes a Boolean value and no value is specified, then the option's default value is `TRUE`. Use the `--h, --help` option to display information about the command.

- json [j]

  `--j, --json = true|false`

  Values are `true|false`. If `true`, prints the result of the registration attempt as a machine-parsable JSON object.
4.5 run

The `run` command executes a MapForce Server execution file (.mfx file) supplied as argument. The MapForce Server execution file is created with MapForce; it essentially represents a mapping compiled for server execution.

Any input files required by the mapping are expected to be at the path specified at mapping design time in MapForce. If MapForce Server does not run on the same operating system as MapForce, the input files required by the mapping must be copied to the target machine alongside with the .mfx file, and must be referenced using a relative path. For information about configuring a mapping with respect to relative or absolute paths, refer to MapForce documentation (https://www.altova.com/documentation#mapforce). Other prerequisites may apply, depending on how you designed the mapping, see Preparing Mappings for Server Execution.

If the mapping returns a simple value such as string, this output is written in the `stdout` (standard output) stream. On the other hand, the success and error messages are available in the `stderr` (standard error) stream. If you do not want the standard output stream to be displayed on the screen together with the success or error messages, redirect either the standard output or the standard error stream (or both) to files. If neither the `stdout` nor the `stderr` streams are redirected, they are both displayed on the screen, combined.

For example, to redirect the standard output stream to a file, use:

```
mapforceserver run MyMapping.mfx > MyOutput.txt
```

To redirect the standard error stream to a file, use:

```
mapforceserver run MyMapping.mfx 2> Diagnostics.log
```

To redirect both streams simultaneously, use:

```
mapforceserver run MyMapping.mfx > MyOutput.txt 2> Diagnostics.log
```

For further information about stream redirection, refer to the documentation of your operating system's command shell.

**Syntax**

Windows

```
MapForceServer run [options] MfxFile
```

Linux

```
mapforceserver run [options] MfxFile
```

Mac

```
mapforceserver run [options] MfxFile
```

**Command options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--catalog</code></td>
<td>Specifies the absolute path to a root catalog file that is not the installed root catalog file. The default value is the</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>--catalog=FILE</strong></td>
<td>This option is applicable to MapForce Server running on Linux. It specifies the absolute path to the installed root catalog file, see Catalog Files.</td>
</tr>
<tr>
<td><strong>--cert --certificatespath</strong></td>
<td>This option is applicable to MapForce Server running on Linux. It specifies the path to the directory where any certificate files required by the mapping are stored. Form: <strong>--certificatespath=DIRECTORY</strong></td>
</tr>
</tbody>
</table>
| **--cred --credential=KEY:VALUE** | Setting this option is meaningful if the mapping contains credentials that you have defined at mapping design time in MapForce. KEY is the credential name as it was defined in MapForce. VALUE consists of one or more name-value pairs, for example: \n\nname1=value1&name2=value2 \n
Where name is any of the following: \n\n- username \n- password \n- oauth:token \n
When using multiple name-value pairs separated by ampersand, enclose the KEY:VALUE part within quotes. For examples, see Credentials. |
| **--gc --globalresourceconfig** | This option is applicable if the mapping consumes Global Resources. It specifies the name of the global resource configuration (alias). This option must be used together with the **--globalresourcefile** option. Form: **--gc=VALUE** |
| **--gr --globalresourcefile** | This option is applicable if the mapping consumes Global Resources. It specifies the path of the global resource definition file. This option must be used together with the --globalresourceconfig option. Form: **--gr=FILE**. |
| **--lang=VALUE** | The language used for displaying messages. Form: **--lang=VALUE** (en, de, ja, es, fr) |
| **--param** | This option is applicable if the mapping was designed to take input parameters. It assigns a value to a parameter defined in the mapping. Form: **--param=ParamName:ParamValue**. The --param switch must be used before each parameter. Use quotes if ParamName or ParamValue contains a space. For example: **--p=company:"Nanonull Inc"**. |
Examples

This example shows you how to run a mapping execution file (.mfx) with MapForce Server on Windows. The mapping used in this example reads an input file, *Employees.xml*, and produces two output files (*PersonList.xml* and *Contacts.xml*).

First, let's generate the MapForce Server execution (.mfx) file, as follows:

1. Run MapForce and open the following MapForce design file (.mfd file): C:\Users\<username>\Documents\Altova\MapForce2021\MapForceExamples\ChainedPersonList.mfd.
3. When prompted, save the .mfx file to C: \temp directory. This will be the working directory where the mapping will be executed by MapForce Server.

Next, let's open a command line prompt and change the working directory to C:\temp.\.

```bash
cd C:\temp
```

Finally, run the following command to execute *ChainedPersonList.mfx*. In this example, MapForce Server is called using an absolute path. (To call it with a relative path, add the executable's path to your system's PATH environment variable).

```
"C:\Program Files (x86)\Altova\MapForceServer2021\bin\MapForceServer.exe" run ChainedPersonList.mfx
```

The two output files (*PersonList.xml* and *Contacts.xml*) are generated in the working directory. Importantly, this mapping is configured to use absolute paths, which is why the mapping ran successfully and did not require that the input *Employees.xml* file exists in the working directory. The *Employees.xml* file actually exists in the MapForce Examples folder mentioned above and is referenced through an absolute path. To
specify whether paths should be treated as absolute or relative, right-click the mapping in MapForce, select **Mapping Settings**, and then select or clear the **Make paths absolute in generated code** check box. Whenever you change the mapping settings, make sure to re-compile the mapping to .mfx. For more information, see [Preparing Mappings for Server Execution](#).
4.6 setdeflang

Syntax and description

The `setdeflang` command (short form is `sdl`) sets the default language of MapForce Server. Available languages are English (`en`), German (`de`), Spanish (`es`), French (`fr`), and Japanese (`ja`). The command takes a mandatory `LanguageCode` argument.

```
mapforceserver setdeflang [options] LanguageCode
```

- The `LanguageCode` argument is required and sets the default language of MapForce Server. The respective values to use are: `en`, `de`, `es`, `fr`, `ja`.
- Use the `--h, --help` option to display information about the command.

Examples

Examples of the `setdeflang` (sdl) command:

```
mapforceserver sdl de
mapforceserver setdeflang es
```

- The first command sets the default language of MapForce Server to German.
- The second command sets the default language of MapForce Server to Spanish.

Options

Use the `--h, --help` option to display information about the command.
### 4.7 verifylicense (Windows only)

#### Syntax and description

The `verifylicense` command checks whether the current product is licensed. Additionally, the `--license-key` option enables you to check whether a specific license key is already assigned to the product. This command is supported only on Windows systems. It is not supported on Linux or Mac systems.

```
mapforceserver verifylicense [options]
```

- To check whether a specific license is assigned to MapForce Server, supply the license key as the value of the `--license-key` option.

For details about licensing, see the LicenseServer documentation (https://www.altova.com/manual/AltovaLicenseServer/).

#### Examples

Example of the `verifylicense` command:

```
mapforceserver verifylicense
mapforceserver verifylicense --license-key=ABCD123-ABCD123-ABCD123-ABCD123-ABCD123-ABCD123
```

- The first command checks whether MapForce Server is licensed.
- The second command checks whether MapForce Server is licensed with the license key specified with the `--license-key` option.

#### Options

Options are listed in short form (if available) and long form. You can use one or two dashes for both short and long forms. An option may or may not take a value. If it takes a value, it is written like this: `--option=value`. Values can be specified without quotes except in two cases: (i) when the value string contains spaces, or (ii) when explicitly stated in the description of the option that quotes are required. If an option takes a Boolean value and no value is specified, then the option's default value is `TRUE`. Use the `--h, --help` option to display information about the command.

- **license-key [l]**
  - `--l, --license-key = Value`
    - Checks whether MapForce Server is licensed with the license key specified as the value of this option.
4.8 version

Syntax and description
The version command displays the version number of MapForce Server.

```
mapforceserver version
```

Example
Example of the version command:

```
mapforceserver version
```
5 MapForce Server API

MapForce Server provides an application programming interface (API) that you can access programmatically from your .NET, Java, or COM-based code.

For an introduction to each platform, refer to the following topics:

- .NET Interface 66
- COM Interface 72
- Java Interface 81

For a technical description of the API, refer to the following topics:

- API Reference (COM, .NET) 65
- API Reference (Java) 109
5.1 .NET Interface

The .NET interface is built as a wrapper around the COM interface. It is provided as a primary interop assembly signed by Altova and uses the namespace Altova.MapForceServer. During installation, MapForce Server will be registered automatically as a COM server object, so there is no need for a manual registration.

In order to use MapForce Server in your .NET project, add a reference to the Altova.MapForceServer.dll file, as shown below. The Altova.MapForceServer.dll is located in the bin folder of the MapForce Server installation folder. This .dll file is automatically added to the Global Assembly Cache (GAC) during MapForce Server installation.

Note: Prior to .NET Framework 4.0, the GAC was located in the %windir%assembly directory. Starting with .NET Framework 4.0, the GAC is located in the %windir%Microsoft.NET\assembly directory. The %windir% part represents the Windows operating system directory, typically C:\Windows.

To add a reference to the MapForce Server DLL in a Visual Studio .NET project:

1. With the .NET project open in Visual Studio, click Project | Add Reference.

2. On the Browse tab, browse for the folder: <MapForceServer application folder>/bin, select the Altova.MapForceServer.dll, and click OK.
You can view the structure of the `Altova.MapForceServer` assembly using the Visual Studio Object Browser (to display the Object Browser, click **Object Browser** on the **View** menu).

### 5.1.1 C# Example

The following example illustrates how to run a mapping execution file (.mfx) from C# code. On Windows, the example files are available at the following path: `C:\Program Files\Altova\MapForceServer2021\etc\Examples`.

#### Prerequisites

- MapForce Server is installed and licensed
- If you are creating a new Visual Studio project, add a reference to the MapForce Server assembly (see **.NET Interface**). You can skip this step if you are running the existing MapForce Server API example, because the example already references the MapForce Server assembly.
- On the **Build** menu of Visual Studio, click **Configuration Manager** and set a correct build platform, for example **Debug | x86** (or **Debug | x64**, if applicable). Do not use "Any CPU" as platform.
- If you have installed MapForce Server 64-bit, then the application which calls the API (such as the sample one below) must also be built for the 64-bit platform in Visual Studio. Also, the path to the MapForce server executable must be adjusted accordingly in the code.

#### Running the mapping code

The code below runs three server execution files (.mfx). The table below lists the input files expected by each .mfx file, and the output that will be created after execution.

<table>
<thead>
<tr>
<th>Execution file (.mfx)</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>TokenizeString.mfx</td>
<td>AltovaTools.xml</td>
<td>AltovaToolsFeatures.csv</td>
</tr>
<tr>
<td>SimpleTotal.mfx</td>
<td>ipo.xml</td>
<td>String</td>
</tr>
<tr>
<td>ClassifyTemperatures.mfx</td>
<td>Temperatures.xml</td>
<td>Temperatures_out.xml</td>
</tr>
</tbody>
</table>

If you have Altova MapForce, you can optionally take a look at the original mappings from which the .mfx files were compiled in order to understand them better. These are called `TokenizeString1.mfd`, `SimpleTotal.mfd`, and `ClassifyTemperatures.mfd`, respectively. You can find the mappings in the following directory: `C:\users\<user>\Altova\MapForce2021\MapForceExamples`.

The example below does the following:

- It creates a new instance of `Altova.MapForceServer.Server`. This is the object you will subsequently be working with.
- It sets a working directory where execution takes place. Input files are expected to exist in this directory if you referred to them using a relative path. Output files will also be created in this directory.
(see the table above).

- It runs **TokenizeString.mfx**. The file path is supplied as argument to the Run method (notice that the path is relative to the working directory that was set previously). Upon successful execution, a .csv file representing the mapping output will be created in the working directory.

- It runs **SimpleTotal.mfx**. Again, the file path is relative to the working directory. This mapping produces a string output, so we call the **GetOutputParameter** method to get the string output.

- It runs **ClassifyTemperatures.mfx**. This mapping expects a parameter as input, which was supplied with the help of the **AddParameter** method.

```
namespace MapForceServerAPI_sample
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                // Create a MapForce Server object

                // Set a working directory - used as a base for relative paths (you may need to adapt the path to the installation folder)
                objMFS.WorkingDirectory = "..\..\..";

                // Default path to the MapForce Server executable is the installation path (same dir with the MapForceServer.dll)
                // In case you moved the binaries on the disk, you need to explicitly set the path to the .exe file
                // objMFS.ServerPath = "C:\\Program Files (x86)\\Altova\MapForceServer2020\\bin\\MapForceServer.exe";
                // objMFS.ServerPath = "C:\\Program Files\\Altova\\MapForceServer2020\\bin\\MapForceServer.exe";

                System.Console.WriteLine("Running " + objMFS.ProductNameAndVersion + ".\n");

                // Set global resource file and configuration, if your mapping uses global resources
                // objMFS.SetOption( "globalresourcefile", "GlobalResources.xml" ); // "gr" can be used as short name for "globalresourcefile"
                // objMFS.SetOption( "globalresourceconfig", "Default" ); // "gc" can be used as short name for "globalresourceconfig"

                // An example with input and output paths stored inside the MFX file
                System.Console.WriteLine("\nExecuting TokenizeString.mfx...\n");
                if (objMFS.Run("TokenizeString.mfx"))
                {
                    System.Console.WriteLine("Successfully generated file 'AltovaToolFeatures.csv'.");
                }
                else
                {
                    // execution failed. maybe no write permissions in working directory?
                    Run this program as administrator.
                }
            }
        }
    }
}
```
System.Console.WriteLine(objMFS.LastExecutionMessage);
}

// An example creating a simple output so that we can retrieve the result explicitly
System.Console.WriteLine("\nExecuting SimpleTotal.mfx...\n");
if (objMFS.Run("SimpleTotal.mfx"))
    System.Console.WriteLine("Mapping result is: " +
    objMFS.GetOutputParameter("total");
else
{
    // execution failed (e.g. somebody deleted file ipo.xml)
    System.Console.WriteLine(objMFS.LastExecutionMessage);
}

// an example with parameterized input
// the default of 'lower = 5' gets changed to the value '10'
// mfx reads file Temperatures.xml and writes its output to Temperatures_out.xml.
System.Console.WriteLine("\nExecuting ClassifyTemperatures.mfx with parameter 'lower' set to '10' ...");
objMFS.AddParameter("lower", "10");
if (objMFS.Run("ClassifyTemperatures.mfx"))
    System.Console.WriteLine("File Temperatures_out.xml has been written successfully.\n");
else
{
    // execution failed. maybe no write permissions in working directory?
    Run this program as administrator.
    System.Console.WriteLine(objMFS.LastExecutionMessage);
}
}

catch (System.Runtime.InteropServices.COMException ex)
{
    System.Console.WriteLine("Internal Error - " + ex.Message);
}

5.1.2 Visual Basic .NET Example

The following example illustrates how to run a mapping execution file (.mfx) from VB.NET code. On Windows, the example files are available at the following path: C:\Program Files\Altova\MapForceServer2021\etc\Examples.

Prerequisites

- MapForce Server is installed and licensed
If you are creating a new Visual Studio project, add a reference to the MapForce Server assembly (see .NET Interface). You can skip this step if you are running the existing MapForce Server API example, because the example already references the MapForce Server assembly.

On the Build menu of Visual Studio, click Configuration Manager and set a correct build platform, for example Debug | x86 (or Debug | x64, if applicable). Do not use "Any CPU" as platform.

If you have installed MapForce Server 64-bit, then the application which calls the API (such as the sample one below) must also be built for the 64-bit platform in Visual Studio. Also, the path to the MapForce server executable must be adjusted accordingly in the code.

The example solution is in the "Program Files" directory, which requires administrative rights. Either run Visual Studio as administrator, or copy the solution to a different folder where you don’t need administrative rights.

### Running the mapping code

The code below runs three server execution files (.mfx). The table below lists the input files expected by each .mfx file, and the output that will be created after execution.

