Universal Tracker 1.0 Installation Guide

A guide to installing Sitecore Universal Tracker

December 14, 2021

Table of Contents

1. About Universal Tracker

The Sitecore Universal Tracker (UT) lets you collect interaction data from any channel, including Sitecore websites (web tracker), other websites (JS tracker), and mobile applications (mobile tracker).

The difference between xConnect and UT is that xConnect collects completed interactions, while UT collects live interactions as they happen, and submits them to xConnect when they are completed.

Universal Tracker consists of four parts:

- The collection service, which collects interaction data.
- The processing service, which processes the collected data.
- Database schemas for implementing *ITrackingInteractionCache* in Microsoft SQL or Azure databases. This component is a plugin for UT and has its own configuration.
- The Status plugin, which shows basic status information for the server, and can be used to ensure the service is running and can reach its dependencies. This plugin is included for both the collection service and the processing service and has its own configuration.

1.1. The Universal Tracker collection service

The Universal Tracker (UT) collection service is a .Net core web API service. It collects events from interactions as they occur, or soon after they occurred, and stores them temporarily until the interaction completes. When the interaction completes, the processing service processes it.

1.2. The Universal Tracker processing service

The Universal Tracker (UT) processing service is a .Net core web API service. It gets data from storage, runs it through pre-filtering, enrichment, and post-filtering, and sends it to xConnect. It is based on .Net core and the Proto.Actor model. You can run UT processing as a web application or as a Windows service.

The processing service consists of the following three parts:

- ChannelManagement
- Engine
- WebHost

1.2.1. ChannelManagement

ChannelManagement lets you define the channels that will send interactions. Each channel has various settings to let you configure it to process the interactions in a specific way.

Each channel is described by the *Channel* and *ChannelDefinition* types. The *Channel* type describes a channel in general. The *ChannelDefinition* type describes the pipelines for a channel.

Base types for pipelines and processors are described in the *Tracking.Processing.Abstractions* project. You can define your custom pipelines and processors based on *BasePrefilterPipeline*, *BaseEnrichmentPipeline*, *BasePostFilteringPipeline*, and *Processor*.

1.2.2. Engine

The engine is based on the Proto.Actor model. It retrieves interactions from the *ITrackingInteractionCache*, runs them through defined pipelines, and pushes them to the xConnect client. When the engine has pushed an interaction to xConnect, it deletes the interaction from the *ITrackingInteractionCache*.

1.2.3. WebHost

The WebHost is an entry point that defines how the processing service works. You can run it as either a Windows service or a web application. The WebHost contains the `sitecorehost.xml` configuration file that is a part of Sitecore.Host, and which contains the logic responsible for configuring *logger* and *aggregation service*. In `sitecorehost.xml` you can define your own commands for running WebHost. You can find more about this in the Sitecore.Host documentation.

2. Prerequisites for installing Universal Tracker

Before you install Universal Tracker (UT) you must have:

- Sitecore 9.1 or 9.1.1.

You can install UT on-premise or on Azure. Regardless of which installation you choose, you must first:

- Download the Sitecore Universal Tracker package
- Change the execution policy in PowerShell

2.1. Download the Sitecore Universal Tracker package

You must download the Sitecore Universal Tracker package from the Sitecore download page. This package contains the following files:

- `Sitecore.Tracking.Collection.Service.1.0.0-r00045.deploy.zip`
- `Sitecore.Tracking.Collection.Service.1.0.0-r00045.wdp.zip`
- `Sitecore.Tracking.Processing.Service.1.0.0-r00070.deploy.zip`
- `Sitecore.Tracking.Processing.Service.1.0.0-r00070.wdp.zip`
- `Sitecore.Tracking.Sql.1.0.0-r00056.dacpac.zip`
- `Sitecore.Tracking.Sql.1.0.0-r00056.deploy.zip`
- `deployAllAzure.ps1` - script for deploying all UT parts to Azure in one go
- `readme.md`

After downloading the UT package, unzip the files to a folder on your local machine.

2.2. Change the execution policy in PowerShell

To execute the PowerShell scripts that install the UT service, you must ensure that you can execute signed scripts in PowerShell.

To change the execution policy in PowerShell:

1. Start a PowerShell session with administrator rights.
2. Execute the following line in the PowerShell console:

```
Set-ExecutionPolicy RemoteSigned
```

You can now run scripts signed by a trusted publisher on your local computer.

3. Install the Universal Tracker on-premise

Before you install Universal Tracker (UT) on-premise, make sure you have all the prerequisites in place, and that your system fulfills the following environment requirements:

- Windows 10 or Windows Server 2016.
- IIS 10. For further information, see the recommendation from Microsoft.
- .Net 4.7.1 and later.
- IIS .Net Core module .Net Core 2.1 runtime and Hosting Bundle for Windows.
- Database SQL 2016R2 or SQL 2017.
- Web Deploy version 3 or later.
- URL Rewrite IIS module version 2.1 or later.

