# Table of Contents

## 1 Altova MapForce Server 2019 Advanced Edition
1.1 How It Works .......................................................... 9  
1.2 Preparing Mappings for Server Execution .................. 12  
1.3 Global Resources .................................................. 18  
1.4 Join Optimization .................................................. 19

## 2 Setup on Windows
2.1 Installation on Windows ........................................... 25  
2.2 Licensing on Windows ............................................... 27

## 3 Setup on Linux
3.1 Installation on Linux ............................................... 33  
3.2 Licensing on Linux .................................................. 36

## 4 Setup on macOS
4.1 Installation on macOS ............................................. 41  
4.2 Licensing on macOS ............................................... 44

## 5 MapForce Server Command Line
5.1 assignlicense (Windows only) ................................. 50  
5.2 exportresourcestrings ............................................. 52  
5.3 help ........................................................................ 54  
5.4 licenseserver .......................................................... 56  
5.5 run ........................................................................ 58  
5.6 setdeflang ................................................................ 62  
5.7 verifylicense (Windows only) ................................. 63

## 6 MapForce Server API

---

6.1 About the .NET Interface ................................................................. 67
6.2 About the COM Interface ............................................................... 69
6.3 About the Java Interface ................................................................. 70
6.4 Code Examples ............................................................................... 73
  6.4.1 C# ......................................................................................... 73
  6.4.2 C++ ....................................................................................... 74
  6.4.3 Java ....................................................................................... 76
  6.4.4 VBScript ............................................................................... 77
  6.4.5 Visual Basic ........................................................................... 78
  6.4.6 Visual Basic for Applications (VBA) .......................................... 80
6.5 API Reference (COM, .NET) ............................................................ 83
  6.5.1 Interfaces .............................................................................. 83
6.6 API Reference (Java) ...................................................................... 97
  6.6.1 Classes .................................................................................. 97

7 Digital Certificate Management ....................................................... 106
  7.1 Trusting Server Certificates on Linux ............................................ 109
  7.2 Trusting Server Certificates on macOS ......................................... 111
  7.3 Trusting Server Certificates on Windows ....................................... 112
  7.4 Accessing the Certificate Stores on Windows .............................. 113
  7.5 Exporting Certificates from Windows .......................................... 115
  7.6 Client Certificates on Linux ........................................................ 121
  7.7 Client Certificates on macOS ...................................................... 123
  7.8 Client Certificates on Windows ................................................... 125

Index .................................................................................................. 127
Chapter 1

Altova MapForce Server 2019 Advanced Edition
1 Altova MapForce Server 2019 Advanced Edition

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 intermediary 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.

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 6 or newer
- RedHat 6 or newer
- Debian 8 or newer
- Ubuntu 14.04 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 10.12 or newer

Note: On Windows and macOS, it is possible to install multiple versions of MapForce Server on the same machine. In such cases, be aware that only one version can be registered to the same Altova LicenseServer at a time. Hence, two different versions of MapForce Server cannot run concurrently on the same machine. Multiple instances of the same product version can run in parallel, however. For more information, see the Altova License Server documentation (https://www.altova.com/documentation#licenseserver).

Last updated: 29 April 2019
1.1 How It Works

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 intermediary 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 can roughly be regarded as 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.
How to execute mappings compiled as MapForce Server Execution files

1. Open in MapForce Enterprise or Professional the mapping to be compiled.
2. On the File menu, click Compile to MapForce Server Execution file, and select a destination directory.
3. Copy the .mfx file to the destination directory or server.
4. Call the "run" command of the command line interface (see Command Line Usage), or the equivalent method of the MapForce Server API (see 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 at a later time (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](https://www.altova.com/help/MapForce/Preparing-Mappings-for-Server-Execution).
1.2 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 and output files and schemas 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 on the “Options” dialog box in MapForce, for example:

![Options](image)

- 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. For example, you can copy all input or output files required by the mapping into the same directory as the mapping, and then reference them just by file name. 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><code>/opt/oracle/instantclient_11_2/ojdbc6.jar:/opt/oracle/instantclient_11_2/ojdbc5.jar</code></td>
</tr>
<tr>
<td>TNS_ADMIN</td>
<td><code>/opt/oracle/NETWORK_ADMIN</code></td>
</tr>
<tr>
<td>ORACLE_HOME</td>
<td><code>/opt/oracle/instantclient_11_2</code></td>
</tr>
<tr>
<td>DYLD_LIBRARY_PATH</td>
<td><code>/opt/oracle/instantclient_11_2</code></td>
</tr>
<tr>
<td>PATH</td>
<td><code>$PATH:/opt/oracle/instantclient_11_2</code></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 working directory. For an example, see "Exposing a Job as a Web Service" (https://www.altova.com/manual/FlowForceServer/FlowForceServerAdvanced/index.html?fs_example_web_service.htm).
- 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, such references are preserved when you compile the mapping to a server execution file (.mfx), or when you deploy the mapping to FlowForce Server. For information about running such mappings with MapForce Server, see Global Resources.

Note: FlowForce Server does not currently support Global Resources. Do not use Global Resources if you intend to execute the mapping with MapForce Server running under FlowForce Server management.

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 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.
1.3 Global Resources

Altova Global Resources represent a way to refer to files, folders, or databases so as to make these resources reusable, configurable and available across multiple Altova applications. 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 the MapForce mapping was designed and then reuse the same connection on the machine where MapForce Server runs the mapping. Global Resources have various other uses, including the ability to make the input or output file or folder paths configurable. For an introduction to Global Resources, refer to the "Global Resources" chapter of MapForce documentation.

If a mapping designed with MapForce uses Altova Global Resources, the same mapping can be run in MapForce Server, and the same Global Resources can be called from the machine where MapForce Server runs. This requires the following:

1. The file where the Global Resource is defined. By default, all Global Resources are defined in a file called `GlobalResources.xml`, which is available in the folder `Documents\Altova` folder on the machine where MapForce is installed. It is possible to create multiple Global Resource files if necessary, in order to make them portable, see also the MapForce documentation.

2. The Global Resource configuration name. Each Global Resource is identified by a configuration name. The default name is "Default" but you can create additional configurations if so required.

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:

  ```
  C:\Program Files (x86)\Altova\MapForceServer2019\bin\MapForceServer.exe run SomeMapping.mfx --globalresourcefile="C:\Users\me\Documents\Altova\GlobalResources.xml" --globalresourceconfig="Default"
  ```

- If you run the mapping through the MapForce Server API, call the method `SetOption` 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 (see also the Code Examples section).
1.4 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\MapForce2019\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).
2 Setup on Windows

This section describes the installation and licensing of MapForce Server on Windows systems.

Installation on Windows
- System requirements
- Installing MapForce Server
- Altova LicenseServer
- LicenseServer versions
- Trial license
- Application folder location

Licensing on Windows
- Start ServiceController
- Start LicenseServer
- Register MapForce Server
- Assign a license
2.1 Installation on Windows

MapForce Server is available for installation on Windows systems. Its installation and setup procedure is described below.

- **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

- **Installing MapForce Server**
  MapForce Server can be installed on Windows systems as follows:
  - As a separate standalone server product called MapForce Server. To install MapForce Server, download and run the MapForce Server installer. Follow the on-screen instructions.
  - As part of the FlowForce Server installation package. To install MapForce Server as part of the FlowForce Server package, download and run the FlowForce Server installer. Follow the on-screen instructions and make sure you check the option for installing MapForce Server.

The installers of both MapForce Server and FlowForce Server are available at the Altova Download Center (http://www.altova.com/download.html).

After installation, the MapForce Server executable will be located by default at:

```
<ProgramFilesFolder>\Altova\MapForceServer2019\bin\MapForceServer.exe
```

All the necessary registrations to use MapForce Server via a COM interface, as a Java interface, and in the .NET environment will be done by the installer.

- **Altova LicenseServer**
  - In order for MapForce Server to work, it must be licensed via an Altova LicenseServer on your network.
  - When you install MapForce Server or FlowForce Server on Windows systems, an option is available that allows you to download and install Altova LicenseServer together with MapForce Server or FlowForce Server.
  - If an Altova LicenseServer is already installed on your network, you do not need to install another one—unless a newer version of Altova LicenseServer is required. (See next point, LicenseServer versions.)
  - During the installation process of MapForce Server or FlowForce Server, check or uncheck the option for installing Altova LicenseServer as appropriate.