<table>
<thead>
<tr>
<th>Execution file (.mfx)</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>TokenizeString.mfx</td>
<td>AltovaTools.xml</td>
<td>AltovaToolsFeatures.csv</td>
</tr>
<tr>
<td>SimpleTotal.mfx</td>
<td>ipo.xml</td>
<td>String</td>
</tr>
<tr>
<td>ClassifyTemperatures.mfx</td>
<td>Temperatures.xml</td>
<td>Temperatures_out.xml</td>
</tr>
</tbody>
</table>

If you have Altova MapForce, you can optionally take a look at the original mappings from which the .mfx files were compiled in order to understand them better. These are called TokenizeString1.mfd, SimpleTotal.mfd, and ClassifyTemperatures.mfd, respectively. You can find the mappings in the following directory: C: \users\<user>\Altova\MapForce2021\MapForceExamples.

The example below does the following:

- It creates a new instance of Altova.MapForceServer.Server. This is the object you will subsequently be working with.
- It sets a working directory where execution takes place. Input files are expected to exist in this directory if you referred to them using a relative path. Output files will also be created in this directory (see the table above).
- It runs TokenizeString.mfx. The file path is supplied as argument to the Run method (notice that the path is relative to the working directory that was set previously). Upon successful execution, a .csv file representing the mapping output will be created in the working directory.
- It runs SimpleTotal.mfx. Again, the file path is relative to the working directory. This mapping produces a string output, so we call the GetOutputParameter method to get the string output.
- It runs ClassifyTemperatures.mfx. This mapping expects a parameter as input, which was supplied with the help of the AddParameter method.
Sub Main()

Try
  'Create a MapForce Server object;

  'Set a working directory - used as a base for relative paths for the MapForce server execution (.mfx) file.
  'objMFS.WorkingDirectory = "C:\Program Files (x86)\Altova\MapForceServer2020\etc\Examples"
  objMFS.WorkingDirectory = "..\..\.."

  'Default path to the MapForce Server executable is the installation path
  (same dir with the MapForceServer.dll)
  'In case you moved the binaries on the disk, you need to explicitly set the
  path to the .exe file
  'objMFS.ServerPath = "C:\Program Files (x86)\Altova\MapForceServer2021\bin\MapForceServer.exe"
  'objMFS.ServerPath = "C:\Program Files\Altova\MapForceServer2021\bin\MapForceServer.exe"

  'Set global resource file and configuration, if your mapping uses global resources
  'objMFS.SetOption("globalresourcefile", "GlobalResources.xml") "gr" can be used as short name for "globalresourcefile"
  'objMFS.SetOption("globalresourceconfig", "Config2") "gc" can be used as short name for "globalresourceconfig"

  '----------------------------------------------------------------------------
 ------
  'An example with input and output paths stored inside the MFX file
  System.Console.WriteLine(vbCrLf & "Executing TokenizeString.mfx...")
  If (objMFS.Run("TokenizeString.mfx")) Then
    System.Console.WriteLine("Successfully generated file 'AltovaToolFeatures.csv'.")
  Else
    'execution failed. maybe no write permissions in working directory? Run this program as administrator.
    System.Console.WriteLine(objMFS.LastExecutionMessage)
  End If
  '----------------------------------------------------------------------------
 ------
  'An example creating a simple output so that we can retrieve the result explicitly
  System.Console.WriteLine(vbCrLf & "Executing SimpleTotal.mfx...")
  If (objMFS.Run("SimpleTotal.mfx")) Then
    System.Console.WriteLine("Mapping result is: " & objMFS.GetOutputParameter("total"))
  Else
    'execution failed (e.g. somebody deleted file ipo.xml)
    System.Console.WriteLine(objMFS.LastExecutionMessage)
End If

' an example with parameterized input
' the default of 'lower=5' gets changed to the value '10'
' mfx reads file Temperatures.xml and writes its output to Temperatures_out.xml.

    System.Console.WriteLine(vbCrLf & "Executing ClassifyTemperatures.mfx with parameter 'lower' set to '10' ...")
    objMFS.AddParameter("lower", "10")
    If (objMFS.Run("ClassifyTemperatures.mfx")) Then
        System.Console.WriteLine("File Temperatures_out.xml has been written successfully.")
    Else
        ' execution failed. maybe no write permissions in working directory? Run this program as administrator.
        System.Console.WriteLine(objMFS.LastExecutionMessage)
    End If

Catch ex As Exception
End Try

End Sub

End Module
5.2 COM Interface

MapForce Server is automatically registered as a COM server object during installation. To check whether the registration was successful, open the Registry Editor (for example, by typing `regedit.exe` command at the command line). If registration was successful, the Registry will contain the following classes:

- `MapForce.Server` (for 32-bit MapForce Server)
- `MapForce_x64.Server` (for 64-bit MapForce Server)

These classes are found under `HKEY_LOCAL_MACHINE\SOFTWARE\Classes`.

Once the COM server object is registered, you can invoke it from within applications and scripting languages that have programming support for COM calls. If you wish to change the location of the MapForce Server installation package, it is best to uninstall MapForce Server and then reinstall it at the required location. In this way, the necessary de-registration and registration are carried out by the installer process.

5.2.1 C++ Example

The following example illustrates how to run a mapping execution file (.mfx) from C++ code. On Windows, the example files are available at the following path: `C:\Program Files\Altova\MapForceServer2021\etc\Examples`.

Prerequisites

Before running the code below, ensure the following prerequisites are met:

- MapForce Server is installed and licensed
- MapForce Server is available as a COM server object (normally, this process takes place automatically during MapForce Server installation; to check if registration was successful, see About the COM Interface).

Running the mapping code

The code below runs three server execution files (.mfx). The table below lists the input files expected by each .mfx file, and the output that will be created after execution.

<table>
<thead>
<tr>
<th>Execution file (.mfx)</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>TokenizeString.mfx</td>
<td>AltovaTools.xml</td>
<td>AltovaToolsFeatures.csv</td>
</tr>
<tr>
<td>SimpleTotal.mfx</td>
<td>ipo.xml</td>
<td>String</td>
</tr>
<tr>
<td>ClassifyTemperatures.mfx</td>
<td>Temperatures.xml</td>
<td>Temperatures_out.xml</td>
</tr>
</tbody>
</table>

If you have Altova MapForce, you can optionally take a look at the original mappings from which the .mfx files were compiled in order to understand them better. These are called `TokenizeString1.mfd`, `SimpleTotal.mfd`, and `ClassifyTemperatures.mfd`, respectively. You can find the mappings in the following directory: `C:\users\<user>\Altova\MapForce2021\MapForceExamples`.

The example below does the following:
• It creates a new instance of `Altova.MapForceServer.Server`. This is the object you will subsequently be working with.

• It sets a working directory where execution takes place. Input files are expected to exist in this directory if you referred to them using a relative path. Output files will also be created in this directory (see the table above).

• It runs `TokenizeString.mfx`. The file path is supplied as argument to the `Run` method (notice that the path is relative to the working directory that was set previously). Upon successful execution, a `.csv` file representing the mapping output will be created in the working directory.

• It runs `SimpleTotal.mfx`. Again, the file path is relative to the working directory. This mapping produces a string output, so we call the `GetOutputParameter` method to get the string output.

• It runs `ClassifyTemperatures.mfx`. This mapping expects a parameter as input, which was supplied with the help of the `AddParameter` method.

```cpp
// MapForceServerAPI_sample.cpp : Defines the entry point for the console application.

#include <iostream>
#include "atlbase.h"

#ifndef _WIN64
#include "progid:MapForce.Server"
#else
#include "progid:MapForce_x64.Server"
#endif

int _tmain(int argc, _TCHAR* argv[])
{
    CoInitialize( NULL );

    try
    {
        // Create a MapForce Server object
        MapForceServerLib::IServerPtr pMFS;
        CoCreateInstance( __uuidof( MapForceServerLib::Server ), NULL, CLSCTX_ALL,
                         reinterpret_cast<void** >( &pMFS ) );

        // Set a working directory - used as a base for relative paths (you may need to adapt the path to the installation folder)
        pMFS->WorkingDirectory = ".."; // this is relative to this applications' working directory (the project folder)

        // Default path to the MapForce Server executable is the installation path (same dir with the MapForceServer.dll)
        // In case you moved the binaries on the disk, you need to explicitly set the path to the .exe file
        // pMFS.ServerPath = "C:\Program Files (x86)\Altova\MapForceServer2021\bin\MapForceServer.exe";
        // pMFS.ServerPath = "C:\Program Files\Altova\MapForceServer2021\bin\MapForceServer.exe";
```
// Set global resource file and configuration, if your mapping uses global resources
// pMFS->SetOption( "globalresourcefile", "GlobalResources.xml" ); // "gr" can be used as short name for "globalresourcefile"
// pMFS->SetOption( "globalresourceconfig", "Default" ); // "gc" can be used as short name for "globalresourceconfig"

// An example with input and output paths stored inside the MFX file
std::cout <<  "\nExecuting TokenizeString.mfx..." << std::endl;
if ( pMFS->Run( "TokenizeString.mfx" ) == VARIANT_TRUE )
  std::cout <<  "Successfully generated file 'AltovaToolFeatures.csv'." << std::endl;
else
  // execution failed. maybe no write permissions in working directory? Run this program as administrator.
  std::cout <<  pMFS->LastExecutionMessage << std::endl;

// An example creating a simple output so that we can retrieve the result explicitly
std::cout <<  "\nExecuting SimpleTotal.mfx..." << std::endl;
if ( pMFS->Run( "SimpleTotal.mfx" ) )
  std::cout <<  "Mapping result is: " + pMFS->GetOutputParameter( "total" ) << std::endl;
else
  // execution failed (e.g. somebody deleted file ipo.xml)
  std::cout <<  pMFS->LastExecutionMessage << std::endl;

// An example with parameterized input
// the default of 'lower = 5' gets changed to the value '10'
// mfx reads file Temperatures.xml and writes its output to Temperatures_out.xml.
std::cout <<  "\nExecuting ClassifyTemperatures.mfx with parameter 'lower' set to '10'..." << std::endl;
pMFS->AddParameter("lower", "10");
if ( pMFS->Run( "ClassifyTemperatures.mfx" ) )
  std::cout <<  "File Temperatures_out.xml has been written successfully." << std::endl;
else
  // execution failed. maybe no write permissions in working directory? Run this program as administrator.
  std::cout <<  pMFS->LastExecutionMessage << std::endl;
}

catch (_com_error& err )
5.2.2 VBScript Example

The following example illustrates how to run a mapping execution file (.mfx) from VBScript code. On Windows, the example files are available at the following path: `C:\Program Files\Altova\MapForceServer2021\etc\Examples`.

Before running the code below, ensure the following prerequisites are met:

- MapForce Server is installed and licensed
- MapForce Server is available as a COM server object (normally, this process takes place automatically during MapForce Server installation; to check if registration was successful, see About the COM Interface).

```vb
Option Explicit

REM This script produces extensive output.
REM It is best called from a cmd.exe console with "cscript MapForceServerAPI_sample.vbs"

' Create the MapForce Server object
Dim objMFS
' Since we load a COM-DLL we need care about the process architecture
On Error Resume Next ' ignore any COM errors avoiding uncontrolled script termination
Dim WshShell
Dim WshProcEnv
Set WshShell = CreateObject("WScript.Shell")
Set WshProcEnv = WshShell.Environment("Process")
Dim process_architecture
process_architecture = WshProcEnv("PROCESSOR_ARCHITECTURE")
If process_architecture = "x86" Then
    Set objMFS = WScript.GetObject( ",", "MapForce.Server" )
    If Err.Number <> 0 then
        WScript.Echo("You are running in a 32-bit process but MapForce Server COM-API 32-bit seems not to be installed on your system.")
        WScript.Quit -1
    End If
Else
    Set objMFS = WScript.GetObject( ",", "MapForce_x64.Server" )
    If Err.Number <> 0 then
```
WScript.Echo("You are running in a 64-bit process but MapForce Server COM-API 64-bit seems not to be installed on your system.")
WScript.Echo("If you have installed 32-bit MapForce Server consider calling your script from the 32-bit console 'C:\Windows\SysWOW64\cmd.exe.'")
WScript.Quit -1

End If
End If

On Error Goto 0  ' re-enable default error promotion

' Set a working directory - used as a base for relative paths (you may need to adapt the path to the installation folder)
REM objMFS.WorkingDirectory = "C:\"Program Files (x86) \Altova\MapForceServer2020\etc\Examples"
Dim currDir
Dim fso
Set fso = CreateObject("Scripting.FileSystemObject")
currDir = fso.GetParentFolderName(Wscript.ScriptFullName)
' set working folder to parent of this script
objMFS.WorkingDirectory = fso.GetParentFolderName( currDir )

'Default path to the MapForce Server executable is the installation path (same dir with the MapForceServer.dll)
' In case you moved the binaries on the disk, you need to explicitly set the path to the .exe file
' objMFS.ServerPath = "C:\Program Files (x86) \Altova\MapForceServer2021\bin\MapForceServer.exe"

'Set global resource file and configuration, if your mapping uses global resources
' Call objMFS.SetOption("globalresourcefile", "GlobalResources.xml") ' "gr" can be used as short name for "globalresourcefile"
' Call objMFS.SetOption("globalresourceconfig", "Config2") ' "gc" can be used as short name for "globalresourceconfig"

WScript.Echo( "Running " & objMFS.ProductNameAndVersion & vbCrLf )

' The Run method will return 'True' if the execution of the mfx file was successful otherwise 'False'.
' In the case of fundamental errors like termination of the server process a COM error will be raised which ' can be handled using the VBScript Err object.
On Error Resume Next  ' ignore any COM errors avoiding uncontrolled script termination
Err.Clear

REM ----------------------------------------------------------------------------
REM run an example with input and output paths stored inside the MFX file
'REM the path to the mfx file can be absolute or relative to the working directory
'REM depends on existence of file AltovaTools.xml in working directory
'REM creates output file AltovaToolFeatures.csv in working directory
WScript.Echo( "Processing TokenizeString.mfx..." )
If ( objMFS.Run( "TokenizeString.mfx" ) ) Then
    WScript.Echo( objMFS.LastExecutionMessage )  ' execution log
WScript.Echo("Successfully generated file AltovaToolFeatures.csv."")
Else
  ' execution failed (e.g. somebody deleted file AltovaTools.xml)
  WScript.Echo( objMFS.LastExecutionMessage )
End If
WScript.Echo(""")
' handle COM errors
If Err.Number <> 0 Then
  WScript.Echo("Internal error - " & Err.Description )
  WScript.Quit -1
End If
REM

REM this is an example creating a simple output so that we can retrieve the result explicitly
' depends on input XML file ipo.xml
WScript.Echo( "Processing SimpleTotal.mfx..." )
If ( objMFS.Run( "SimpleTotal.mfx" ) ) Then
  WScript.Echo( objMFS.LastExecutionMessage )
  WScript.Echo( "Mapping result is: " & objMFS.GetOutputParameter("total") )
Else
  ' execution failed (e.g. somebody deleted file ipo.xml)
  WScript.Echo( objMFS.LastExecutionMessage )
End If
WScript.Echo(""")
' handle COM errors
If Err.Number <> 0 Then
  WScript.Echo("Internal error - " & Err.Description )
  WScript.Quit -1
End If
REM

REM this is an example with parameterized input
' the default of "lower=5" gets changed to the value '10'
' mfx reads file Temperatures.xml and writes its output to Temperatures_out.xml.
WScript.Echo( "Processing ClassifyTemperatures.mfx with parameter 'lower' set to '10'
..." )
call objMFS.AddParameter("lower", "10")
If ( objMFS.Run( "ClassifyTemperatures.mfx" ) ) Then
  WScript.Echo( objMFS.LastExecutionMessage )
  WScript.Echo( "File Temperatures_out.xml has been written successfully." )
Else
  ' execution failed (e.g. somebody locks file Temperatures_out.xml)
  WScript.Echo( objMFS.LastExecutionMessage )
End If
call objMFS.ClearParameterList()
WScript.Echo(""")
' handle COM errors
If Err.Number <> 0 Then
  WScript.Echo("Internal error - " & Err.Description )
  WScript.Quit -1
End If
5.2.3 VBA Example

Microsoft Visual Basic for Applications (VBA) is primarily used for automating tasks in Microsoft Office. However, it is also possible to call the MapForce Server API from VBA and execute mappings. The following instructions have been tested on MapForce Server and the VBA for Microsoft Office 2013. Instructions may differ if you are using another VBA development environment.

