Ubisense Installation Guide

for SmartSpace with Site connector

Version 3.8

Part Number: SS/SC_INS_3.8_EN

Ubisense Limited, St Andrew's House, St Andrew's Road, Cambridge CB4 1DL, United Kingdom.

Telephone: +44 (0)1223 535170. Website: www.ubisense.com.

Ubisense Limited
St Andrew’s House
St Andrew’s Road
Cambridge CB4 1DL
United Kingdom

Tel: +44 (0)1223 535170

WWW: https://www.ubisense.com

All contents of this document are subject to change without notice and do not represent a commitment on the part of Ubisense. Reasonable effort is made to ensure the accuracy of the information contained in the document. However, due to on-going product improvements and revisions, Ubisense and its subsidiaries do not warrant the accuracy of this information and cannot accept responsibility for errors or omissions that may be contained in this document.

Information in this document is provided in connection with Ubisense products. No license, express or implied to any intellectual property rights is granted by this document.

Ubisense encourages all users of its products to procure all necessary intellectual property licenses required to implement any concepts or applications and does not condone or encourage any intellectual property infringement and disclaims any responsibility related thereto. These intellectual property licenses may differ from country to country and it is the responsibility of those who develop the concepts or applications to be aware of and comply with different national license requirements.

Contents

Ubisense Installation Guide for SmartSpace with Site connector

A SmartSpace installation comprises the core Ubisense software required to run SmartSpace along with any further software for additional SmartSpace features you have licensed.

Site connector connects Ubisense platforms on separate networks using TCP/IP.

These instructions guide you through installing Ubisense SmartSpace software. They include a description of the organization and installation of Ubisense software across server, admin and client machines; prerequisites for installation; and the stages of the installation process.

SmartSpace installations can range in size from a single computer running all SmartSpace software, for example for evaluation purposes, to several servers with associated admin and client machines in a large industrial setting. SmartSpace can be installed and run on Windows and Linux computers and an installation may include a mix of these, for example with Linux servers and Windows client machines, and the instructions that follow include information for both Linux and Windows.

For Windows computers, installation files (.msi) are supplied. However, because of the variations between Linux distributions and between package management systems used by different Enterprise configurations, Ubisense do not provide an automatic installation method for Linux. Instead the instructions describe the necessary state of a Ubisense platform installation on Linux, prerequisites for the installation, the layout and permissions expected, and example scripts.

⚠ **From SmartSpace version 3.8, installed Ubisense applications are 64 bit. For interoperability between 32-bit and 64-bit applications, you require the new 64-bit Ubisense Application Manager. Using an older 32-bit version of the application will prevent you from downloading other 64-bit Ubisense applications, for example SmartSpace Config.**

The next section provides an overview of the installation process and subsequent sections take you through installing SmartSpace with Site connector step by step.

The Ubisense Installation Process

Ubisense requires software to be installed on three types of machine: *server*, *admin* and *client*.

- Servers run the core and controller software, and the Ubisense platform from which you can start and stop the Ubisense servers. Ubisense servers can run on either Windows or Linux (see *Requirements* for supported versions).
- The software installed on an admin machine enables you to manage the installation and deployment of SmartSpace features across your entire SmartSpace installation.
- On client machines, you will find the SmartSpace applications available to users according to the SmartSpace features you have licensed.

Depending on your requirements, you might install all three SmartSpace machines on a single computer or you might spread the installation over a number of machines.

The following are the steps required to set up a new installation:

Install the Ubisense Software

1. Install the Ubisense server software onto the relevant machine(s).
2. Start the core server and service controllers.
3. Install licenses.
4. Install the Ubisense admin software onto the relevant machines.
5. Install and deploy licensed SmartSpace features.
6. Install the Ubisense client software onto the relevant machines.
7. Download SmartSpace software to client machines.

Depending on the features you have licensed, users may access SmartSpace via a web interface, for example to view Web maps. In this case additional steps are required to set up a web server. You do not need to install additional software for end users to access the browser-based features.

Install Site connector

After you have installed the core software, you can install Site connector by installing and deploying additional services. Follow these steps to install Site connector:

1. Install the Site Connector for Servers service.
2. Install the Site Connector Client or the Site Connector Client for Servers services.

Requirements

This section describes the hardware and software prerequisites for a SmartSpace with Site connector installation.

Server Hardware Requirements

Exact requirements for server hardware will depend on such things as the number of sensors and tags in your installation or the number of users querying any browser-based applications you have licensed. Contact Ubisense for further guidance on the specific requirements for your installation.

The following is an illustration of an installation with two servers running DIMENSION4 and SmartSpace with the Visibility component.

Feature	Server 1 (DIMENSION4 + SmartSpace core)	Server 2 (Windows server server running IIS)
Processor	Quad-core Intel® Xeon® processors 3400 series	16-core Intel® Xeon® processors
Memory	8 GB	8 GB
Ethernet Interface	Gigabit Network Adapter	Gigabit Network Adapter
Virtualization	For information about virtualization, contact Ubisense Support.	For information about virtualization, contact Ubisense Support.

Server Software Requirements

Ubisense supports the following operating systems:

Windows

- All versions of Windows server currently supported by Microsoft from Windows Server 2012 onwards.
- Additionally, for client machines (and for proofs of concept, with the agreement of Ubisense Support), versions of Windows from Windows 8.0 which are currently supported by Microsoft.

Note: "Home" versions of Windows, e.g. Windows 10 Home, are *not* suitable for use as SmartSpace *servers* in production environments, although they may be acceptable for evaluation purposes.

Linux

- SUSE Linux Enterprise Server (SLES) 12 SP2
- Red Hat® Enterprise Linux® (RHEL) 8+

If you are installing a web server for use with the Visibility component, see the additional considerations discussed in *Linux Requirements for SmartSpace Web*.

Microsoft .NET Runtimes

Some SmartSpace features use Microsoft .NET runtimes. These are:

- Reporting
- External data connector
- Typed API
- Real-time rules engine
- SmartSpace Web site, includes:
  - Web maps
  - Web forms
  - HMIs
  - ObjectView API
  - Operations web interface
- Application REST API
- .NET API
- Application .NET API (Managed Browser)
- AVL/GPS connect (deprecated from version 3.9 and above)
- the translation tool used in localization and tailoring

Note: On Linux, if Microsoft .NET is already installed on the server, the websites will use that runtime provided it is the version of .NET required by that release of SmartSpace. If a server-wide installation of .NET is *not* provided, then a Ubisense-supplied isolated local runtime will be used, shared only by the platform services that use it. It is important to note that if a version of .NET

Core is installed that is different to the version required by SmartSpace (for example 3.1 if SmartSpace wanted 6.0), then the website will not work, and there is no fallback to the isolated local runtime.

Note: If Microsoft .NET is installed on a Linux server, and is *not* in the default path of **/usr/share/dotnet**, then it is necessary to ensure that the environment variable **DOTNET_ROOT** is set to point to where the .NET runtime is located in your Linux file system. This is usually added to the user profiles automatically, but may not be available to systemd services, so if running core and controller from systemd, it is important to add **DOTNET_ROOT** to the systemd service definitions. See the sample systemd scripts in *Installing the Server Software on Linux* for an example.

For links to the requisite downloads, see *https://dotnet.microsoft.com/en-us/download/dotnet/6.0*.

Current versions of .NET for SmartSpace are:

- For Windows, you *must* install the latest ASP.NET Core Runtime 6.0.x Hosting bundle.
- For Linux, we *strongly recommend* you install the latest ASP.NET Core Runtime 6.0.x.

For a list of SmartSpace releases and their respective .NET versions, see the Ubisense Documentation Portal.

Visual C++

With the release of SmartSpace 3.8, Windows and Linux 32-bit executables were replaced with 64-bit executables. If you are installing or upgrading to SmartSpace 3.8 or higher, you need to ensure you have the latest C++ runtime dlls by installing the Visual C++ Redistributable Package from *https://aka.ms/vs/17/release/vc_redist.x64.exe*.

Typed API modules (.NET API)

If you built Typed API modules with versions of SmartSpace earlier than 3.8, you need to rebuild these modules using x64.

C++ API

If you have been using a version of the C++ API earlier than SmartSpace 3.8, you must upgrade to a 64-bit version of Microsoft Visual Studio 2015 or later.

ManagedBrowser (Application .NET API)

If you have ManagedBrowser clients /services that you want to rebuild against the latest SmartSpace release (not required), you will have to switch to x64.

Database Server for Reporting or RDBMS map

Either of the following are required *only* if the Reporting component or the RDBMS map feature is licensed:

SQL Server

Database versions: 2008 R2 or higher.

- Windows servers using ODBC and SQL Server Native Client library
- Linux servers using Microsoft ODBC Driver 17 for SQL Server

Oracle

Database versions: 11G R2 or higher.

- Windows servers using Oracle Instant Client 21.x library
- Linux servers using Oracle Instant Client 21.x library

For information on configuring servers for use with the Reporting component, see SmartSpace Reporting on the Ubisense Documentation Portal.

For information on configuring servers for use with the RDBMS map feature, see RDBMS map configuration on the Ubisense Documentation Portal.

Web Server Requirements

If you have also licensed any SmartSpace features which provide web-based functionality, you must configure a web server. For example, for:

- Web maps
- Web forms
- HMIs
- Operations web interface
- Web reports

Additionally, if you have licensed the feature, you must configure a separate web server for the Application REST API.

From release 3.4, SmartSpace Web and the SmartSpace REST API can be deployed on either Windows or Linux web servers.

- Under Windows, the websites are installed using Windows Installer, and are controlled and hosted under IIS (described in *Installing SmartSpace Web on Windows*).
- On Linux, the websites are controlled by the Ubisense platform controller, as services, and by default require a reverse proxy (such as Apache2) to make them available to the network (see *Installing SmartSpace Web on Linux*).

If you are using unicast cluster support in your Ubisense platform, see also the *SmartSpace Unicast Cluster Setup Guide* on the Ubisense Documentation Portal for additional configuration requirements.