See the section, Licensing on Windows, for more information about how to register and
license MapForce Server with Altova LicenseServer.

- **LicenseServer versions**
  - Altova server products must be licensed either with the version of LicenseServer that is appropriate to the installed MapForce Server version, or with a later version of LicenseServer.
  - The LicenseServer version that is appropriate for a particular version of MapForce Server is displayed during the installation of MapForce Server. You can install this version of LicenseServer along with MapForce Server, or you can install LicenseServer separately.
  - Before installing a newer version of LicenseServer, any older one must be de-installed. The LicenseServer installer will do this automatically if it detects an older version.
  - LicenseServer versions are backwards compatible. They will work with older versions of MapForce Server.
  - If you install a new version of MapForce Server and if your installed LicenseServer version is older than the appropriate LicenseServer, install the latest version available on the Altova website.
  - At the time of LicenseServer de-installation, all registration and licensing information held in the older version of LicenseServer will be saved to a database on your server machine. This data will be imported automatically into the newer version when the newer version is installed.
  - The version number of the currently installed LicenseServer is given at the bottom of the LicenseServer configuration page (all tabs).

  **Current version: 3.2**

- **Trial license**
  During the installation process, you will be given the option of requesting a 30-day trial license for MapForce Server. After submitting the request, a trial license will be sent to the email address you registered.

- **Application folder location**
  The application will be installed in the following folder:

<table>
<thead>
<tr>
<th>Windows 7, 8, 10</th>
<th>C:\Program Files\Altova\</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 bit Version on 64-bit OS</td>
<td>C:\Program Files (x86)\Altova\</td>
</tr>
</tbody>
</table>
2.2 Licensing on Windows

MapForce Server must be licensed with Altova LicenseServer. Licensing is a two-step process:

1. **Register MapForce Server** with LicenseServer. Registration is done from MapForce Server.
2. **Assign a license** to MapForce Server from LicenseServer. Download the latest version of LicenseServer from the [Altova website](https://www.altova.com), and install it on your local machine or a machine on your network.

The steps to carry out are given below in brief. For detailed information, see the [LicenseServer user manual](https://www.altova.com) at the [Altova website](https://www.altova.com).

### Start ServiceController

Altova ServiceController is started in order to start Altova LicenseServer.

Altova ServiceController (ServiceController for short) is an application for conveniently starting, stopping and configuring Altova services on Windows systems.

ServiceController is installed with Altova LicenseServer and with *Altova server products that are installed as services* (FlowForce Server, RaptorXML(+XBRL) Server, and Mobile Together Server). It can be started by clicking **Start | Altova LicenseServer | Altova ServiceController**. (This command is also available in the **Start** menu folders of *Altova server products that are installed as services* (FlowForce Server, RaptorXML(+XBRL) Server, and Mobile Together Server).) After ServiceController has been started, it can be accessed via the system tray ([screenshot below](#)).

To specify that ServiceController starts automatically on logging in to the system, click the **ServiceController** icon in the system tray to display the **ServiceController** menu ([screenshot below](#)), and then toggle on the command **Run Altova ServiceController at Startup**. (This command is toggled on by default.) To exit ServiceController, click the **ServiceController** icon in the system tray and, in the menu that appears ([see screenshot below](#)), click **Exit Altova ServiceController**.

![ServiceController menu](#)
Setup on Windows
Licensing on Windows

▼ Start LicenseServer
To start LicenseServer, click the ServiceController icon in the system tray, hover over Altova LicenseServer in the menu that pops up (see screenshot below), and then select Start Service from the LicenseServer submenu. If LicenseServer is already running, the Start Service option will be disabled.

![Start LicenseServer](image)

▼ Register MapForce Server
To register MapForce Server from the command line interface, use the licenseserver command:

```
MapForceServer licenseserver [options] ServerName-Or-IP-Address
```

For example, if localhost is the name of the server on which LicenseServer is installed:

```
MapForceServer licenseserver localhost
```

If MapForce Server was installed as part of a FlowForce Server installation, registering FlowForce Server with LicenseServer will automatically also register MapForce Server. Essentially: (i) Start Altova FlowForce Web as a service via ServiceController (see previous point); (ii) Enter your password to access the Setup page; (iii) Select the LicenseServer name or address and click Register with LicenseServer. For more information, see Register FlowForce Server.

After successful registration, go to the Server Management tab of LicenseServer's configuration page to assign a license to MapForce Server.

▼ Assign a license
After successfully registering MapForce Server, it will be listed in the Server Management tab of the configuration page of LicenseServer. Go there and assign a license to MapForce Server.

The licensing of Altova server products is based on the number of processor cores available on the product machine. 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 (an octa-core processor), you must purchase at least one 8-core license. You can also combine licenses to achieve the core count. So, two 4-core licenses can also be used for an octa-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 on the server were utilized.

**Note:** Each Altova server product license can be used for only one client machine—the machine on which the Altova server product is installed—at a time, 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.

**Note:** FlowForce Server Advanced Edition and MapForce Server Advanced Edition will run only on machines with two or more cores.

When assessing the number of cores you should license, this decision should take into account the data volumes you need to process and the processing time your business environment is expected to allow for. In most scenarios, a larger number of cores means more volumes of data processed in less time. Below are just a few application-specific tips:

- FlowForce Server runs as a multi-threaded application. If the number of concurrent requests to the server is big, an insufficient number of cores will lead to latency (waiting times). For example, if you are exposing jobs as Web services, there may be hundreds of concurrent requests from clients. In this case, FlowForce Server will significantly benefit from a larger number of cores.

- MapForce Server will utilize a single core at a time, per mapping. Therefore, if you need to run multiple mappings simultaneously, a larger number of cores is highly recommended. For example, when MapForce Server runs under FlowForce Server management, several mapping jobs may overlap and run concurrently, depending also on the setup. Note, however, that if the volumes processed by your mappings are extremely big, latency could still occur.

In addition to the above, note that there are various external factors that typically influence the processing volumes and times that your server is capable to handle (for example, the hardware, the current load on the CPU, memory allocation of other applications running on the server). In order to get the most accurate performance measurements, it is generally advisable to first run the tools in your environment and expose them to actual factors and data specific to your business.
Chapter 3

Setup on Linux
3 Setup on Linux

This section describes the installation and licensing of MapForce Server on Linux systems (Debian, Ubuntu, CentOS, RedHat).

Installation on Linux

- System requirements
- Uninstall old versions of Altova server products
- Download the Linux package
- Install MapForce Server
- Altova LicenseServer
- LicenseServer versions

Licensing on Linux

- Start LicenseServer
- Register MapForce Server
- Assign a license
3.1 Installation on Linux

MapForce Server is available for installation on Linux systems. Its installation and setup procedure is described below.

- System requirements
  - Linux
    - CentOS 6 or newer
    - RedHat 6 or newer
    - Debian 8 or newer
    - Ubuntu 14.04 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</td>
<td>libidn, krb5-libs</td>
<td>libidn11, libgssapi-krb5-2</td>
<td>libidn11, libgssapi-krb5-2</td>
</tr>
<tr>
<td>Advanced Edition</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- FlowForce Server integration

If you are installing MapForce Server together with FlowForce Server, it is recommended that you install FlowForce Server first. Otherwise, after having installed both MapForce Server and FlowForce Server, run the following command:

```
cp /opt/Altova/MapForceServer2019/etc/*.tool /opt/Altova/FlowForceServer2019/tools
```

This command 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.

- Uninstall old versions of Altova server products

If you need to uninstall a previous version, do this as follows. On the Linux command line interface (CLI), you can check which Altova server products are installed with the following command:

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

If MapForce Server is not installed, go ahead with the installation as documented below in *Installing MapForce Server*. 
If MapForce Server is installed and you wish to install a newer version of MapForce Server, uninstall the old version with the command:

[Debian, Ubuntu]:  sudo dpkg --remove mapforcecserveradv
[CentOS, RedHat]:  sudo rpm -e mapforcecserveradv

If you need to uninstall an old version of Altova LicenseServer, do this with the following command:

[Debian, Ubuntu]:  sudo dpkg --remove licenseserver
[CentOS, RedHat]:  sudo rpm -e licenseserver

**Download the Linux package**

MapForce Server installation packages for the following Linux systems are available at the Altova website.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Package extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debian</td>
<td>.deb</td>
</tr>
<tr>
<td>Ubuntu</td>
<td>.deb</td>
</tr>
<tr>
<td>CentOS</td>
<td>.rpm</td>
</tr>
<tr>
<td>RedHat</td>
<td>.rpm</td>
</tr>
</tbody>
</table>

After downloading the Linux package, copy it to any directory on the Linux system. Since you will need an Altova LicenseServer in order to run MapForce Server, you may want to download LicenseServer from the Altova website at the same time as you download MapForce Server, rather than download it at a later time.