Prerequisites
Before you can call the MapForce Server API functions from your VBA project, note the following prerequisites:

1. Microsoft Office and MapForce Server must be installed on the same machine.
2. The architecture of MapForce Server (32-bit or 64-bit) must match that of Microsoft Office. For example, if you run VBA on Microsoft Office 32-bit, make sure that you use MapForce Server 32-bit. To find out whether your Office product runs on 64-bit, click the File tab, click Account, and then click "About Excel" (or "About Word").
3. The MapForce Server library must be referenced from your VBA project (see instructions below).

How to add a reference to the MapForce Server Library from your VBA project

1. In a macro-enabled Microsoft Office document (.docm, .xlsm), on the Developer tab, click Visual Basic.

   By default, the Developer tab is not enabled in Microsoft Office. To enable the Developer tab in an Office 2013 program, right-click the ribbon and select Customize the Ribbon from the context menu. Then, in the Options dialog box, select the Developer check box under "Main Tabs".

2. In the VBA development environment, in the Tools menu, click References.
How to call the MapForce Server API

Once you have added a reference to the MapForce Server Library in your VBA project, you can enter the VBA code in the Code Editor window. For example, the following sample code calls MapForce Server and runs a mapping executable file (`mapping.mfx`) that takes an input parameter called "outfile" having the value "output.csv".

```vba
Sub RunMapping()
    ' Create a new instance of the MapForce Server
    Dim objMFS As New MapForceServerLib.Server
    With objMFS
        ' Set the working directory
        ' (used as base if the mapping has relative paths)
        .WorkingDirectory = "C:\temp\"
        ' If the mapping has parameters, supply them
        Call .AddParameter("outfile", "output.csv")
        ' Run the mapping
        .Run ("C:\temp\mapping.mfx")
    End With
End Sub
```

Press **F5** to debug the VBA code and run the mapping.
Sub RunMapping()
    ' Create a new instance of the MapForce Server
    Dim objMFS As New MapForceServerLib.Server
    With objMFS
        ' Set the working directory
        .WorkingDirectory = "C:\temp\"
        ' If the mapping has parameters, supply them
        Call .AddParameter("outfile", "output.csv")
        ' Run the mapping
        .Run("C:\temp\mapping.mfx")
    End With
End Sub
5.3 Java Interface

To access the MapForce Server API from Java, a reference to `MapForceServer.jar` library must be added to the Java CLASSPATH. You can find the `MapForceServer.jar` file in the `bin` folder of the MapForce Server installation folder. You can either reference the file from its original location or copy it to another location if this fits your project setup.

If you are using Eclipse as Java development environment, you can add the required library reference to the CLASSPATH by editing the properties of the Java project, as shown below. Optionally, you can also attach documentation in JavaDoc format to the .jar library. You can find the JavaDoc in the `bin` folder of the MapForce Server installation folder; the instructions below illustrate how to make the JavaDoc documentation visible from Eclipse.

To add a reference to the MapForce Server library:

1. With the project open in Eclipse, on the **Project** menu, click **Properties**.
2. Click **Java Build Path**.
3. On the **Libraries** tab, click **Add External JARs**, and then browse for the `MapForceServer.jar` file located in the MapForce Server installation folder.

4. Optionally, to add the JavaDoc archive, expand the `MapForceServer.jar` record, and then double-click the **Javadoc location: (None)** record.
5. Ensure that the **Javadoc in archive** and **External file** options are selected, and then browse for the `MapForceServer_JavaDoc.zip` file located in the MapForce Server installation folder.

6. Click OK.

Below is an example of how the Eclipse .classpath file might look if you are referencing the files from the original installation folder, on a 64-bit Windows running 64-bit MapForce Server (the relevant lines are highlighted in yellow):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<classpath>
  <classpathentry kind="src" path=""/>
  <classpathentry kind="lib" path="C:\Program Files\Altova\MapForceServer2019\bin\M">
```
5.3.1 Java Example

The following example illustrates how to run a mapping execution file (.mfx) from Java code. On Windows, all the example files are available at the following path: `C:\Program Files\Altova\MapForceServer2021\etc\Examples`. They are as follows:

- **TokenizeString.mfx** - A MapForce Server execution file that will be run by the Java program.
- **AltovaTools.xml** - An XML file that provides input data to the mapping program.

On Linux and Mac platforms, no examples are pre-installed; however, you can prepare an executable .mfx file like the one run in this example as follows:

1. Open the desired mapping design file (.mfd) with MapForce on Windows.
2. On the File menu, select **Mapping Settings**, and clear the **Make paths absolute in generated code** check box if it is selected.
3. For each mapping component, open the Properties dialog box (by double-clicking the component's title bar, for example), and change all file paths from absolute to relative. Also, select the **Save all file paths relative to MFD file** check box. For convenience, you can copy all input files and schemas in the same folder as the mapping itself, and reference them just by the file name. Refer to MapForce documentation for more information about dealing with relative and absolute paths while designing mappings.
4. On the File menu, select **Compile to MapForce Server Execution file**. This generates the .mfx file that you will subsequently run with MapForce Server, as shown in the code listing below.

### Prerequisites

Before running the code below, ensure the following prerequisites are met:

- MapForce Server is installed and licensed
- The Java CLASSPATH includes a reference to the `MapForceServer.jar` library (for an example, see About the Java Interface).

If you are using a custom .mfx file as shown above, there may be other prerequisites, depending on the kind of data processed by the mapping. For more information, see Preparing Mappings for Server Execution.
On Windows, the example Java project is in the "Program Files" directory, which requires administrative rights. You will either need to run your Java development environment (for example, Eclipse) as administrator, or copy the example to a different folder where you don't need administrative rights.

Running the Java program
The code listing below first creates a MapForce Server object. Next, it sets the working directory where the application should look for any files that act as input to the mapping, and where it should generate the mapping output files. As mentioned above, the example mapping file reads data from a source XML file—so make sure that both the XML file and its schema exist in the working directory.

The `setServerPath` method specifies the path to the MapForce Server executable. For example, on Ubuntu, this would be `/opt/Altova/MapForceServer2021/bin/mapforceserver`. You can omit the `setServerPath` if you did not move the `MapForceServer.jar` from its default location.

Finally, the `run` method runs a mapping (.mfx file) that was compiled with MapForce (in this example, `TokenizeString.mfx`). On success, the program below generates a CSV file in the working directory. On error, the program attempts to print out the last execution message generated by MapForce Server.

```java
public class Program {
    public static void main(String[] args) {
        try {
            //Create a MapForce Server object

            //Set a working directory - used as a base for relative paths (you may need to adapt the path to the installation folder)
            objMFS.setWorkingDirectory("C:\Program Files (x86)\Altova\MapForceServer2021\etc\Examples");

            //Default path to the MapForce Server executable is the installation path (same dir with the MapForceServer.jar)
            //In case you copied the JAR file to a new location, you need to explicitly set the path to the .exe file
            objMFS.setServerPath("C:\Program Files (x86)\Altova\MapForceServer2021\bin\MapForceServer.exe");

            //Set global resource file and configuration, if your mapping uses global resources
            //objMFS.setOption( "globalresourcefile", "GlobalResources.xml" ); // "gr" can be used as short name for "globalresourcefile"
            //objMFS.setOption( "globalresourceconfig", "Default" ); // "gc" can be used as short name for "globalresourceconfig"

            //Prepare the parameters, if your design uses parameters
            //objMFS.addParameter( "testparam1", "value 1" );
        }
```
// Run the mapping; input and output paths are stored inside the MFX file
// NOTE Please adapt the path to the input file in order to run the sample
if ( objMFS.run("C:\Program Files (x86)\Altova\MapForceServer2021\etc\Examples\TokenizerString.mfx") )
    System.out.println("Success - finished execution");
else
    System.out.println(objMFS.getLastExecutionMessage());
}
catch ( Exception e )
{
    e.printStackTrace();
}
System.out.println("Finished - exiting");
}
5.4 Example: Run Mapping with Parameters

This example shows you how to compile a MapForce mapping to a MapForce Server execution file (.mfx) and run it from the MapForce API. The example specifically illustrates the scenario when the mapping takes the input file name as parameter. See also any of the previous C#, C++, VB.NET, VBScript, or Java examples.

In this example, MapForce is used so that you can view and understand the original mapping design. MapForce is also used to compile the mapping to a MapForce Server execution file (.mfx) and configure settings such as relative versus absolute paths.

The server platform used in the example is Windows. This could be either the same machine where MapForce is installed, or a different one. You can also run this example on a Linux or Mac machine (in Java), provided that you adjust the Windows-style paths as applicable to your platform.

Prerequisites

Running this mapping has the same prerequisites as described in the previous C#, C++, VB.NET, VBScript, or Java examples.

Preparing the mapping for server execution

After installing MapForce and running MapForce for the first time, several demo mapping design files are available at the following path:

C:\Users\<username>\Documents\Altova\MapForce2021\MapForceExamples

Make sure to change the path above accordingly if you have a different version of MapForce.

The mapping design used in this example is called FileNamesAsParameters.mfd. As illustrated below, this is a straightforward mapping that copies data from a source to a target XML file almost unchanged. Only the PrimaryKey and Name fields of the target XML file are populated with constants from the mapping.
Example: Run Mapping with Parameters

FileNamesAsParameters.mfd

The mapping has two input parameters: **InputFileName** and **OutputFileName**, respectively. In order to make it possible to preview the mapping in MapForce, the parameter values are supplied by two constants. As further described below, you will be able to replace the parameter values with your own when the mapping runs with MapForce Server.

Notice that both the source and target mapping components are configured to get the file name dynamically from the mapping. Namely, a **File: <dynamic>** input connector is available at the very top of the component, and it reads data from the respective mapping parameter (**InputFileName** or **OutputFileName**). In MapForce, you can set or change this option by clicking the **File/String** button in the top-right corner of the component. Be aware that the input file name supplied as parameter must be a reference to a physical XML file that provides input data to the mapping (in this example, **Altova_Hierarchical.xml** from the same folder as the mapping design file). The output file name can be any valid name, for example **Output.xml**.

Before compiling the mapping to a MapForce Server Execution file (.mfx), you will typically want to review the mapping settings. Right-click on an empty area on the mapping, and select **Mapping Settings** from the context menu.
For the scope of this example, change the settings as illustrated above. Specifically, when you clear the **Make paths absolute in generated code** check box, any absolute paths to input or output files used by the mapping are resolved as relative to the mapping design file (.mfd). At mapping runtime, MapForce Server will look for these paths in the program’s working directory. The default working directory depends on the platform that you use to access the MapForce Server API. For example, in case of a C# application, it is the same directory as the executable. In addition, as further illustrated below, you can change the working directory with the help of an API property called **WorkingDirectory** (or **setWorkingDirectory** Java method, if applicable).

Notes:

- On the Mapping Settings dialog box, the only settings that directly affect the compilation of the .mfx file are **Make paths absolute in generated code** and **Line ends**.
- In this mapping, we did not need to edit each mapping component in MapForce so as to change absolute paths to relative because all paths were already relative. Otherwise, you would need to perform this extra step as well, see [Preparing Mappings for Server Execution](#).

In this example, we will use `C:\MappingExample` as working directory. Therefore, copy the input file `Altova_Hierarchical.xml` referenced by the mapping to from `C:\Users\<username>\Documents\Altova\MapForce2021\MapForceExamples` to the working directory.

In this example, both the source and target are XML files, and there is no need to copy the XML schema file to the server, because information derived from it is embedded into the .mfx file during compilation. If your mapping uses other component types (for example, databases), there could be additional prerequisites, as described in [Preparing Mappings for Server Execution](#).

Finally, to compile the mapping to a server execution file, do the following:

- On the **File** menu, click **Compile to MapForce Server Execution file** and select a target directory. In this example, the target directory is the same as working directory, `C:\MappingExample`. 

![Mapping Settings](image)
The following code listings illustrate how to run the mapping from various environments, using the MapForce Server API. In the code listings below, notice that both parameters are referenced by the same name that they have in the mapping design. Also, the parameter value has the same data type as defined on the mapping (in this case, string).

Running the mapping from C++
You can now run the .mfx file by using C++ code such as the one below. To run this code listing successfully, make sure that MapForce Server has a valid license, see also the verifylicense CLI command.

C++

```cpp
#include <iostream>
#include "atlbase.h"

// 32-bit MapForce Server
#import "progid:MapForce.Server"

int _tmain(int argc, _TCHAR* argv[])
{
    CoInitialize( NULL );

    try
    {
        //Create a MapForce Server object
        MapForceServerLib::IServerPtr pMFS;
        CoCreateInstance( __uuidof( MapForceServerLib::Server ), NULL, CLSCTX_ALL,
                        __uuidof( MapForceServerLib::IServer ), reinterpret_cast<void**>( &pMFS ) );

        //Set a working directory - used as a base for relative paths
        pMFS->WorkingDirectory = "C:\MappingExample";

        //Prepare the parameters
        pMFS->AddParameter("InputFileName", "Altova_Hierarchical.xml");
        pMFS->AddParameter("OutputFileName", "Output.xml");

        //Run the mapping
        if (pMFS->Run("FileNamesAsParameters.mfx"))
            std::cout << "Success - finished execution" << std::endl;
        else
            std::cout << pMFS->LastExecutionMessage << std::endl;
    }
    catch (_com_error& err )
    {
        BSTR bstrMessage;
        (err).ErrorInfo()->GetDescription( &bstrMessage );
        std::cout << "Exception occurred: " << _com_util::ConvertBSTRToString( bstrMessage ) << std::endl;
    }

    CoUninitialize();
    return 0;
}
```
Running the mapping from C#

You can now run the .mfx file by using C# code such as the one below. To run this code listing successfully, first add a reference to the MapForce Server DLL in Visual Studio, as described in [NET Interface](#), and make sure that MapForce Server has a valid license, see also the [verifylicense](#) CLI command.

C#

```csharp
static void Main(string[] args)
{
    try
    {
        // Create a MapForce Server object

        // Set the working directory - all relative paths will be resolved against it
        objMFS.WorkingDirectory = "C:\\MappingExample";

        // Prepare the parameters
        objMFS.AddParameter("InputFileName", "Altova_Hierarchical.xml");
        objMFS.AddParameter("OutputFileName", "Output.xml");

        // Run the mapping
        if (objMFS.Run("FileNamesAsParameters.mfx"))
        {
            System.Console.WriteLine("Success - finished execution");
        }
        else
        {
            System.Console.WriteLine(objMFS.LastExecutionMessage);
        }
    }
    catch(System.Runtime.InteropServices.COMException ex)
    {
        System.Console.WriteLine("Internal Error: " + ex.Message);
    }
}
```

Running the mapping from Java

You can now run the .mfx file by using Java code such as the one below. To run this code listing successfully, make sure that:

- the Java CLASSPATH environment variable includes a reference to the MapForceServer.jar library, as described in [About the Java Interface](#)
- MapForce Server has a valid license, see also the [verifylicense](#) CLI command.

Java