Supported Browsers

Supported browsers for use with SmartSpace's web-enabled features are recent versions of:

- Microsoft Edge
- Chrome
- Chrome for Android
- iOS Safari

Additionally, recent versions of the following browsers work but are not explicitly supported:

- Firefox
- Opera
- Safari

Internet Explorer 11 is deprecated and SmartSpace's web-enabled features are not guaranteed to work with this browser.

Additional Requirements for Linux Installations

The following are requirements for Linux computers onto which server or admin machines are to be installed:

- Platform executables require a 64-bit libstdc++.so.6
- The firewall should be disabled on the server
- In order to work around kernel bind(0) behavior, the local dynamicport range should be changed

  - **Either**: place the following in an init script such as /etc/rc.d/rc.local: sysctl -w net.ipv4.ip_local_port_range=32768 49978
  - **Or**: place the following in /etc/sysctl.conf: net.ipv4.ip_local_port_range=32768 49978

After reboot or applying sysctl -p, the property net.ipv4.ip_local_port_range can be checked with sysctl -a

Platform user

A user should be configured to execute platform services. We will refer to this as the *platform user*.

Operations group

A group should be configured for operations. Users in this group should be able to perform production operations, including starting and stopping the platform services, making and restoring backups, and performing other diagnostic and support roles, such as license management and platform service upgrades. The platform user might be in the operations group.

Windows client machine

In order to run the SmartSpace Config software, you will need access to a Windows computer to install a client machine.

Unzipping Software to a Distribution Directory

The SmartSpace software is supplied as a zipfile with the name SmartSpace followed by numbers indicating the version of the software, for example **SmartSpace_3_3_669.zip**. Before you install SmartSpace, you need to unzip this file into a *distribution directory* accessible to the machines on which you will be installing the software.

Installing the Server Software on Windows

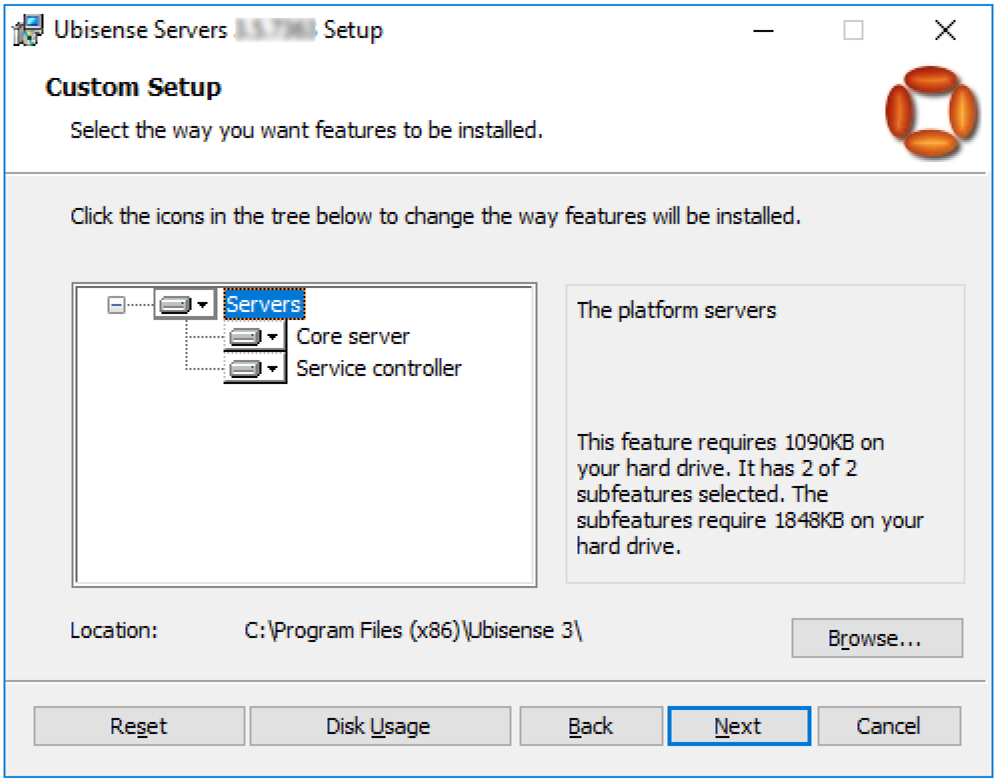
Installing the Server Software

On each machine you want to use as a server, you must install the Ubisense server software.

During the installation process, for each server machine you can choose to install either the core server or service controller or both. If you intend to run SmartSpace on a single server, you need to install both the core server and service controller on that machine. For an installation with more than one server, you need to run the core server on one machine only and the service controller on the rest, and you can install the components accordingly.

To install the server software:

1. Go to the **servers\windows** directory of your SmartSpace distribution directory.
2. Double-click the **UbisenseServers.msi** file and the Ubisense Servers Setup wizard appears.
3. Click **Next** to display the Custom Setup dialog.



Ubisense Servers [redacted] Setup

Custom Setup

Select the way you want features to be installed.

Click the icons in the tree below to change the way features will be installed.

- Servers
  - Core server
  - Service controller

The platform servers

This feature requires 1090KB on your hard drive. It has 2 of 2 subfeatures selected. The subfeatures require 1848KB on your hard drive.

Location: C:\Program Files (x86)\Ubisense 3\

4. Choose the components to install.

By default, all features are selected. Choose whether to install or exclude items using the dropdowns beside their names. **Reset** returns you to the default selection.

5. Choose the location for the software.

   You can accept the default **C:\Program Files (x86)\Ubisense 3** or click **Browse** to select another destination.

6. Click **Next** and click **Install**.
7. When the installation is complete, click **Finish** to close the Ubisense Servers Setup wizard.

You have now installed Ubisense Platform Control and the Ubisense server software onto your computer. Using Platform Control to start the server software is described in the next section.

Starting the Server Software

After you have installed the server software, you need to start the core server and service controller(s).

To start the server software:

1. From a SmartSpace server, run Platform Control.
2. For a new installation, you need to choose a location for your dataset:

   In the **Properties** section, browse to the required location (creating a new folder, if needed) and click **OK (new)**.

3. Start the core server and service controller by:
   a. Selecting **UbisenseCoreServer 3** and then clicking **Start**.
   b. Selecting **UbisenseServiceController 3** and then clicking **Start**.

      See the information below for information on starting services with a single server or multiple servers.

      You are offered only the server components installed on the machine (see *Installing the Server Software*).

   The status of each service changes to **to be started**.

4. Click **Apply**. The status of each service changes to **running**.

Running SmartSpace on a Single Server

If you want to configure SmartSpace to run on a single server, run Platform Control on the server and:

- In Services, ensure you have started *both* the core server and the service controller.
- *Don't* select **run in standalone mode** if you want access to your network (and to sensors).

Running SmartSpace on Multiple Servers

If you want to configure SmartSpace to run on more than one server, you must:

- Assign one server to be the core server and *on this machine only* run Platform Control and in Services start the core server.
- On all other server machines, run Platform Control and in Services start the service controller.
- *Don't* select **run in standalone mode** if you want access to your network (and to sensors).

Backing up your Dataset

After you have set up your SmartSpace installation, ensure that you back up your dataset occasionally, so that you can recover your data. To take a backup, use the **Backup Dataset** option, and then compress the folder.

You can also use the **ubisense_backup.exe** command-line tool from the **tools\windows** folder of your distribution directory to backup your dataset.

Installing the Server Software on Linux

Installing the Server Software

For Linux servers, there are two executables: **ubisense_core_server** and **ubisense_local_control**. You can find them in the following locations in the distribution directory:

```
    servers/linux/ubisense_core_server
    servers/linux/ubisense_local_control
```

If you want to run SmartSpace on a single server, copy both of these files to that machine.

If you want to run SmartSpace on several servers, copy **ubisense_core_server** onto one server machine only and **ubisense_local_control** onto the remainder of the machines.

Starting the Server Software

On each server machine, one or both of the **ubisense_core_server** and **ubisense_local_control** services should be executed on startup, depending on whether the machine is to act as a core server, a service controller, or both. These services should be executable by the platform user, and no other user, to avoid accidental execution. Because of the variations between Linux distributions, Ubisense do not ship standard startup scripts for these executables, but examples are provided:

Sample init.d scripts for core server and service controller

Core Server

```
#!/bin/bash
#
# Init file for Ubisense core platform server
#
# chkconfig: 345 98 02
# description: Ubisense core platform for linux
# processname: ubisense_core_server
# config: /etc/ubisense.conf

# source function library
```

```
if [ -e /etc/rc.d/init.d/functions ]
then
   . /etc/rc.d/init.d/functions
else
 # steal status() from /etc/rc.d/init.d/functions on a RH box
 status() {
       local base=${1##*/}
       local pid

       # Test syntax.
       if [ "$#" = 0 ] ; then
               echo $"Usage: status {program}"
               return 1
       fi

       # First try "pidof"
       pid=`pidof -o $$ -o $PPID -o %PPID -x $1 || \
            pidof -o $$ -o $PPID -o %PPID -x ${base}`
       if [ -n "$pid" ]; then
               echo $"${base} (pid $pid) is running..."
               return 0
       fi

       # Next try "/var/run/*.pid" files
       if [ -f /var/run/${base}.pid ] ; then
               read pid < /var/run/${base}.pid
               if [ -n "$pid" ]; then
                       echo $"${base} dead but pid file exists"
                       return 1
               fi
       fi
       # See if /var/lock/subsys/${base} exists
       if [ -f /var/lock/subsys/${base} ]; then
               echo $"${base} dead but subsys locked"
               return 2
       fi
       echo $"${base} is stopped"
       return 3
 }
fi
```

```
# pull in sysconfig settings
[ -f /etc/ubisense.conf ] && . /etc/ubisense.conf

PLATFORM_USER=${PLATFORM_USER:-platform}
UBISENSE_CORE_SERVER=/home/platform/bin/i586_linux_2.6/ubisense_core_server
export UCONFIG=/etc/ubisense/platform.conf

RETVAL=0
prog="ubisense"

start()
{
       echo -n $"Starting ubisense_core_server:"
       if [ -e /etc/rc.d/init.d/functions ]
       then
          daemon --check ubisense_core_server --user=platform ${UBISENSE_CORE_SERVER} -d
       else
          startproc -u platform ${UBISENSE_CORE_SERVER} -d
       fi
        touch /var/lock/subsys/ubisense_core_server
       echo
}

stop()
{
       echo -n $"Stopping ubisense_core_server:"
        if [ -e /etc/rc.d/init.d/functions ]
       then
           killproc ubisense_core_server
       else
           killproc ${UBISENSE_CORE_SERVER}
       fi
        rm -f /var/lock/subsys/ubisense_core_server
       echo
}


case "$1" in
       start)
               start
               ;;
       stop)
               stop
               ;;
       restart)
                stop
               start
               ;;
       status)
                status ubisense_core_server
               ;;
       *)
               echo $"Usage: $0 {start|stop|restart|status}"
               RETVAL=1
esac
exit $RETVAL
```