**Install MapForce Server**

In a terminal window, switch to the directory where you have copied the Linux package. For example, if you copied it to a user directory called MyAltova (that is located, say, in the /home/User directory), then switch to this directory as follows:

cd /home/User/MyAltova

Install MapForce Server with the following command:

[Debian]:  sudo dpkg --install mapforceserveradv-2019-debian.deb
[Ubuntu]:  sudo dpkg --install mapforceserveradv-2019-ubuntu.deb
[CentOS]:  sudo rpm -ivh mapforceserveradv-2019-1.86_64.rpm
[RedHat]:  sudo rpm -ivh mapforceserveradv-2019-1.86_64.rpm

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

The MapForce Server package will be installed in the folder:

/opt/Altova/MapForceServer2019
Altova LicenseServer

In order for any Altova Server product—including MapForce Server—to run, that server product must be licensed via an Altova LicenseServer on your network.

On Linux systems, Altova LicenseServer will need to be installed separately. Download LicenseServer from the Altova website and copy the package to any directory on the Linux system. Install it just like you installed MapForce Server (see previous step).

[Debian]: `sudo dpkg --install licenseserver-3.2-debian.deb`
[Ubuntu]: `sudo dpkg --install licenseserver-3.2-ubuntu.deb`
[CentOS]: `sudo rpm -ivh licenseserver-3.2-1.x86_64.rpm`
[RedHat]: `sudo rpm -ivh licenseserver-3.2-1.x86_64.rpm`

The LicenseServer package will be installed in:

/opt/Altova/LicenseServer

For information about how to register MapForce Server with Altova LicenseServer and license it, see the section, Licensing on Linux. Also see the LicenseServer documentation for more detailed information.

LicenseServer versions

- Altova server products must be licensed either with the version of LicenseServer that is appropriate to the installed MapForce Server version, or with a later version of LicenseServer.
- The LicenseServer version that is appropriate for a particular version of MapForce Server is displayed during the installation of MapForce Server. You can install this version of LicenseServer along with MapForce Server, or you can install LicenseServer separately.
- Before installing a newer version of LicenseServer, any older one must be de-installed. The LicenseServer installer will do this automatically if it detects an older version.
- LicenseServer versions are backwards compatible. They will work with older versions of MapForce Server.
- If you install a new version of MapForce Server and if your installed LicenseServer version is older than the appropriate LicenseServer, install the latest version available on the Altova website.
- At the time of LicenseServer de-installation, all registration and licensing information held in the older version of LicenseServer will be saved to a database on your server machine. This data will be imported automatically into the newer version when the newer version is installed.
- The version number of the currently installed LicenseServer is given at the bottom of the LicenseServer configuration page (all tabs).

Current version: 3.2
3.2 Licensing on Linux

MapForce Server must be licensed with Altova LicenseServer. Licensing is a two-step process:

1. **Register MapForce Server** with LicenseServer. Registration is done from MapForce Server.
2. **Assign a license** to MapForce Server from LicenseServer. Download the latest version of LicenseServer from the Altova website, and install it on your local machine or a machine on your network.

The steps to carry out are given below in brief. For detailed information, see the LicenseServer user manual at the Altova website.

▼ **Start LicenseServer**

To correctly register and license MapForce Server with LicenseServer, LicenseServer must be running as a daemon on the network. Start LicenseServer as a daemon with the following command:

```
[< Debian 8]  sudo /etc/init.d/licenseserver start
[≥ Debian 8]  sudo systemctl start licenseserver
[< CentOS 7]  sudo initctl start licenseserver
[≥ CentOS 7]  sudo systemctl start licenseserver
[< Ubuntu 15]  sudo initctl start licenseserver
[≥ Ubuntu 15]  sudo systemctl start licenseserver
[RedHat]  sudo initctl start licenseserver
```

If at any time you need to stop LicenseServer, replace `start` with `stop` in the above commands. For example:

```
sudo /etc/init.d/licenseserver stop
```

▼ **Register MapForce Server**

To register MapForce Server from the command line interface, use the `licenseserver` command:

```
sudo /opt/Altova/MapForceServer2019/bin/mapforceserver licenseserver [options] ServerName-Or-IP-Address
```

For example, if `localhost` is the name of the server on which LicenseServer is installed:

```
sudo /opt/Altova/MapForceServer2019/bin/mapforceserver licenseserver localhost
```

In the command above, `localhost` is the name of the server on which LicenseServer is installed. Notice also that the location of the MapForce Server executable is:

```
/opt/Altova/MapForceServer2019/bin/
```

After successful registration, go to the Server Management tab of LicenseServer's
configuration page to assign a license to MapForce Server.

**Assign a license**

After successfully registering MapForce Server, it will be listed in the Server Management tab of the configuration page of LicenseServer. Go there and assign a license to MapForce Server.

The licensing of Altova server products is based on the number of processor cores available on the product machine. 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 (an octa-core processor), you must purchase at least one 8-core license. You can also combine licenses to achieve the core count. So, two 4-core licenses can also be used for an octa-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 on the server were utilized.

**Note:** Each Altova server product license can be used for only one client machine—the machine on which the Altova server product is installed—at a time, 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.

**Note:** FlowForce Server Advanced Edition and MapForce Server Advanced Edition will run only on machines with two or more cores.

When assessing the number of cores you should license, this decision should take into account the data volumes you need to process and the processing time your business environment is expected to allow for. In most scenarios, a larger number of cores means more volumes of data processed in less time. Below are just a few application-specific tips:

- FlowForce Server runs as a multi-threaded application. If the number of concurrent requests to the server is big, an insufficient number of cores will lead to latency (waiting times). For example, if you are exposing jobs as Web services, there may be hundreds of concurrent requests from clients. In this case, FlowForce Server will significantly benefit from a larger number of cores.
- MapForce Server will utilize a single core at a time, per mapping. Therefore, if you need to run multiple mappings simultaneously, a larger number of cores is highly recommended. For example, when MapForce Server runs under FlowForce Server management, several mapping jobs may overlap and run concurrently, depending also on the setup. Note, however, that if the volumes processed by your mappings are extremely big, latency could still occur.

In addition to the above, note that there are various external factors that typically influence the processing volumes and times that your server is capable to handle (for example, the hardware, the current load on the CPU, memory allocation of other applications running on
the server). In order to get the most accurate performance measurements, it is generally advisable to first run the tools in your environment and expose them to actual factors and data specific to your business.
4 Setup on macOS

This section describes the installation and licensing of MapForce Server on macOS systems.

**Installation on macOS**
- System requirements
- Uninstall old versions of Altova server products
- Download the macOS package
- Install MapForce Server
- Altova LicenseServer
- LicenseServer versions

**Licensing on macOS**
- Start LicenseServer
- Register MapForce Server
- Assign a license
4.1 Installation on macOS

MapForce Server is available for installation on macOS systems. Its installation and setup procedure is described below.

▼ System requirements
  - macOS
    - macOS 10.12 or newer

▼ FlowForce Server integration

If you install MapForce Server together with FlowForce Server, it is recommended that you install FlowForce Server first. If you install MapForce Server before FlowForce Server, then, after having installed both MapForce Server and FlowForce Server, run the following command:

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

This command 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.

▼ Uninstall old versions of Altova server products

In the Applications folder in Finder, right-click the MapForce Server icon and select Move to Trash. The application will be moved to Trash. You will, however, still need to remove the application from the usr folder. Do this with the command:

```bash
sudo rm -rf /usr/local/Altova/MapForceServer2019/
```

If you need to uninstall an old version of Altova LicenseServer, you must first stop it running as a service. Do this with the following command:

```bash
sudo launchctl unload /Library/LaunchDaemons/
com.altova.LicenseServer.plist
```

To check whether the service has been stopped, open the Activity Monitor in Finder and make sure that LicenseServer is not in the list. Then proceed to uninstall in the same way as described above for MapForce Server.