To install UT on-premise, install the following components:

- The UT SQL database
- The UT collection service
- The UT processing service

> **NOTE**
> You must install the UT SQL database before installing the services.

You can install all three parts on one server. To scale the installation vertically, you can install the collection service, the processing services, and the database on separate, dedicated servers. The collection and processing services must connect to the same database.

You can scale the installation horizontally by adding more collection services, processing services, or databases. If you have more than one database, each pairing of collection and processing services must use the same database. A processing service can only process data from a collection service that is connected to the same database as the processing service.

3.1. Install the SQL database on-premise

You install the SQL database using a PowerShell script included in the Universal Tracker (UT) package. There are a number of parameters you must specify when you run the script:

Parameter	Description	Note
sourceDacFile	The absolute path to the unzipped `*.dacpac` file. For example: `c:\Sitecore.Tracking.Sql.<version>.dacpac`.	Required **IMPORTANT** The path must not contain whitespace characters.
serverName	The name of the SQL server	Required
dbName	The name of the SQL database	Required
dbUser	A user name for access to the SQL database	Required
dbPassword	The password for the database user name	Required

To install the SQL database:

1. Perform the required pre-installation steps described in Prerequisites for installing Universal Tracker.
2. Copy these archives from the Sitecore UT package to a new folder on your machine:
   - `Sitecore.Tracking.Sql.1.0.0-r00056.deploy.zip`
   - `Sitecore.Tracking.Sql.1.0.0-r00056.dacpac.zip`

   NOTE
   The folder name must not contain whitespace characters.
3. Unpack the `*.deploy.zip` and `*.dacpac.zip` files to new folders on your machine.
4. Open a PowerShell console with administrator rights. Navigate to the folder where you unpacked the `*.deploy.zip` files. In this folder, navigate to the `OnPremDeploymentInfrastructure` folder.

   NOTE
   The `OnPremDeploymentInfrastructure` folder contains the `onPremDeploy.ps1` file, which is a script that deploys the SQL database using the MSDeploy tool.

   You must not edit this file.
5. Run the `onPremDeploy.ps1` file with the required parameters. Parameters must be enclosed in double quotation marks ("`<parameter>`"). For example:

```
.\onPremDeploy.ps1
    -sourceDacFile "c:\Sitecore.Tracking.Sql.<version>.dacpac"
```

```
-serverName "my-db-server"
-dbName "my-db"
-dbUser "my-db-User"
-dbPassword "<password>"
```

3.2. Install the collection service on-premise

You install the collection service using a PowerShell script included in the Universal Tracker package. The PowerShell script takes several parameters. The following table shows which parameters require you to specify a value, and which parameters you only need to specify if you need to change the default value.

Parameter	Description	Note
licenseXmlPath	The absolute path to your `license.xml` file. For example: `c:\license.xml`	Required
wdpPackagePath	The absolute path to the `*.wdp.zip` file. For example: "`c:\Sitecore.Tracking.Collection.Service.<version>.wdp.zip`".	Required **IMPORTANT** The path must not contain whitespace characters.
databaseConnectionString	The connection string to the SQL database. For example: `user id={your_name}; password={your_pass}; data source={your_sql_server}; database={your_database}; ConnectRetryCount=5; ConnectRetryInterval=10; Connection Timeout=50;`	You can set the `databaseConnectionString` when you run `onPremDeploy.ps1`, or afterwards by editing the `config.xml` file. The database connection string must be the same for the collection and processing services.
siteFolderPath	The path to the root folder of your websites. The default value is `c:\inetpub\wwwroot`.	Use only if you need to change the default value.
instanceName	The website instance name. The default value is *Sitecore.Tracking.Collection.Service*.	Use only if you need to change the default value. Make sure the website name is not already in use on your machine.
IISAppPollName	The IIS application pool for new websites. The default value is *Sitecore.Tracking.Collection.Service*.	Use only if you need to change the default value. Make sure the pool name is not already in use on your machine.
siteHttpPort	The HTTP port that will be used for new websites. The default value is *80*.	Use only if you need to change the default value.
siteHttpsPort	The HTTPS port that will be used for new websites. The default value is *443*.	Use only if you need to change the default value.