Local Controller

```
#!/bin/bash
#
# Init file for Ubisense local controller
#
# chkconfig: 345 99 01
# description: Ubisense local controller for linux
# processname: ubisense_local_control
# config: /etc/ubisense.conf

# source function library
if [ -e /etc/rc.d/init.d/functions ]
then
   . /etc/rc.d/init.d/functions
else
 # steal status() from /etc/rc.d/init.d/functions on a RH box
 status() {
       local base=${1##*/}
       local pid

       # Test syntax.
       if [ "$#" = 0 ] ; then
               echo $"Usage: status {program}"
               return 1
       fi

       # First try "pidof"
       pid=`pidof -o $$ -o $PPID -o %PPID -x $1 || \
            pidof -o $$ -o $PPID -o %PPID -x ${base}`
       if [ -n "$pid" ]; then
               echo $"${base} (pid $pid) is running..."
               return 0
       fi

       # Next try "/var/run/*.pid" files
       if [ -f /var/run/${base}.pid ] ; then
               read pid < /var/run/${base}.pid
               if [ -n "$pid" ]; then
                       echo $"${base} dead but pid file exists"
                       return 1
               fi
       fi
       # See if /var/lock/subsys/${base} exists
       if [ -f /var/lock/subsys/${base} ]; then
               echo $"${base} dead but subsys locked"
               return 2
       fi
       echo $"${base} is stopped"
       return 3
 }
fi

# pull in sysconfig settings
[ -f /etc/ubisense.conf ] && . /etc/ubisense.conf
```

```
PLATFORM_USER=${PLATFORM_USER:-platform}
UBISENSE_LOCAL_CONTROL=/home/platform/bin/i586_linux_2.6/ubisense_local_control
export UCONFIG=/etc/ubisense/platform.conf

RETVAL=0
prog="ubisense"

start()
{
       echo -n $"Starting ubisense_local_control:"
       if [ -e /etc/rc.d/init.d/functions ]
       then
             daemon --check ubisense_local_control --user=platform ${UBISENSE_LOCAL_CONTROL}
-d
        else
             startproc -u platform ${UBISENSE_LOCAL_CONTROL} -d
       fi
       touch /var/lock/subsys/ubisense_local_control
       echo
}

stop()
{
       echo -n $"Stopping ubisense_local_control:"
       if [ -e /etc/rc.d/init.d/functions ]
       then
          killproc ubisense_local_control
       else
         killproc ${UBISENSE_LOCAL_CONTROL}
       fi
        rm -f /var/lock/subsys/ubisense_local_control
       echo
}


case "$1" in
       start)
               start
               ;;
       stop)
               stop
               ;;
       restart)
                stop
               start
               ;;
       status)
                status ubisense_local_control
               ;;
       *)
               echo $"Usage: $0 {start|stop|restart|status}"
               RETVAL=1
esac
```

```
exit $RETVAL
```

Sample systemd scripts for a RedHat Linux machine

The following example illustrates the use of systemd scripts for SmartSpace with the core and controller executables on a single Red Hat® Linux machine.

> The instructions assume the core server and local controller executables (**ubisense_core_server** and **ubisense_local_control**) are in **/home/platform/bin/i586_linux**. If this is not the case, the service files (**ubisense_core_server.service** and **ubisense_local_control.service**) will have to be updated to reflect the location of the executable.

1. Add a target file **ubisense_service.target** in **/etc/systemd/system** containing the following:

```
[Unit]
Description=ubisense_service Target
Requires=multi-user.target
After=multi-user.target
AllowIsolate=yes

[Install]
WantedBy=multi-user.target
```

2. Run the following commands:

```
systemctl list-units --type service
systemctl daemon-reload
systemctl enable ubisense_service.target
systemctl isolate ubisense_service.target
ln -sf /etc/systemd/system/ubisense_service.target
/etc/systemd/system/default.target.wants/
```

3. Reboot the machine.
4. Check the status of the target using the command below to make sure the target is active and running:

```
systemctl list-units --type target
```

5. Add a service file **ubisense_core_server.service** in **/etc/systemd/system** containing the following:

```
[Unit]
Description=ubisense_core_server daemon
After=multi-user.target

[Service]
Type=forking
ExecStart=/home/platform/bin/i586_linux_2.6/ubisense_core_server -d
Environment=DOTNET_ROOT=/usr/lib64/dotnet
User=platform
Group=operations

[Install]
WantedBy=ubisense_service.target
```

6. Add a service file **ubisense_local_control.service** in **/etc/systemd/system** containing the following:

   ```
   [Unit]
   Description=ubisense_local_control Daemon
   After=multi-user.target

   [Service]
   Type=forking
   ExecStart=/home/platform/bin/i586_linux_2.6/ubisense_local_control -d
   User=platform
   Group=operations
   TasksMax=3072

   [Install]
   WantedBy=ubisense_service.target
   ```

   Note: The **TasksMax** value highlighted above is an example only and may need to be changed depending on the version of Linux you are using and the particular Ubisense products installed.

7. Run the following commands:

   ```
   systemctl daemon-reload
   systemctl enable ubisense_local_control.service
   systemctl enable ubisense_core_server.service
   ```

8. Reboot the machine.
9. To list the status of the services run the following command:

   ```
   systemctl list-units --type service
   ```

Service security and Authentication using Cached Service Credentials on Linux

If you are running on a Linux server and configure a security policy in Security Manager that requires services to authenticate as a user, using **ubisense_cache_service_credentials**, then you must run the core and controller software with the **-d** flag (as shown in the examples above). Otherwise all services will still have a connected stdin/stdout and will attempt to prompt for credentials rather than reading the cached service credentials. See *Ubisense Security Manager* on the Ubisense Documentation Portal for information on configuring security.

Configuration Parameters

On Linux, the local configuration parameters for each core or controller machine are set, by default, in a configuration file. This file contains configuration parameters for the local platform processes, such as the location of the dataset and the networking mode. The default location expected by all platform executables is **/etc/ubisense/platform.conf**. If another location is to be used, then the environment variable **UCONFIG** should be defined: it is recommended this be set in startup scripts for all users on the server, but it certainly is required for the platform user and all users in the operations group. **UCONFIG** should be the full path of **platform.conf** in its desired location.

Configuration parameters are each on a single line in the file, with a colon and white-space separating the name of the parameter from the value. For example:

```
platform_dataset: /mnt/syn013/ubisense/production/dataset
no_multicast_mode: 1
server_unicast_addresses: 10.1.5.207,10.1.16.73
```

Warnings and Errors

Immediate warnings and errors when starting the two platform service executables are logged to the Linux syslog. On a typical Linux distribution they will either be in **/var/log/messages** or **/var/log/warn**. If the services will not run, check these locations for more information.

Platform Dataset

The platform dataset is the directory where both the **ubisense_core_server** and **ubisense_local_control** services store platform state. Files in this directory comprise the configuration and ongoing operational state of the platform core, and of all the services configured to run on the local controller.

This directory should be owned by the platform user with full control. The operations group should also have read permission, to allow backup. Restore requires that the backup be copied here and all files set to have platform ownership. See *Configuring Operations Permissions*.

The default platform dataset location is **/home/platform/dataset**. To set a different location, set **platform_datase** in the **platform.conf** file.

Configuring Operations Permissions

If your Linux distribution supports sudo, then the operations group can be assigned permission to start and stop the platform services, and to change ownership of files to the platform user. For example, the following lines might be added to the end of the **/etc/sudoers** file using visudo.

```
%operations ALL = (root) NOPASSWD: /sbin/service ubisense_core_server *, \
/sbin/service ubisense_local_control *, \
/bin/chown -R platform *, /bin/chown platform *
```

With this configuration, any user who is in the operations group will be able to run **sudo /sbin/service ...** to stop, start and get the status of just the platform services. They will also be able to restore platform dataset backups and set the ownership of the restored files back to the platform user.

Backing up your Dataset

After you have set up your SmartSpace installation, ensure that you back up your dataset occasionally, so that you can recover your data. Use the **ubisense_backup** command-line tool from the **tools\linux** folder of your distribution directory to backup your dataset.

Installing Licenses on Windows

SmartSpace feature licenses are supplied as a zipfile with the name **FeatureSetup.zip**. Before you install the licenses, you need to unzip this file into a directory accessible to a server machine from which **ubisense_core_server** will be executed.

To install SmartSpace licenses:

1. Go to the directory where you unzipped the licenses.
2. Double-click the **FeatureSetup.msi** file and the Ubisense Feature Licenses Setup Wizard appears.
3. Click **Next** and the Ubisense Feature Licenses Setup wizard appears.
4. By default all licenses are selected for installation to the default location **C:\Program Files (x86)\Ubisense 3\bin**.
    - Click on the directory tree of licenses, click on individual features and choose whether or not they are to be installed
    - Click **Reset** to return the licenses selection to its default setting
    - Click **Browse** to navigate to a different directory to install the licenses in
5. When you have selected the files and location you require, click Next and then click **Install**.
6. When installation is complete, click **Finish** to close the wizard.

Installing Licenses on Linux

License files must be placed on the server so that the platform can find them. The default location is in the directory **/etc/ubisense**. If a different location is required, then the **license_search_path** can be defined in **platform.conf** (see *Configuration Parameters* for information on the location of this file). Each program also searches for licenses in the same directory as its executable. Licenses should be readable by both the platform user and by the operations group.