▼ Download the disk image file

Download the disk image (.dmg) file from the Altova website (http://www.altova.com/download.html).

▼ Install MapForce Server
Click to open the downloaded disk image (.dmg). This causes the MapForce Server installer to appear as a new virtual drive on your computer. On the new virtual drive, double-click the installer package (.pkg). Go through the successive steps of the installer wizard. These are self-explanatory and include one step in which you have to agree to the license agreement before being able to proceed. To eject the drive after installation, right-click it and select Eject.

The MapForce Server package will be installed in the folder:
- /usr/local/Altova/MapForceServer2019 (application binaries)
- /var/Altova/MapForceServer (data files: database and logs)

Altova LicenseServer

In order for any Altova Server product—including MapForce Server—to run, that server product must be licensed via an Altova LicenseServer on your network.

The Altova LicenseServer installation package is available on the virtual drive you have mounted in the previous step. To install Altova LicenseServer, double-click the installer package included on the virtual drive and follow the on-screen instructions. You will need to accept the license agreement for installation to proceed.

Altova LicenseServer can also be downloaded and installed separately from the Altova website (http://www.altova.com/download.html).

The LicenseServer package will be installed in the folder:
- /usr/local/Altova/LicenseServer

For information about how to register MapForce Server with Altova LicenseServer and license it, see the section, Licensing on macOS.

LicenseServer versions

- Altova server products must be licensed either with the version of LicenseServer that is appropriate to the installed MapForce Server version, or with a later version of LicenseServer.
- The LicenseServer version that is appropriate for a particular version of MapForce Server is displayed during the installation of MapForce Server. You can install this version of LicenseServer along with MapForce Server, or you can install LicenseServer separately.
- Before installing a newer version of LicenseServer, any older one must be de-installed. The LicenseServer installer will do this automatically if it detects an older version.
- LicenseServer versions are backwards compatible. They will work with older versions of MapForce Server.
- If you install a new version of MapForce Server and if your installed LicenseServer version is older than the appropriate LicenseServer, install the latest version available on the Altova website.
- At the time of LicenseServer de-installation, all registration and licensing information held in the older version of LicenseServer will be saved to a database on your server machine. This data will be imported automatically into the newer version when the
newer version is installed.

- The version number of the currently installed LicenseServer is given at the bottom of the LicenseServer configuration page (all tabs).

**Current version: 3.2**
4.2 Licensing on macOS

MapForce Server must be licensed with Altova LicenseServer. Licensing is a two-step process:

1. **Register MapForce Server** with LicenseServer. Registration is done from MapForce Server.
2. **Assign a license** to MapForce Server from LicenseServer. Download the latest version of LicenseServer from the Altova website, and install it on your local machine or a machine on your network.

The steps to carry out are given below in brief. For detailed information, see the LicenseServer user manual at the Altova website.

**Start LicenseServer**

To correctly register and license MapForce Server with LicenseServer, LicenseServer must be running as a daemon. Start LicenseServer as a daemon with the following command:

```
sudo launchctl load /Library/LaunchDaemons/com.altova.LicenseServer.plist
```

If at any time you need to stop LicenseServer, replace `load` with `unload` in the above command:

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.LicenseServer.plist
```

**Register MapForce Server**

To register MapForce Server from the command line interface, use the `licenseserver` command:

```
sudo /usr/local/Altova/MapForceServer2019/bin/MapForceServer licenseserver [options] ServerName-Or-IP-Address
```

For example, if `localhost` is the name of the server on which LicenseServer is installed:

```
sudo /usr/local/Altova/MapForceServer2019/bin/MapForceServer licenseserver localhost
```

In the command above, `localhost` is the name of the server on which LicenseServer is installed. Notice also that the location of the MapForce Server executable is:

```
/usr/local/Altova/MapForceServer2019/bin/
```

After successful registration, go to the Server Management tab of LicenseServer's configuration page to assign a license to MapForce Server.

**Assign a license**

After successfully registering MapForce Server, it will be listed in the Server Management tab of the configuration page of LicenseServer. Go there and assign a license to MapForce Server.

The licensing of Altova server products is based on the number of processor cores available
Licensing on macOS

on the product machine. 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 (an octa-core processor), you must purchase at least one 8-core license. You can also combine licenses to achieve the core count. So, two 4-core licenses can also be used for an octa-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 on the server were utilized.

**Note:** Each Altova server product license can be used for only one client machine—the machine on which the Altova server product is installed—at a time, 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.

**Note:** FlowForce Server Advanced Edition and MapForce Server Advanced Edition will run only on machines with two or more cores.

When assessing the number of cores you should license, this decision should take into account the data volumes you need to process and the processing time your business environment is expected to allow for. In most scenarios, a larger number of cores means more volumes of data processed in less time. Below are just a few application-specific tips:

- FlowForce Server runs as a multi-threaded application. If the number of concurrent requests to the server is big, an insufficient number of cores will lead to latency (waiting times). For example, if you are exposing jobs as Web services, there may be hundreds of concurrent requests from clients. In this case, FlowForce Server will significantly benefit from a larger number of cores.

- MapForce Server will utilize a single core at a time, per mapping. Therefore, if you need to run multiple mappings simultaneously, a larger number of cores is highly recommended. For example, when MapForce Server runs under FlowForce Server management, several mapping jobs may overlap and run concurrently, depending also on the setup. Note, however, that if the volumes processed by your mappings are extremely big, latency could still occur.

In addition to the above, note that there are various external factors that typically influence the processing volumes and times that your server is capable to handle (for example, the hardware, the current load on the CPU, memory allocation of other applications running on the server). In order to get the most accurate performance measurements, it is generally advisable to first run the tools in your environment and expose them to actual factors and data specific to your business.
Chapter 5

MapForce Server Command Line
## 5 MapForce Server Command Line

*This topic:*
- Default location of MapForce Server executable
- Usage and list of CLI commands

### Default location of MapForce Server executable
Given below are the default locations of the MapForce Server executable:

- **Windows**  
  `<ProgramFilesFolder>\Altova\MapForceServer2019\bin\MapForceServer.exe`

- **Linux**  
  `/opt/Altova/MapForceServer2019/bin/mapforceserver`

- **Mac**  
  `/usr/local/Altova/MapForceServer2019/bin/mapforceserver`

### Usage and list of CLI commands
General command line syntax for `MapForceServer` is:

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

▼ Casing and slashes on the command line

- **MapForceServer on Windows**
- **mapforceserver on Unix (Linux, Mac)**

  * Note that lowercase (`mapforceserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`MapForceServer`) works only on Windows and Mac.
  * Use forward slashes on Linux and Mac, backslashes on Windows.

where

<table>
<thead>
<tr>
<th>--h</th>
<th>--help</th>
<th>Displays the help text.</th>
</tr>
</thead>
<tbody>
<tr>
<td>--version</td>
<td></td>
<td>Displays the version of MapForce Server.</td>
</tr>
</tbody>
</table>

Valid CLI commands are listed below and are explained in the sub-sections of this section.

<table>
<thead>
<tr>
<th>licenseserver</th>
<th>Register MapForce Server with LicenseServer on the local network.</th>
</tr>
</thead>
<tbody>
<tr>
<td>assignlicense</td>
<td>Upload a license to LicenseServer and assign it to MapForce Server on this machine.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>verifylicense</td>
<td>Check if the current MapForce Server is licensed; optionally, check if a given license key is assigned.</td>
</tr>
<tr>
<td>run</td>
<td>Executes a MapForce Server Execution file (.mfx)</td>
</tr>
<tr>
<td>exportresourcestrings</td>
<td>Exports all application resource strings to an XML file.</td>
</tr>
<tr>
<td>setdeflang</td>
<td>_sd1</td>
</tr>
<tr>
<td>help</td>
<td>Displays help for a specific command. For example: help run</td>
</tr>
</tbody>
</table>
5.1 assignlicense (Windows only)

On execution, the assignlicense command uploads the license file specified by the FILE argument to the registered LicenseServer, and assigns the license to MapForce Server on this machine. The FILE argument takes the file path of the license file. The --test-only option allows you to upload to LicenseServer and validate the license, without assigning the license to MapForce Server. For details about licensing, see the LicenseServer documentation (https://www.altova.com/documentation).

Note: This command is supported only on Windows systems. It is not supported on Linux or Mac systems.

Syntax

MapForceServer assignlicense [options] FILE

Casing and slashes on the command line

MapForceServer on Windows
mapforceserver on Unix (Linux, Mac)

* Note that lowercase (mapforceserver) works on all platforms (Windows, Linux, and Mac), while upper-lower (MapForceServer) works only on Windows and Mac.

* Use forward slashes on Linux and Mac, backslashes on Windows.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>The path to the license file</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--t, --test-only</td>
<td>Values are true</td>
</tr>
<tr>
<td>--h, --help</td>
<td>Displays information about the command</td>
</tr>
</tbody>
</table>

Options are listed in their short forms (if available) and long forms. You can use one or two dashes for both short and long forms. An option that takes a value is written like this: --option=value.

Note: On Windows systems: Avoid using the end backslash and closing quote on the command line "", for example, as in: "C:\My directory\". These two characters are interpreted by the command line parser as a literal double-quotation mark. Use the double backslash \ if spaces occur in the command line and you need the quotes (for example: "C:\My Directory\"). Alternatively, try to avoid using spaces and, therefore, quotes at all.
**Examples**

Examples of the `assignlicense` command:

```bash
mapforceserver assignlicense C:\MapForce Server12345.altova_licenses
mapforceserver assignlicense --test-only=true C:\MapForce Server12345.altova_licenses
```

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

The `exportresourcestrings` command outputs an XML file containing the resource strings of the MapForce Server application. It takes two arguments:

- 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 path and name of the output XML file

How to create localizations is described below.

Syntax

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

▼ Casing and slashes on the command line

- MapForceServer on Windows
- mapforceserver on Unix (Linux, Mac)

* Note that lowercase (`mapforceserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`MapForceServer`) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LanguageCode</td>
<td>Specifies the language of resource strings in the exported XML file. Is a string of length 2. Supported languages are: <code>en</code>, <code>de</code>, <code>es</code>, <code>fr</code>, <code>ja</code></td>
</tr>
<tr>
<td>XMLOutputFile</td>
<td>Specifies the location and name of the exported XML file. Is a string of length 256.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--h, --help</td>
<td>Displays information about the command</td>
</tr>
</tbody>
</table>

Options are listed in their short forms (if available) and long forms. You can use one or two dashes for both short and long forms. An option that takes a value is written like this: `--option=value`.

**Note:** On Windows systems: Avoid using the end backslash and closing quote on the command line `\`, for example, as in: "C:\My directory\". These two characters are interpreted by the command line parser as a literal double-quotation mark. Use the double backslash `\` if spaces occur in the command line and you need the quotes (for example: "C:\My Directory\"). Alternatively, try to avoid using spaces and, therefore, quotes at all.
Examples
An example of the exportresourcestrings command:

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

This command creates a file called Strings.xml at c:\ that contains all 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\MapForceServer2019\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 four 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\%APPFOLDER\bin folder. Your DLL file will have a name of the form MapForceServer2019_lc.dll. The _lc part of the name contains the language code. For example, in MapForceServer2019_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.
5.3 help

The `help` command takes a single argument \((\text{Command})\): the name of the command for which help is required. It displays the correct syntax of the command and other information relevant to the correct execution of the command.

Syntax

```
mapforceserver help [options] Command
```

Casing and slashes on the command line

- **MapForceServer** on Windows
- **mapforceserver** on Unix (Linux, Mac)

* Note that lowercase (`mapforceserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`MapForceServer`) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>The name of the command for which you want help information</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--h, --help</td>
<td>Displays information about the command</td>
</tr>
</tbody>
</table>

Options are listed in their short forms (if available) and long forms. You can use one or two dashes for both short and long forms. An option that takes a value is written like this: `--option=value`.

Examples

An example of the `help` command:

```
mapforceserver help licenseserver
```

The command above contains one argument: the command `licenseserver`, for which help is required. When the example command above is executed, information about the `licenseserver` command will be displayed in the terminal.

The --help option

Help information about a command is also available by using the --help option with the command for which help information is required. For example: Using the --help option with the `licenseserver` command, as follows:
mapforceserver licenseserver --help

achieves the same result as does using the help command with an argument of licenseserver:

mapforceserver help licenseserver

In both cases, help information about the licenseserver command is displayed.
5.4 licenseserver

On execution, the licenseserver command registers MapForce Server with the LicenseServer specified by the Server-Or-IP-Address argument. For the licenseserver command to be executed successfully, the two servers must be connected on the network and LicenseServer must be running. You must have administrator privileges to be able to register MapForce Server with LicenseServer.

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/documentation).

Syntax

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

Casing and slashes on the command line

- **MapForceServer on Windows**
- **mapforceserver on Unix (Linux, Mac)**

* Note that lowercase (mapforceserver) works on all platforms (Windows, Linux, and Mac), while upper-lower (MapForceServer) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server-Or-IP-Address</td>
<td>Identifies the machine on the network on which Altova LicenseServer is installed and running. It can be the machine’s name or its IP address.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--h, --help</td>
<td>Displays information about the command</td>
</tr>
<tr>
<td>--j, --json</td>
<td>Values are true</td>
</tr>
</tbody>
</table>

Options are listed in their short forms (if available) and long forms. You can use one or two dashes for both short and long forms. An option that takes a value is written like this: **--option=value**.

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.
5.5 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:

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

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

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

To redirect both streams simultaneously, use:

```bash
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`

**Casing and slashes on the command line**

- `MapForceServer` on Windows
**mapforceserver on Unix (Linux, Mac)**

* Note that lowercase (`mapforceserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`MapForceServer`) works only on Windows and Mac.

* Use forward slashes on Linux and Mac, backslashes on Windows.

On Windows systems, avoid using the end backslash and closing quote on the command line `"`, for example, as in: "C:\My directory". These two characters are interpreted by the command line parser as a literal double-quotation mark. Use the double backslash `\` if spaces occur in the command line and you need the quotes (for example: "C:\My Directory\`). Alternatively, try to avoid using spaces and, therefore, quotes at all.

**Command options**

Options are listed in their short forms (if available) and long forms. You can use one or two dashes for both short and long forms. An option that takes a value is written like this: `--option=value`.

<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 absolute path to the installed root catalog file. Form: <code>--catalog=FILE</code></td>
</tr>
<tr>
<td><code>--cert</code></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: <code>--certificatespath=DIRECTORY</code></td>
</tr>
<tr>
<td></td>
<td>See also <a href="#">Digital Certificate Management</a>.</td>
</tr>
<tr>
<td><code>--gc</code></td>
<td>This option is applicable if the mapping consumes Altova Global Resources (see Global Resources). It specifies the name of the global resource configuration (alias). This option must be used together with the <code>--globalresourcefile</code> option. Form: <code>--gc=VALUE</code></td>
</tr>
<tr>
<td><code>--gr</code></td>
<td>This option is applicable if the mapping consumes Altova Global Resources (see Global Resources). It specifies the path of the global resource definition file. This option must be used together with the <code>--globalresourceconfig</code> option. Form: <code>--gr=FILE</code></td>
</tr>
<tr>
<td><code>--l</code></td>
<td>The language used for displaying messages. Form: <code>--lang=VALUE</code> (en, de, ja, es, fr)</td>
</tr>
</tbody>
</table>
| `--p`          | 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
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--taxonomy-package</code></td>
<td>Specifies the absolute path to an additional XBRL taxonomy package as described in the Taxonomy Packages 1.0 recommendation. The value of FILE gives the location of the taxonomy package. Add the option multiple times to specify more than one taxonomy package. Form: <code>--taxonomy-package=FILE</code></td>
</tr>
<tr>
<td><code>--taxonomy-packages-config-file</code></td>
<td>Specifies the path to a configuration file called TaxonomyPackagesConfig.json, used to load XBRL taxonomy packages. This configuration file is updated every time when you add, remove, activate, or deactivate XBRL taxonomy packages from the graphical user interface of Altova XMLSpy, MapForce, or StyleVision. If you have added custom XBRL taxonomy packages using one of the products above, the file is located at C:\Users&lt;username&gt;\Documents\Altova. Form: <code>--taxonomy-packages-config-file=FILE</code></td>
</tr>
</tbody>
</table>

### 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\<user>\Documents\Altova\MapForce2019\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\MapForceServer2019\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.
5.6 **setdeflang**

The `setdeflang` command (short form is `sdl`) sets the default language of MapForce Server. It takes a mandatory `LanguageCode` argument.

