1spatial
YOUR WORLD SMARTER

1Integrate

Wildfly

Installation Guide

Product Version: v 4.2

Document Version: v 3.2

Document Date: 15/12/2023

1Spatial

Tennyson House

Cambridge Business Park

Cambridge

CB4 0WZ

United Kingdom

Phone: +44 (0)1223 420414

Fax: +44 (0)1223 420044

Web: www.1spatial.com

Contents

1 Introduction

This guide explains how to install 1Integrate.

The procedures apply to both Windows and Linux environments, unless specifically indicated.

For more information about new features and changes in this release, and hardware and software requirements, refer to the *1Integrate Release Notes*.

Audience

This guide is intended for personnel responsible for the installation, configuration, and administration of software.

The procedures detailed in the guide should be performed by a system administrator who is familiar with the application environment of the organisation.

Licences

1Integrate licences will be issued via email.

1Spatial Product Support

If assistance is required while installing 1Integrate, please call 1Spatial support on +44 (0)1223 423069, or visit the support section of the 1Spatial website via the Services menu at www.1spatial.com.

2 Prerequisites

Before installing 1Integrate, please ensure you have met all system requirements and installed all necessary prerequisite components:

- Check the release notes to find the system requirements for your 1Integrate version.
- "Microsoft Visual C++ Redistributable (Windows only)" on the next page
- "Configuring a Database Server for 1Integrate's repository" on the next page
- "ICU Libraries (Linux only)" on page 5
- "Required Packages (Linux only)" on page 5
- Loading Data Formats

Sizing a Server for 1Integrate

There are a number of minimum requirements for server size in order to install 1Integrate

Depending on your intended configuration you will need, as a minimum:

PER INTERFACE

- 2 CPU Cores
- 2GB RAM

PER ENGINE

- 1 CPU Core
- 1GB RAM

> **Note:** In order to achieve optimal performance, a dedicated core per engine is recommended.

While being processed, data is stored on disk. So the recommended available storage varies on a case by case basis.

While improving CPU and memory will increase performance, the biggest gains can be made from using fast disk storage, e.g. SSDs.

Storage Requirements

The database storage requirements for the 1Integrate repository (1Integrate's internal usage, not the source of spatial data to be processed) depend on the usage of 1Integrate.

- The size and number of 1Integrate entities (rules, actions, data stores and sessions).
- How many sessions will be running in parallel, and how many non-conformances or reports they will generate, before they are stopped/reset.
- The size of any data that is uploaded as files to a data store or via 1Data Gateway when 1Integrate is configured to store uploaded files in the repository.

A small set of rules and actions which read data from a database or service and do not run many sessions in parallel will need much less repository database storage (e.g. 500 MB) than one in which many huge data files are uploaded in parallel (which might need 500GB).

Microsoft Visual C++ Redistributable (Windows only)

Microsoft Visual C++ 2015-2019 64-bit Redistributable packages are required for Windows installations of 1Integrate.

These can be downloaded from the Microsoft website (**vcredist_x64.exe**).

For more information about the version numbers of pre-requisites, please refer to the relevant release notes for your installation.

Configuring a Database Server for 1Integrate's repository

1Integrate requires access to an database server to store configuration data such as rule or session definitions, and conformance results. This is referred to as the "1Integrate repository".

By default, 1Integrate will use a H2 database as the Repository. This will store the 1Integrate metadata and entities on the same file system as the 1Integrate installation.

⚠ **Warning:** This is intended for local, development work and will enable you to get started quickly. Do not use the default H2 database in production environments. See the following database options for recommended repository servers.

Supported Database Servers

The following additional Database Servers are supported:

- *Oracle* Enterprise
- *PostgreSQL (WildFly only)*
- *Microsoft SQL Server (WildFly only)*

Note: Please refer to the relevant product documentation when installing your chosen database server.

Creating the repository

You will need to create a database and user (Oracle only requires a User) for your 1Integrate repository on your selected server.

Note: When deploying 1Integrate to multiple servers, one database user is required per environment.

ORACLE

Using Oracle, 1Integrate requires a database user. The setup should be performed as recommended by your Database Administrator who must grant the user access to the database and at least the following permissions:

- `create session`
- `create sequence`
- `create table`

MICROSOFT SQL SERVER DATABASE

Using SQL Server Database, you will need a Database and a User that can access that database.

The setup should be performed as recommended by your Database Administrator who must grant the user access to the database and at least the following permissions:

- `create sequence`
- `create table`

> **Note:** Microsoft SQL server supports two methods of authenticating users, but these will differ depending on your operating system:
>
> - Windows: SQL Server authentication and Integrated Windows Authentication
> - Linux: SQL Server authentication

CREATE A POSTGRESQL USER

Create a user that can access that database with the default properties. This will ensure the user has all required privileges. Alternatively the set up should be performed as recommended by your Database Administrator who must grant the user access to the database and at least the following permissions:

- `create sequence`
- `create table`

Required Packages (Linux only)

The following package is a prerequisite for Linux installations of 1Integrate:

- **libX11**

You must install **libX11** via the system's package manager.