Installing Admin Machines on Windows

To install the Ubisense software for an admin machine:

1. Go to the **clients\windows** directory of your SmartSpace distribution directory.
2. Double-click the **UbisenseServiceManager.msi** file and the Service Manager Setup wizard appears.
3. Click **Next**.
4. Choose the Destination Folder for the software.
   You can accept the default **C:\Program Files (x86)\Ubisense 3** or change to another destination.
5. Click **Next** and click **Install**.
6. When the installation is complete, click **Finish** to close the Service Manager Setup wizard.

You have now installed Service Manager onto your computer and you can use it to install and deploy SmartSpace features.

> ℹ Before you can install and deploy features, you must install their licenses.

1. From an admin machine, run Service Manager 3.
2. Click **Install services...**.
3. Specify the directory from which to install.

   This is generally the **packages** folder in your SmartSpace distribution directory. Click **<Recently used directories>** to select previous locations of features.
4. Select the features you want to install.

   Use **Select all** or **Clear all** or click on individual features to indicate which items you want to install.

   - All SmartSpace features are listed.
   - All licensed features are selected by default.
   - Unlicensed features are shown preceded by **[Unlicensed]**. You cannot select these features.
5. By default **Deploy services** is selected.

This means that any services you select for installation will also be automatically deployed during the installation process. Deselect **Deploy services** if you want to manually deploy the services after installation.

6. Click **Install**.
7. When installation is complete, click **Finish**.

You have now installed your SmartSpace features. In Service Manager you can see which services have been deployed by the installed features.

Installing Admin Machines on Linux

Administrative executables, used to configure and maintain the running state of the Ubisense platform, should be executable by the operations group.

Your distribution directory contains the following admin executables:

```
tools/linux/ubisense_backup
tools/linux/ubisense_cache_service_credentials
tools/linux/ubisense_configuration_client
tools/linux/ubisense_file_downloader
tools/linux/ubisense_installer
tools/linux/ubisense_machine_id
tools/linux/ubisense_multicast_test
tools/linux/ubisense_proxyconfig_admin
tools/linux/ubisense_restore_dataset
tools/linux/ubisense_save_dataset
tools/linux/ubisense_service_admin
tools/linux/ubisense_service_ping
tools/linux/ubisense_trace_receiver
tools/linux/ubisense_transfer_config
```

Installing Client Machines on Windows

In Windows, the Ubisense Application Manager allows you to perform the following configuration activities on a client machine:

- Set up Start menu shortcuts for client applications
- Download various command-line tools and SmartSpace documents to a specified location on a client machine

To install the Ubisense software for a client machine:

1. Go to the **clients\windows** directory of your SmartSpace distribution directory.
2. Double-click the **UbisenseApplicationManager.msi** file and the Ubisense Application Manager Setup wizard appears.
3. Click **Next**.
4. Choose the Destination Folder for the software.
   You can accept the default **C:\Program Files (x86)\Ubisense 3** or change to another destination.
5. Click **Next** and click **Install**.
6. When the installation is complete, click **Finish** to close the Ubisense Application Manager Updater Setup wizard.

You have now installed the Ubisense Application Manager and can now configure shortcuts to client applications and download documents and other files to your client machine.

Managing applications

To create shortcuts to SmartSpace applications:

1. Run the Ubisense Application Manager and click on **APPLICATIONS**.
2. Available applications are listed, with their version numbers and, where applicable, location on the Start menu.

   Choose the applications you want to install.

   - Double-click a single application
   - Select several applications and press Enter

   The following SmartSpace client program is available:

- SmartSpace Config (the main SmartSpace configuration GUI)

3. Click **Create shortcuts for selected applications**.

   Shortcuts are created in the Start menu in the locations indicated.

Managing tools and documents

To download SmartSpace command-line tools and documents to a selected directory:

1. Run the Ubisense Application Manager and click on **DOWNLOADABLES**.
   Command-line tools and documents are listed in different categories. The tools and documents available to you depend on the features you have installed.
2. Choose the tools or documents you want to download.
3. Specify the directory to install the files in and click **Start download**.

   The files are downloaded to the specified directory.

> ℹ Whenever you upgrade your SmartSpace installation, you must follow the process described above to replace your existing tools and documents with upgraded versions.

Installing Client Machines on Linux

In order to avoid the use of incompatible versions of SmartSpace administrative and configuration tools, these tools are installed into the platform along with service upgrades. You can then download the current version of each tool onto your Linux client machine using the **ubisense_file_downloader**.

Run the tool with no arguments for help.

For example, to download *all* Linux tools currently available to the current directory, run:

```
> ubisense_file_downloader download --linux-only .
```

To force the overwriting of existing downloads, add `--force`.

Installing SmartSpace Web on Windows

If you have licensed SmartSpace features that are accessed in a browser, such as Web maps or Web forms, you need to set up a web server before installing these features.

To install and configure the Web Server:

1. Enable Internet Information Services.
2. Install the Windows ASP.NET Core Runtime 6.0.x Hosting bundle.
3. Install the websites required.
4. Modify the website configuration files, if required.

Enable Internet Information Services (IIS) on Windows Server 2016

1. Open the Server Manager, click Add roles and features.
2. Click through to Server Roles, and select Web Server (IIS).
3. Click through to Web Server Role (IIS) and under Role Services, ensure that, in addition to the default features, you have enabled Security/Windows Authentication.
4. Click through to Confirm the installation.

ℹ Due its increased vulnerabilities, NTLM is no longer added as a provider for Windows authentication, when SmartSpace Web is installed. If you need to use NTLM, you can manually add it as a provider in IIS Manager.

Firefox does not support Windows authentication by default without NTLM. If you need to support Firefox, you can either add NTLM manually or use this workaround:
The workaround for Firefox requires the browser settings to be changed:

1. Open Firefox and enter **about:config** in the address bar. Click to confirm advanced configuration.
2. In the Filter field, enter **negotiate**.
3. Double-click **network.negotiate-auth.trusted-uris**. This preference lists the trusted sites for Kerberos authentication.
4. Enter your domain (e.g. **company.local**)
5. Click Save (the tick button).

Install the Windows ASP.NET Core Runtime Hosting Bundle

1. Download the Microsoft ASP.NET Core Runtime 6.0.x Hosting bundle from *https://dotnet.microsoft.com/en-us/download/dotnet/6.0* which includes the .NET Runtime and IIS support.
2. Run the installer.
3. If .NET was not previously installed on the server, then a reboot is required for IIS to pick up the path to the .NET Runtime.

Install the Website and/or REST API

Follow these instructions to install the SmartSpace Web application.

When you install the SmartSpace Web application, the following components are created as part of the installation process:

Component	Description
Application Pool:	SmartSpace has its own application pool.
Website:	The entry point to SmartSpace via a browser.

To install the SmartSpace Web application:

1. Go to the **web\windows** directory of your SmartSpace distribution directory.
2. Double-click the **SmartSpaceWeb.msi**.
3. Enter a Website Name: this name will form part of the URL when accessing SmartSpace in a browser.
4. Choose the location for the software.

   You can accept the default **C:\Program Files (x86)\Ubisense 3\SmartSpace\** or click **Change** to select another destination.
5. Enter an Application Pool name.
6. Click **Next** and **Install**.

To install the SmartSpace Web API:

1. Go to the **web\windows** directory of your SmartSpace distribution directory.
2. Double-click the **SmartSpaceWebApi.msi**.

3. Enter a Website Name: this name will form part of the URL when accessing SmartSpace in a browser.
4. Choose the location for the software.

   You can accept the default **C:\Program Files (x86)\Ubisense 3\WebApiCore\** or click **Change** to select another destination.
5. Enter an Application Pool name.
6. Click **Next** and **Install**.

By default, the SmartSpace website can be accessed by navigating to **http://localhost/smartspace** and the REST API can be accessed by navigating to **http://localhost/smartspaceapi**.

Workaround for Failed Installation due to PowerShell Configuration

Installation of SmartSpace Web or the Application REST API can sometimes fail and rollback. This can be because the local machine has been configured to require signed PowerShell scripts. The workaround is to temporarily remove this configuration from the Windows Registry in order to run the installer successfully.

In **HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\PowerShell**, ensure you have the following values set:

Key	Type	Value
EnableScripts	REG_DWORD	1
ExecutionPolicy	REG_SZ	Unrestricted

Modify the website configuration files if required

Advanced configuration of the websites is done by creating and editing configuration files in the installation folders. By default, on Windows, these files are in:

- SmartSpace website: **C:\Program Files (x86)\Ubisense 3\SmartSpace\Web\localsettings.json**
- REST API website: **C:\Program Files (x86)\Ubisense 3\WebApiCore\Web\localsettings.json**

These files should be created if you wish to make advanced configurations, rather than modifying the installed defaults in **appsettings.json**. This is because **appsettings.json** can be overwritten on a software upgrade.

Using OpenID Connect to enable external authentication

OpenID Connect support allows an external authentication authority to handle login and provide user ID and roles to SmartSpace Web. The external authority takes care of ensuring that the user is allowed to access the application, and sends claims about the user back to SmartSpace Web. SmartSpace can then be configured to extract the user ID, and optionally the roles, from the claims provided by the authority.

To use OpenID Connect, SmartSpace must be accessed via https, so make sure you have configured SSL in IIS.

There are two places where OpenID Connect must be configured: at the authority, and in SmartSpace Web.

Configuration at the Authority

The authority provides an application client ID and secret, and is configured to accept the redirect URIs that will be used during authentication. The authority will also be configured to control which users/groups are allowed to access SmartSpace, and which properties should be sent to SmartSpace for it to use as the user ID and/or roles. Generally this involves creating a new "application" instance or registration at the authority, and then configuring the application to work with SmartSpace.

In order to ensure that only valid granted authorities are used, both the authority and SmartSpace share a configured "client id" and "client secret". The ID identifies the specific SmartSpace web site application at the authority, and the secret is used by SmartSpace to verify that the returned claims are valid. So the first step to setting up the authority is to create a new client ID in the authority, and to generate the client secret. The application must have the "code flow" or "code" response type enabled (also known as "Authorization code grant").

Authorities have different ways to do this, but for example, in Microsoft Azure Active Directory (AD), at the time of writing:

1. Go to App Registrations.
2. Click New registration.
3. Give the application a name and the set of users who can log in, and then click Register.
4. Click Certificates and secrets, and create a new secret.
5. Take a copy of the Value (it can only be viewed/copied on creation).