```java
public static void main(String[] args) {
    try {
        // Create a MapForce Server object
        com.altova.mapforceserver.MapForceServer objMFS = new 
        com.altova.mapforceserver.MapForceServer();
```
// Set the working directory - all relative paths will be resolved against it
// objMFS.setWorkingDirectory("/home/ubuntu/Downloads/MappingExample");
objMFS.setWorkingDirectory("C:\\MappingExample");

// Add the mapping parameters
objMFS.addParameter("InputFileName", "Altova_Hierarchical.xml");
objMFS.addParameter("OutputFileName", "Output.xml");

// Run the mapping
if ( objMFS.run("FileNamesAsParameters.mfx") )
    System.out.println("Success - finished execution");
else
    System.out.println( objMFS.getLastExecutionMessage() );

} catch (Exception e) {
    e.printStackTrace();
}
System.out.println("Finished execution");
}
5.5 API Reference (COM, .NET)

This section provides general reference to the MapForce Server API elements (such as interfaces and methods) applicable to code written for the COM or .NET platforms.

5.5.1 Interfaces

5.5.1.1 IServer

The IServer interface creates a new MapForce Server object instance, and provides access to the MapForce Server engine.

If you are using C++ under COM platform, the name of the main interface is IServer. If you are using a .NET language such as C# or VB.NET, the name of the main interface is Server.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APIMajorVersion</td>
<td>Read-only. Gets the major version of the MapForce Server API. This can be different from the product version if the API is connected to another server.</td>
</tr>
<tr>
<td>APIMinorVersion</td>
<td>Read-only. Gets the minor version of the MapForce Server API. This can be different from the product version if the API is connected to another server.</td>
</tr>
<tr>
<td>APIServicePackVersion</td>
<td>Read-only. Gets the service pack version of the MapForce Server API. This can be different from the product version if the API is connected to another server.</td>
</tr>
<tr>
<td>Is64Bit</td>
<td>Read-only. Returns true if the MapForce Server engine is a 64-bit executable.</td>
</tr>
<tr>
<td>LastExecutionMessage</td>
<td>Read-only. Gets the message received during the last Run command.</td>
</tr>
<tr>
<td>MajorVersion</td>
<td>Read-only. Gets the major version of the product, as number of years starting from 1998 (for example, &quot;20&quot; for Altova MapForce Server 2018).</td>
</tr>
<tr>
<td>MinorVersion</td>
<td>Read-only. Gets the minor version of the product (for example, &quot;2&quot; for Altova MapForce Server 2018 r2).</td>
</tr>
</tbody>
</table>
### Name | Description
--- | ---
**ProductName** | Read-only. Gets the name of the product (for example, "Altova MapForce Server").

**ProductNameAndVersion** | Read-only. Gets the complete name and version of the product (for example, "Altova MapForce Server 2018 r2 sp1 (x64)").

**ServerPath** | Gets or sets the path of the MapForce Server executable.

**ServicePackVersion** | Read-only. Gets the service pack version of the product (for example, "1" for Altova MapForce Server 2018 r2 sp1).

**WorkingDirectory** | Gets or sets the current directory for running jobs (relative paths will be evaluated against the working directory).

### Methods

| Name | Description |
--- | --- |
**AddCredentialProperty** | Adds a property to the current credential (for example, the username, the password, or both). The first argument specifies the property name, and the second argument specifies the property value. Valid property names: `username`, `password`. In MapForce Server Advanced Edition, the property name `oauth:token` is additionally supported. This method must be called after calling `BeginCredential()` and before calling `EndCredential()`.

**AddParameter** | Assigns a value to a parameter defined in the mapping. The first argument specifies the name of the parameter as defined on the mapping; the second argument specifies the parameter value.

**BeginCredential** | Creates a new credential with the name supplied as argument. If you call this method, you must also add properties to it using `AddCredentialProperty()`, and finally close the credential by calling `EndCredential()`.

**ClearCredentialList** | Clears the list of credentials set previously. All credentials are valid for the lifetime of the object. Call this method if you need to explicitly clear all of the previously set credentials.

**ClearOptions** | Clears the list of options previously set through the `SetOption` method. All options set through the `SetOption` method are valid for the lifetime of the object. Call this method if you need to explicitly clear all of the previously set options.

**ClearParameterList** | Clears the list of parameters that were previously set using the `AddParameter` method.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EndCredential</td>
<td>Closes a credential object that was previously created using the BeginCredential method.</td>
</tr>
<tr>
<td>GetOutputParameter</td>
<td>Gets the string output generated by the last run command. Returns null if no output was generated. This function requires a string parameter which identifies the name of the output component in MapForce.</td>
</tr>
<tr>
<td>Run</td>
<td>Executes a MapForce Server Execution file (.mfx file). Returns true in case of success; false otherwise.</td>
</tr>
<tr>
<td>SetOption</td>
<td>Sets an option before running the mapping. The first argument specifies the name of the option, while the second argument specifies the option value. This method is particularly useful when a mapping was designed to consume Global Resources (see Altova Global Resources). The currently supported options are as follows:</td>
</tr>
<tr>
<td></td>
<td>- globalresourcefile (or gr) - A Global Resource file path. (When this option is specified, then a Global Resource configuration name must also be specified, see next item).</td>
</tr>
<tr>
<td></td>
<td>- globalresourceconfig (or gc) - A Global Resource configuration name. (When this option is specified, then a Global Resource file path must also be specified, see previous item).</td>
</tr>
<tr>
<td></td>
<td>- catalog - The path to a custom RootCatalog.xml file. This option enables you to specify a custom catalog file used to resolve URLs used by the mapping. The default catalog is in the etc subdirectory of the program installation directory.</td>
</tr>
<tr>
<td></td>
<td>- taxonomy-package - The path to a custom XBRL taxonomy package, if one is required by the mapping.</td>
</tr>
<tr>
<td></td>
<td>- taxonomy-packages-config-file - The path to a custom XBRL taxonomy package configuration, if one is required by the mapping.</td>
</tr>
</tbody>
</table>

All set options are valid for the lifetime of the object. If you set an option with the same name twice, the previous option will be overridden. To explicitly clear all options, call the ClearOptions() method.

| StopServerProcess   | This method stops explicitly the process connected with the COM object, without releasing the object. The process stops implicitly when the COM object is released. |

Examples
See the following examples that illustrate how to initialize and run MapForce Server in various languages:

- C++ example

---
5.5.1.1.1 Properties

5.5.1.1.1.1 APIMajorVersion

Gets the major version of the MapForce Server API. This can be different from the product version if the API is connected to another server.

Signature

```c
APIMajorVersion : Integer
```

Generic signature

```c
int APIMajorVersion { get; }
```

C#

```c
HRESULT APIMajorVersion([out, retval] INT* pnVal);
```

C++

```c
ReadOnly Property APIMajorVersion As Integer
```

VB.NET

5.5.1.1.1.2 APIMinorVersion

Gets the minor version of the MapForce Server API. This can be different from the product version if the API is connected to another server.

Signature

```c
APIMinorVersion : Integer
```

Generic signature

```c
int APIMinorVersion { get; }
```

C#

```c
HRESULT APIMinorVersion([out, retval] INT* pnVal);
```
5.5.1.1.3  APIServicePackVersion

Gets the service pack version of the MapForce Server API. This can be different from the product version if the API is connected to another server.

**Signature**

```csharp
APIServicePackVersion : Integer
```

**Generic signature**

```c#
int APIServicePackVersion { get; }
```

**C#**

```c#
HRESULT APIServicePackVersion([out, retval] INT* pnVal);
```

**C++**

```c++
ReadOnly Property APIServicePackVersion As Integer
```

**VB.NET**

```vbnet
5.5.1.1.4  Is64Bit

Returns true if the MapForce Server engine is a 64-bit executable.

**Signature**

```csharp
Is64Bit : Boolean
```

**Generic signature**

```c#
bool Is64Bit { get; }
```

**C#**

```c#
HRESULT Is64Bit([out, retval] VARIANT_BOOL* pbVal);
```
5.5.1.1.1.5 LastExecutionMessage

Gets the message received during the last Run command.

**Signature**

LastExecutionMessage : String

**Generic signature**

string LastExecutionMessage { get; }

C#

HRESULT LastExecutionMessage([out, retval] BSTR* pbstrResult);

VB.NET

ReadOnly Property LastExecutionMessage As String

5.5.1.1.1.6 MajorVersion

Gets the major version of the product, as number of years starting from 1998 (for example, "20" for Altova MapForce Server 2018).

**Signature**

MajorVersion : Integer

**Generic signature**

int MajorVersion { get; }

C#

HRESULT MajorVersion([out, retval] INT* pnVal);


**C++**

```
ReadOnly Property MajorVersion As Integer
```

**VB.NET**


---

**5.5.1.1.7 MinorVersion**

Gets the minor version of the product (for example, "2" for Altova MapForce Server 2018 r2).

**Signature**

```
MinorVersion : Integer
```

**Generic signature**

```
int MinorVersion { get; }
```

**C#**

```
HRESULT MinorVersion([out, retval] INT* pnVal);
```

**C++**

```
ReadOnly Property MinorVersion As Integer
```

**VB.NET**


---

**5.5.1.1.8 ProductName**

Gets the name of the product (for example, "Altova MapForce Server").

**Signature**

```
ProductName : String
```

**Generic signature**

```
string ProductName { get; }
```

**C#**

```
HRESULT ProductName([out, retval] BSTR* pstrVal);
```
5.5.1.1.9  `ProductNameAndVersion`

Gets the complete name and version of the product (for example, "Altova MapForce Server 2018 r2 sp1 (x64)").

**Signature**

```csharp
HRESULT ProductNameAndVersion([out, retval] BSTR* pstrVal);
```

```cpp
ReadOnly Property ProductNameAndVersion As String
```

**VB.NET**

```vbnet
5.5.1.1.10  `ServerPath`

Gets or sets the path of the MapForce Server executable.

**Signature**

```csharp
HRESULT ServerPath([in] BSTR bstrServerFile);
HRESULT ServerPath([out, retval] BSTR* pbstrServerFile );
```

```cpp
ReadOnly Property ProductNameAndVersion As String
```

**VB.NET**
5.5.1.1.11 ServicePackVersion

Gets the service pack version of the product (for example, "1" for Altova MapForce Server 2018 r2 sp1).

Signature

ServicePackVersion : Integer

Generic signature

int ServicePackVersion { get; }

C#

HRESULT ServicePackVersion([out, retval] INT* pnVal);

C++

ReadOnly Property ServicePackVersion As Integer

VB.NET

5.5.1.1.12 WorkingDirectory

Gets or sets the current directory for running jobs (relative paths will be evaluated against the working directory).

Signature

WorkingDirectory : String

Generic signature

string WorkingDirectory { set; get; }

C#

HRESULT WorkingDirectory([in] BSTR bstrWorkingDirectory );
HRESULT WorkingDirectory([out, retval] BSTR* pbstrWorkingDirectory);

C++

Property WorkingDirectory As String

VB.NET

5.5.1.1.2 Methods

5.5.1.1.2.1 AddCredentialProperty

Adds a property to the current credential (for example, the username, the password, or both). The first argument specifies the property name, and the second argument specifies the property value. Valid property names: username, password. In MapForce Server Advanced Edition, the property name oauth:token is additionally supported.

This method must be called after calling BeginCredential() and before calling EndCredential().

Signature

AddCredentialProperty(in bstrName: System.String, in bstrValue: System.String) -> Void

Generic signature

void AddCredentialProperty(string bstrName, string bstrValue)

C#

HRESULT AddCredentialProperty([in] BSTR bstrName, [in] BSTR bstrValue);

C++

Sub AddCredentialProperty(ByVal bstrName As String, ByVal bstrValue As String)

VB.NET

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrName</td>
<td>System.String</td>
<td>Specifies the name of the credential property.</td>
</tr>
</tbody>
</table>
Name | Type | Description
--- | --- | ---
bstrValue | System.String | Specifies the value of the credential property.

**Examples**
The following code listing illustrates how to declare a credential called "mycredential" in C#. The credential name must be the one given to the credential in MapForce at design time.

```csharp
//Create a MapForce Server object

objMFS.BeginCredential("mycredential");
objMFS.AddCredentialProperty("username", "altova");
objMFS.AddCredentialProperty("password", "b45ax78!");
objMFS.EndCredential();
```

### 5.5.1.1.2.2 AddParameter
Assigns a value to a parameter defined in the mapping. The first argument specifies the name of the parameter as defined on the mapping; the second argument specifies the parameter value.

**Signature**

```csharp
AddParameter(in bstrName:String, in bstrValue:String) -> Void
```

**Generic signature**

```csharp
void AddParameter(string bstrName, string bstrValue)
```

**C#**

```csharp
HRESULT AddParameter([in] BSTR bstrName, [in] BSTR bstrValue);
```

**C++**

```csharp
Sub AddParameter(ByVal bstrName As String, ByVal bstrValue As String)
```

**VB.NET**

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
bstrName | String   | Specifies the parameter name. |
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrValue</td>
<td>String</td>
<td>Specifies the parameter value.</td>
</tr>
</tbody>
</table>

### 5.5.1.1.2.3 BeginCredential

Creates a new credential with the name supplied as argument. If you call this method, you must also add properties to it using AddCredentialProperty(), and finally close the credential by calling EndCredential().

**Signature**

```csharp
void BeginCredential(string bstrCredentialName)
```

**C#**

```csharp
HRESULT BeginCredential([in] BSTR bstrCredentialName);
```

**C++**

```csharp
Sub BeginCredential(ByVal bstrCredentialName As String)
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrCredentialName</td>
<td>String</td>
<td>Specifies the name of the credential as it was defined in MapForce.</td>
</tr>
</tbody>
</table>

### 5.5.1.1.2.4 ClearCredentialList

Clears the list of credentials set previously. All credentials are valid for the lifetime of the object. Call this method if you need to explicitly clear all of the previously set credentials.

**Signature**

```csharp
void ClearCredentialList()
```

**Generic signature**

```csharp
void ClearCredentialList()
```
5.5.1.1.2.5  ClearOptions

Clears the list of options previously set through the `SetOption` method. All options set through the `SetOption` method are valid for the lifetime of the object. Call this method if you need to explicitly clear all of the previously set options.

**Signature**

```plaintext
void ClearOptions()
C#
HRESULT ClearOptions();
C++
Sub ClearOptions()
VB.NET
```

**Generic signature**

```plaintext
void ClearOptions()
C#
HRESULT ClearOptions();
C++
Sub ClearOptions()
VB.NET
```
5.5.1.1.2.6  **ClearParameterList**
Clears the list of parameters that were previously set using the `AddParameter` method.

**Signature**

```
ClearParameterList() -> Void
```

**Generic signature**

```
void ClearParameterList()
```

*C#*

```
HRESULT ClearParameterList();
```

*C++*

```
Sub ClearParameterList()
```

*VB.NET*


5.5.1.1.2.7  **EndCredential**
Closes a credential object that was previously created using the `BeginCredential` method.

**Signature**

```
EndCredential() -> Void
```

**Generic signature**

```
void EndCredential()
```

*C#*

```
HRESULT EndCredential();
```

*C++*

```
Sub EndCredential()
```

*VB.NET*
5.5.1.1.2.8  GetOutputParameter

Gets the string output generated by the last run command. Returns null if no output was generated. This function requires a string parameter which identifies the name of the output component in MapForce.

**Signature**

GetOutputParameter(\texttt{in bstrName: String}) $\rightarrow$ \texttt{String}

*Generic signature*

\texttt{string GetOutputParameter(string bstrName)}

*C#*

\texttt{HRESULT GetOutputParameter([in] BSTR bstrName, [out, retval] BSTR* pbstrValue )};

*C++*

\texttt{Function GetOutputParameter(bstrName As String) As String}

*VB.NET*

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrName</td>
<td>String</td>
<td>Specifies the name of the output component as it appears in MapForce. This name is displayed in the title bar of each component on the mapping (or when you right-click the component header, and select Properties).</td>
</tr>
</tbody>
</table>

5.5.1.1.2.9  Run

Executes a MapForce Server Execution file (.mfx file). Returns \texttt{true} in case of success; \texttt{false} otherwise.

**Signature**

Run(\texttt{in bstrMappingPath: String}) $\rightarrow$ \texttt{Boolean}

*Generic signature*

\texttt{bool Run(string bstrMappingPath)}
C#  

HRESULT Run([in] BSTR bstrMappingPath, [out, retval] VARIANT_BOOL* pbSuccess);

C++  

Function Run(ByVal bstrMappingPath As String) As Boolean

VB.NET  

Parameters  

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrMappingPath</td>
<td>String</td>
<td>Specifies the path to the MapForce Server execution (.mfx) file. If you specify a relative path, then it will be resolved against the working directory. You can set the working directory from the WorkingDirectory property.</td>
</tr>
</tbody>
</table>

5.5.1.1.2.10 SetOption  

Sets an option before running the mapping. The first argument specifies the name of the option, while the second argument specifies the option value. This method is particularly useful when a mapping was designed to consume Global Resources (see Altova Global Resources). The currently supported options are as follows:

- **globalresourcefile (or gr)** - A Global Resource file path. (When this option is specified, then a Global Resource configuration name must also be specified, see next item).
- **globalresourceconfig (or gc)** - A Global Resource configuration name. (When this option is specified, then a Global Resource file path must also be specified, see previous item).
- **catalog** - The path to a custom RootCatalog.xml file. This option enables you to specify a custom catalog file used to resolve URLs used by the mapping. The default catalog is in the etc subdirectory of the program installation directory.
- **taxonomy-package** - The path to a custom XBRL taxonomy package, if one is required by the mapping.
- **taxonomy-packages-config-file** - The path to a custom XBRL taxonomy package configuration, if one is required by the mapping.

All set options are valid for the lifetime of the object. If you set an option with the same name twice, the previous option will be overridden. To explicitly clear all options, call the ClearOptions() method.