ICU Libraries (Linux only)

ICU libraries are required for data and timestamp support in Linux.

A root user (or a user with root access privileges) is required to copy the files from the installation package and run the `ld_config` command.

INSTALL THE ICU LIBRARIES

1. Run the `su` command to switch to the root user.
2. Copy the **.so** files from the installation folder (within the ICU folder) to the `/usr/local/lib64/` folder.
3. Create a new configuration file: **/etc/ld.so.conf.d/integrate.conf**

> **Note:** Both the .so files and the .conf file must have read permissions for all users.

4. Inside the configuration file, reference the location of the ICU libraries, for example: `/usr/local/lib64/*`
5. Enter the following `ld_config` command:

```
/sbin/ldconfig -v /usr/local/lib64/
```

Alternatively, edit **~/.bashrc** or similar for the user used to run 1Integrate to include `/usr/local/lib64` on the `LD_LIBRARY_PATH`, as in the following example:

```
# User specific
if [ -z "$LD_LIBRARY_PATH" ]; then
export LD_LIBRARY_PATH="/usr/local/lib64"
else
export LD_LIBRARY_PATH="/usr/local/lib64:$LD_LIBRARY_PATH"
fi
```

Data Store Prerequisites

1Integrate supports the following types of Data Stores for Read and Commit mapping and Copy To mapping. Pay particular attention to the prerequisites attached to certain Data Store types.

There are two methods for writing data in 1Integrate, however these are not supported by all Data Store types:

- Commit Task
- Copy To Task

Data Store Type	Read	Write	
		Commit	Copy To
Autodesk DWG	✓	✗	✗
Bentley Microstation Design V8 DGN	✓	✗	✗
Contextual Data Store	✓[1]	✓	✓
Delimiter Separated Values (DSV)	✓	✗	✓
Esri ArcGIS Services	✓	✓	✓
Esri File Geodatabase	✓	✓	✓
Esri Shapefile	✓	✗	✓
FME Flow	✓	✗	✗
GeoJSON	✓	✗	✓
Geography Markup Language (GML)	✓	✗	✗
Google BigQuery	✓[2]	✗	✓[3]
MapInfo TAB	✓	✗	✓
MariaDB	✓	✓	✓

[1]This format is only available as an Extension.
[2]This format is only available as an Extension.
[3]This format is only available as an Extension.

Data Store Type	Read	Write	
		Commit	Copy To
Microsoft Access	✓	✗	✗
Microsoft SQL Server	✓	✓	✓
Non-Conformance Report	✓	✗	✗
OGC GeoPackage	✓	✓	✓
OGC Web Feature Service (WFS)	✓ [1]	✗	✗
Oracle	✓	✓	✓
PostgreSQL/PostGIS	✓	✓	✓
Schema Only	✓	✗	✗

[1] This format only supports 2D geometries.

3 Installing 1Integrate on WildFly

> **Note:** Before proceeding, ensure you have completed all pre-requisite steps (see "Prerequisites" on page 2).

1Integrate runs as several separate applications: an interface application and one or more engine applications.

The following instructions describe how to configure an interface and an engine. This is suitable for running both parts of 1Integrate on one host.

Deployment Preparation

The following information is required prior to deploying 1Integrate:

- JDBC URL of the repository created
- Username and password for the database user
- Path of the **tnsnames.ora** file (if used)
- Location of a directory that can be used to store 1Integrate temporary files

> **Note:** On both Windows and Linux, paths must be entered with forward slashes (/) as the path separator instead of backslashes (\). Ensure that no spaces are entered after each parameter as this will result in an unsuccessful installation.

Copying the WildFly directory

Copy the WildFly directory (typically called `1Integrate-[version]-wildfly`) from the installation package to a local folder, then extract the files to a location such as `C:\Program Files\1Spatial\1Integrate\`. These files will be used to configure settings for 1Integrate in the following procedures.

In the WildFly installation, the **server-interface** directory contains the configuration for the 1Integrate interface and the **server-engine** directory contains the configuration for the engine.

Configuring System Properties

The following parameters can be configured in the **settings.properties** file, located within the **SETTINGS** folder within the installation location (e.g.

`C:\Program Files\1Spatial\1Integrate\1Integrate-[version]-wildfly\SETTINGS`).

> **Note**: On Windows, folder paths within the **settings.properties** file must be specified using a forward slash. Spaces in folder names are allowed (e.g. `C:/Program Files/1Spatial/1Integrate`).