Set up the redirect URIs used during login and logout. These must match the redirect requests that the SmartSpace website generates when it attempts to login or logout a user by directing the browser to the authority.

- Sign in or callback URI: **https://<your server host as seen by the end user>/SmartSpace/signin-oidc**
- Sign out or front channel logout URI: **https://<your server host as seen by the end user>/SmartSpace/**

You can also define roles, or put a set of AD groups as claims. See the Azure AD documentation for how this is done.

Configuration in SmartSpace Web

There is an OpenID Connection configuration section in the web settings JSON files. Depending on your operating system this file is either **localsettings.json**, or **/etc/ubisense/web.json**.

```
"OpenIDC" : {
         "ClientId": "865b3f07-3f30-4881-895b-a831d11917ac",
         "ClientSecret": "<your secret shared with the authority goes here>",
         "MetadataAddress": "https://login.microsoftonline.com/fec9c4e9-d7de-4363-a4e3-
039b6f2606d2/v2.0/.well-known/openid-configuration",
         "IdClaim": "preferred_username",
         "RoleClaim": "http://schemas.microsoft.com/ws/2008/06/identity/claims/role",
         "LogoutURL": "",
         "DebugClaims": true
   },
```

The settings in this section are case sensitive, and have the following meanings:

Setting	Meaning
ClientId	The identity of this app at the OpenID Connect authority
ClientSecret	The shared secret used to generate and validate claims about the logged-in user
MetadataAddress	The OpenID Connect configuration meta-data document from your authority. For Azure AD, for example, go to Endpoints under your application registration, and copy "OpenID Connect metadata document".
IdClaim	The claim type to be used as the logged-in user in SmartSpace

Setting	Meaning
RoleClaim	The claim type to be used as a role for the user in SmartSpace. Optional
LogoutURL	Some OpenID Connect authorities do not provide an "end_session_endpoint", so logout will be unsuccessful. In this case, provide a URL to log out the user from the client ID at the authority. Optional
DebugClaims	If set true, this will provide a page within SmartSpace Web that you can visit once authentication has redirected back to SmartSpace, and see the claims provided. The page can be found at SmartSpace/Auth/Claims. It is recommended you disable this configuration in production. The default is false. Optional

After the configuration has been saved, restart the web site. Go to IIS Manager -> Application Pools -> SmartSpaceCore -> Start.

You can test log in by visiting the SmartSpace Web site and clicking Login. You should be redirected to the authority login page. On successfully logging in, you will be asked to confirm sharing your profile with SmartSpace, and then be redirected back to SmartSpace Web. Note that if you are already logged in at the authority (for example if you are logged into Microsoft 365), there may be no need to provide any user/password – you may not even see the redirects as they can happen very quickly.

On success, the user ID should be displayed in the top left of the SmartSpace Web site. If this is not the case, enable DebugClaims, restart SmartSpace Web, and authenticate again. Then visit /SmartSpace/Auth/Claims to see what the authority returned, and pick a suitable IdClaim.

Using Roles from OpenID Connect

Claims returned by the authority that have a claim type configured in RoleClaim will be treated as user roles. These will be passed transparently through to SmartSpace. To use them within SmartSpace they should also be created as roles using the USERS / ROLES task in SmartSpace Config. You do not need to add members of the role within SmartSpace Config (though this is allowed if necessary). Users that have the role according to the OpenID Connect authority will automatically be treated as having the role within SmartSpace Web.

For example, if the authority passes a claim "SmartSpace Supervisor", you should create a "SmartSpace Supervisor" role in the USERS / ROLES task in SmartSpace Config. After you have created the role within SmartSpace, you can:

- assign this role as a member of other roles
- specify the searches/views/properties that this role can access directly
- assign the role to reports, HMIs, and in ObjectView API views

Again, you can use *DebugClaims* to see what the authority has returned to SmartSpace for the currently authenticated user. Remember to disable DebugClaims and restart the web site before going into production.

Header Authentication (SiteMinder)

The websites can read the authenticated user from a header passed to them by a proxy server, such as SiteMinder. To configure this, set "AuthOptions/UserHeader" to be the name of the header from which the logged in user is to be extracted.

```
{
    "AuthOptions": {
       "UserHeader": "SITEMINDER_USER"
   }
}
```

> If you configure this option, it is vital that users cannot access the website except through the proxy server, because otherwise they could add their own header as part of the request and gain unauthorized access. Typically in this case you would configure IIS bindings to only listen on the loopback interface, or configure IP Address and Domain Restrictions. See Microsoft IIS documentation for details.

Header Authentication with Other External Authentication Systems

Upstream reverse proxies can provide some other external authentication system, and pass both user and roles from that external system to SmartSpace.

AuthOptions.RolesHeader can be used with AuthOptions.UserHeader to specify a header from which to read roles for passthrough to SmartSpace. Roles are comma separated in the value of the header provided. The config is optional: if not specified in the settings file then no roles will be passed through. The option is ignored if UserHeader is not set.

For example:

```
  "AuthOptions": {
       "UserHeader": "AUTH_USER",
       "RolesHeader": "AUTH_ROLES",
        ...
   }
```

This would read the user (authenticated by the up-stream reverse proxy) from header AUTH_USER, and the roles from header AUTH_ROLES. If these are used, the web site should not be accessible except from the reverse proxy, which should filter out any user-supplied values for these headers.

As with roles in *OpenID Connect*, the roles that can be passed in the header should also be created in the USERS / ROLES task in SmartSpace Config so they can be assigned as members of other roles, or given searchs/views, etc.

Switching to Forms Authentication

By default the Windows website installation uses Windows authentication for login. You can switch to forms authentication by configuring LDAP parameters and setting up "AuthOptions/UseCookiesOnWindows". For production or integration deployment, you will need an SSL certificate signed by a suitable root authority for your users. For development or test deployment, a test certificate can be used instead (e.g. generated locally using OpenSSL). If you configure forms authenticatication without SSL, it will not work.

In the following examples, the first example shows the configuration for an Active Directory server, whilst the second shows the configuration for an LDAP server that does not require a login for searching.

LDAP authentication with an Active Directory server

```
"LDAPAuth": {
      "Server": "adserver.company.com",
      "Port": "389",
      "User": "",
      "Password": "",
      "SearchStart": "dc=company,dc=com",
      "AccountId": "sAMAccountName"
   },

   "AuthOptions": {
      "UseCookiesOnWindows" : true,
      "ExpiryTimeSpan": "00:30",
      "SlidingExpiry":  true
   }
}
```

LDAP authentication with no login for searching

```
"LDAPAuth": {
      "Server": "ldap.company.com",
      "Port": "389",
      "User": "",
      "Password": "",
      "SearchStart": "ou=people,dc=company,dc=com",
      "AccountId": "uid",
      "ObjectClass": "account"
   },

   "AuthOptions": {
      "UseCookiesOnWindows" : true,
      "ExpiryTimeSpan": "00:30",
      "SlidingExpiry":  true
   }
}
```

Note: The example given above works with the default OpenLDAP schema: other servers and schema might require different parameters.

How the LDAP validator behaves

The LDAP validator does the following:

1. If User option set, bind using this user/password.
2. If SearchStart is set, use the SearchStart, AccountId and ObjectClass to search for the DN of the entered user name.
3. Bind using the DN, if the search succeeded, or the entered user name. Use the entered password. If this bind succeeds, the user is authenticated successfully.

The validator reports what it is doing on the website trace (always on). for example:

```
[Tue Sep 24 14:23:03 2019, 127.0.0.1:42943] website: LDAPValidator: Is user valid
marcin
[Tue Sep 24 14:23:03 2019, 127.0.0.1:42943] website: LDAPValidator: searching (&
(objectclass=nsAccount)(uid=marcin)) at base ou=people,dc=ubisense,dc=aws
[Tue Sep 24 14:23:03 2019, 127.0.0.1:42943] website: LDAPValidator: binding as user
uid=marcin,ou=People,dc=ubisense,dc=aws
```

Enabling Secure LDAP (LDAPS)

To enable secure LDAP (LDAPS) for the connection you must use port 636. By using this port, SSL/TLS is enabled for the connection to the LDAP server (all other port numbers use TCP).

```
"LDAPAuth": {
      "Server": "adserver.company.com",
      "Port": "636",
      "User": "",
      "Password": "",
      "SearchStart": "dc=company,dc=com",
      "AccountId": "sAMAccountName"
   },

   "AuthOptions": {
      "UseCookiesOnWindows" : true,
      "ExpiryTimeSpan": "00:30",
      "SlidingExpiry":  true
   }
}
```

Configuring the Authentication Timespan

The cookies authentication uses an expiry time of 30 minutes and a sliding timespan. This means that the authentication will expire 30 minutes after the user closes the website, but will continue to be refreshed while the user is still visiting the website.

You can disable this sliding expiry and set an absolute time after which login will need to be repeated. For example, to log out after three hours:

```
{
   "AuthOptions": {
      "ExpiryTimeSpan": "03:00",
      "SlidingExpiry":  false
   }
}
```

Disable Hardened Headers

By default the website injects headers in each response for penetration security. These disable cross-site/cross-frame scripting, prevent content type sniffing, etc. If necessary, these headers can be disabled, and IIS configured manually to add appropriate headers instead.

```
{
    "SecurityOptions": {
       "HardenHeaders": false
    }
}
```

Enable Cross Origin Scripting

CORS is supported for the SmartSpace REST API. For features using SmartSpace Web where CORS is not supported, there are workarounds such as making configuration changes so that the browser treats localhost requests and the remote site as if they come from the same domain, disabling hardened headers (see above), or using IIS or Apache.

By default, no CORS headers are sent, so browsers will refuse to execute the API web methods from a page served from a different web server.