Signature  

SetOption(in bstrName: String, in bstrValue: String) -> Void
**Generic signature**

```c#
void SetOption(ByName bstrName As String, ByVal bstrValue As String)
```

**C++**

```c++
Sub SetOption(ByName bstrName As String, ByVal bstrValue As String)
```

**VB.NET**

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bstrName</td>
<td>String</td>
<td>Specifies the name of the option to set.</td>
</tr>
<tr>
<td>bstrValue</td>
<td>String</td>
<td>Specifies the value of the option to set.</td>
</tr>
</tbody>
</table>

### 5.5.1.1.2.11  StopServerProcess

This method stops *explicitly* the process connected with the COM object, without releasing the object. The process stops implicitly when the COM object is released.

**Signature**

```c#
StopServerProcess() -> System.Void
```

**Generic signature**

```c#
void StopServerProcess()
```

**C#**

```c#
Sub StopServerProcess()
```

**VB.NET**
5.6 API Reference (Java)

This section provides general reference to the MapForce Server API elements (such as classes and methods) applicable to code written for the Java platform.

5.6.1 Classes

5.6.1.1 MapForceServer

The MapForceServer class creates a new MapForce Server object instance, and provides access to the MapForce Server engine.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addCredentialPropertiesFromMap</td>
<td>Adds properties from a credential property map to the current credential. This method takes as argument a credential property map (property_name, property_value). This method must be called after calling beginCredential() and before calling endCredential(). As an alternative to calling this method, you can also call AddCredentialProperty().</td>
</tr>
<tr>
<td>addCredentialProperty</td>
<td>Adds a property to the current credential (for example, the username, the password, or both). The first argument specifies the property name, and the second argument specifies the property value. Valid property names: username, password. In MapForce Server Advanced Edition, the property name oauth:token is additionally supported. This method must be called after calling beginCredential() and before calling endCredential(). As an alternative to calling this method, you can also call addCredentialPropertiesFromMap().</td>
</tr>
<tr>
<td>addParameter</td>
<td>Assigns a value to a parameter defined in the mapping.</td>
</tr>
<tr>
<td>beginCredential</td>
<td>Creates a new credential with the name supplied as argument. If you call this method, you must also add properties to it using addCredentialProperty() or addCredentialPropertiesFromMap(), and finally close the credential by calling endCredential().</td>
</tr>
<tr>
<td>clearCredentialList</td>
<td>Clears the list of credentials set previously. All credentials are valid for the lifetime of the object. Call this method if you need to explicitly clear all of the previously set credentials.</td>
</tr>
<tr>
<td>clearOptions</td>
<td>Clears the list of options previously set through the setOption() method. All options set through the setOption method are cleared.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>clearParameterList</td>
<td>Clears the list of parameters that were previously set using the <code>addParameter</code> method.</td>
</tr>
<tr>
<td>endCredential</td>
<td>Closes a credential object that was previously created using the <code>beginCredential</code> method.</td>
</tr>
<tr>
<td>getAPIMajorVersion</td>
<td>Gets the major version of the MapForce Server API. This can be different from the product version if the API is connected to another server.</td>
</tr>
<tr>
<td>getAPIMinorVersion</td>
<td>Gets the minor version of the MapForce Server API. This can be different from the product version if the API is connected to another server.</td>
</tr>
<tr>
<td>getAPIServicePackVersion</td>
<td>Gets the service pack version of the MapForce Server API. This can be different from the product version if the API is connected to another server.</td>
</tr>
<tr>
<td>getLastExecutionMessage</td>
<td>Gets the message received during the last <code>run</code> command.</td>
</tr>
<tr>
<td>getMajorVersion</td>
<td>Gets the major version of the product, as number of years starting from 1998 (for example, &quot;20&quot; for Altova MapForce Server 2018).</td>
</tr>
<tr>
<td>getMinorVersion</td>
<td>Gets the minor version of the product (for example, &quot;2&quot; for Altova MapForce Server 2018 r2).</td>
</tr>
<tr>
<td>getOutputParameter</td>
<td>Gets the string output generated by the last <code>run</code> command. Returns null if no output was generated. This function requires a string parameter which identifies the name of the output component in MapForce.</td>
</tr>
<tr>
<td>getName</td>
<td>Gets the name of the product (for example, &quot;Altova MapForce Server&quot;).</td>
</tr>
<tr>
<td>getProductNameAndVersion</td>
<td>Gets the complete name and version of the product (for example, &quot;Altova MapForce Server 2018 r2 sp1 (x64)&quot;).</td>
</tr>
<tr>
<td>getServicePackVersion</td>
<td>Gets the service pack version of the product (for example, &quot;1&quot; for Altova MapForce Server 2018 r2 sp1 (x64)).</td>
</tr>
<tr>
<td>is64bit</td>
<td>Returns <code>true</code> if the MapForce Server engine is a 64-bit executable.</td>
</tr>
<tr>
<td>run</td>
<td>Executes a MapForce Server Execution file (.mfx file). Returns <code>true</code> in case of success; <code>false</code> otherwise.</td>
</tr>
<tr>
<td>setOption</td>
<td>Sets an option before running the mapping. The first argument specifies the name of the option, while the second argument specifies the option value. This method is particularly useful</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>globalresourcefile (or gr)</td>
<td>A Global Resource file path. (When this option is specified, then a Global Resource configuration name must also be specified, see next item).</td>
</tr>
<tr>
<td>globalresourceconfig (or gc)</td>
<td>A Global Resource configuration name. (When this option is specified, then a Global Resource file path must also be specified, see previous item).</td>
</tr>
<tr>
<td>catalog</td>
<td>The path to a custom RootCatalog.xml file. This option enables you to specify a custom catalog file used to resolve URLs used by the mapping. The default catalog is in the etc subdirectory of the program installation directory.</td>
</tr>
<tr>
<td>taxonomy-package</td>
<td>The path to a custom XBRL taxonomy package, if one is required by the mapping.</td>
</tr>
<tr>
<td>taxonomy-packagess-config-file</td>
<td>The path to a custom XBRL taxonomy package configuration, if one is required by the mapping.</td>
</tr>
</tbody>
</table>

All set options are valid for the lifetime of the object. If you set an option with the same name twice, the previous option will be overridden. To explicitly clear all options, call the `clearOptions()` method.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>setServerPath</td>
<td>Sets the path of the MapForce Server executable.</td>
</tr>
<tr>
<td>setWorkingDirectory</td>
<td>Sets the current directory for running jobs (relative paths will be evaluated against the working directory).</td>
</tr>
</tbody>
</table>

**Examples**

For an example of creating a new instance of MapForceServer in Java, see the [Java example](#).

### 5.6.1.1.1 Methods

#### 5.6.1.1.1 addCredentialPropertiesFromMap

Adds properties from a credential property map to the current credential. This method takes as argument a credential property map (`property_name`, `property_value`). This method must be called after calling `beginCredential()` and before calling `endCredential()`. As an alternative to calling this method, you can also call `AddCredentialProperty()`.
addCredentialPropertiesFromMap(arg0: Map) -> void

Generic signature

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg0</td>
<td>Map</td>
<td>A map that supplies the name of the credential property and its value.</td>
</tr>
</tbody>
</table>

Examples

The following code listing illustrates adding a credential called "mycredential" to the current context, using the `addCredentialPropertiesFromMap` method.

```java
//Create a MapForce Server object
objMFS.beginCredential("mycredential");
java.util.Map<String, String> credentialMap = new java.util.HashMap<String,String>();
credentialMap.put("username", "altova");
credentialMap.put("password", "b45ax78!");
objMFS.addCredentialPropertiesFromMap(credentialMap);
objMFS.endCredential();
```

5.6.1.1.1.2 addCredentialProperty

Adds a property to the current credential (for example, the username, the password, or both). The first argument specifies the property name, and the second argument specifies the property value. Valid property names: `username`, `password`. In MapForce Server Advanced Edition, the property name `oauth:token` is additionally supported.

This method must be called after calling `beginCredential()` and before calling `endCredential()`. As an alternative to calling this method, you can also call `addCredentialPropertiesFromMap()`.

addCredentialProperty(arg0: String, arg1: String) -> void

Generic signature
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg0</td>
<td>String</td>
<td>The name of the credential property (for example, &quot;username&quot; or &quot;password&quot;).</td>
</tr>
<tr>
<td>arg1</td>
<td>String</td>
<td>The value of the credential property.</td>
</tr>
</tbody>
</table>

Examples

The following code listing illustrates how to declare a credential called "mycredential". The credential name must be the one given to the credential in MapForce at design time.

```java
//Create a MapForce Server object
objMFS.beginCredential("mycredential");
objMFS.addCredentialProperty("username", "altova");
objMFS.addCredentialProperty("password", "b45ax78!");
objMFS.endCredential();
```

5.6.1.1.3  addParameter

Assigns a value to a parameter defined in the mapping.

Signature

```
addParameter(arg0:String, arg1:String) -> void
```

Generic signature

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg0</td>
<td>String</td>
<td>Specifies the parameter name.</td>
</tr>
<tr>
<td>arg1</td>
<td>String</td>
<td>Specifies the parameter value.</td>
</tr>
</tbody>
</table>
**5.6.1.1.1.4  beginCredential**

Creates a new credential with the name supplied as argument. If you call this method, you must also add properties to it using `addCredentialProperty()` or `addCredentialPropertiesFromMap()`, and finally close the credential by calling `endCredential()`.

**Signature**

```
beginCredential(arg0: String) -> void
```

*Generic signature*

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg0</td>
<td>String</td>
<td>The name of the credential as it was defined in MapForce.</td>
</tr>
</tbody>
</table>

**5.6.1.1.1.5  clearCredentialList**

Clears the list of credentials set previously. All credentials are valid for the lifetime of the object. Call this method if you need to explicitly clear all of the previously set credentials.

**Signature**

```
clearCredentialList() -> void
```

*Generic signature*

**5.6.1.1.1.6  clearOptions**

Clears the list of options previously set through the `setOption()` method. All options set through the `setOption` method are valid for the lifetime of the object. Call this method if you need to explicitly clear all of the previously set options.

**Signature**

```
clearOptions() -> void
```

*Generic signature*
5.6.1.1.1.7 clearParameterList

Clears the list of parameters that were previously set using the addParameter method.

**Signature**

```java
void clearParameterList()
```

*Generic signature*

5.6.1.1.1.8 endCredential

Closes a credential object that was previously created using the beginCredential method.

**Signature**

```java
void endCredential()
```

*Generic signature*

5.6.1.1.1.9 getAPIMajorVersion

Gets the major version of the MapForce Server API. This can be different from the product version if the API is connected to another server.

**Signature**

```java
int getAPIMajorVersion()
```

*Generic signature*

5.6.1.1.1.10 getAPIMinorVersion

Gets the minor version of the MapForce Server API. This can be different from the product version if the API is connected to another server.

**Signature**

```java
int getAPIMinorVersion()
```

*Generic signature*
5.6.1.1.1.11  getAPIServicePackVersion

Gets the service pack version of the MapForce Server API. This can be different from the product version if the API is connected to another server.

Signature

getAPIServicePackVersion() -> int

Generic signature

5.6.1.1.1.12  getLastExecutionMessage

Gets the message received during the last run command.

Signature

getLastExecutionMessage() -> java.lang.String

Generic signature

5.6.1.1.1.13  getMajorVersion

Gets the major version of the product, as number of years starting from 1998 (for example, "20" for Altova MapForce Server 2018).

Signature

getMajorVersion() -> int

Generic signature

5.6.1.1.1.14  getMinorVersion

Gets the minor version of the product (for example, "2" for Altova MapForce Server 2018 r2).

Signature

getMinorVersion() -> int

Generic signature
5.6.1.1.1.15  getOutputParameter

Gets the string output generated by the last run command. Returns null if no output was generated. This function requires a string parameter which identifies the name of the output component in MapForce.

**Signature**

```java
getOutputParameter(String arg0) -> java.lang.String
```

**Generic signature**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg0</td>
<td>String</td>
<td>Specifies the name of the output component as it appears in MapForce. This name is displayed in the title bar of each component on the mapping (or when you right-click the component header, and select Properties).</td>
</tr>
</tbody>
</table>

5.6.1.1.1.16  getProductName

Gets the name of the product (for example, "Altova MapForce Server").

**Signature**

```java
getProductName() -> java.lang.String
```

**Generic signature**

5.6.1.1.1.17  getProductNameAndVersion

Gets the complete name and version of the product (for example, "Altova MapForce Server 2018 r2 sp1 (x64)").

**Signature**

```java
getProductNameAndVersion() -> java.lang.String
```

**Generic signature**
5.6.1.1.1.18  getServicePackVersion

Gets the service pack version of the product (for example, "1" for Altova MapForce Server 2018 r2 sp1 (x64)).

**Signature**

```
getServicePackVersion() -> int
```

*Generic signature*

5.6.1.1.1.19  is64bit

Returns true if the MapForce Server engine is a 64-bit executable.

**Signature**

```
is64bit() -> boolean
```

*Generic signature*

5.6.1.1.1.20  run

Executes a MapForce Server Execution file (.mfx file). Returns true in case of success; false otherwise.

**Signature**

```
run(arg0: String) -> boolean
```

*Generic signature*

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg0</td>
<td>String</td>
<td>Specifies the path to the MapForce Server Execution file (.mfx file). If you specify a relative path, then it will be resolved against the working directory. You can set the working directory by calling the setWorkingDirectory method.</td>
</tr>
</tbody>
</table>
5.6.1.1.1.21  setOption

Sets an option before running the mapping. The first argument specifies the name of the option, while the second argument specifies the option value. This method is particularly useful when a mapping was designed to consume Global Resources (see Altova Global Resources). The currently supported options are as follows:

- **globalresourcefile (or gr)** - A Global Resource file path. (When this option is specified, then a Global Resource configuration name must also be specified, see next item).
- **globalresourceconfig (or gc)** - A Global Resource configuration name. (When this option is specified, then a Global Resource file path must also be specified, see previous item).
- **catalog** - The path to a custom RootCatalog.xml file. This option enables you to specify a custom catalog file used to resolve URLs used by the mapping. The default catalog is in the etc subdirectory of the program installation directory.
- **taxonomy-package** - The path to a custom XBRL taxonomy package, if one is required by the mapping.
- **taxonomy-packages-config-file** - The path to a custom XBRL taxonomy package configuration, if one is required by the mapping.

All set options are valid for the lifetime of the object. If you set an option with the same name twice, the previous option will be overridden. To explicitly clear all options, call the `clearOptions()` method.

**Signature**

```
setOption(arg0: String, arg1: String) -> void
```

**Generic signature**

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg0</td>
<td>String</td>
<td>Specifies the name of the option to set.</td>
</tr>
<tr>
<td>arg1</td>
<td>String</td>
<td>Specifies the value of the option to set.</td>
</tr>
</tbody>
</table>

5.6.1.1.1.22  setServerPath

Sets the path of the MapForce Server executable.

**Signature**

```
setServerPath(arg0: String) -> void
```

**Generic signature**
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg0</td>
<td>String</td>
<td>Specifies the path to the MapForce Server executable.</td>
</tr>
</tbody>
</table>

5.6.1.1.23  setWorkingDirectory

Sets the current directory for running jobs (relative paths will be evaluated against the working directory).

Signature

setWorkingDirectory(arg0: String) -> void

Generic signature

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg0</td>
<td>String</td>
<td>Specifies the path to the working directory.</td>
</tr>
</tbody>
</table>

5.6.1.2  MapForceServerException

The MapForceServerException class provides programmatic access to exceptions thrown by the MapForceServer class.

public class MapForceServerException extends Exception
6 Digital Certificate Management

Digital certificate management is an integral part of secure data exchange between a client computer and a Web server. Since mappings can be executed not only on Windows by MapForce, but also on a Windows, Linux or macOS server by MapForce Server (either standalone or in FlowForce Server execution), this section deals with managing HTTPS certificates on various platforms.

In the context of secure HyperText Transport Protocol (HTTPS), it is important to distinguish between server and client certificates.

Server certificates
A server certificate is what identifies a server as a trusted entity to a client application such as MapForce. The server certificate may be digitally signed by a commercial Certificate Authority, or it may be self-signed by your organization. In either case, while designing the mapping in MapForce, you can specify the following settings:

- Whether the server certificate must be checked.
- Whether the request must proceed if a mismatch has been detected between the name certificate and the name of the host.