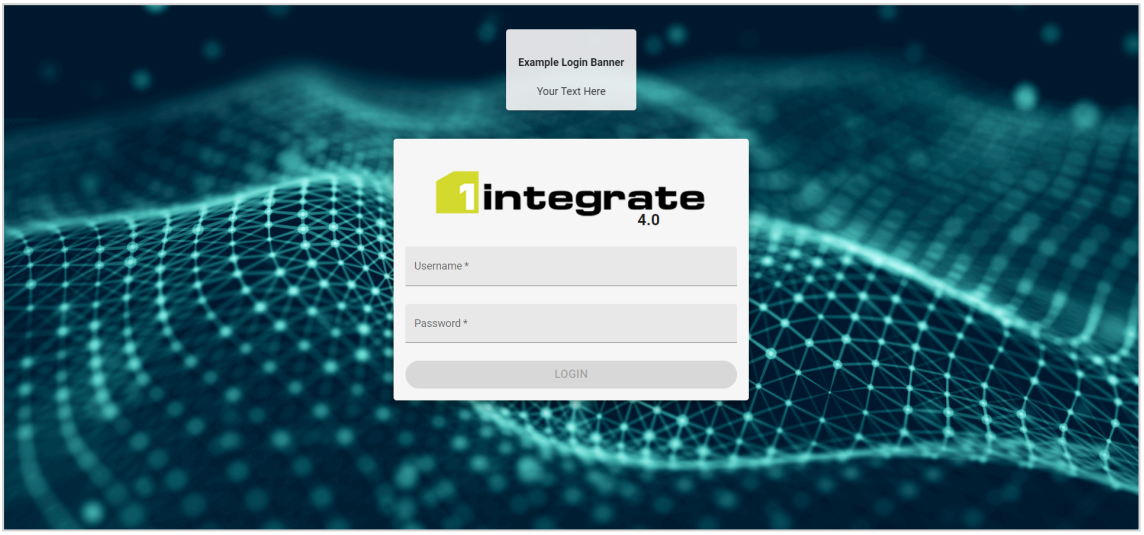
Required

Property	Type	Description
`repository.driver`	string	The database type. Defaults to a H2 database, but Oracle, SQLserver or PostGresql are also supported. **Note**: Do not use the default H2 database in production environments. See the following database options for recommended repository servers. For other supported databases the format is: • For Oracle: `jdbc:oracle:thin:@[HOSTNAME]:[PORT]/[Service]Orjdbc:oracle:thin:@[HOSTNAME]:[PORT]:[SID]` • For MS SQL: `jdbc:sqlserver://[serverName [\\instanceName] [:portNumber]];databaseName=1Integrate` For MS SQL with Windows Authentication: `repository.url=jdbc:sqlserver://[serverName [\\instanceName] [:portNumber]];integratedSecurity=true;databaseName=1Integrate` **Note**: You must define either a `serverName` or `instanceName`. See Microsoft documentation for further guidance. • For PostgreSQL: **Note**: You will only need to specify a port if you are not using the default ◦ To use the *public schema* to store the repository: `jdbc:postgresql://[HOSTNAME]:[PORT(if non-default)]/[DATABASE]` ◦ To use a *specified schema* to store the repository: `jdbc:postgresql://[HOSTNAME]:[PORT(if non-default)]/[DATABASE]?currentSchema=[REPOSITORYSCHEMA]`
`repository.url`	string	The URL for the database in which to store 1Integrate metadata and entities (Rules, Actions, Sessions and so on), and files that are uploaded to the server used by the data store (such as MapInfo Tab files).

Property	Type	Description
		By default, his will be populated by the H2 database details.
`repository.username`	string	Username to connect to the repository schema, in which to store 1Integrate metadata and entities (e.g. Rules, Actions, Sessions) as well as files that are uploaded to the server used by the data store (e.g. MapInfo Tab files).
`repository.password`	string	Password for the user described by `repository.username`.
`ls_license`	string	The location and name of the provided product licence file.
`temp.directory`	string	This is a temporary folder location for files that are eventually stored in the database.
`cache.directory`	string	The location of the cache directory. When a Session is run, a folder is created called "1Integratecache", within which the cache is stored. **Note:** This stores the data cache from data read by 1Integrate . This may require large amounts of disk space depending on the size of data being read into Sessions and the number of concurrent Sessions. **Note:** Multiple engines need to share the same cache location to allow a paused Session to be picked up and continued/rewound without restarting it. For Linux, if left blank this will default to /tmp. For Windows, if left blank this will default to `C:\Users\[user]\AppData\Local\Temp`.
`interface.http.port`	integer	The port to use for the interface. **Note:** If possible, use the default port number specified in the file. If there is a clash with another application that already uses this port number, increment the value by 1 until an unused port number is found.
`engine.http.port`	integer	The port to use for the engine. **Note:** If possible, use the default port number specified in the file. If there is a clash with another application that already uses this port number, increment the value by 1 until an unused port number is found.