To install the collection service:

1. Perform the required pre-installation steps described in Prerequisites for installing Universal Tracker.
2. Copy the following archives from the Sitecore Universal Tracker package to a new folder on your machine:
   - `Sitecore.Tracking.Collection.Service.1.0.0-r00045.deploy.zip`
   - `Sitecore.Tracking.Collection.Service.1.0.0-r00045.wdp.zip`

   NOTE
   The folder name must not contain whitespace characters.
3. Unpack the `*.deploy.zip` file to a new folder on your machine.
4. Open a PowerShell console with administrator rights. Navigate to the folder where you unpacked the `*.deploy.zip` files. In this folder, navigate to the `OnPremDeploymentInfrastructure` folder.

   NOTE
   Do not edit the `onPremDeploy.ps1` file or the `setParameters.xml` file in the `OnPremDeploymentInfrastructure` folder. `onPremDeploy.ps1` is a script that deploys the collection service using the MsDeploy tool, and `setParameters.xml` defines the deployment parameters the MsDeploy tool requires.
5. In the PowerShell console, run the `onPremDeploy.ps1` file with the required parameters and any optional parameters you want to change from their defaults. You must enclose parameters in double quotation marks ("`<parameter>`"). For example:

```
.\onPremDeploy.ps1
    -wdpPackagePath "c:\Sitecore.Tracking.Collection.Sservice.<version>.wdp.zip"
    -licenseXmlPath "c:\license.xml"
    -siteFolderPath "c:\inetpub\wwwroot"
    -siteHttpPort "80"
```

6. If you did not specify the database connection string when you ran the `onPremDeploy.ps1` file, you must set it now in the `<instance_name>\sitecore\Sitecore.Tracking.SqlServer\Config\config.xml` file, in the <*ConnectionString*> node. For example:

```
<ConnectionString>user id=myUser;password=myPassword;data source=my-db-Server;database=my-db;ConnectRetryCount=5;ConnectRetryInterval=10;Connection Timeout=50;</ConnectionString>
```

7. Add the binding for your service in the `%windir%\System32\drivers\etc\hosts` file.
8. In IIS, navigate to the website you specified in `instanceName`. Navigate to **Bindings**. Double-click on the port 443 binding and select the SSL certificate.

NOTE
When you change the configuration, you must restart the Universal Tracker collection service in order for the changes to take effect.

3.3. Install the processing service on-premise

You install the processing service using a PowerShell script included in the Universal Tracker package. The PowerShell script takes several parameters. The following table shows which parameters require you to specify a value, and which parameters you only need to specify if you need to change the default value.

Parameter	Description	Note
licenseXmlPath	The absolute path to your license.xml file. For example: `c:\license.xml`	Required.
wdpPackagePath	The absolute path to the `*.wdp.zip` file. For example: "`c:\Sitecore.Tracking.Processing.Service.<version>.wdp.zip`".	Required. **IMPORTANT** The path must not contain whitespace characters.
databaseConnectionString	The connection string to the SQL database. For example: `user id={your_name}; password={your_pass}; data source={your_sql_server}; database={your_database}; ConnectRetryCount=5; ConnectRetryInterval=10; Connection Timeout=50;`	You can set the databaseConnectionString when you run `onPremDeploy.ps1`, or afterwards by editing the `config.xml` file. The database connection string must be the same for the collection and processing services.
siteFolderPath	The path to the root folder of your websites. The default value is `c:\inetpub\wwwroot`.	Use only if you need to change the default value. Make sure that the combination of `siteFolderPath` and `instanceName` is not already in use on your machine.
instanceName	The website instance name. The default value is *Sitecore.Tracking.Processing.Service*.	Use only if you need to change the default value. Make sure the combination of `siteFolderPath` and `instanceName` is not already in use on your machine.
IISAppPollName	The IIS application pool for new websites. The default value is *Sitecore.Tracking.Processing.Service*.	Use only if you need to change the default value. Make sure the pool name is not already in use on your machine.
siteHttpPort	The HTTP port that will be used for new websites. The default value is *80*.	Use only if you need to change the default value.
siteHttpsPort	The HTTPS port that will be used for new websites. The default value is *443*.	Use only if you need to change the default value.
serviceName	Windows service name. The default value is *Sitecore Universal Tracker Processing Service*.	Use only if you need to change the default value. Make sure the Windows service name is not already in use on your machine.

Parameter	Description	Note
serviceDisplayName	Windows service display name. The default value is *Sitecore Universal Tracker Processing Service.*	Use only if you need to change the default value.
serviceDescription	Windows service description. The default value is *Sitecore Host Based Tracking Processing Service*.	Use only if you need to change the default value.
deployOnlyWebApp	Specify *True* to install only the web application. Specify *False* to install both web application and windows service. The default is to install both.	Use only if you need to change the default value. **NOTE** Due to limitations in Sitecore.Host, the processing service can *not* be installed as a web service.