**Syntax**

```
mapforceserver setdeflang | sdl [options] LanguageCode
```

- **Casing and slashes on the command line**
  - `MapForceServer on Windows`
  - `mapforceserver on Unix (Linux, Mac)`

  * Note that lowercase (`mapforceserver`) works on all platforms (Windows, Linux, and Mac), while upper-lower (`MapForceServer`) works only on Windows and Mac.
  * Use forward slashes on Linux and Mac, backslashes on Windows.

**Argument**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LanguageCode</td>
<td>sets the default language of MapForce Server. Supported languages are: <code>en</code>, <code>de</code>, <code>es</code>, <code>fr</code>, <code>ja</code></td>
</tr>
</tbody>
</table>

**Option**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--h, --help</td>
<td>Displays information about the command</td>
</tr>
</tbody>
</table>

Options are listed in their short forms (if available) and long forms. You can use one or two dashes for both short and long forms. An option that takes a value is written like this: `--option=value`.

**Examples**

An example of the `setdeflang` command:

```
mapforceserver setdeflang DE
```

The command above sets the default language for messages to German.
5.7 verifylicense (Windows only)

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

Syntax

MapForceServer verifylicense [options]

- Casing and slashes on the command line
  - MapForceServer on Windows
  - mapforceserver on Unix (Linux, Mac)

* Note that lowercase (mapforceserver) works on all platforms (Windows, Linux, and Mac), while upper-lower (MapForceServer) works only on Windows and Mac.
* Use forward slashes on Linux and Mac, backslashes on Windows.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--l, license-key</td>
<td>The license key to check: whether it has been assigned to the product</td>
</tr>
<tr>
<td>--h, --help</td>
<td>Displays information about the command</td>
</tr>
</tbody>
</table>

Options are listed in their short forms (if available) and long forms. You can use one or two dashes for both short and long forms. An option that takes a value is written like this: --option=value.

Examples

Examples of the verifylicense command:

  mapforceserver verifylicense
  mapforceserver verifylicense --license-key=ABCDEFG-HIJKLMN-OPQRSTU-VWXYZ12-3456789

The commands above check, respectively, whether the product has been licensed, and whether the product has been licensed with the license key given as the value of the --license-key option.
Chapter 6

MapForce Server API
6 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:

- About the .NET Interface
- About the COM Interface
- About the Java Interface

For code examples, refer to the Code Examples section.

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

- API Reference (COM, .NET)
- API Reference (Java)
6.1 About the .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. If you receive an access error, open the Component Services and give permissions to the same account that runs the application pool containing MapForce Server.

In order to use MapForce Server in your .NET project, add a reference to the Altova.MapForceServer.dll file (see the instructions 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.

Once MapForce Server has been registered as a COM server object, and the Altova.MapForceServer.dll is available to the .NET interface, the MapForce API functionality becomes available in your .NET project.

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**. The Add Reference dialog box pops up.
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).
6.2 About the 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 classes `MapForce.Server`. These two classes will typically be 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.
6.3 About the Java Interface

To access the MapForce Server API from Java code, the following references must be added to the .classpath file of your Java project.

<table>
<thead>
<tr>
<th>MapForceServer.jar</th>
<th>The library that communicates with MapForce Server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MapForceServer_JavaDoc.zip</td>
<td>A Javadoc archive file containing documentation for the Java MapForce Server API.</td>
</tr>
</tbody>
</table>

Both files are available in the bin folder of the MapForce Server installation folder. You can either reference the file from their original location or copy them to another location if this fits your project setup.

In Eclipse, you can add the classpath references by editing the properties of the Java project, as shown in the sample instructions below.

To add the MapForce Server library references in Eclipse:

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. Under JARs and class folders on the build path, expand the MapForceServer.jar
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. The reference to the MapForce Server library and Javadoc archive has now been added to the `.classpath` file of the project.

   Below is an example of how the `.classpath` file might look if you are referencing the files
from the original installation folder, on a 64-bit operating system running 32-bit MapForce Server (the relevant lines are highlighted in yellow):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<classpath>
    <classpathentry kind="src" path="src"/>  
    <classpathentry kind="con" path="org.eclipse.jdt.launching.JRE_CONTAINER/org.eclipse.jdt.internal.debug.ui.launcher.StandardVMType/jre7"/>
    <classpathentry kind="lib" path="C:/Program Files (x86)/Altova/MapForceServer2019/bin/MapForceServer.jar">
        <attributes>
            <attribute name="javadoc_location" value="jar:file:/C:/Program Files (x86)/Altova/MapForceServer2019/bin/MapForceServer_JavaDoc.zip!/"/>
        </attributes>
    </classpathentry>
    <classpathentry kind="output" path="bin"/>
</classpath>
```
6.4 Code Examples

This section includes sample code which illustrates how to run a mapping programmatically with MapForce Server. Samples are included for the following languages:

- C#
- C++
- Java
- VBScript
- Visual Basic
- Visual Basic for Applications (VBA)

6.4.1 C#

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\MapForceServer2019\etc\Examples`.

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.

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 About the .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 .mfx file

The code 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. Namely, the server execution file run by this example (TokenizeString.mfx) reads data from a file called AltovaTools.xml, which is accompanied by the AltovaTools.xsd schema. These files must exist in the working directory in order for the .mfx file to run successfully.
It runs the `TokenizeString.mfx` file. The path to this file is supplied as argument to the `Run` method. Upon successful execution, a `.csv` file representing the mapping output will be created in the working directory.

```csharp
namespace MapForceServerAPI_sample
{
    class Program
    {
        static void Main(string[] args)
        {
            //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 = "C:\Program Files\Altova\MapForceServer2019\etc\Examples";

            //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\MapForceServer2019\bin\MapForceServer.exe";

            //Set global resource file and configuration, if your mapping uses global resources
            //objMFS.SetOption("globalresourcefile", "GlobalResource.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 mapping 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\Altova\MapForceServer2019\etc\Examples\TokenizeString.mfx"))
            {
                System.Console.WriteLine("Success - finished execution");
            }
            else
            {
                System.Console.WriteLine(objMFS.LastExecutionMessage);
            }
        }
    }
}
```

### 6.4.2 C++

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\MapForceServer2019\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](#)).

```cpp
// MapForceServerAPI_sample.cpp : Defines the entry point for the console application.
//
#include <iostream>
#include "atlbase.h"

// 32-bit MapForce Server
#import "progid:MapForce.Server"
// 64-bit MapForce Server
// #import "progid:MapForce_x64.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
        (you may need to adapt the path to the installation folder)
        pMFS->WorkingDirectory = "C:\Program Files (x86)\Altova\MapForceServer2019\etc\Examples";

        //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\MapForceServer2019\bin\MapForceServer.exe";

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

        //Prepare the parameters, if your mapping uses parameters
        //pMFS->AddParameter( "testparam1", "value 1" );

        //Run the mapping; the output will be stored at C:\temp \ExpReport.rtf
        // NOTE Please adapt the path to the input file in order to run
```
6.4.3 Java

The following example illustrates how to run a mapping execution file (.mfx) from Java code. On Windows, the example files are available at the following path: **C:\Program Files\Altova\MapForceServer2019\etc\Examples**.

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

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

- MapForce Server is installed and licensed
- MapForce Server libraries are added to the .classpath file in your project (for an example, see About the Java Interface).