These settings are available on the HTTP Security Settings dialog box of MapForce. When you enable server certificate checks, consider the following:

- If you are calling a Web server whose certificate is signed by a trusted Certificate Authority, your operating system will likely be already configured to trust the server certificate, and no additional configuration is necessary.
- If you are calling a Web server which provides a self-signed certificate (for example, a local network server within your organization), you will need to configure your operating system as well to trust that certificate.

In most cases, you can check the level of trust between your operating system and the Web server by typing the URL of the Web service in the browser's address bar. If the server is not trusted, or if your operating system is not configured to trust the server, your browser will display a message such as "This connection is untrusted", or "There is a problem with this website's certificate". Note that you cannot use the browser to check the level of trust with a Web server if the browser uses a certificate database other than that of the operating system (for example, Firefox on Ubuntu).

On Windows, you can establish trust with the server by following the browser's instructions and importing or installing the required certificates into your system's Trusted Root Authorities store (see Trusting Server Certificates on Windows). On macOS, you can do the equivalent operation in Keychain Access (see Trusting Server Certificates on macOS). For instructions applicable to Linux, see Trusting Server Certificates on Linux.

Client certificates
While server certificates are used to identify a server as a trusted entity, client certificates are primarily used to authenticate the caller against the Web server. If you intend to call a Web server which requires client certificates, you may need to contact the administrator of the Web server for the client configuration instructions. Taking IIS (Internet Information Services) as an example, the Web server may be configured to handle HTTPS and client certificates in one of the following ways:
• Require HTTPS and ignore client certificate
• Require HTTPS and accept client certificate
• Require HTTPS and require client certificate

The success or failure of the Web service request depends both on the configuration of the Web server and the client application. For example, if the Web server is configured to require a client certificate, then, for the call to be successful, the calling application must present a valid client certificate.

From a MapForce perspective, the same is true for mappings which include Web service calls through HTTPS. In particular, to run such mappings successfully, it is assumed that the Web server has been configured to accept or require the client certificate, and that the operating system where the mapping runs provides the correct client certificate to the Web server.

The diagram below illustrates a scenario where a client certificate used in MapForce is transferred to a Linux server running MapForce Server. Once the certificate has been transferred to the target operating system, MapForce Server can use it to authenticate itself against the Web server and execute the mapping successfully.

![Diagram](image_url)

*Deploying mappings with client certificates to another computer*

For HTTPS authentication in Web service calls, MapForce is capable of using Transport Layer Security (TLS) on top of HTTP, which is the successor of Secure Sockets Layer (SSL) protocol. Note that fallback to SSL may occur if either the client implementation or the server does not support TLS.

To support Web calls with client certificate authentication on multiple platforms, MapForce (and MapForce Server) relies on the certificate management implementation of each platform, thus ensuring that certificate management is always in the scope of the underlying operating system. Each operating system provides different support for certificate management, as shown in the table below.
<table>
<thead>
<tr>
<th>Platform</th>
<th>Certificate management and implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>On Windows, you can manage certificates using the Certificate snap-in (see Accessing the Certificate Stores on Windows). TLS support is available through the Secure Channel (also known as SChannel) library.</td>
</tr>
<tr>
<td>Linux</td>
<td>On Linux, you can manage certificates using the OpenSSL (openssl) command line tool and library. If OpenSSL support is not already available on the Linux machine where MapForce Server is installed, you will need to download and install it before you can manage certificates. TLS support is available through the OpenSSL library (<a href="https://www.openssl.org/">https://www.openssl.org/</a>).</td>
</tr>
<tr>
<td>macOS</td>
<td>On macOS, you can manage certificates using the Keychain Access Manager, located under Finder &gt; Applications &gt; Utilities. TLS support is provided by the Secure Transport library native to the operating system.</td>
</tr>
</tbody>
</table>

If you execute the mapping on a Windows operating system where you can already successfully consume the same Web service that you intend to call from MapForce, no additional certificate configuration is normally required (for the conditions to run the mapping successfully on Windows, see Client Certificates on Windows). However, if you design mappings with MapForce on a Windows computer, and then deploy them to another computer (which may run a different operating system), the client certificate is not stored or copied together with the deployed package. For the Web service call (and the mapping) to execute successfully, the client certificate must exist on the target operating system as well.

To transfer a certificate from a Windows system to another Windows-based computer, export the required certificate (with private key) from the source system (see Exporting Certificates from Windows). Then import the same certificate to the Current User\Personal store on the target operation system (see Client Certificates on Windows).

For instructions on how to transfer client certificates to the Linux and macOS platforms, see Client Certificates on Linux and Client Certificates on macOS, respectively.
6.1 Trusting Server Certificates on Linux

On Linux, you can import a trusted certificate into the system's certificate store as shown below.

Perform the following steps only if you are sure of the authenticity of the certificate you want to trust.

On Debian and Ubuntu, follow the steps below:

1. Copy the certificate file of the Web server to the following directory.
   
   ```bash
   sudo cp /home/downloads/server_cert.crt /usr/local/share/ca-certificates/
   ```

2. Update the certificate store as follows:
   
   ```bash
   sudo update-ca-certificates
   ```

On CentOS, follow the steps below:

1. Install the `ca-certificates` package:
   
   ```bash
   yum install ca-certificates
   ```

2. Enable the dynamic certificate authority configuration feature:
   
   ```bash
   update-ca-trust enable
   ```

3. Copy the server certificate to the following directory:
   
   ```bash
   cp server_cert.crt /etc/pki/ca-trust/source/anchors/
   ```

4. Use the command:
   
   ```bash
   update-ca-trust extract
   ```

For cases where you need to access the server only through the browser, it is sufficient to import the certificate into the browser certificate store. The exact instructions will vary for each browser. For example, in Firefox 59.0.2, you can do this as follows:

1. Under Options | Privacy & Security, click View Certificates.
2. On Authorities tab, click Import and browse for the root certificate file created previously.
3. When prompted, select Trust this CA to identify websites.
You have been asked to trust a new Certificate Authority (CA).

Do you want to trust "ODIN CA" for the following purposes?

☑ Trust this CA to identify websites.
☐ Trust this CA to identify email users.
☐ Trust this CA to identify software developers.

Before trusting this CA for any purpose, you should examine its certificate and its policy and procedures (if available).

View Examine CA certificate

Cancel OK
6.2 Trusting Server Certificates on macOS

On macOS, you can import a trusted certificate into Keychain Access as follows.

1. Run Keychain Access.
2. Click **System**, and then click **Certificates**.
3. On the **File** menu, click **Import Items**.
4. Browse for the trusted certificate, and click **Open**.
5. Enter the Keychain Access password when prompted, and then click **Modify Keychain**.
6. Double-click the certificate, expand the **Trust** section, and select **Always Trust**.

![Certificate Trust Settings](image)
6.3 Trusting Server Certificates on Windows

On Windows, you can import a trusted certificate into the system certificates store as follows:

1. Open the Windows certificate store for the computer account, see Accessing Windows Certificate Store.

Perform the following steps only if you are sure of the authenticity of the Web server certificate.

2. Under "Trusted Root Certification Authorities", right-click Certificates, and select All Tasks | Import, and follow the certificate import wizard.

For more information, see https://technet.microsoft.com/en-us/library/cc754489(v=ws.11).aspx.
6.4 Accessing the Certificate Stores on Windows

On Windows, you can manage certificates from the Microsoft Management Console (MMC) snap-in, either for your user account, or for the computer account.

To open the Certificates snap-in (for the current Windows user):

- Run `certmgr.msc` at the command line.

To open the Certificates snap-in (for the computer account):

1. Run `mmc` at the command line.
2. On the File menu, click Add/Remove Snap-in.
3. Click Certificates, and then click Add.

4. Click Computer account, and click Next.
5. Click **Local computer**, and then click **Finish**.
6.5 Exporting Certificates from Windows

For mappings that call Web services through HTTPS and are deployed to a macOS or Linux server running MapForce Server or FlowForce Server, the same client certificate must be available on the non-Windows operating system as the one used on Windows to design and test the mapping. To execute such mappings on a non-Windows operating system with MapForce Server, export the required certificate with private key from Windows and then import it into the target operating system.

To export a certificate with private key from Windows:

1. On Windows, open the Certificates snap-in (see Accessing the Certificate Stores on Windows).
2. Right-click the certificate that you want to export, point to All Tasks, and then click Export.
3. Click Next.

4. Choose to export from Windows the certificate together with its private key, and then click Next.
5. Choose the *Personal Information Exchange - PKCS #12 (.pfx)* file format, and then click **Next**.
Note: Make sure not to select the option **Delete the private key if the export is successful**, otherwise you will not be able to make use of the certificate after it is exported.

6. Enter a password, and then click **Next**. You will need this password after you copy the certificate to the target operating system.
7. Browse for the location of the file to export, and then click **Next**.
8. Click **Finish**.
Completing the Certificate Export Wizard

You have successfully completed the Certificate Export wizard.

You have specified the following settings:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Setting Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>C:\Username</td>
</tr>
<tr>
<td>Export Keys</td>
<td>Yes</td>
</tr>
<tr>
<td>Include all certificates in the certification path</td>
<td>Yes</td>
</tr>
<tr>
<td>File Format</td>
<td>Personal</td>
</tr>
</tbody>
</table>
6.6 Client Certificates on Linux

If your mappings include Web service authentication through HTTPS by means of client certificates, follow these steps to deploy such mappings to a Linux machine running MapForce Server:

1. Open in MapForce the mapping which calls the Web service.
2. Double-click the header of the Web Service component. The Component Settings dialog box appears.
3. Click Edit next to HTTP Security Settings.
4. In the HTTP Security Settings dialog box, click Client Certificate, and then select the required certificate from the Current User\Personal store on Windows.
5. Save the mapping and compile it to a mapping execution file or deploy it to FlowForce Server.
6. Transfer the client certificate required by the Web service call to the target operating system. Make sure that the certificate has a private key, and that the Enhanced Key Usage property of the certificate includes "Client authentication" as purpose.

To transfer the client certificate to Linux:

1. Export the client certificate with private key from Windows, in the Personal Information Exchange - PKCS #12 (.pfx) file format (see Exporting Certificates from Windows).
2. Copy the certificate file to the Linux machine.
3. Convert the .pfx file to .pem format using the command:

   ```bash
   openssl pkcs12 -in cert.pfx -out "John Doe.pem" -nodes
   ```

   This command parses the .pfx file and outputs a .pem file, without encrypting the private key. Certificates with an encrypted private key prompt for password and are not supported in server execution.

Executing the mapping

To instruct MapForce Server to use the .pem file as client certificate, set the --certificatespath parameter when running the mapping. The --certificatespath parameter defines the path of the directory where all certificates required by the current mapping are stored. For example, if the certificate file path is /home/John/John Doe.pem, then --certificatespath must be set to /home/John.

By default, if the --certificatespath parameter is not provided, MapForce Server looks for certificates in the directory $HOME/.config/altova/certificates of the current user.

For the mapping to execute successfully, the certificate file is expected to have the .pem extension and the file name must match the Common Name (CN) of the certificate, including spaces (for example, John Doe.pem). If the CN contains a forward slash (/), it must be replaced with an underscore (_) character.

If you intend to execute the mapping as a FlowForce Server job, copy the certificate file to the $HOME/.config/altova/certificates directory. When running the job, FlowForce Server will use this directory to look for any certificate files required by the mapping.

For security considerations, make sure that certificate files are not readable by other users, since they...
contain sensitive information.
6.7 Client Certificates on macOS

If your mappings include Web service authentication through HTTPS client certificates, follow these steps to deploy such mappings to a macOS running MapForce Server:

1. Open in MapForce the mapping which calls the Web service.
2. Double-click the header of the Web Service component. The Component Settings dialog box appears.
3. Click Edit next to HTTP Security Settings.
4. In the HTTP Security Settings dialog box, click Client Certificate, and then select the required certificate.
5. If the certificate name does not match exactly the host name of the server, select Allow name mismatch between certificate and request.
6. Save and deploy the mapping to the target operating system.
7. Transfer the client certificate required by the Web service call to the target operating system. Make sure that the certificate has a private key, and that the Enhanced Key Usage property of the certificate includes "Client authentication" as purpose.

To transfer the client certificate to macOS:

1. Export the client certificate with private key from Windows, in the Personal Information Exchange - PKCS #12 (.pfx) file format (see Exporting Certificates from Windows) and copy the .pfx file to the macOS.
2. If this hasn't been done already, make sure that the operating system trusts the server certificate (see Trusting Server Certificates on Mac OS).
4. On the File menu, click Import Items.
5. Browse for the client certificate exported from Windows in step 1 and select a destination keychain.
6. Click Open and enter the password with which the certificate was encrypted.

Executing the mapping

You are now ready to run the mapping using the MapForce Server run command. Note the following:

- If you execute the mapping remotely through SSH, first unlock the keychain with the security unlock-keychain command.
- If you execute the mapping through the macOS graphical user interface, when prompted to allow MapForce Server access to the keychain, click Allow.
6.8 Client Certificates on Windows

When you run on Windows a mapping which requires client certificates, the conditions to run the mapping successfully are as follows:

- The client certificate must exist in the Current User\Personal certificate store (also referred to as the My store). For the certificate to exist in this store, it must be imported through the Certificate Import Wizard. For instructions, see https://technet.microsoft.com/en-us/library/cc754489(v=ws.11).aspx.
- The certificate must have a private key.
- The Enhanced Key Usage property of the certificate must include "Client authentication" as purpose.

In the current version of MapForce, due to a limitation of the library used by MapForce, Windows will select the required certificate automatically from the certificate store when you run the mapping. The mapping will execute successfully if, after filtering the Current User\Personal certificate store, the server finds a suitable certificate. Note that the HTTPS authentication (and the certificate selection operation) is managed by Windows and is not controlled by MapForce or MapForce Server. In some cases, if multiple certificates exist in the Current User\Personal store, an unsuitable certificate may be selected automatically by the operating system, which causes the mapping execution to fail. This situation can be avoided by limiting the number of certificates.
available in the **Current User\Personal** store.
7 XBRL Taxonomy Manager

XBRL Taxonomy Manager is a tool that provides a centralized way to install and manage XBRL taxonomies for use across all Altova XBRL-enabled applications, including MapForce Server. On Windows, XBRL Taxonomy Manager has a graphical user interface and is also available at the command line. On Linux and Mac*, the tool is available at the command line only.

* The Linux and macOS operating systems are applicable only if you are running XBRL Taxonomy Manager on those operating systems in conjunction with Altova cross-platform server applications such as MapForce Server, StyleVision Server, or RaptorXML+XBRL Server.

XBRL Taxonomy Manager provides the following features:

- View XBRL taxonomies installed on your computer, and check whether new versions are available for download.
- Download newer versions of XBRL taxonomies independently of the Altova product release cycle. All taxonomies are maintained by Altova on an online-based storage accessible to XBRL Taxonomy Manager, and you can download them as soon as they become available.
- Install or uninstall any of the multiple versions of a given taxonomy (or all versions if necessary).
- A single XBRL taxonomy represents a “package” but it may have dependencies on other taxonomies. Whenever you choose to install or uninstall a particular taxonomy, any dependent taxonomies are detected and also installed or removed automatically. The graphical user interface (or the command line if applicable) informs you when dependencies are being added or removed.
- XBRL taxonomies maintained through XBRL Taxonomy Manager benefit from the XML catalog mechanism that enables URI references in instance or schema documents to be resolved from local files, as opposed to being retrieved from the Internet. This is extremely important in the case of big XBRL taxonomies where schema resolution from remote URIs is not practical or even recommended, mainly for performance reasons.

XBRL Taxonomy Manager provides a way to administer any of the XBRL taxonomies required for use in any one of the Altova XBRL-enabled applications. These include the European Banking Authority Reporting Framework taxonomies, US-GAAP Financial Reporting taxonomies, and various other country- or domain-specific XBRL taxonomies. To view the full list, either run XBRL Taxonomy Manager or run the `list` command at the command line, see also View Installed Packages.