Configure Login Banner

You can configure a login banner to appear at the top of the login page to include additional text for your users. This banner will not replace the 1Integrate login graphics or logo, but will appear at the top of the page.



Login page with an example banner

The banner is defined by a file that includes a snippet of text or other HTML.

> **Note:** The Login box will remain at the same location and in the foreground, so if the banner is too large it will appear behind the login box.

HTML tags, while not required, can be included for further customisations. For a full list of supported tags, see the drop-down below.

SUPPORTED HTML TAGS

<h1> <h2> <h3> <h4> <h5> <p>
 <text> <body> <title> <html> <head> <meta> <b> <i> <mark> <small> <del> <ins> <sub> <sup>

The file can have any extension and must be placed on the filesystem accessible to all interface servers in the deployment.

Configuration

1. Create your HTML file.
2. In the **settings.properties** file, add the following:

Parameter	Description
`loginBanner`	Enter a file path to a HTML document you have defined e.g:`loginBanner=C:/Apps/1Spatial/1Integrate/Banner%20Folder/Heading_Banner.html` **Note:** The HTML file must be accessible by each Interface server.

NIC/Network Adaptor Configuration

The Grid discovery used to find engines by default uses the first found non-loopback address, for example a machine with Ethernet adaptors "eth0" and "eth1" and Local Loopback "lo" will likely use "eth0".

Note: If you do not need to override the adaptor default behaviour, then the following properties do not need to be included.

CONFIGURE NIC/NETWORK ADAPTOR

The following properties must be included in the **settings.properties** file:

```
grid.local.address=[NIC Address]
grid.discovery.tcp.port=[default: 51300]
grid.communication.tcp.port=[default: 51401]
```

Where:

- **grid.local.address** - specifies the IP address of the network adaptor used for grid communication.
- **grid.communication.tcp.port** and **grid.discovery.tcp.port** - allows environments to specify known ports (for example, when using a firewall).

Note: The communication port must be a minimum of 100 greater than the discovery port, in order to avoid conflict.

Reconnect Repository on database restart

By default, when your WildFly deployment of 1Integrate disconnects from a repository (e.g. database restart or network issue) it will attempt to reconnect at the configured interval.

> **Note:** To make changes to this reconnection, see the Configuring System Properties table and locate the `repository.validation` and `repository.validation.millis` parameters detailed there and in the **settings.properties** file.

Custom Extensions

You can define custom extensions in 1Integrate, these come in two formats:

- **Custom Built-ins**

  For information on creating custom built-ins, please refer to the 1Integrate Built-in Function Programmer Guide

- **Custom data stores**

  For information on custom data stores, please contact 1Spatial support.

> **Note:** When upgrading your installation of 1Integrate, follow the steps for upgrade in the appropriate 1Integrate installation guide.

For WildFly, custom extensions can be stored in the `EXTENSIONS` folder in the 1Integrate bundle.

If you want to define your own directory outside of the 1Integrate bundle, then you will need to change the `settings.properties` file. To do this, add the following, replacing the default value:

```
1integrate.extension.dir=<directory location>
```

> **Note:** Manually setting the directory location is recommended if you would like to persist custom extensions between installations and upgrades.

Configure Memory in WildFly

To update the maximum available heap size in a 1Integrate WildFly installation, you can change the start scripts or set an environment variable.

Setting the environment variables

You can set `INTEGRATE_ENGINE_OPTS` and `INTEGRATE_INTERFACE_OPTS` as environment variables.

If you do this, settings will persist between installations as you will not need to edit the start-up scripts.

Changing the start-up scripts

WINDOWS

1. Edit **start-Interface.cmd** or **start-Engine.cmd**.
2. Locate the following lines:

```
SET "INTEGRATE_INTERFACE_OPTS=%INTEGRATE_INTERFACE_
OPTS%"
```

   or

```
SET "INTEGRATE_ENGINE_OPTS=%INTEGRATE_ENGINE_OPTS%"
```

3. Define the required size using a Java system property, e.g. to set the interface to 2GB:

```
SET "INTEGRATE_INTERFACE_OPTS=%INTEGRATE_INTERFACE_
OPTS% -Xms2g -Xmx2g"
```

> **Note**: For performance reasons it is recommended that your initial heap size matches your maximum heap size.

LINUX

1. Edit **start-Interface.sh** or **start-Engine.sh**.
2. Locate the following lines:

```
export "INTEGRATE_INTERFACE_OPTS=%INTEGRATE_
INTERFACE_OPTS%"
```

   or

```
export "INTEGRATE_ENGINE_OPTS=%INTEGRATE_ENGINE_
OPTS%"
```

3. Define the required size using a Java system property, e.g. to set the interface to 2GB:

```
export "INTEGRATE_INTERFACE_OPTS=%INTEGRATE_
INTERFACE_OPTS% -Xms2g -Xmx2g"
```

Note: For performance reasons it is recommended that your initial heap size matches your maximum heap size.