The option AllowOrigins in the **appsettings.json** file which overrides settings in **localsettings.json** enables cross origin scripting:

```
{
...
   "SecurityOptions": {
   "HardenHeaders": true,
   "AllowOrigins": [ "http://example.com", "https://*.mydomain.com" ]
   }
...
}
```

If AllowOrigins is set, and matches the origin of a request, the API will respond with suitable headers, for example:

```
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: X-Requested-With
Access-Control-Allow-Methods: PUT
Access-Control-Allow-Origin: https://server.mydomain.com
Access-Control-Max-Age: 3600
```

The browser will now allow the request. Note that this allows the browser to cache the pre-flight OPTIONS responses for up to an hour, to reduce load on the API server. Thus changes to the allowed origins may not be picked up by browsers for an hour.

Adding a custom theme for the website

You can customize the look of the SmartSpace website to better reflect your corporate identity, for example by replacing the Ubisense logo with your own or using a custom stylesheet. To prevent such customization from being overwritten during website upgrades, you should store your local theme in an external folder on the host machine, for example **C:\Ubisense\localtheme**. You specify the folder using the website configuration setting ContentOptions / LocalThemePath.

```
{
    "ContentOptions": {
       "LocalThemePath": "C:\\Ubisense\\localtheme\\"
    }
}
```

If the custom theme folder contains any **.min.css** files, they are loaded in place of the **wwwroot/bundle/base.css**. If the folder contains **custom.css**, it will be loaded in addition to other **css** files.

The following files can also be overridden by adding corresponding files in the custom theme folder:

- **images/logo.png**
- **images/background.png**
- **images/ubisense_large.png**
- **images/ubisense_small.png**
- **manifest.json**

Errors

When the website is loaded, you may see 502.5 "ANCM Out-of-Process Startup Failure".

This is normally because the .NET Runtime could not be found. Make sure you restarted the server after installing the ASP.NET Core Runtime 6.0.x Hosting bundle.

Also, make sure that you included the Security/Windows Authentication feature when deploying IIS, as this is required by the websites on Windows unless forms or header authentication is configured.

Security Configuration for SmartSpace Web with Security Manager

If you are using a non-trivial security manager configuration to force authentication for services (as is the case for ACS installations) then you must run the **ubisense_cache_service_credentials** tool on the web server host. This is because the **credentials.dat** file created by the tool (in the latest version) allows IIS_IUSRS as a reader. Without this, the web site code cannot read the credentials, and therefore cannot connect to the platform services it needs.

For a Windows server, you *must* use the version of the **ubisense_cache_service_credentials** tool from the 3.4 sp1 distribution or above.

Installing SmartSpace Web on Linux

If you have licensed SmartSpace features that are accessed in a browser, such as Web maps or Web forms, you need to set up a web server before installing these features.

In this section, we will describe configuring the websites using Apache2 as a reverse proxy. *Advanced configuration* options will then be covered later.

Linux Requirements for SmartSpace Web

We support recent enterprise Linux distributions, such as SUSE Linux Enterprise Server 11+ or Red Hat® Enterprise Linux® v7+.

The following instructions assume you are configuring a reverse proxy in Apache 2.4.23 or above. For Red Hat® Enterprise Linux® Apache 2.4.23 is only available for version 8+. Whilst configuring a reverse proxy on earlier versions of Red Hat® Enterprise Linux® is possible, instructions for this are beyond the scope of this guide.

The following packages must be installed on the server.

- ldap 2.4 libraries

For production or integration deployment, you will need an SSL certificate signed by a suitable root authority for your users. This certificate will be installed for Apache2. SSL (TLS) is required for Linux installation because the website uses forms-based authentication, and so must have transport level security configured. For development or test deployment, a test certificate can be used instead (e.g. generated locally using OpenSSL).

Microsoft .NET Runtimes

See *Microsoft .NET Runtimes* for information on .NET runtimes for Linux.

Server Configuration

Ensure the web server is connected to the platform, using multicast, unicast cluster (see Smart Space Using Unicast Cluster Setup Guide on the Ubisense Documentation Portal), or a site connector.

If you have not already done so, on the web server install the platform servers for Linux, configure a dataset directory, and start a local controller. The website will be deployed on this controller. See *Installing the Ubisense SmartSpace software on Linux*.

Authentication Options

There are two authentication methods supported in the web sites when deployed on Linux: Forms authentication, and Header authentication.

Forms authentication directs the user to a login page when they attempt to access a part of the web sites that requires authentication. This login page gathers the user and password, which are then validated using a configured LDAP server. If this succeeds, a cookie is returned to the user's browser, which is used to authenticate in subsequent requests. Forms authentication requires the web site to be accessed via HTTPS for all authenticated or login traffic, to protect the credentials and cookies. This is configured in the reverse proxy that handles the HTTPS protocol.

Header authentication relies on another system, such as SiteMinder, doing authentication before passing the request to the website. A special header is set by this up-stream system on each request that reaches the web site, indicating the authenticated user. The web site simply assumes that the given header is authoritative. This is a commonly used method in enterprise environments.

Either method can be used for the SmartSpace website, but the Rest API only supports Header authentication (or no authentication).

Configuration Files

Set up the configuration files for the SmartSpace website and REST API. On Linux these are placed in **/etc/ubisense**. The files should all be readable only by the user that runs the platform controller. The following configuration files are used:

web.json

This contains configuration specific to the website. In the following examples, we will set up the parameters to access the LDAP server used to validate the user's credentials. We also set a proxy base path matching the reverse proxy path we will configure in Apache2 for the website. The first example shows the configuration for an Active Directory server, whilst the second shows the configuration for an LDAP server that does not require a login for searching.

LDAP authentication with an Active Directory server

```
{
    "LDAPAuth": {
      "Server": "adserver.company.com",
      "Port": "389",
      "User": "",
      "Password": "",
      "SearchStart": "dc=company,dc=com",
      "AccountId": "sAMAccountName"
   },

     "ProxyOptions": {
        "Base": "/SmartSpace"
    }
 }
```

LDAP authentication with no login for searching

```
{
    "LDAPAuth": {
      "Server": "ldap.company.com",
      "Port": "389",
      "User": "",
      "Password": "",
      "SearchStart": "ou=people,dc=company,dc=com",
      "AccountId": "uid",
      "ObjectClass": "account"
   },


     "ProxyOptions": {
        "Base": "/SmartSpace"
    }
 }
```

Note: The example given above works with the default OpenLDAP schema: other servers and schema might require different parameters.

How the LDAP validator behaves

The LDAP validator does the following:

1. If User option set, bind using this user/password.
2. If SearchStart is set, use the SearchStart, AccountId and ObjectClass to search for the DN of the entered user name.
3. Bind using the DN, if the search succeeded, or the entered user name. Use the entered password. If this bind succeeds, the user is authenticated successfully.

The validator reports what it is doing on the website trace (always on). for example:

```
[Tue Sep 24 14:23:03 2019, 127.0.0.1:42943] website: LDAPValidator: Is user valid
marcin
[Tue Sep 24 14:23:03 2019, 127.0.0.1:42943] website: LDAPValidator: searching (&
(objectclass=nsAccount)(uid=marcin)) at base ou=people,dc=ubisense,dc=aws
[Tue Sep 24 14:23:03 2019, 127.0.0.1:42943] website: LDAPValidator: binding as user
uid=marcin,ou=People,dc=ubisense,dc=aws
```

Enabling Secure LDAP (LDAPS)

To enable secure LDAP (LDAPS) for the connection you must use port 636. By using this port, SSL/TLS is enabled for the connection to the LDAP server (all other port numbers use TCP).

Note: If the LDAP server uses LDAPS (port 636), then the web site server must be able to verify the certificate that the LDAP server uses. For example, in active directory, the root certificate used to sign the LDAP server certificate should be imported into the web server host as a certificate authority (CA). How this is done depends on the specific operating system, but if it is not done, then LDAPS authentication will fail.

```
"LDAPAuth": {
      "Server": "adserver.company.com",
      "Port": "636",
      "User": "",
      "Password": "",
      "SearchStart": "dc=company,dc=com",
      "AccountId": "sAMAccountName"
   },

   "AuthOptions": {
      "UseCookiesOnWindows" : true,
      "ExpiryTimeSpan": "00:30",
      "SlidingExpiry":  true
   }
}
```

Using OpenID Connect to enable external authentication

OpenID Connect support allows an external authentication authority to handle login and provide user ID and roles to SmartSpace Web. The external authority takes care of ensuring that the user is allowed to access the application, and sends claims about the user back to SmartSpace Web. SmartSpace can then be configured to extract the user ID, and optionally the roles, from the claims provided by the authority.

To use OpenID Connect, SmartSpace must be accessed via https, so make sure you have configured SSL in your reverse proxy.

There are two places where OpenID Connect must be configured: at the authority, and in SmartSpace Web.

Configuration at the Authority

The authority provides an application client ID and secret, and is configured to accept the redirect URIs that will be used during authentication. The authority will also be configured to control which users/groups are allowed to access SmartSpace, and which properties should be sent to SmartSpace for it to use as the user ID and/or roles. Generally this involves creating a new "application" instance or registration at the authority, and then configuring the application to work with SmartSpace.

In order to ensure that only valid granted authorities are used, both the authority and SmartSpace share a configured "client id" and "client secret". The ID identifies the specific SmartSpace web site application at the authority, and the secret is used by SmartSpace to verify that the returned claims are valid. So the first step to setting up the authority is to create a new client ID in the authority, and to generate the client secret. The application must have the "code flow" or "code" response type enabled (also known as "Authorization code grant").

Authorities have different ways to do this, but for example, in Microsoft Azure Active Directory (AD), at the time of writing:

1. Go to App Registrations.
2. Click New registration.
3. Give the application a name and the set of users who can log in, and then click Register.
4. Click Certificates and secrets, and create a new secret.
5. Take a copy of the Value (it can only be viewed/copied on creation).

Set up the redirect URIs used during login and logout. These must match the redirect requests that the SmartSpace website generates when it attempts to login or logout a user by directing the browser to the authority.

- Sign in or callback URI: **https://<your server host as seen by the end user>/SmartSpace/signin-oidc**
- Sign out or front channel logout URI: **https://<your server host as seen by the end user>/SmartSpace/**

You can also define roles, or put a set of AD groups as claims. See the Azure AD documentation for how this is done.