To install the processing service:

1. Perform the required pre-installation steps described in Prerequisites for installing Universal Tracker.
2. Copy these archives from the Sitecore Universal Tracker package to a new folder on your machine:
   - `Sitecore.Tracking.Processing.Service.1.0.0-r00070.deploy.zip`
   - `Sitecore.Tracking.Processing.Service.1.0.0-r00070.wdp.zip`

   NOTE
   The folder name must not contain whitespace characters.
3. Unpack the `*.deploy.zip` file to a new folder on your machine.
4. Open a PowerShell console with administrator rights. Navigate to the folder where you unpacked the `*.deploy.zip` files. In this folder, navigate to the `OnPremDeploymentInfrastructure` folder.

   NOTE
   The folder contains the `onPremDeploy.ps1` file, which is a script that deploys the processing service using the MsDeploy tool, and the `setParameters.xml` file, which defines the deployment parameters the MsDeploy tool requires.

   Do not edit these two files.
5. Run the `onPremDeploy.ps1` file with the required parameters and any optional parameters you want to change from their defaults. Parameters must be enclosed in double quotation marks ("`<parameter>`"). For example:

```
.\onPremDeploy.ps1
    -wdpPackagePath "c:\Sitecore.Tracking.Processing.Service.<version>.wdp.zip"
    -licenseXmlPath "c:\license.xml"
```

```
    -siteFolderPath "c:\inetpub\wwwroot"
    -siteHttpPort "80"
```

6. If you did not specify the database connection string, you must set it now in the `<instance_name>\sitecore\Sitecore.Tracking.SqlServer\Config\config.xml` file, in the <*ConnectionString*> node. For example:

```
<ConnectionString>user id=myUser;password=myPassword;data source=my-db-server;database=my-db;ConnectRetryCount=5;ConnectRetryInterval=10;Connection Timeout=50;</ConnectionString>
```

7. Set the connection string to the XConnect client for sending analytics data to XConnect, by opening the `<instance_name>\sitecore\Sitecore.Tracking.Processing.Engine\Config\config.xml` file and updating the *XConnect.ServiceUrl* and *XConnect.ClientCertificate* nodes.
8. If you deployed the processing service as a Windows service, you must open the Windows Services desktop application and start the *Sitecore Universal Processing Service* Windows service.
9. Add the binding for your service in the `%windir%\System32\drivers\etc\hosts` file.
10. Open a browser and navigate to `<instance_name>/status` to check the status of the processing service.

4. Prepare to deploy the Universal Tracker on Azure

Before you deploy any parts of Universal Tracker to Azure, you must:

- Install the Azure PowerShell modules
- Ensure the Azure modules are updated
- Register the Azure application
- Set Azure application permissions
- Prepare the deployment parameters

4.1. Install the Azure PowerShell modules

To install the Azure PowerShell modules:

1. Open a PowerShell console with administrator rights.
2. In the PowerShell console command line, enter the command `Install-Module AzureRM`.

> **NOTE**
> If you want to specify a special PowerShell repository, use the `-Repository "repository name"` parameter. For example, `Install-Module AzureRM -Repository "local-repo"`

4.2. Ensure the Azure modules are updated

Carry out the following steps to make sure the Azure modules are updated:

1. Open a PowerShell console with administrator rights.
2. In the PowerShell command line, enter the command `Update-Module AzureRM`.
3. Restart your PowerShell session.
4. Verify that you have AzureRM.Profile version 5.6.0 or later by entering the command `Find-Module AzureRM.Profile` in the command line.
5. Verify that you have AzureRM.Storage version 5.2.0 or later by entering the command `Find-Module AzureRM.Storage` in the command line.

4.3. Register the Azure application

To set up the Azure Active Directory (AD) application:

1. Navigate to your Azure portal (`https://portal.azure.com/`) and sign in.
2. On the **Azure Active Directory** tab, click **App registrations**.
3. At the top of **App registrations** window, click **+ New application registration**.
4. In the **Create** window, fill in the following parameters:

Parameter	Value
Name	any
Application type	Web app / API
Sign-on URL	any

5. Click **Create**.

> **NOTE**
> For more information on the Azure AD, please refer to the Microsoft documentation on Azurethe Microsoft documentation on Azure.

4.4. Set Azure application permissions

When you have created the new application, you must add the application to the `Contributor` role of the subscription.

To set the Azure application permissions:

1. Navigate to your Azure portal (`https://portal.azure.com/`) and sign in.
2. To open your profile , in the top-right corner of the window, click your profile name.
3. In the profile panel, click **More links** (···) and in the drop-down list click **My permissions**.
4. In the **Subscription** drop-down list, click your subscription.
5. Click the **Click here to view complete access details for this subscription** link.
6. On the **Access control (IAM)** tab, click **Add**.
7. In the **Add permissions** panel, in the **Role** drop-down list, click the **Contributor** role.
8. In the **Assign access to** drop-down list, click **azure AD user, group, or application**.
9. In the **Select** drop-down list, click the application you created and then click **Save**.