Configuring Users and Roles

Users and Roles can be edited within 1Integrate.

Note: You will need to restart 1Integrate for any changes to user and roles to take effect.

By default, 1Integrate is deployed with example users and passwords included. This enables a quick set-up process, but for security reasons it is HIGHLY RECOMMENDED that:

- As a minimum, on installation, change all passwords from the default to unique values.
- change the user names to ones relevant to your organisation.

For stronger security and management, consider using other authentication mechanisms such as using your organisation's Lightweight Directory Access Protocol (LDAP) Service e.g. Microsoft Active Directory. This ensures that passwords and usernames are not stored in the application server but managed, as normal, by an IT department.

Roles

The following roles are available in 1Integrate:

Group Permission	Description
`1int-user`	The User is designed to be applied to standard users, this role includes: • `1int-datastores-read` • `1int-datastores-write` • `1int-rules-read` • `1int-rules-write` • `1int-actions-read` • `1int-actions-write` • `1int-actionmaps-read` • `1int-actionmaps-write` • `1int-sessions-read` • `1int-sessions-write` • `1int-sessions-control` • `1int-sessions-results` • `1int-grid-read`
`1int-admin`	The Admin to includes all permissions and is designed for those that will be performing administrative functions. Includes all the permissions of 1int-user with the addition of: • `1int-grid-write` • `1int-api-keys` • `1int-access-groups` • `1int-repository`

Users

The following users are and roles are created by default upon installation:

Username	Password	Assigned permissions
INTFull	integrate1	This default User has the **1int-admin** Group Permission applied.
INTAdmin	integrate101	This default User has the **1int-admin** Group Permission applied.
INTUser	integrate102	This default User has the **1int-user** Group Permission applied.

WildFly Users

To configure Users and Permissions, navigate to the `\wildfly-[version]\SETTINGS` folder. This folder contains the following files:

- **users.properties** contains a list of usernames and passwords, in the form `username=password`.

> **Note**: All users listed in the previous table are included as default.

- **roles.properties** contains a mapping from user names to 1Integrate permissions in the form `username=permission1,permission2,permission3`

LDAP

For stronger security and management, Consider using other authentication and authorisation mechanisms such as your organisation's Lightweight Directory Access Protocol (LDAP) Service e.g. Microsoft Active Directory. This ensures that passwords and usernames are not stored in the application server but managed, as normal, by an IT department.

Authenticate using LDAP

The default WildFly configuration of storing passwords as plain text is not recommended for production use. To configure 1Integrate to use your organisation's LDAP service in WildFly, perform the following configuration:

CONFIGURE AN LDAP SERVICE

1. Find the **settings.properties** file, to locate this go to: `[1Integrate_Directory]\SETTINGS\`.
2. Add the following and fill with your LDAP values:

```
#ldap.authentication.enabled=true
#ldap.host=
#ldap.principal=
#ldap.credential=
#ldap.username.attribute=
#ldap.user.base.dn=
#ldap.group.attribute=
#ldap.group.base.dn=
```

Parameter	Expected Value
`ldap.host`	The hostname for the LDAP server.

Parameter	Expected Value
ldap.principal	The principal (username) used for authentication with the LDAP server.
ldap.credential	The credential (password) used for authentication with the LDAP server.
ldap.username.attribute	The LDAP attribute to be used for 1Integrate login names e.g. samAccountName
ldap.user.base.dn	The LDAP search root for users.
ldap.group.attribute	The group name attribute to be used from your LDAP server e.g. samAccountName
ldap.group.base.dn	The LDAP search root for groups.

Change default ldap settings

Depending on your particular LDAP implementation, you may need to change some of the default LDAP settings.

```
#ldap.protocol=ldap
#ldap.port=389
#ldap.referral.mode=IGNORE
#ldap.direct.verification=true
#ldap.recursive.search=true
#ldap.group.member.attribute=member
#ldap.authentication.level=simple
```

To do this:

1. Find the **settings.properties** file, to locate this go to: `[1Integrate_Directory]\SETTINGS\`.
2. Enter the settings that need changing for your active directory.

Parameter	Expected Value
ldap.protocol	The protocol the connection will use (ldap or ldaps).
ldap.port	The port number for your LDAP.
ldap.referral.mode	Set to FOLLOW if you have multiple LDAP servers that refer to eachother.
ldap.direct.verification	Set to FALSE if 1Integrate should query the users password out of the LDAP server for

Parameter	Expected Value
	user credential verification. **Note:** For most installations direct verification will be sufficient.
`ldap.recursive.search`	Recursive search will search through nested groups. If this setting is unnecessary for your environment, setting to `false` may improve performance.
`ldap.group.member.attribute`	The attribute on the group object that contains the members of that group.
`ldap.authentication.level`	Set to none if you want to bind to the LDAP server anonymously. **Note:** For security we would recommend all connections are authenticated.