Configuration in SmartSpace Web

There is an OpenID Connection configuration section in the web settings JSON files. Depending on your operating system this file is either **localsettings.json**, or **/etc/ubisense/web.json**.

```
"OpenIDC" : {
         "ClientId": "865b3f07-3f30-4881-895b-a831d11917ac",
         "ClientSecret": "<your secret shared with the authority goes here>",
         "MetadataAddress": "https://login.microsoftonline.com/fec9c4e9-d7de-4363-a4e3-
039b6f2606d2/v2.0/.well-known/openid-configuration",
         "IdClaim": "preferred_username",
         "RoleClaim": "http://schemas.microsoft.com/ws/2008/06/identity/claims/role",
         "LogoutURL": "",
         "DebugClaims": true
   },
```

The settings in this section are case sensitive, and have the following meanings:

Setting	Meaning
ClientId	The identity of this app at the OpenID Connect authority
ClientSecret	The shared secret used to generate and validate claims about the logged-in user
MetadataAddress	The OpenID Connect configuration meta-data document from your authority. For Azure AD, for example, go to Endpoints under your application registration, and copy "OpenID Connect metadata document".
IdClaim	The claim type to be used as the logged-in user in SmartSpace
RoleClaim	The claim type to be used as a role for the user in SmartSpace. Optional
LogoutURL	Some OpenID Connect authorities do not provide an "end_session_endpoint", so logout will be unsuccessful. In this case, provide a URL to log out the user from the client ID at the authority. Optional
DebugClaims	If set true, this will provide a page within SmartSpace Web that you can visit once authentication has redirected back to SmartSpace, and see the claims provided. The page can be found at SmartSpace/Auth/Claims. It is recommended to disable this configuration in production. The default is false. Optional

After the configuration has been saved, restart the web site by using Service Manager to restart the Ubisense/Visibility/Web site service.

You can test log in by visiting the SmartSpace Web site and clicking Login. You should be redirected to the authority login page. On successfully logging in, you will be asked to confirm sharing your profile with SmartSpace, and then be redirected back to SmartSpace Web. Note that if you are already logged in at the authority (for example if you are logged into Microsoft 365), there may be no need to provide any user/password – you may not even see the redirects as they can happen very quickly.

On success, the user ID should be displayed in the top left of the SmartSpace Web site. If this is not the case, enable DebugClaims, restart SmartSpace Web, and authenticate again. Then visit /SmartSpace/Auth/Claims to see what the authority returned, and pick a suitable IdClaim.

Using Roles from OpenID Connect

Claims returned by the authority that have a claim type configured in RoleClaim will be treated as user roles. These will be passed transparently through to SmartSpace. To use them within SmartSpace they should also be created as roles using the USERS / ROLES task in SmartSpace Config. You do not need to add members of the role within SmartSpace Config (though this is allowed if necessary). Users that have the role according to the OpenID Connect authority will automatically be treated as having the role within SmartSpace Web.

For example, if the authority passes a claim "SmartSpace Supervisor", you should create a "SmartSpace Supervisor" role in the USERS / ROLES task in SmartSpace Config. After you have created the role within SmartSpace, you can:

- assign this role as a member of other roles
- specify the searches/views/properties that this role can access directly
- assign the role to reports, HMIs, and in ObjectView API views

Again, you can use *DebugClaims* to see what the authority has returned to SmartSpace for the currently authenticated user. Remember to disable DebugClaims and restart the web site before going into production.

Header Authentication with Other External Authentication Systems

Upstream reverse proxies can provide some other external authentication system, and pass both user and roles from that external system to SmartSpace.

AuthOptions.RolesHeader can be used with AuthOptions.UserHeader to specify a header from which to read roles for passthrough to SmartSpace. Roles are comma separated in the value of the header provided. The config is optional: if not specified in the settings file then no roles will be passed through. The option is ignored if UserHeader is not set.

For example:

```
"AuthOptions": {
     "UserHeader": "AUTH_USER",
     "RolesHeader": "AUTH_ROLES",
      ...
 }
```

This would read the user (authenticated by the up-stream reverse proxy) from header AUTH_USER, and the roles from header AUTH_ROLES. If these are used, the web site should not be accessible except from the reverse proxy, which should filter out any user-supplied values for these headers.

As with roles in *OpenID Connect*, the roles that can be passed in the header should also be created in the USERS / ROLES task in SmartSpace Config so they can be assigned as members of other roles, or given searchs/views, etc.

restapi.json

This contains configuration specific to the REST API. In this example, we will set up a proxy base path matching the reverse proxy path we will configure in Apache2 for the API:

```
{
    "ProxyOptions": {
        "Base": "/SmartSpaceApi"
    }
}
```

Here we are allowing anonymous access to the API. For header authentication see the *advanced configuration*.

shared.json

shared.json is a configuration file that is loaded by both web sites, where options shared by the two can be set up. This is not used in our example configuration.

Deploy the Platform Services

The platform services should be deployed using the Service Manager client or the **ubisense_installer** command-line tool.

To deploy the services using Service Manager:

1. Run Service Manager 3 on a Windows client connected to the platform.
2. Click **Install services...**, click **Browse** and navigate to the extracted SmartSpace release, then go to the **web\linux\packages** folder.
3. If you already have a .NET Core Runtime installed on the web server, unselect **Shared runtime** in the list of features to install.
4. Click **Install** and the services are installed. You can click **Finish** when the "Service installation complete" message displays to return to the main Service Manager screen.

5. If you have multiple Linux controllers, deploy the website services onto the prepared Linux server:
    a. Expand the **Controllers** entry under HIERARCHY, so you can see the web server controller.
    b. Expand the **Services** entry under HIERARCHY, find the “Ubisense/Visibility/Web site” service. Drag this service and drop it onto the web server controller. It should deploy onto that controller.
    c. Repeat the above for the service “Ubisense/Application integration/RestAPI site”.
6. The Web site and Rest API site services should now be running on the Linux web server, but will not be visible from the wider network.

To deploy the services using the **ubisense_installer** command-line tool:

1. Go to the **web\linux\packages** directory of your SmartSpace distribution directory.
2. Run the following commands to install and deploy the Web site and REST API site:

```
ubisense_installer -ud SmartSpaceWeb.xml
```

```
ubisense_installer -ud SmartSpaceRestApi.xml
```

3. If you do not have a .NET Core Runtime installed on the web server, you must also run the following command:

```
ubisense_installer -ud SharedRuntime.xml
```

Configure Apache2 Reverse Proxy

Now install and configure Apache2 as a reverse proxy. The instructions below are targeted at SLES, and will need to be adapted for other Linux distributions.

Install Apache2

Use the package management software for your Linux distribution to install Apache2. For example:

```
sudo zypper in apache2
```

You will also need to ensure all required Apache modules are enabled using the following command:

```
sudo a2enmod <module name>
```

The required modules include:

```
mod_proxy
mod_proxy_http
mod_ssl
mod_headers
mod_rewrite
```

On some variants of Linux, drop the "mod_" prefix when enabling a module for Apache. You can see the list of enabled modules with the command:

```
sudo apache2ctl –M
```

Modules may be displayed with slightly different names in the output generated by this command, with a **_module** suffix instead of a **mod_** prefix.

Enable Outbound Connections

For SELinux servers such as RHEL 7, you may find that the Apache service cannot connect to another website. To correct this you need to enable outbound connections by running the following command:

```
/usr/sbin/setsebool -P httpd_can_network_connect 1
```

Install the SSL certificates

Place the SSL certificate and key in a suitable location:

```
/etc/apache2/ssl.crt/localhost.crt
/etc/apache2/ssl.key/localhost.key
```

Creating the service configuration file

Apache is configured by **.conf** files located in **/etc/apache2/vhosts.d/**.

Create a file **smartspace.conf**. The example below matches the proxy paths configured in the **web.json** and **restapi.json** example files above.

```
<VirtualHost *:*>
        RequestHeader set "X-Forwarded-Proto" expr=%{REQUEST_SCHEME}
</VirtualHost>

<VirtualHost *:80>
    # Rewrite http to https
    RewriteEngine On
    RewriteCond %{HTTPS} !=on
    RewriteRule ^/?(.*) https://%{SERVER_NAME}/$1 [R,L]
</VirtualHost>

<VirtualHost *:443>
        SSLProxyEngine on
        ProxyPreserveHost On

        # We proxy SmartSpaceApi to the REST api http port
        ProxyPass /SmartSpaceApi http://127.0.0.1:5002/SmartSpaceApi
        ProxyPassReverse /SmartSpaceApi http://127.0.0.1:5002/SmartSpaceApi

        # We proxy SmartSpace to the web site http port
        ProxyPass /SmartSpace http://127.0.0.1:5000/SmartSpace
        ProxyPassReverse /SmartSpace http://127.0.0.1:5000/SmartSpace

        # Add the forwarded protocol header
        RequestHeader set "X-Forwarded-Proto" expr=%{REQUEST_SCHEME}

       # Using localhost as server hostname
        ServerName mywebserverhost.domain.com
        ServerAlias mywebserverhost

        # Set logging_dir to a suitable logging directory
        ErrorLog /var/log/apache2/smartspace_error.log
        CustomLog /var/log/apache2/smartspace_custom.log common

        # Give away as little as possible on 404.
        ErrorDocument 404 "Not found"

        # Redirect top level to the SmartSpace web site
        RedirectMatch ^/$ /SmartSpace

        # Configure the https options to be suitably secure
        SSLEngine on
        SSLProtocol all -SSLv2
        SSLCipherSuite ALL:!ADH:!EXPORT:!SSLv2:!RC4+RSA:+HIGH:+MEDIUM:!LOW:!RC4
        SSLCertificateFile /etc/apache2/ssl.crt/localhost.crt
        SSLCertificateKeyFile /etc/apache2/ssl.key/localhost.key
</VirtualHost>
```

In this example, **SSLCertificateFile** should be the primary certificate file for the domain name. **SSLCertificateKeyFile** should be the key file generated when CSR is created. **SSLCertificateChainFile** should be the intermediate certificate file (if any) that was supplied by the certificate authority.