4.5. Prepare the deployment parameters

Before the deployment, you must get the following values from your Azure portal (https://portal.azure.com/):

- Azure subscription ID
    1. In the search box, enter *subscriptions*.
    2. In the search results, click *Subscriptions*.
    3. Find the relevant subscription and copy the *Subscription ID* value.
- Azure tenant IDAzure tenant ID
    1. Navigate to **Azure Active Directory**.
    2. Click **Properties**.
    3. Copy the *Directory ID* value.
- Azure client idAzure client id
    1. Navigate to the **Azure Active Directory**.
    2. On the **App Registrations** tab, click your application.
    3. Copy the *Application ID* value.
- Azure application secretAzure application secret
    1. Navigate to the **Azure Active Directory**.
    2. On the **App Registrations** tab, click your application.
    3. Click **Certificates & secrets**.
    4. In the **Client secrets** section, click **New secret**. Enter a description of the secret, and the duration you want it to be valid for, and click **Add**. The window now displays the key value of the client secret.

> **IMPORTANT**
> You must copy the generated key value before closing the tab. You will *not* be able to see the key again after you have closed the tab.

5. Deploy Universal Tracker to Azure

Before you deploy Universal Tracker (UT) to Azure, make sure you have all the prerequisites in place.

To deploy a new, complete UT installation:

- Use the `deployAllAzure.ps1` script to deploy all the elements of UT at once.

Alternatively, deploy the following elements of UT individually:

- The UT SQL database

> **NOTE**
> If you deploy the UT elements individually, you must deploy the UT SQL database before deploying the services.

- The UT collection service
- The UT processing service

5.1. Deploy to Azure using deployAllAzure.ps1

You can deploy all three parts of the Universal Tracker service to Azure at the same time using the `deployAllAzure.ps1` PowerShell script included in the Universal Tracker package. You must specify the following parameters when you run the script:

Parameter	Description	Note
subscriptionId	Your Azure subscription id.	Required
resourceGroupName	The name of the Azure resource group.	Required. The default value is *scuniversaltracking-rg*.
resourceGroupLocation	The location of the Azure resource group.	Required. The default value is *North Europe*.
licenseXmlPath	The absolute path to your `license.xml` file. For example: `c:\license.xml`	Required
collectionWdpPackagePath	The absolute path to the `Sitecore.Tracking.Collection.service.<version>.wdp.zip` file. For example: `c:\Sitecore.Tracking.Collection.Service.<version>.wdp.zip`	Required **IMPORTANT** The path must not contain whitespace characters.

Parameter	Description	Note
collectionInfrastructureDirPath	The path to the `AzureDeploymentInfrastructure` folder in the folder where you unpack the `Sitecore.Tracking.Collection.Service.<version>.deploy.zip` file.	Required
processingWdpPackagePath	The absolute path to the `Sitecore.Tracking.Processing.Service.<version>.wdp.zip` file. For example: `c:\Sitecore.Tracking.Processing.Service.<version>.wdp.zip`	Required
processingInfrastructureDirPath	The path to the `AzureDeploymentInfrastructure` folder in the folder where you unpack the `Sitecore.Tracking.Processing.Service.<version>.deploy.zip` file.	Required
storageContainerName	The name of the Azure storage container.	Required. The default name is *scuniversaltracking-sa*.
azureSecurityKey	The Azure security key.	Required
azureClientID	Your Azure client ID.	Required
azureTenantID	The Azure active directory ID.	Required
targetServerName	The name of the SQL Server instance to use. If it does not exist, it will be created.	Required
targetDatabaseName	The name of the SQL Database to create. If it already exists on server, it will be re-created.	Required
targetuser	Administrator username to access SQL Server instance.	Required
targetUserPassword	The password for the administrator user.	Required
sourceDacFile	The absolute path to the unzipped `dacpac` file. For example: `c:\Sitecore.Tracking.Sql.<version>.dacpac`.	Required **IMPORTANT** The path must not contain whitespace characters.
sqlInfrastructureDirPath	The path to the `AzureDeploymentInfrastructure` folder in the folder where you unpack the `Sitecore.Tracking.Sql.<version>.deploy.zip` file.	Required

To deploy Universal Tracker to Azure:

1. Perform the required pre-installation steps described in Preparing to deploy the Universal Tracker to Azure.
2. Unpack the Sitecore Universal Tracker package to a new folder on your machine.

   NOTE
   The folder name must not contain whitespace characters.