Mapping LDAP groups to 1Integrate roles

Once you have completed your LDAP configuration, any users assigned to LDAP groups named the same as 1Integrate Permissions (e.g.`1int-repository`) will be authenticated and authorised by your LDAP system.

Authorisation in Wildfly

It is possible to configure authentication so that usernames and passwords are managed by LDAP and the authorisation (permissions assigned to specific LDAP groups) is managed within the configuration files. This removes the need for the LDAP service to know anything about 1Integrate specific permissions.

Note: In order to authorise in LDAP and authenticate in WildFly, the **roles.properties** file must be populated with the **groups** that match those from the LDAP directory with the associated permissions(s). e.g. `LDAPGroup = 1int-repository.`

Note: You can only map **LDAP Groups** to roles if using this method.

Example of the **roles.properties** file which is mapping LDAP groups to the relevant roles:

```
ADMIN\ USERS= 1int-admin
ENGINEERS=1int-user,1int-grid-write
WORKERS=1int-user
```

Password Encryption

1Integrate stores all Data Store configuration in the repository, including sensitive parameters, such as database passwords. Sensitive information is encrypted, with two options available:

- **Advanced Encryption Standard (AES)**

  This is the standard method of encryption in 1Integrate and will be used as default.

- **AES with a custom key**

  This method requires additional configuration. For more see Configuring AES with a custom Key below.

> **Note:** AES with a custom key is the most secure method of encryption in 1Integrate. Be aware that sensitive information (e.g. passwords) created on a custom key installation will be invalidated if the key is removed or changed. Sensitive information will also only be compatible with other 1Integrate installations configured with the same key.

Configuring AES with a Custom Key

AES with custom key can be configured both at the point of installation or after.

> **Note:** It is recommended you carry out the implementation of a custom key at the point of installation to avoid invalidating already stored passwords.

1. Open the `settings.properties` file for your 1Integrate installation.
2. Add and set:

   ```
   encryption.key=<Input your custom key>
   ```

3. If 1Integrate is running, restart your installation.

4 Deployment

Once you have installed 1Integrate, it is now ready to be deployed. The method of deployment will differ depending on your Operating system or deployment type.

Standard Deployment

Before running your start-up scripts, you must first ensure your deployment is configured to refer to the correct version of Java. For the Java version supported for your release of 1Integrate, please refer to the release notes.

JAVA_HOME

Once you are sure you have the correct version of Java installed, you will need to ensure that the JAVA_HOME environment variable points to this for the user running 1Integrate.

> **Note:** If the JAVA_HOME variable is already defined on your system with the correct version of Java, then you do not need to do anything.

If the JAVA_HOME environment variable is not set, you have the following options:

- Set the JAVA_HOME environment variable for the user running 1Integrate to point to the correct version of Java.
- Follow the steps below for Windows and Linux to set JAVA_HOME from the 1Integrate start-up script.

To set the JAVA_HOME environment variable in the start-up scripts, carry out the following steps:

WINDOWS

1. Edit **start-Interface.cmd** and **start-Engine.cmd**.
2. Locate the following lines:

```
@REM Optionally uncomment and set this path to be
the correct for your environment
REM SET "JAVA_HOME=C:\PROGRA~1\Java\[jdk_version]
```

3. Uncomment the second line by removing "REM", so that it begins with "SET". Ensure that it is pointing to the location of Java home for the correct version of Java installed as part of the pre-requisites. The resulting line should appear as follows:

```
SET "JAVA_HOME=C:\PROGRA~1\Java\[jdk_version]"
```

4. Once configured, run the scripts to start the 1Integrate components.

LINUX

1. Edit **start-Interface.sh** and **start-Engine.sh**.
2. Uncomment the line just below "#Optionally uncomment and set this path to be the correct for your environment":

```
export JAVA_HOME="/opt/jdk"
```

3. Once configured, run the scripts to start the 1Integrate components.

Starting 1Integrate

You can now run the start-up scripts:

- **start-Engine**
- **start-Interface**

Once run, 1Integrate will be ready to use. For information on ensuring your installation and deployment has been successful, please refer to "Testing the Installation" on page 29.

Configure additional Engines

1Integrate can be deployed with multiple engines each of which can run either on the same server as the interface or on separate machines (as long as they are on the same network as the interface).

Engines can be run in parallel, deployed using different port numbers. This requires a license from 1Spatial that enables the required number of parallel engines.

> **Note:** On a machine running Windows, this may present a path name length error. If this occurs, copy the **server-engine** folder to a higher level directory, rename it, and paste it back into the **wildfly-[version]** folder.

> **Note:** For more advanced environments, contact either your distributor or 1Spatial prior to continuing to discuss an appropriate deployment plan.

You will need to copy and edit a different file, depending on if you are using Windows or Linux.