Enable SSL for Apache

Additionally SSL must be enabled for Apache by adding the **SSL** flag to **APACHE_SERVER_FLAGS** in **/etc/sysconfig/apache2**:

```
APACHE_SERVER_FLAGS="SSL"
```

You will need to restart Apache to reload any new or changed configuration files, using the following commands:

```
sudo systemctl restart apache2
sudo systemctl enable apache2
```

Check the status of Apache with the command:

```
sudo systemctl status apache2
```

Test the website

Visit the website in a browser. You should be redirected to the top level SmartSpace website page.

Advanced Configuration

Header Authentication (SiteMinder)

The websites can read the authenticated user from a header passed to them by a proxy server, such as SiteMinder. To configure this, in **shared.json**, **web.json** or **restapi.json**, set **AuthOptions/UserHeader** to be the name of the header from which the logged in user is to be extracted. If this option is configured, then the LDAPAuth options do not need to be set.

```
{
    "AuthOptions": {
       "UserHeader": "SITEMINDER_USER"
       "RequireAuth": true
   }
}
```

If UserHeader has been configured, it would be normal to also require authentication for all pages. This is what the RequireAuth configuration setting does. Since the UserHeader assumes that the header passed in the request has been checked, it is important that the user should not be able to bypass the proxy server and pass their own user header directly to the web sites.

Configuring the Authentication Timespan

The default authentication on Linux uses an expiry time of 30 minutes and a sliding timespan. This means that the authentication will expire 30 minutes after the user closes the website, but will continue to be refreshed while the user is still visiting the website.

You can disable this sliding expiry and set an absolute time after which login will need to be repeated. For example, to log out after three hours:

```
{
   "AuthOptions": {
      "ExpiryTimeSpan": "03:00",
      "SlidingExpiry":  false
   }
}
```

Disable Hardened Headers

By default the website injects headers in each response for penetration security. These disable cross-site/cross-frame scripting, prevent content type sniffing, etc. If necessary, these headers can be disabled, and the reverse proxy configured manually to add appropriate headers instead.

```
{
    "SecurityOptions": {
       "HardenHeaders": false
    }
}
```

Enable Cross Origin Scripting

CORS is supported for the SmartSpace REST API. For features using SmartSpace Web where CORS is not supported, there are workarounds such as making configuration changes so that the browser treats localhost requests and the remote site as if they come from the same domain, disabling hardened headers (see above), or using IIS or Apache.

By default, no CORS headers are sent, so browsers will refuse to execute the API web methods from a page served from a different web server.

The option AllowOrigins in the **appsettings.json** file which overrides settings in **localsettings.json** enables cross origin scripting:

```
{
...
   "SecurityOptions": {
   "HardenHeaders": true,
   "AllowOrigins": [ "http://example.com", "https://*.mydomain.com" ]
   }
...
}
```

If AllowOrigins is set, and matches the origin of a request, the API will respond with suitable headers, for example:

```
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: X-Requested-With
Access-Control-Allow-Methods: PUT
Access-Control-Allow-Origin: https://server.mydomain.com
Access-Control-Max-Age: 3600
```

The browser will now allow the request. Note that this allows the browser to cache the pre-flight OPTIONS responses for up to an hour, to reduce load on the API server. Thus changes to the allowed origins may not be picked up by browsers for an hour.

Adding a custom theme for the website

You can customize the look of the SmartSpace website to better reflect your corporate identity, for example by replacing the Ubisense logo with your own or using a custom stylesheet. To prevent such customization from being overwritten during website upgrades, you should store your local theme in an external folder on the host machine, for example **/home/platform/localtheme**. You specify the folder using the website configuration setting ContentOptions / LocalThemePath.

```
{
    "ContentOptions": {
       "LocalThemePath": "//home//platform//localtheme//"
    }
}
```

If the custom theme folder contains any **.min.css** files, they are loaded in place of the **wwwroot/bundle/base.css**. If the folder contains **custom.css**, it will be loaded in addition to other **css** files.

The following files can also be overridden by adding corresponding files in the custom theme folder:

- **images/logo.png**
- **images/background.png**
- **images/ubisense_large.png**
- **images/ubisense_small.png**
- **manifest.json**

Removing the Shared Runtime after Undeploying the Websites

If you deploy the website and/or the REST API without installing the .NET Runtime on the server, and subsequently move these website services to another controller, the shared runtime will be left in the dataset. You can simply delete this directory when it is no longer needed by any locally deployed services. The folder is:

```
<dataset path>/Ubisense/Platform/Shared\ runtime/x.x.x/
```

where `x.x.x` is the .NET Runtime version number.

Security Configuration for SmartSpace Web with Security Manager

If you are using a non-trivial security manager configuration to force authentication for services (as is the case for ACS installations) then you must run the **ubisense_cache_service_credentials** tool on the web server host. This is because the **credentials.dat** file created by the tool (in the latest version) allows IIS_IUSRS as a reader. Without this, the web site code cannot read the credentials, and therefore cannot connect to the platform services it needs.

Installing Site connector

The Site connector consists of a server application and a set of client applications. There are two Site connector clients: which one you install depends on the configuration of your installation.

Site connector requires both the Site connector server and at least one client to be installed. The steps you follow to install the server and the client software are described below. Additional configuration steps might be required.

For an overview of different uses for Site connector and detailed information on configuration, see the Ubisense Site Connector Guide on the Ubisense Documentation Portal.

The Site connector software is supplied as a zipfile with the name SiteConnector followed by numbers indicating the version of the software, for example **SiteConnector_2_1_11_7160.zip**. Before you install Site connector, you need to unzip this file into a *distribution directory* accessible to the machines on which you will be installing the software.

Installing Site connector server

The Site connector server is an independent service rather than a package deployed via the Ubisense platform. On Windows, this is installed as a Windows Service. On Linux, it should be started in the same way as core and controller, via a startup script or systemd, depending on the Linux distribution.

Installing Site connector server on Windows

1. Go to the **UbisenseSiteConnectorForServers** directory of your Site connector distribution directory.
2. Double-click the UbisenseSiteConnectorForServers.msi file and the Ubisense Site Connector Service Setup wizard appears.
3. Click **Next**.
4. Choose the Destination Folder for the software.
   You can accept the default **C:\Program Files (x86)\Ubisense 2.1** or change to another destination.
5. Click **Next** and click **Install**.
6. When the installation is complete, click **Finish** to close the Ubisense Site Connector Service Setup wizard.

After installation is complete, start the service using Windows Services manager:

1. Open Services by typing **View local services** in the Start menu.
2. Start the service **UbisenseSiteConnectorServer 2.1**.

   The service is configured to start automatically on reboot.

You can also stop and start the site connector service from the command prompt (as an administrator):

```
net stop "UbisenseSiteConnectorServer 2.1"
net start "UbisenseSiteConnectorServer 2.1"
```

Installing Site connector server on Linux

For Linux, you can find the Site connector server executable in your distribution directory under **linux/server**. To install the executable:

1. Copy **ubisense_site_connector_server** onto your server
2. Create a startup script or systemd configuration to run the executable on startup.

Sample systemd startup script for the Site connector server

The following is a sample startup script for systemd.

```
[Unit]
Description=ubisense_site_connector_server daemon
After=multi-user.target

[Service]
User=platform
Group=platform
Type=simple
ExecStart=/home/platform/bin/ubisense_site_connector_server

[Install]
WantedBy=ubisense_service.target
```

Installing Site connector clients

Which client software you install depends on how you are using Site connector. See the Introduction to Site connector in the *Ubisense Site Connector Guide* available from the the Ubisense Documentation Portal for information on intended uses for Site connector and their configuration.

Installing the Site connector Client on Windows

1. Go to the **UbisenseSiteConnectorClient** directory of your Site connector distribution directory.
2. Double-click the UbisenseSiteConnectorClient.msi file and the Ubisense Site Connector Client Setup wizard appears.
3. Click **Next**.
4. Choose the Destination Folder for the software.
   You can accept the default **C:\Program Files (x86)\Ubisense 2.1** or change to another destination.
5. Click **Next**.
6. Input the server IPaddress and site connector port number (these can be changed after installation). 49983 is the default port number which will be used in most installations. By default **Run in standalone mode** is selected. In standalone mode the client will only connect to the network of the Site Connector server it connects to. When not in standalone mode Site Connector will connect the networks of the client and server together. This should only be done if the effects are fully understood. After installation the standalone mode setting can only be changed with Platform Control (providing you have access to it).
7. Click **Next** and click **Install**.
8. When the installation is complete, click **Finish** to close the Ubisense Site Connector Client Setup wizard.

Installing the Site connector Client for Servers on Windows

1. Go to the **UbisenseSiteConnectorClientForServers** directory of your Site connector distribution directory.
2. Double-click the UbisenseSiteConnectorClientForServers.msi file and the Ubisense Site Connector Client for Servers Setup wizard appears.
3. Click **Next**.
4. Choose the Destination Folder for the software.
   You can accept the default **C:\Program Files (x86)\Ubisense 2.1** or change to another destination.
5. Click **Next**.

6. Input the server IPaddress and site connector port number (these can be changed after installation). 49983 is the default port number which will be used in most installations. By default **Run in standalone mode** is selected. In standalone mode the client will only connect to the network of the Site Connector server it connects to. When not in standalone mode Site Connector will connect the networks of the client and server together. This should only be done if the effects are fully understood. After installation the standalone mode setting can only be changed with Platform Control (providing you have access to it).
7. Click **Next** and click **Install**.
8. When the installation is complete, click **Finish** to close the Ubisense Site Connector Client for Servers Setup wizard.

Installing the Site connector Client on Linux

There is no particular step to install the client. Launch the application **ubisense_site_connector_client**. Following are examples of scripts for starting or restarting the client.

Sample cron script to restart the Site connector client

To ensure that the daemon will restart in the event of a fatal failure, you can write a cron script such as this one:

```
#!/bin/bash
if [[ ! `pidof -s ubisense_site_connector_client ` ]]; then
    invoke-rc.d ubisense_site_connector_client start
fi
```

Sample systemd startup script for the Site connector client

The following is a sample startup script for systemd.

```
[Unit]
Description=ubisense_site_connector_client daemon
After=multi-user.target

[Service]
User=platform
Group=platform
Type=simple
ExecStart=/home/platform/bin/ubisense_site_connector_client

[Install]
WantedBy=ubisense_service.target
```