Custom XBRL Taxonomies

If you need to work with custom XBRL taxonomies that are not included with XBRL Taxonomy Manager, you can enable those as follows:

- From your Altova XBRL-enabled desktop application, run the Tools | Options menu command, and then go to the XBRL | Taxonomy Packages settings page. From the settings page, browse for the .zip package of your custom XBRL taxonomy.
- From your Altova XBRL-enabled server application, provide the `--taxonomy-package` or `--taxonomy-package-config-file` options when running commands or API methods that support them. For example, these options are supported by XBRL validation commands such as `valxbrl` or `valxbrltaxonomy` in RaptorXML+XBRL Server, or by the `run` command (or the equivalent API method) in MapForce Server.
Altova XBRL-enabled applications

The following Altova applications are XBRL-enabled and thus benefit from the features provided by XBRL Taxonomy Manager:

- Altova XBRL Add-ins for Excel (EBA, Solvency II)
- MapForce Enterprise Edition
- MapForce Server
- MapForce Server Advanced Edition
- RaptorXML+XBRL Server
- StyleVision Server
- StyleVision Enterprise Edition
- XMLSpy Enterprise Edition

Changes in XBRL taxonomies using XBRL Taxonomy Manager take effect for all the applications listed above if they are installed on the same computer.

Installation

XBRL Taxonomy Manager is installed automatically whenever you install any Altova XBRL-enabled application or the Altova Mission Kit Enterprise Edition. Likewise, it is removed automatically when you uninstall the last Altova XBRL-enabled application from your computer or the Altova Mission Kit Enterprise Edition.

How it works

Altova maintains an online taxonomy storage where all XBRL taxonomies used in Altova products are stored. This taxonomy storage is updated on a periodical basis, for example, shortly after new versions of relevant taxonomies are released by their issuing organizations.

Whenever you run XBRL Taxonomy Manager at the graphical user interface, information about the latest available taxonomies is displayed in a dialog box, where you can view, install, upgrade or uninstall them. You can also perform the same actions at the command line interface. Note that, on Linux and macOS, the command line interface is the only means to interact with XBRL Taxonomy Manager.
You may also install taxonomies by running .altova_taxonomies files downloaded from the Altova website (https://www.altova.com/taxonomy-manager). In addition, when you take an action that requires loading XBRL taxonomies, MapForce Server prompts you to install the missing XBRL taxonomies.

Regardless of the manner in which taxonomies were installed, all information about installed taxonomies is tracked in a centralized location on your computer, also known as the local cache directory. The local cache directory contains information about Altova packages (except for the actual taxonomy files, which are installed on demand). The local cache directory is at the following path:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>/var/opt/Altova/pkgs</td>
</tr>
<tr>
<td>macOS</td>
<td>/var/Altova/pkgs</td>
</tr>
</tbody>
</table>
The local cache directory gets updated automatically from time to time, so as to propagate the latest state of the online storage to the local computer. More specifically, the cache is updated as follows:

- When you run the XBRL Taxonomy Manager.
- When you run MapForce Server for the first time in the same calendar day.
- If MapForce Server is already running, the cache directory gets updated every 24 hours.
- You can also update the local cache from the online storage on demand, by running the update command at the command line interface.

As you install or uninstall taxonomies, the local cache directory gets automatically updated with information about the available and installed taxonomies, as well as the taxonomy files themselves.

The local cache directory is maintained automatically based on the taxonomies you install or uninstall; it should not be altered or deleted manually. If you ever need to reset XBRL Taxonomy Manager to the original “pristine” state, run the reset command of the command line interface, and then run the initialize command. (Alternatively, run the reset command with the -i option.)
7.1 Run XBRL Taxonomy Manager

You can run XBRL Taxonomy Manager by using the approaches listed below, as applicable.

Graphical user interface (Windows only)
If MapForce Server runs on Windows, you can run XBRL Taxonomy Manager as follows:

- Run (double-click) a file with .altova_taxonomies extension downloaded from the Altova website.
- From the Windows Control Panel, right-click the Altova Taxonomy Manager entry and select Change or Uninstall from the context menu.

In addition, the check box Invoke Altova Taxonomy Manager is available on the last page of the installation wizard, after you complete the installation of MapForce Server.

Command line interface (Windows)
To run XBRL Taxonomy Manager from a command line interface:

1. Open a command prompt window and change directory to C:\ProgramData\Altova\SharedBetweenVersions.
2. To display help at the command line, run:

   TaxonomyManager.exe --help

Command line interface (Linux, macOS)
To run XBRL Taxonomy Manager from a command line interface:

1. Open a terminal window and change directory to %INSTALLDIR%/bin, where %INSTALLDIR% is the program's installation directory, see Important Paths.
2. To display help at the command line, run:

   sudo ./taxonomymanager --help
7.2 Apply Patches

Occasionally, XBRL taxonomies may receive patches from their issuers. When the XBRL Taxonomy Manager detects that patches are available, the following happens:

- If you use XBRL Taxonomy Manager through the Windows graphical user interface, the respective XBRL taxonomies are shown with the ❗ icon.
- If you use the command line or a Linux/macOS system, any XBRL taxonomies that have patches are listed when you run the executable with the `list -u` command.

To apply a patch on Windows:

1. Click the **Patch Selection** button. The icon of each XBRL taxonomy that qualifies changes from ❗ to ✅, and the dialog box informs you about the patches that are to be applied, for example:
**Note:** The Patch Selection button is enabled only when there are patches available for any of the currently installed XBRL taxonomies.

2. Click **Apply**.

**To apply a patch at the command line interface:**

1. Run the `list -u` command. This lists any taxonomies where patch upgrades are available.
2. Run the `upgrade` command to install the patches.

For more information, see the reference to the [Command Line Interface](#).
7.3 Install a Taxonomy

You can install a taxonomy by using the approaches listed below, as applicable.

Note the following:

- Installing or uninstalling a taxonomy from XBRL Taxonomy Manager takes effect for all users accounts on the same computer.
- Installing or uninstalling a taxonomy from XBRL Taxonomy Manager takes effect in all Altova XBRL-enabled applications installed on the same computer.
- If the current taxonomy has dependencies on other taxonomies, the dependent taxonomies are also installed (or uninstalled, as applicable).

Graphical user interface (Windows only)

To install a taxonomy:

1. Run XBRL Taxonomy Manager.
2. Select the check box next to the taxonomies or taxonomy versions you want to install, and click Apply.

Alternatively, if you have downloaded a file with .altova_taxonomies extension from the Altova website (https://www.altova.com/taxonomy-manager), double-click the .altova_taxonomies file to run it. XBRL Taxonomy Manager opens when you run the .altova_taxonomies file.

Command line interface (Windows)

To install a taxonomy, run:

```
TaxonomyManager.exe install FILTER...
```

Where FILTER means one of the following:

1. A taxonomy identifier in the format <name>-<version>, for example: `eba-2.10, us-gaap-2020.0`. To view all the available taxonomy identifiers and versions, run the list command.
2. An .altova_taxonomies file downloaded from the Altova website.

Command line interface (Linux, macOS)

To install a taxonomy, run:

```
sudo ./taxonomymanager install FILTER...
```

Where FILTER means one of the following:

1. A taxonomy identifier in the format <name>-<version>, for example: `eba-2.10, us-gaap-2020.0`. To view all the available taxonomy identifiers and versions, run the list command.
2. An .altova_taxonomies file downloaded from the Altova website.
Installing taxonomies on demand
Whenever MapForce Server detects that certain XBRL taxonomies must be installed on your computer, you may be prompted to install taxonomies on demand. For example, if you run a command that requires loading XBRL taxonomies, and if no XBRL taxonomies are currently installed, the command is cancelled and an error message similar to the following appears:

Unable to load a schema with target namespace [...] from [...].
Details:
I/O Error: File [...] is part of missing taxonomy [...]. Call 'taxonomymanager install [...]'.

To install the missing taxonomy, run the install command as suggested by the error message. You can always view all of the previously installed taxonomies by running the list command.

Applying patches
Occasionally, XBRL taxonomies may receive patches from their issuers. When the XBRL Taxonomy Manager detects that patches are available, the following happens:

- If you use XBRL Taxonomy Manager through the Windows graphical user interface, the respective XBRL taxonomies are shown with the icon.
- If you use the command line or a Linux/macOS system, any XBRL taxonomies that have patches are listed when you run the executable with the list -u command.

To apply a patch on Windows:

1. Click the Patch Selection button. The icon of each XBRL taxonomy that qualifies changes from to , and the dialog box informs you about the patches that are to be applied, for example:
Note: The Patch Selection button is enabled only when there are patches available for any of the currently installed XBRL taxonomies.

2. Click Apply.

To apply a patch at the command line interface:

1. Run the list -u command. This lists any taxonomies where patch upgrades are available.
2. Run the upgrade command to install the patches.

For more information, see the reference to the Command Line Interface.
7.4 View Installed Taxonomies

You can view the currently installed taxonomies by using the approaches listed below, as applicable.

Graphical user interface (Windows only)
To view all installed taxonomies from a graphical user interface, run XBRL Taxonomy Manager. A selected check box next to a taxonomy (or a taxonomy version) indicates that that taxonomy is installed.

Command line interface (Windows)
To view all available taxonomies from a command line interface, run:

```
TaxonomyManager.exe list
```

To view only installed taxonomies, run:

```
TaxonomyManager.exe list -i
```

To view only taxonomies where a newer version is available, run:

```
TaxonomyManager.exe list -u
```

Command line interface (Linux, macOS)
To view all available taxonomies, run:

```
sudo ./taxonomymanager list
```

To view only installed taxonomies, run:

```
sudo ./taxonomymanager list -i
```

To view only taxonomies where a newer version is available, run:

```
sudo ./taxonomymanager list -u
```
7.5 Uninstall a Taxonomy

You can uninstall a taxonomy by using the approaches listed below, as applicable.

Graphical user interface (Windows only)
To uninstall a taxonomy:

1. Run XBRL Taxonomy Manager.
2. Clear the check box next to the taxonomies or taxonomy versions you want to uninstall. If the selected taxonomy is dependent on other taxonomy packages, a dialog box opens, informing you that the dependencies will be removed as well, for example:

   ![Package Dependencies](image)

   The following packages depend on the selected package and therefore have to be removed too:

   US GAAP 2020 - Financial reporting taxonomy provided by the US Security and Exchange Commission

3. Click Apply to remove the taxonomy and its dependencies.

Command line interface (Windows)
To uninstall a taxonomy, run:

```
TaxonomyManager.exe uninstall FILTER...
```

Where `FILTER` means one of the following:

1. A taxonomy identifier in the format `<name>-<version>`, for example: `eba-2.10, us-gaap-2020.0`. To view all the available taxonomy identifiers and versions, run the `list` command.
2. An `.altova_taxonomies` file downloaded from the Altova website.

Command line interface (Linux, macOS)
To uninstall a package, run:

```
sudo ./taxonomymanager uninstall FILTER...
```

Where `FILTER` means one of the following:

1. A taxonomy identifier in the format `<name>-<version>`, for example: `eba-2.10, us-gaap-2020.0`. To view all the available taxonomy identifiers and versions, run the `list` command.
2. An `.altova_taxonomies` file downloaded from the Altova website.
7.6 Command Line Interface

To call XBRL Taxonomy Manager at the command line, you need to know the path of the executable. By default, the XBRL Taxonomy Manager executable is installed at the following path:

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux*</td>
<td>/opt/Altova/MapForceServer2021/bin/taxonomymanager</td>
</tr>
<tr>
<td>macOS*</td>
<td>/usr/local/Altova/MapForceServer2021/bin/taxonomymanager</td>
</tr>
<tr>
<td>Windows</td>
<td>C:\ProgramData\Altova\SharedBetweenVersions\TaxonomyManager.exe</td>
</tr>
</tbody>
</table>

* The Linux and macOS paths are applicable only if you are running XBRL Taxonomy Manager on those operating systems in conjunction with Altova cross-platform server applications such as MapForce Server, StyleVision Server, or RaptorXML+XBRL Server.

By convention, this documentation omits the full path of the executable when describing a given command, and uses `<exec>` instead of the executable name, for example:

```
<exec> help
```

Where `<exec>` is the path or name of the executable.

Calling XBRL Taxonomy Manager from the installation directory

To call the executable without having to type the full path, change the current directory to the one below:

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>cd /opt/Altova/MapForceServer2021/bin</td>
</tr>
<tr>
<td>macOS</td>
<td>cd /usr/local/Altova/MapForceServer2021/bin</td>
</tr>
<tr>
<td>Windows</td>
<td>cd &quot;C:\ProgramData\Altova\SharedBetweenVersions&quot;</td>
</tr>
</tbody>
</table>

You can now run a command by calling the executable with a relative path, for example:

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>sudo ./taxonomymanager help</td>
</tr>
<tr>
<td>macOS</td>
<td>sudo ./taxonomymanager help</td>
</tr>
<tr>
<td>Windows</td>
<td>TaxonomyManager.exe help</td>
</tr>
</tbody>
</table>

**Note:** On Linux and macOS systems, the prefix `./` indicates that the executable is in the current directory. The prefix `sudo` indicates that the command must be run with root privileges.

Calling XBRL Taxonomy Manager from any directory

To call the executable from any directory, refer to it using the absolute path. Alternatively, if you want to call the program by typing just the executable name, you can edit the PATH environment variable of your operating
system so that it includes the full path to the executable's directory. For ways to change the PATH environment variable, refer to the documentation of your operating system.

Notes:

- After changing the PATH environment variable, you may need to close the terminal window and open a new one, in order for the changes to take effect.
- On Linux and macOS, using `sudo` does not take into account the user's PATH.

Command line syntax
The general syntax for using the command line is as follows:

```
<exec> -h | --help | --version | <command> [options] [arguments]
```

In the listing above, the vertical bar `|` separates a set of mutually exclusive items. The square brackets `[]` indicate optional items. Essentially, you can type the executable path followed by either `--h`, `--help`, or `--version` options, or by a command. Each command may have options and arguments. The list of commands is described in the following sections.

### 7.6.1 help

This command provides contextual help about commands pertaining to XBRL Taxonomy Manager executable.

**Syntax**

```
<exec> help [command]
```

Where `[command]` is an optional argument which specifies any valid command name.

**Remarks**

You can also invoke help by typing a command followed by `-h` or `--help`, for example:

```
<exec> list -h
```

You can also invoke general help by typing `-h` or `--help` directly after the executable, for example:

```
<exec> --help
```

**Example**

The following command displays help about the `list` command:

<table>
<thead>
<tr>
<th>Linux, macOS</th>
<th><code>./taxonomymanager help list</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td><code>TaxonomyManager.exe help list</code></td>
</tr>
</tbody>
</table>
7.6.2 info

This command displays detailed information for each of the taxonomies supplied as argument. The detailed information includes the title, version, description, publisher, and dependency references.

Syntax

<exec> info [options] FILTER...

To specify multiple taxonomies, repeat FILTER as many times as necessary.

Arguments

<table>
<thead>
<tr>
<th>FILTER</th>
<th>List only taxonomies that contain this string in their name (for example, eba or us-gaap-2020.0). You can specify this argument multiple times.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>To view the full list of identifiers, run the list command.</td>
</tr>
</tbody>
</table>

Options

|--help, --h | Display help about this command at the command line. |

Example

The following command displays detailed information about the eba-2.10 and us-gaap-2020.0 taxonomies:

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux, macOS</td>
<td>./taxonomymanager info eba-2.10 us-gaap-2020.0</td>
</tr>
<tr>
<td>Windows</td>
<td>TaxonomyManager.exe info eba-2.10 us-gaap-2020.0</td>
</tr>
</tbody>
</table>

7.6.3 initialize

This command initializes XBRL Taxonomy Manager environment. It creates a cache directory where information about all taxonomies is stored. Initialization is performed automatically the first time when you install an Altova application that includes supports for XBRL Taxonomy Manager, so you don't need to run this command under normal circumstances. You typically need to run this command after executing the reset command.

Syntax

<exec> initialize [options]

The alias of this command is init.
7.6.4 install

This command installs one or more taxonomies. Note the following:

- Installing or uninstalling a taxonomy from XBRL Taxonomy Manager takes effect for all users accounts on the same computer.
- Installing or uninstalling a taxonomy from XBRL Taxonomy Manager takes effect in all Altova XBRL-enabled applications installed on the same computer.
- If the current taxonomy has dependencies on other taxonomies, the dependent taxonomies are also installed (or uninstalled, as applicable).