3. Unpack each of the following files to a separate new folder on your machine:
   - `Sitecore.Tracking.Sql.1.0.0-r00056.deploy.zip`

- `Sitecore.Tracking.Collection.Service.1.0.0-r00045.deploy.zip`
- `Sitecore.Tracking.Processing.Service.1.0.0-r00070.deploy.zip`
- `Sitecore.Tracking.Sql.1.0.0-r00056.dacpac.zip`

4. Navigate to the folder where you unpacked the `Sitecore.Tracking.Collection.Service.<version>.deploy.zip` file, then navigate to the `AzureDeploymentInfrastructure\templates` folder within.
5. In the `serviceDeploy.parameters.json` file, change `webAppName` to your unique web app name.

> **NOTE**
> When you select a name for your web app, you must follow the naming conventions for Microsoft Azurenaming conventions for Microsoft Azure.

6. In the `storageDeploy.parameters.json` file, change the `name` parameter to your unique storage account name.

> **NOTE**
> When you select a name for your storage account, you must follow the naming conventions for Microsoft Azurenaming conventions for Microsoft Azure.

7. Navigate to the folder where you unpacked the `Sitecore.Tracking.Processing.Service.<version>.deploy.zip` file, then navigate to the `AzureDeploymentInfrastructure\templates` folder within.
8. In the `serviceDeploy.parameters.json` file, change `webAppName` to your unique web app name.

> **NOTE**
> When you select a name for your web app, you must follow the naming conventions for Microsoft Azurenaming conventions for Microsoft Azure.

9. In the `storageDeploy.parameters.json` file, change the `name` parameter to the same name that you set in step 6.
10. Open a PowerShell console with administrator rights. Navigate to the folder where you unpacked the Sitecore Universal Tracker package. This folder contains the `deployAllAzure.ps1` file.
11. Run the `deployAllAzure.ps1` file with the required parameters. You must enclose parameters in double quotation marks (`"<parameter>"`). For example:

```
.\deployAllAzure.ps1
    -subscriptionId "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    -azureSecurityKey "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
    -azureClientID "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    -azureTenantID "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    -targetServerName "my-db-server"
    -targetDatabaseName "my-db"
```

```
    -targetUser "SQLAdminUser"
    -targetUserPassword "adminpw"
    -licenseXmlPath "c:\license.xml"
    -collectionWdpPackagePath "c:\Sitecore.Tracking.Collection.Service.<version>.wdp.zip."
    -collectionInfrastructureDirPath "c:\collection\AzureDeploymentInfrastructure"
    -processingWdpPackagePath "c:\Sitecore.Tracking.Processing.Service.<version>.wdp.zip"
    -processingInfrastructureDirPath "c:\processing\AzureDeploymentInfrastructure"
    -sourceDacFile "c:\Sitecore.Tracking.Sql.<version>.dacpac"
    -sqlInfrastructureDirPath  "c:\sql\AzureDeploymentInfrastructure"
```

5.2. Deploy the SQL database to Azure

You deploy the SQL database to Azure using a PowerShell script included in the Universal Tracker package. You must specify the following parameters when you run the script:

Parameter	Description	Note
subscriptionId	Your Azure subscription ID.	Required
resourceGroupName	The name of the Azure resource group.	Required. The default value is *scuniversaltracking-rg*.
resourceGroupLocation	The location of the Azure resource group.	Required. The default value is *North Europe*.
sourceDacFile	The absolute path to the unzipped `dacpac` file. For example: "`c:\Sitecore.Tracking.Sql.<version>.dacpac`".	Required **IMPORTANT** The path must not contain whitespace characters.
storageContainerName	The name of the Azure storage container.	Required. The default is *scuniversaltracking-sa*.
azureSecurityKey	The Azure security key.	Required
azureClientID	Your Azure client ID.	Required
azureTenantID	The Azure active directory ID.	Required
targetServerName	The name of the SQL Server instance to use. If it does not exist, it will be created.	Required
targetDatabaseName	The name of the SQL Database to create. If it already exists on the server, it will be re-created.	Required
targetuser	Administrator username to access SQL Server instance.	Required
targetUserPassword	The password for the administrator user.	Required

To deploy the SQL database to Azure:

1. Perform the required pre-installation steps described in Preparing to deploy the Universal Tracker to Azure.

2. Unpack the following archives from the Sitecore Universal Tracker package to separate new folders on your machine:
   - `Sitecore.Tracking.Sql.1.0.0-r00056.deploy.zip`
   - `Sitecore.Tracking.Sql.1.0.0-r00056.dacpac.zip`

   NOTE
   The folder names must not contain whitespace characters.