```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\\MapForceServer2019\\etc\\Examples");
        } catch (_com_error& err) {
            BSTR bstrMessage;
            (err).ErrorInfo()->GetDescription(&bstrMessage);
            std::cout << "Exception occurred: " << _com_util::ConvertBSTRToString(bstrMessage) << std::endl;
        }
        CoUninitialize();
        return 0;
    }
}
```
//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\\MapForceServer2019\\bin\\MapForceServer.exe");

//Set global resource file and configuration, if your mapping uses global resources
//objMFS.setOption( "globalresourcefile", "GlobalResource.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\\MapForceServer2019\\etc\\Examples\\TokenizeString.mfx" ) )
    System.out.println( "Success - finished execution" );
else
    System.out.println( objMFS.getLastExecutionMessage() );
}
catch ( Exception e )
{
    e.printStackTrace();
}
System.out.println( "Finished - exiting" );

6.4.4 VBScript

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\MapForceServer2019\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).

Option Explicit
'Create a MapForce Server object; use "MapForce_x64.Server" if you want to use the 64-bit version.
Dim objMFS
Set objMFS = WScript.GetObject( "", "MapForce.Server" )

'Set a working directory - used as a base for relative paths (you may need to adapt this to your system)
objMFS.WorkingDirectory = "C:\Program Files (x86)\Altova\MapForceServer2019\etc\Examples"

'Default path to the MapForce Server executable is the installation path (same dir with the MapForceServer.dll)
'Default path to the MapForce Server executable is the installation path (same dir with the MapForceServer.dll)
'Default path to the MapForce Server executable is the installation path (same dir with the MapForceServer.dll)
'Default path to the MapForce Server executable is the installation path (same dir with the MapForceServer.dll)
'Default path to the MapForce Server executable is the installation path (same dir with the MapForceServer.dll)
'Default path to the MapForce Server executable is the installation path (same dir with the MapForceServer.dll)

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

'Prepare the parameters, if your mapping uses parameters
Call 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\MapForceServer2019\etc\Examples\TokenizeString.mfx" ) ) Then
WScript.Echo( "Success - finished execution" )
Else
WScript.Echo( objMFS.LastExecutionMessage )
End If

6.4.5 Visual Basic

The following example illustrates how to run a mapping execution file (.mfx) from Visual Basic code. On Windows, the example files are available at the following path: C:\Program Files \Altova\MapForceServer2019\etc\Examples.

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.

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 About the .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 .mfx file

The code 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. Namely, the server execution file run by this example (TokenizeString.mfx) reads data from a file called `AltovaTools.xml`, which is accompanied by the `AltovaTools.xsd` schema. These files must exist in the working directory in order for the .mfx file to run successfully.
- It runs the `TokenizeString.mfx` file. The path to this file is supplied as argument to the `Run` method. Upon successful execution, a .csv file representing the mapping output will be created in the working directory.

```vba
Option Explicit On
Module Program
    Sub Main()
        '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 = "C:\Program Files (x86)\Altova\MapForceServer2019\etc\Examples"

        '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\MapForceServer2019\bin\MapForceServer.exe"

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

        'Prepare the parameters, if your mapping 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\MapForceServer2019\etc\Examples\TokenizeString.mfx")) Then
    End Sub
End Module
```
System.Console.WriteLine("Success - finished execution")
Else
    System.Console.WriteLine(objMFS.LastExecutionMessage)
End If
End Sub
End Module

6.4.6 Visual Basic for Applications (VBA)

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.
3. Click to select the **MapForce Server Type Library**.

**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
    ' 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 Sub
6.5 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.

6.5.1 Interfaces

6.5.1.1 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>
</tbody>
</table>
| MinorVersion             | Read-only. Gets the minor version of the product (for example, "2"
## Name

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ProductName</strong></td>
<td>Read-only. Gets the name of the product (for example, &quot;Altova MapForce Server&quot;).</td>
</tr>
<tr>
<td><strong>ProductNameAndVersion</strong></td>
<td>Read-only. Gets the complete name and version of the product (for example, &quot;Altova MapForce Server 2018 r2 sp1 (x64)&quot;).</td>
</tr>
<tr>
<td><strong>ServerPath</strong></td>
<td>Gets or sets the path of the MapForce Server executable.</td>
</tr>
<tr>
<td><strong>ServicePackVersion</strong></td>
<td>Read-only. Gets the service pack version of the product (for example, &quot;1&quot; for Altova MapForce Server 2018 r2 sp1).</td>
</tr>
<tr>
<td><strong>WorkingDirectory</strong></td>
<td>Gets or sets the current directory for running jobs (relative paths will be evaluated against the working directory).</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddParameter</strong></td>
<td>Assigns a value to a parameter defined in the mapping.</td>
</tr>
<tr>
<td></td>
<td>The first argument specifies the name of the parameter as defined on the mapping; the second argument specifies the parameter value.</td>
</tr>
<tr>
<td><strong>ClearOptions</strong></td>
<td>Clears the list of options previously set through the SetOption method.</td>
</tr>
<tr>
<td></td>
<td>All options set through the SetOption method are valid for the lifetime of the object.</td>
</tr>
<tr>
<td></td>
<td>Call this method if you need to explicitly clear all of the previously set options.</td>
</tr>
<tr>
<td><strong>ClearParameterList</strong></td>
<td>Clears the list of parameters that were previously set using the AddParameter method.</td>
</tr>
<tr>
<td><strong>GetOutputParameter</strong></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><strong>Run</strong></td>
<td>Executes a MapForce Server Execution file (.mfx file).</td>
</tr>
<tr>
<td></td>
<td>Returns <strong>true</strong> in case of success; <strong>false</strong> otherwise.</td>
</tr>
<tr>
<td><strong>SetOption</strong></td>
<td>Sets an option before running the mapping.</td>
</tr>
<tr>
<td></td>
<td>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).</td>
</tr>
</tbody>
</table>
MapForce Server API

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Global Resources). The currently supported options are as follows:</td>
</tr>
<tr>
<td></td>
<td>- <code>globalresourcefile (or gr)</code> - 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>- <code>globalresourceconfig (or gc)</code> - 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>- <code>catalog</code> - The path to a custom <code>RootCatalog.xml</code> 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>- <code>taxonomy-package</code> - The path to a custom XBRL taxonomy package, if one is required by the mapping.</td>
</tr>
<tr>
<td></td>
<td>- <code>taxonomy-packages-config-file</code> - 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.

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

- C++ example
- C# example
- VBScript example
- VB.NET example.

6.5.1.1.1 Properties

6.5.1.1.1 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

\[ \text{APIMajorVersion} : \text{Integer} \]

*Generic signature*

\[ \text{int APIMajorVersion} \{ \text{get; } \} \]

*C#*

\[ \text{HRESULT APIMajorVersion([out, retval] INT* pnVal);} \]

*C++*

\[ \text{ReadOnly Property APIMajorVersion As Integer} \]

*VB.NET*

\[ \text{APIMinorVersion} \]

6.5.1.1.2  \( \text{APIMinorVersion} \)

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

\[ \text{APIMinorVersion} : \text{Integer} \]

*Generic signature*

\[ \text{int APIMinorVersion} \{ \text{get; } \} \]

*C#*

\[ \text{HRESULT APIMinorVersion([out, retval] INT* pnVal);} \]

*C++*

\[ \text{ReadOnly Property APIMinorVersion As Integer} \]

*VB.NET*
6.5.1.1.3  **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**

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>APIServicePackVersion</strong> : Integer</td>
<td></td>
</tr>
</tbody>
</table>

*Generic signature*

```csharp
int APIServicePackVersion { get; }
```

*C#*

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

*C++*

```cpp
ReadOnly Property APIServicePackVersion As Integer
```

*VB.NET*


6.5.1.1.4  **Is64Bit**

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

**Signature**

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Is64Bit</strong> : Boolean</td>
<td></td>
</tr>
</tbody>
</table>

*Generic signature*

```csharp
bool Is64Bit { get; }
```

*C#*

```csharp
HRESULT Is64Bit([out, retval] VARIANT_BOOL* pbVal);
```

*C++*

```cpp
ReadOnly Property Is64Bit As Boolean
```

*VB.NET*
6.5.1.1.5  LastExecutionMessage

Gets the message received during the last Run command.

**Signature**

<table>
<thead>
<tr>
<th>LastExecutionMessage : String</th>
</tr>
</thead>
</table>

*Generic signature*

| string LastExecutionMessage { get; } |

*C#*

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

*C++*

| ReadOnly Property LastExecutionMessage As String |

*VB.NET*

6.5.1.1.6  MajorVersion

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

**Signature**

<table>
<thead>
<tr>
<th>MajorVersion : Integer</th>
</tr>
</thead>
</table>

*Generic signature*

| int MajorVersion { get; } |

*C#*

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

*C++*

| ReadOnly Property MajorVersion As Integer |

*VB.NET*
6.5.1.1.7  MinorVersion

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

Signature

| MinorVersion : Integer

Generic signature

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

C#

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

C++

```csharp
ReadOnly Property MinorVersion As Integer
```

VB.NET

6.5.1.1.8  ProductName

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

Signature

| ProductName : String

Generic signature

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

C#

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

C++