Syntax

```<exec> install [options] FILTER...```

To specify multiple taxonomies to install, repeat `FILTER` as many times as necessary.

Arguments

<table>
<thead>
<tr>
<th>FILTER</th>
<th>Where <code>FILTER</code> means one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1. A taxonomy identifier in the format <code>&lt;name&gt;-&lt;version&gt;</code>, for example: <code>eba-2.10</code>, <code>us-gaap-2020.0</code>. To view all the available taxonomy identifiers and versions, run the <code>list</code> command.</td>
</tr>
<tr>
<td></td>
<td>2. An <code>.altova_taxonomies</code> file downloaded from the Altova website.</td>
</tr>
<tr>
<td></td>
<td>You can also use abbreviated identifiers if they are unique, for example, <code>eba</code>. If you use an abbreviated identifier, this will install the latest available version of that taxonomy.</td>
</tr>
</tbody>
</table>

Options

| --help, --h | Display help about this command at the command line. |
| --silent, --s | Display only error messages. The default value is `false`. |
| --verbose, --v | Display more information during execution. The default value is `false`. |
Example
The following command installs the latest **eba** (European Banking Authority) and **us-gaap** (US Generally Accepted Accounting Principles) taxonomies:

<table>
<thead>
<tr>
<th>Linux, macOS</th>
<th>./taxonomymanager install eba us-gaap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>TaxonomyManager.exe install eba us-gaap</td>
</tr>
</tbody>
</table>

### 7.6.5 list

Use this command to list taxonomies at the command line, in one of the following ways:

- list all available taxonomies
- list specific taxonomies
- list only installed taxonomies
- list only taxonomies that require upgrade.

**Syntax**

```
<exec> list [options] [FILTER...]
```

This command can be abbreviated with `ls`.

**Arguments**

<table>
<thead>
<tr>
<th>FILTER</th>
<th>List only taxonomies that contain this string in their name. You can specify this argument multiple times.</th>
</tr>
</thead>
</table>

**Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--help, --h</td>
<td>Display help about this command at the command line.</td>
</tr>
<tr>
<td>--installed, --i</td>
<td>List only installed taxonomies. The default value is false.</td>
</tr>
<tr>
<td>--upgradeable, --u</td>
<td>List only taxonomies where patch upgrades are available. The default value is false.</td>
</tr>
</tbody>
</table>

**Examples**

To list all available taxonomies, run:

<table>
<thead>
<tr>
<th>Linux, macOS</th>
<th>./taxonomymanager list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>TaxonomyManager.exe list</td>
</tr>
</tbody>
</table>

To list only installed taxonomies, run:
To list all taxonomies that contain either "eba" or "us-gaap" in their name, run:

<table>
<thead>
<tr>
<th></th>
<th>Linux, macOS</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>./taxonomymanager list -i</td>
<td>TaxonomyManager.exe list -i</td>
</tr>
</tbody>
</table>

7.6.6     reset

This command removes all installed taxonomies and the cache directory.

**Warning:** This command deletes all installed taxonomies and their information.

After running this command, make sure to run the initialize command, in order to recreate the cache directory. Alternatively, run the reset command with the -i option.

Note that reset -i restores the original installation of the product, so it's recommended to run the update command as well, after performing a reset. Alternatively, run the reset command with the -i and -u options.

**Syntax**

```bash
<exec> reset [options]
```

**Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h, --help</td>
<td>Display help about this command at the command line.</td>
</tr>
<tr>
<td>-i, --init</td>
<td>Initialize the XBRL Taxonomy Manager environment after reset. Valid values are true and false. The default value is false.</td>
</tr>
<tr>
<td>-s, --silent</td>
<td>Display only error messages. The default value is false.</td>
</tr>
<tr>
<td>-u, --update</td>
<td>Initialize and update the XBRL Taxonomy Manager environment after reset. Valid values are true and false. The default value is false.</td>
</tr>
<tr>
<td>-v, --verbose</td>
<td>Display additional information during execution. The default value is false.</td>
</tr>
</tbody>
</table>

**Examples**

To reset the XBRL Taxonomy Manager, run:

<table>
<thead>
<tr>
<th></th>
<th>Linux, macOS</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>./taxonomymanager reset</td>
<td>TaxonomyManager.exe reset</td>
</tr>
</tbody>
</table>
### 7.6.7  setdeflang

This command sets the language of XBRL Taxonomy Manager.

**Syntax**

```
<exec> setdeflang language
```

Where `language` is a mandatory argument supplying the language code. The alias of this command is `sdl`.

**Arguments**

| language | The language to be set. Valid values are English (`en`), French (`fr`), German (`de`), Japanese (`ja`), and Spanish (`es`). |

**Options**

```
--help, --h
```

Display help about this command at the command line.

**Examples**

To set the language to Spanish, run:

```
Linux, macOS  ./taxonomymanager setdeflang es
Windows    TaxonomyManager.exe setdeflang es
```

### 7.6.8  uninstall

This command uninstalls one or more taxonomies. By default, any taxonomies referenced by the current one are uninstalled as well. To uninstall just the current taxonomy and keep the referenced taxonomies, set the option `--k`.

**Syntax**

```
<exec> uninstall FILTER...
```

To specify multiple taxonomies, repeat `FILTER` as many times as necessary.

**Arguments**

| FILTER | Where `FILTER` means one of the following: |
1. A taxonomy identifier in the format `<name>-<version>`, for example: `eba-2.10, us-gaap-2020.0`. To view all the available taxonomy identifiers and versions, run the `list` command.

2. An `.altova_taxonomies` file downloaded from the Altova website.

### Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--help, --h</code></td>
<td>Display help about this command at the command line.</td>
</tr>
<tr>
<td><code>--keep-references, --k</code></td>
<td>If this option is set, then referenced taxonomies are not uninstalled. The default value is <code>false</code>.</td>
</tr>
<tr>
<td><code>--silent, --s</code></td>
<td>Display only error messages. The default value is <code>false</code>.</td>
</tr>
<tr>
<td><code>--verbose, --v</code></td>
<td>Display additional information during execution. The default value is <code>false</code>.</td>
</tr>
</tbody>
</table>

### Example

The following command uninstalls the `eba-2.10` and `us-gaap-2020.0` taxonomies and their dependencies:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux, macOS</td>
<td><code>./taxonomymanager uninstall eba-2.10 us-gaap-2020.0</code></td>
</tr>
<tr>
<td>Windows</td>
<td><code>TaxonomyManager.exe uninstall eba-2.10 us-gaap-2020.0</code></td>
</tr>
</tbody>
</table>

### 7.6.9 update

This command queries the list of taxonomies available from the online storage and updates the local cache directory. The update of this information takes place implicitly and you shouldn't need to run this command unless you have performed a reset and initialize.

#### Syntax

```
<exec> update [options]
```

#### Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--help, --h</code></td>
<td>Display help about this command at the command line.</td>
</tr>
<tr>
<td><code>--silent, --s</code></td>
<td>Display only error messages. The default value is <code>false</code>.</td>
</tr>
<tr>
<td><code>--verbose, --v</code></td>
<td>Display additional information during execution. The default value is <code>false</code>.</td>
</tr>
</tbody>
</table>

#### Example

The following command updates the taxonomies information explicitly.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux, macOS</td>
<td><code>./taxonomymanager update</code></td>
</tr>
</tbody>
</table>
7.6.10 upgrade

This command upgrades all eligible taxonomies to the latest available patch version. In other words, it performs only upgrades at patch level of a specific release. Running this command is meaningful only if there are upgradeable taxonomies available. You can identify upgradeable taxonomies by running the `list -u` command.

Syntax

```
<exec> upgrade [options]
```

Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--help, --h</td>
<td>Display help about this command at the command line.</td>
</tr>
<tr>
<td>--silent, --s</td>
<td>Display only error messages. The default value is false.</td>
</tr>
<tr>
<td>--verbose, --v</td>
<td>Display additional information during execution. The default value is false.</td>
</tr>
</tbody>
</table>
8 Catalog Files

MapForce Server supports a subset of the OASIS XML catalogs mechanism (https://www.oasis-open.org/committees/entity/spec-2001-08-06.html). The catalog mechanism enables MapForce Server to retrieve commonly used DTDs and XML schemas (as well as stylesheets and other files) from local folders instead of resolving them from a public URI. This increases the overall processing speed, enables you to work offline (that is, not connected to a network), and improves the portability of documents.

How catalogs work

Catalogs are commonly used to redirect a public DTD or schema reference to a local URI (typically, a local file path). To achieve this, a catalog file in XML format defines a mapping between the public schema URI and a local URI. Whenever MapForce Server parses an XML document, it looks for the schema URI (or public or system identifier of a DTD, if applicable) inside the catalog file first. If a mapping is found in the catalog file, then that reference will be used and the schema will be read from a local file. If no mapping is found in the catalog file, then the URI of the XML document will be resolved as is.

For example, let’s suppose that the following XML file must be processed by MapForce Server.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Articles xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Articles.xsd">
  <Article>
    <Number>1</Number>
    <Name>T-Shirt</Name>
    <SinglePrice>25</SinglePrice>
  </Article>
</Articles>
```

Let’s also suppose that a catalog.xml file exists somewhere in a local directory (of which MapForce Server is aware, as further discussed below), and it contains the following line:

```xml
<catalog>
  <!--...-->
  <uri name="http://www.w3.org/2001/XMLSchema-instance.xsd" uri="files/XMLSchema-instance.xsd"/>
  <!--...-->
</catalog>
```

On parsing the XML file, MapForce Server will detect a match for the schema reference http://www.w3.org/2001/XMLSchema-instance.xsd in the catalog file. Consequently, the schema will be loaded from files/XMLSchema-instance.xsd (which is a local path relative to the catalog file). If no mapping were found in the catalog file, then the schema would be loaded from http://www.w3.org/2001/XMLSchema-instance.

Root catalog

When MapForce Server starts, it loads a file called RootCatalog.xml from the %ProgramFilesFolder%\Altova\MapForceServer2021\etc directory. RootCatalog.xml contains a list of catalog files, each in a nextCatalog element. These catalog files are looked up and the URIs in them are resolved by MapForce.
Server according to the mappings specified in them.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
   xmlns:spy="http://www.altova.com/catalog_ext"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:oasis:names:tc:entity:xmlns:xml:catalog Catalog.xsd">
  <nextCatalog catalog="CustomCatalog.xml"/>
  <nextCatalog catalog="CoreCatalog.xml"/>
  <!-- Include all catalogs under common schemas folder on the first directory level -->
  <nextCatalog spy:recurseFrom="Schemas" catalog="catalog.xml" spy:depth="1"/>
</catalog>
```

### RootCatalog.xml

In the listing above, notice that the following catalogs are listed for lookup:

- **CustomCatalog.xml** is the file in which you can create your own mappings. This file is in the `%ProgramFilesFolder%\Altova\MapForceServer2021\etc` directory. You can add mappings to CustomCatalog.xml for any custom schema that is not already addressed by the Altova-configured catalog files (see the next bullets).
- **CoreCatalog.xml** contains certain Altova-specific mappings for locating schemas. This file is also in the `%ProgramFilesFolder%\Altova\MapForceServer2021\etc` directory.
- Multiple `catalog.xml` files from the Schemas directory relative to the RootCatalog.xml file. Each `catalog.xml` file is inside the directory of a specific schema (such as SVG, DITA, DocBook, WSDL, and so on), and each maps public and/or system identifiers to URIs that point to locally saved copies of the respective schemas.

Note the following:

- If you intend to modify the CustomCatalog.xml, use only the Supported elements. Also, make sure not to duplicate the already existing mappings, as this could lead to errors.

### Supported elements

When creating entries in CustomCatalog.xml, use only the elements listed below. Other elements of the OASIS XML catalog specification are not supported.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>publicId specifies the public identifier of a resource&lt;br&gt;uri specifies a URI reference (for example, a relative path to a local file)</td>
<td><code>&lt;public publicId=&quot;-//W3C//DTD XMLSCHEMA 200102//EN&quot; uri=&quot;files/XMLSchema.dtd&quot;/&gt;</code></td>
</tr>
<tr>
<td>system</td>
<td>systemId specifies the system identifier of a resource&lt;br&gt;uri specifies a URI reference (for example, a relative path to a local file)</td>
<td><code>&lt;system systemId=&quot;http://www.w3.org/2009/XMLSchema/datatypes.dtd&quot; uri=&quot;files/datatypes.dtd&quot;/&gt;</code></td>
</tr>
<tr>
<td>Element</td>
<td>Attributes</td>
<td>Example</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>• uri specifies an alternate URI reference</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(for example, a relative path to a local file)</td>
<td></td>
</tr>
<tr>
<td>rewriteURI</td>
<td>• uriStartString specifies the starting part of</td>
<td><code>&lt;rewriteURI uriStartString='http://www.altova.com/schemas/svg/' rewritePrefix='files/'/&gt;</code></td>
</tr>
<tr>
<td></td>
<td>a URI to rewrite</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• uri specifies the replacement string (for</td>
<td></td>
</tr>
<tr>
<td></td>
<td>example, a relative path to a local directory)</td>
<td></td>
</tr>
<tr>
<td>rewriteSystem</td>
<td>• systemIdStartString specifies the starting</td>
<td><code>&lt;rewriteSystem systemIdStartString='http://www.altova.com/schemas/svg/' rewritePrefix='files/'/&gt;</code></td>
</tr>
<tr>
<td></td>
<td>part of a system identifier to rewrite</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• rewritePrefix specifies the replacement string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(for example, a relative path to a local directory)</td>
<td></td>
</tr>
</tbody>
</table>

The **public**, **system**, and **uri** elements can also take the `xml:base` attribute, which is used to specify a base URI with respect to which a relative URI would be resolved. For more information, see the XML Catalogs specification ([http://www.oasis-open.org/committees/entity/spec-2001-08-06.html](http://www.oasis-open.org/committees/entity/spec-2001-08-06.html)).
Index

A
ADO, 6, 20
ADO.NET, 6, 20
assignlicense,
   as CLI command, 52

C
C#,
   example, 66
   running mappings with, 66
C++,
   example, 72
   running mappings with, 72
CLI commands,
assignlicense, 52
exportresourcestrings, 53
help, 55
licenseserver, 56
run, 57
setdeflang, 61
verifylicense, 62, 63
Command Line Interface,
   how to use, 49

D
Digital certificates,
   exporting from Windows, 130
   in MapForce mappings, 121
   managing on Windows, 128
   transferring to Linux, 136
   transferring to Mac, 138
   trusting on Linux, 124
   trusting on Mac, 126
   trusting on Windows, 127

E
exportresourcestrings,
   as CLI command, 53

F
FlowForce Server, 6, 17, 20

G
Global Resources,
   definition, 26
   using in mappings, 26

H
help,
   as CLI command, 55
HTTPS,
   calling Web services through, 121

J
Java, 20
   adding MapForce Server to CLASSPATH, 81
   calling MapForce Server from, 83
   example, 83
JDBC, 6
Join optimization,
   how it works, 28

L
licenseserver,
   as CLI command, 56
Linux,
Linux,
executing mappings with Web service calls through HTTPS, 136
installing MapForce Server on, 10
support info, 7
transferring client certificates to, 136
trusting server certificates on, 124
uninstalling MapForce Server from, 11
viewing currently installed Altova products, 11

Mac,
executing mappings with Web service calls through HTTPS, 138
transferring client certificates to, 138
trusting server certificates on, 126
macOS,
installing MapForce Server on, 12
support info, 7
uninstalling MapForce Server from, 13

MapForce Server API,
C# example, 66
C++ example, 72
for .NET, 65
for COM, 72
for Java, 81
introduction, 64
Java example, 83
VB.NET example, 68
VBA example, 78
VBScript example, 75

Mappings,
compiling to execution files, 17
deploying to FlowForce Server, 17
preparing for execution, 20
running at the command line, 57
running with C#, 66
running with C++, 72
running with Java, 83
running with VB.NET, 68
running with VBA, 78
running with VBScript, 75

ODBC, 6, 20

run,
as CLI command, 57

Server cores,
requirements, 8
setdeflang,
as CLI command, 61

VB.NET,
example, 68
running mappings with, 68
VBA,
example, 78
running mappings with, 78
VBScript,
example, 75
running mappings with, 75
verifylicense,
as CLI command, 62
version,
as CLI command, 63

Web services,
calling through HTTPS, 121

Windows,
executing mappings with Web service calls through HTTPS, 139
Index

Windows,
   installing MapForce Server on, 14
   support info, 7
   trusting server certificates on, 127
   uninstalling MapForce Server from, 16

XML Catalogs,
   configuring, 162
   how it works, 162