3. Open a PowerShell console with administrator rights. Navigate to the folder where you unpacked the `*.deploy.zip` files.
4. Run the `azureDeploy.ps1` file with the required parameters. You must enclose parameters in double quotation marks (`"<parameter>"`). For example:

```
.\azureDeploy.ps1
    -subscriptionId "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    -resourceGroupName "myResourceGroup"
    -resourceGroupLocation "North Europe"
    -storageContainerName "myStorageContainer"
    -azureSecurityKey "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
    -azureClientID "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    -azureTenantID "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    -sourceDacFile "c:\Sitecore.Tracking.Sql.<version>.dacpac"
    -targetServerName "myServer"
    -targetDatabaseName "myDB"
    -targetUser "myUser"
    -targetUserPassword "myPassword"
```

5.3. Deploy the collection service to Azure

You deploy the collection service to Azure using a PowerShell script included in the Universal Tracker package. You can specify the following parameters when you run the script:

Parameter	Description	Note
subscriptionId	Your Azure subscription ID.	Required
resourceGroupName	The name of the Azure resource group.	Required. The default value is *scuniversaltracking-rg*.
resourceGroupLocation	The location of the Azure resource group.	Required. The default value is *North Europe*.
licenseXmlPath	The absolute path to your license.xml file. For example: `c:\license.xml`	Required

Parameter	Description	Note
wdpPackagePath	The absolute path to the `*.wdp.zip` file. For example: `"c:\Sitecore.Tracking.Collection.Service.<version>.wdp.zip"`.	Required **IMPORTANT** The path must not contain whitespace characters.
storageContainerName	The name of the Azure storage container.	Required
azureSecurityKey	The Azure security key.	Required
azureClientID	Your Azure client ID.	Required
azureTenantID	The Azure active directory ID.	Required

dbConnectionString	The connection string to the SQL database.
	
	For example:
	
	`user id={your_name};`
	`password={your_pass};`
	`data source={your_sql_server};`
	`database={your_database};`
	`ConnectRetryCount=5;`
	`ConnectRetryInterval=10;`
	`Connection Timeout=50;`
		The database connection string must be the same for the collection and processing services.

storageTemplate	The path to the ARM template for deploying the storage account. By default, this is the `templates` folder within the folder where you unzip the deployment package.	Required, if the file is not in the default folder.
storageParameters	The path to the ARM parameters file for deploying storage account. By default, this is the `templates` folder within the folder where you unzip the deployment package.	Required, if the file is not in the default folder.
serviceTemplate	The path to the ARM template for deploying the service to a WebApp. By default, this is the `templates` folder within the folder where you unzip the deployment package.	Required, if the file is not in the default folder.
serviceParameters	The path to the ARM parameters file for deploying the service to a WebApp. By default, this is the `templates` folder within the folder where you unzip the deployment package.	Required, if the file is not in the default folder.

To deploy the collection service:

1. Perform the required pre-installation steps described in Preparing to deploy the Universal Tracker to Azure.
2. Copy the following archives from the Sitecore Universal Tracker package to a new folder on your machine:
   - `Sitecore.Tracking.Collection.Service.1.0.0-r00045.deploy.zip`
   - `Sitecore.Tracking.Collection.Service.1.0.0-r00045.wdp.zip`

   NOTE
   The folder name must not contain whitespace characters.
3. Unpack the `*.deploy.zip` file to a new folder on your machine.
4. Navigate to the folder where you unpacked the deploy file. Navigate to the `AzureDeploymentInfrastructure` folder. This folder contains:

- `azureDeploy.ps1`, which is a script that deploys the collection service using the MSDeploy tool.
- The `templates` folder, which contains the following files that are used by the installation script:
  - `serviceDeployParameters.json`
  - `storageDeployParameters.json`
  - ARM templates for deploying web apps

5. In the `storageDeploy.parameters.json` file, change the `name` parameter to your unique storage account name.

> **NOTE**
> When you select a name for your storage account, you must follow the naming conventions for Microsoft Azurenaming conventions for Microsoft Azure.

6. In the `serviceDeploy.parameters.json` file, change `webAppName` to your unique web app name.

> **NOTE**
> When you select a name for your web app, you must follow the naming conventions for Microsoft Azurenaming conventions for Microsoft Azure.

7. Open a PowerShell console with administrator rights. Navigate to the folder where you unpacked the `*.deploy.zip` files. Navigate to the `AzureDeploymentInfrastructure` folder.
8. In the PowerShell console, run the `azureDeploy.ps1` file with the required parameters. You must enclose parameters in double quotation marks (`"<parameter>"`). For example:

```
.\azureDeploy.ps1
    -subscriptionId "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    -resourceGroupName "myResourceGroup"
    -resourceGroupLocation "North Europe"
    -wdpPackagePath "C:\deployment-package
\Sitecore.Tracking.Collection.Service.<version>.wdp.zip"
    -storageContainerName "myStorageContainer"
    -azureSecurityKey "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
    -azureClientID "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    -azureTenantID "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    -storageTemplate ".\storageTemplate.json"
    -storageParameters ".\storageParameters.json"
    -serviceTemplate ".\serviceTemplate.json"
    -serviceParameters ".\serviceParameters.json"
```

> **NOTE**
> If you change the configuration for the Universal Tracker collection service, you must restart the service for the changes to take effect.