WINDOWS

1. Create a copy of the **server-engine** folder and rename it **server-engine2** etc.

> **Note:** You must suffix server-engine with an integer.

2. Inside the cloned directory, open the **configuration\engine.conf.bat**
3. Update `SET "ENGINE_NUMBER=1"` to the corresponding number e.g `SET "ENGINE_NUMBER=2"`
4. Create a copy of the **start-Engine.cmd** file and rename it.

> **Note:** When deploying multiple engines, ensure you run each **start-Engine.cmd** file you have created.

5. Edit this file to refer to `"ENGINE_NUMBER=2"` instead of `"ENGINE_NUMBER=1"` (or `"ENGINE_NUMBER=3"` for the third engine etc.)

LINUX

1. Create a copy of the **server-engine** folder and rename it **server-engine2**.

> **Note:** You must suffix server-engine with an integer.

2. Create a copy of the **start-Engine.sh** file and rename it **start-Engine2.sh**.

   Edit this file to refer to **server-engine2** instead of **server-engine** (or **server-engine3** for the third engine etc.) and change the port offset at the end of the WildFly startup line from 100 to 200 (or 300 for the third engine etc.).

Example: The line
`./bin/standalone.sh -P=SETTINGS/settings.properties -Djboss.server.base.dir=server-engine =Djboss.socket.binding.port-offset=100`
would be changed to
`./bin/standalone.sh -P=SETTINGS/settings.properties -Djboss.server.base.dir=server-engine2 =Djboss.socket.binding.port-offset=200`.

Engine Labels

You can label engines you are deploying to group them, and control which Sessions run on different groups.

WINDOWS

1. Open the `server-engine` directory for the engine you will be applying labels to.
2. Within the `configuration` directory, open the **engine.conf.bat** file.
3. Locate and uncomment the following lines:

```
SET "engine.labels="
```

4. Add engine labels as a comma separated list:

```
SET "engine.labels=label1,label2,label3"
```

5. Repeat the steps for each engine being deployed.

LINUX

1. Open the **start-Engine.sh** script for the engine you are applying labels to.
2. Locate and uncomment the following lines:

```
export engine.labels=""
```

3. Add engine labels as a comma separated list:

```
export engine.labels="label1,label2,label3"
```

4. Repeat the steps for each engine being deployed.

Service Deployment (Windows Only)

You can deploy 1Integrate as a service through additional configuration and by running two scripts supplied in the installation package.

JAVA_HOME

Before running your start-up scripts, you must first ensure your deployment is configured to refer to the correct version of Java. For the Java version supported for your release of 1Integrate, please refer to the release notes.

Once you are sure you have the correct version of Java installed, you will need to ensure that the JAVA_HOME variable is defined.

> **Note:** If the JAVA_HOME variable is already defined on your system with the correct version of Java, then you do not need to do anything.

If the JAVA_HOME environment variable is not set, you have the following options:

- Set the JAVA_HOME environment variable for the user running 1Integrate to point to the correct version of Java.
- Follow the steps below to set the JAVA_HOME environment variable from the 1Integrate service start-up scripts.

To set the JAVA_HOME environment variable in the start-up scripts, carry out the following steps:

1. Edit **start-Interface-Service.cmd** and **install-Engine-Service.cmd**.
2. Locate the following lines:

```
@REM Optionally uncomment and set this path to be
the correct for your environment
REM SET "JAVA_HOME=C:\PROGRA~1\Java\[jdk_version]
```

3. Uncomment the second line and remove "REM", so that it begins with "SET". Ensure that it is pointing to the location of Java home for the correct version of Java installed as part of the pre-requisites. The resulting line should appear as follows:

```
SET "JAVA_HOME=C:\PROGRA~1\Java\[jdk_version]"
```

4. Once configured, run the scripts to start the 1Integrate components.

Using Specific User Accounts

By default, the services run as the local system user. To use a specific user account you will need to run the **start-Interface-Service.cmd** and **install-Engine-Service.cmd** with the following parameters:

```
start-Interface-Service.cmd <domain>\<username>
<password>
```

```
start-Engine-Service.cmd <domain>\<username>
<password>
```

> **Note:** Using a named user account is required to use AzureAD authentication for SQL Server connections made by 1Integrate

Deploying the Service

Once you have checked your `JAVA_HOME` variable, you can deploy 1Integrate as a service by running the following scripts:

> **Note:** Both scripts must be run by an admin user account.

install-Interface-Service.cmd

The interface script will create a service with the name "**1INTService**" and the display name "**1Integrate Interface [version]**".

install-Engine-Service.cmd

The engine script will automatically create *N* services with the name "**1INTEngine[N]Service**" and the display name "**1Integrate Engine [version] #[N]**" based on how many engines have already been configured.