```csharp
ReadOnly Property ProductName As String
```

VB.NET
6.5.1.1.9  

**ProductNameAndVersion**

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

**Signature**

```
ProductNameAndVersion : String
```

**Generic signature**

```
string ProductNameAndVersion { get; }
```

**C#**

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

**C++**

```
ReadOnly Property ProductNameAndVersion As String
```

---

6.5.1.1.10  

**ServerPath**

Gets or sets the path of the MapForce Server executable.

**Signature**

```
ServerPath : String
```

**Generic signature**

```
string ServerPath { set; get; }
```

**C#**

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

**C++**

```
Property ServerPath As String
```
6.5.1.1.11  ServicePack Version

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
```

6.5.1.1.12  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

6.5.1.1.2 Methods

6.5.1.1.2.1 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

```AddParameter(in bstrName:String, in bstrValue:String) : Void```

**Generic signature**

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

C#

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

C++

```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>
<tbody>
<tr>
<td>bstrName</td>
<td>String</td>
<td>Specifies the parameter name.</td>
</tr>
<tr>
<td>bstrValue</td>
<td>String</td>
<td>Specifies the parameter value.</td>
</tr>
</tbody>
</table>
6.5.1.2.2  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**

```csharp
void ClearOptions()
```

**Generic signature**

```csharp
void ClearOptions()
```

**C#**

```csharp
HRESULT ClearOptions();
```

**C++**

```csharp
Sub ClearOptions()
```

**VB.NET**

```csharp
Sub ClearOptions()
```

6.5.1.2.3  ClearParameterList

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

**Signature**

```csharp
void ClearParameterList()
```

**Generic signature**

```csharp
void ClearParameterList()
```

**C#**

```csharp
HRESULT ClearParameterList();
```

**C++**

```csharp
Sub ClearParameterList()
```
### 6.5.1.2.4 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**

```vbnet
GetOutputParameter(in bstrName:String) : String
```

**Generic signature**

```c
string GetOutputParameter(string bstrName)
```

**C#**

```c
HRESULT GetOutputParameter([in] BSTR bstrName, [out, retval] BSTR* pbstrValue);
```

**C++**

```c
Function GetOutputParameter(bstrName As String) As String
```

**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 <strong>Properties</strong>)</td>
</tr>
</tbody>
</table>

### 6.5.1.2.5 Run

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

Run(in bstrMappingPath:String) : Boolean

Generic signature

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.</td>
</tr>
</tbody>
</table>

6.5.1.2.6 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**

```plaintext
SetOption(in bstrName:String, in bstrValue:String) : Void
```

**Generic signature**

```plaintext
void SetOption(ByVal bstrName As String, ByVal bstrValue As String)
```

**C#**

```plaintext
HRESULT SetOption([in] BSTR bstrName, [in] BSTR bstrValue );
```

**C++**

```plaintext
Sub SetOption(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>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>
6.6 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.

6.6.1 Classes

6.6.1.1 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>addParameter</td>
<td>Assigns a value to a parameter defined in the mapping.</td>
</tr>
<tr>
<td>clearOptions</td>
<td>Clears the list of options previously set through the SetOption method.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>clearParameterList</td>
<td>Clears the list of parameters that were previously set using the AddParameter 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 Run 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 run.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>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><code>getProductName</code></td>
<td>Gets the name of the product (for example, &quot;Altova MapForce Server&quot;).</td>
</tr>
<tr>
<td><code>getProductNameAndVersion</code></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><code>getServicePackVersion</code></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><code>is64bit</code></td>
<td>Returns <code>true</code> if the MapForce Server engine is a 64-bit executable.</td>
</tr>
<tr>
<td><code>run</code></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><code>setOption</code></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>• <code>globalresourcefile (or gr)</code> - 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>• <code>globalresourceconfig (or gc)</code> - 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>• <code>catalog</code> - The path to a custom <code>RootCatalog.xml</code> 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>• <code>taxonomy-package</code> - The path to a custom XBRL taxonomy package, if one is required by the mapping.</td>
</tr>
<tr>
<td></td>
<td>• <code>taxonomy-packages-config-file</code> - 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
### MapForce Server API

**Name**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set the path of the MapForce Server executable.</td>
</tr>
<tr>
<td>Set 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](#).

#### 6.6.1.1.1 Methods

6.6.1.1.1.1 **addParameter**

Assigns a value to a parameter defined in the mapping.

**Signature**

```java
addParameter(arg0: String, arg1: String) : void
```

**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>

6.6.1.1.1.2 **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**

```java
clearOptions() : void
```
6.6.1.1.3  clearParameterList

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

**Signature**

clearParameterList() : void

6.6.1.1.4  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**

getAPIMajorVersion() : int

6.6.1.1.5  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**

getAPIMinorVersion() : int

6.6.1.1.6  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

6.6.1.1.7  getLastExecutionMessage

Gets the message received during the last Run command.

**Signature**

getLastExecutionMessage() : java.lang.String
6.6.1.1.8  

**getMajorVersion**

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

**Signature**

```java
getMajorVersion() : int
```

6.6.1.1.9  

**getMinorVersion**

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

**Signature**

```java
getMinorVersion() : int
```

6.6.1.1.10  

**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(arg0: String) : java.lang.String
```

**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 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 <strong>Properties</strong>).</td>
</tr>
</tbody>
</table>
6.6.1.1.11  getProductName

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

Signature

getProductName() : java.lang.String

6.6.1.1.12  getProductNameAndVersion

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

Signature

getProductNameAndVersion() : java.lang.String

6.6.1.1.13  getServicePackVersion

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

Signature

getServicePackVersion() : int

6.6.1.1.14  is64bit

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

Signature

is64bit() : boolean

6.6.1.1.15  run

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

Signature

run(arg0: String) : boolean
### 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).</td>
</tr>
</tbody>
</table>

#### 6.6.1.1.16  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

```java
setOption(arg0: String, arg1: String) : void
```

### 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>
6.6.1.1.17  setServerPath

Sets the path of the MapForce Server executable.

**Signature**

```java
setServerPath(arg0:String) : void
```

**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>

6.6.1.1.18  setWorkingDirectory

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

**Signature**

```java
setWorkingDirectory(arg0:String) : void
```

**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>

6.6.1.2  MapForceServerException

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

```java
public class MapForceServerException extends Exception
```
Chapter 7

Digital Certificate Management
7 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 35.0.1 on Ubuntu 14.04).

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.

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.
7.1 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.
   
   `sudo cp /home/downloads/server_cert.crt /usr/local/share/ca-certificates/`

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

On CentOS, follow the steps below:

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

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

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

4. Use the command:

   `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
7.2 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**.
7.3 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.
7.4 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**.
7.5 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>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>C:\User</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>
7.6 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:

   ```
   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.
7.7 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 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.
mapforceserver wants to sign using key "privateKey" in your keychain.

The authenticity of "mapforceserver" cannot be verified. Do you want to allow access to this item?

? Always Allow Deny Allow
### 7.8 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](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.
Index

C
Command line usage, 48
Commands,
  exportresoucrstrings, 52
  help, 54
  licenseserver, 56
  run, 58
  setdeflang, 62

D
Digital certificates,
  exporting from Windows, 115
  in MapForce mappings, 106
  managing on Windows, 113
  transferring to Linux, 121
  transferring to Mac, 123
  trusting on Linux, 109
  trusting on Mac, 111
  trusting on Windows, 112

E
exportresoucrstrings command, 52

H
help command, 54
HTTPS,
  calling Web services through, 106

I
Installation on Linux, 33

L
licenseserver command, 56
Licensing on Linux, 36
Licensing on macOS, 44
Licensing on Windows, 27
Linux,
  executing mappings with Web service calls through HTTPS, 121
  installation on, 33
  licensing on, 36
  transferring client certificates to, 121
  trusting server certificates on, 109
Localization, 52, 62

M
Mac,
  executing mappings with Web service calls through HTTPS, 123
  transferring client certificates to, 123
  trusting server certificates on, 111
macOS,
  installation on, 41
  licensing on, 44

R
run command, 58

S
setdeflang command, 62
Setup,
  on Linux, 32
  on macOS, 40
  on Windows, 24
W

Web services,
calling through HTTPS, 106

Windows,
executing mappings with Web service calls through HTTPS, 125
installation on, 25
licensing on, 27
trusting server certificates on, 112