Configure additional engines (Service)

1. Create a copy of the **server-engine** folder and rename it **server-engine2** etc.

> **Note:** You must suffix server-engine with an integer.

2. Inside the cloned directory, open the **configuration\engine.conf.bat**
3. Update `SET "ENGINE_NUMBER=1"` to the corresponding number e.g `SET "ENGINE_NUMBER=2"`
4. You are now ready to start the **install-Engine-Service.cmd** script, which will deploy all configured engines.

> **Note:** Along with the installation there are two uninstallation scripts. You will not need to define `JAVA_HOME`, as per the installation script. Uninstallation must be carried out prior to an upgrade, using the scripts packaged with the current installation, not the upgrade.

You can label engines you are deploying to group them, and control which Sessions run on different groups.

WINDOWS

1. Open the `server-engine` directory for the engine you will be applying labels to.
2. Within the `configuration` directory, open the **engine.conf.bat** file.
3. Locate and uncomment the following lines:

```
SET "engine.labels="
```

4. Add engine labels as a comma separated list:

```
SET "engine.labels=label1,label2,label3"
```

5. Repeat the steps for each engine being deployed.

5 Testing the Installation

> **Note:** Empty your browser cache before testing your installation.

1Integrate can be accessed through the following site:
http://[machine]:[service_port]/1Integrate

> **Note:** Use the port number specified during installation, by default this is 8080.

Log in as a user with administrator permissions.

To verify the engine installations, click the **Admin** tab and check that the Grid Topology matches the number of interfaces and engines installed.

6 Upgrading an Installation

> **Note:** Before performing an upgrade to an existing installation, ensure **all sessions are stopped** and perform a **backup** of your repository.

> **Note:** These instructions apply when upgrading from one version to the immediately subsequent release only (e.g. from 1.1 to 1.2). If you are performing an upgrade from any older version, please consult your release notes or contact 1Spatial Support.

UPGRADE AN INSTALLATION

Upgrading an installation on Wildfly consists of unpacking your new release folder, copying across your existing settings files, and editing the Java location in your interface and engine component files.

> **Note:** The following instructions use an example where currently version 1.A is installed, and we want to upgrade to 1.B. We have used `C:\1Spatial` as a directory within which to store our installation package files, organised into `C:\1Spatial\Product-1.A` and `C:\1Spatial\Product-1.B`.

1. Unzip your new Release Package, and the Wildfly folder within it (e.g. `C:\Program Files\1Spatial\Product-1.B\Product-1.B_wildfly\wildfly-[version]`).
2. Copy the following properties files from the **SETTINGS** folder in your *existing* installation directory (e.g. `C:\Program Files\1Spatial\Product-1.A\Product-1.A_wildfly\wildfly-[version]\SETTINGS`, into the **SETTINGS** folder in your *new* installation directory:
   - **settings.properties**
   - **roles.properties**
   - **users.properties**
3. Copy any custom extensions such as built-ins or data stores (**.jar** files) from your *existing* **EXTENSIONS** directory into the corresponding **EXTENSIONS** folder within your new installation directory. If you have previously defined a directory containing custom extensions, ensure the `1integrate.extension.dir` parameter in your new

settings.properties file points to the correct location.

4. Before running your start-up scripts, you must first ensure your deployment is configured to refer to the correct version of Java. For the Java version supported for your release of 1Integrate, please refer to the release notes.

JAVA_HOME

Once you are sure you have the correct version of Java installed, you will need to ensure that the JAVA_HOME environment variable points to this for the user running 1Integrate.

> **Note:** If the JAVA_HOME variable is already defined on your system with the correct version of Java, then you do not need to do anything.

If the JAVA_HOME environment variable is not set, you have the following options:

- Set the JAVA_HOME environment variable for the user running 1Integrate to point to the correct version of Java.
- Follow the steps below for Windows and Linux to set JAVA_HOME from the 1Integrate start-up script.

To set the JAVA_HOME environment variable in the start-up scripts, carry out the following steps:

WINDOWS

1. Edit **start-Interface.cmd** and **start-Engine.cmd**.
2. Locate the following lines:

```
@REM Optionally uncomment and set this path to be
the correct for your environment
REM SET "JAVA_HOME=C:\PROGRA~1\Java\[jdk_version]
```

3. Uncomment the second line by removing "REM", so that it begins with "SET". Ensure that it is pointing to the location of Java home for the correct version of Java installed as part of the pre-requisites. The resulting line should appear as follows:

```
SET "JAVA_HOME=C:\PROGRA~1\Java\[jdk_version]"
```

4. Once configured, run the scripts to start the 1Integrate components.

LINUX

1. Edit **start-Interface.sh** and **start-Engine.sh**.
2. Uncomment the line just below "`#Optionally uncomment and set this path to be the correct for your environment`":

```
export JAVA_HOME="/opt/jdk"
```

3. Once configured, run the scripts to start the 1Integrate components.