5.4. Deploy the processing service to Azure

You deploy the processing service to Azure using a PowerShell script included in the Universal Tracker package.

> **NOTE**
> The Universal Tracker processing service cannot be hosted as an Azure Web Job.

You can specify the following parameters when you run the script:

Parameter	Description	Note
subscriptionId	Your Azure subscription ID.	Required
resourceGroupName	The name of the Azure resource group.	Required. The default value is *scuniversaltracking-rg*.
resourceGroupLocation	The location of the Azure resource group.	Required. The default value is *North Europe*.
licenseXmlPath	The absolute path to your `license.xml` file. For example: `c:\license.xml`	Required
wdpPackagePath	The absolute path to the `*.wdp.zip` file. For example: `c:\Sitecore.Tracking.Processing.Service.<version>.wdp.zip`.	Required **IMPORTANT** The path must not contain whitespace characters.
storageContainerName	The name of the Azure storage container.	Required
azureSecurityKey	The Azure security key.	Required
azureClientID	Your Azure client ID.	Required
azureTenantID	The Azure active directory ID.	Required

dbConnectionString	The connection string to the SQL database.
	
	For example:
	
	`user id={your_name};`
	`password={your_pass};`
	`data source={your_sql_server};`
	`database={your_database};`
	`ConnectRetryCount=5;`
	`ConnectRetryInterval=10;`
	`Connection Timeout=50;`
		The database connection string must be the same for the collection and processing services.

storageTemplate	The path to the ARM template for deploying the storage account. By default, this is the `templates` folder within the folder where you unzip the deployment package.	Required, if the file is not in the default folder.
storageParameters	The path to the ARM parameters file for deploying the storage account. By default, this is the `templates` folder within the folder where you unzip the deployment package.	Required, if the file is not in the default folder.
serviceTemplate	The path to the ARM template for deploying the processing service to a WebApp. By default, this is the `templates` folder within the folder where you unzip the deployment package.	Required, if the file is not in the default folder.

Parameter	Description	Note
serviceParameters	The path to the ARM parameters file for deploying the processing service to a WebApp. By default, this is the `templates` folder within the folder where you unzip the deployment package.	Required, if the file is not in the default folder.

To deploy the processing service:

1. Perform the required pre-installation steps described in Preparing to deploy the Universal Tracker to Azure.
2. Copy the following archives from the Sitecore Universal Tracker package to a new folder on your machine:
   - `Sitecore.Tracking.Processing.Service.1.0.0-r00070.deploy.zip`
   - `Sitecore.Tracking.Processing.Service.1.0.0-r00070.wdp.zip`

   NOTE
   The folder name must not contain whitespace characters.
3. Unpack the `*.deploy.zip` file to a new folder on your machine.
4. Navigate to the folder where you unpacked the deploy file. Navigate to the `AzureDeploymentInfrastructure` folder.
   This folder contains:
   - `azureDeploy.ps1`, which is a script that deploys the collection service using the MSDeploy tool.
   - The `templates` folder, which contains the following files that are used by the installation script:
     - `serviceDeploy.Parameters.json`
     - `storageDeploy.Parameters.json`
     - ARM templates for deploying web apps
5. In the `serviceDeploy.parameters.json` file, change `webAppName` to your unique web app name.

   NOTE
   When you select a name for your web app, you must follow the naming conventions for Microsoft Azurenaming conventions for Microsoft Azure.
6. In the `storageDeploy.parameters.json` file, change the `name` parameter to your unique storage account name.

   NOTE
   When you select a name for your storage account, you must follow the naming conventions for Microsoft Azurenaming conventions for Microsoft Azure.

7. Open a PowerShell console with administrator rights. Navigate to the folder where you unpacked the `*.deploy.zip` files. Navigate to the folder `AzureDeploymentInfrastructure`.
8. In the PowerShell console, run the `azureDeploy.ps1` file with the required parameters. You must enclose parameters in double quotation marks (`"<parameter>"`). For example:

```
.\azureDeploy.ps1
    -subscriptionId "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    -resourceGroupName "myResourceGroup"
    -resourceGroupLocation "North Europe"
    -wdpPackagePath "C:\Sitecore.Tracking.Processing.Service.<version>.wdp.zip"
    -storageContainerName "myStorageContainer"
    -azureSecurityKey "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
    -azureClientID "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
    -azureTenantID "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
```

NOTE

If you change the configuration for the Universal Tracker processing service, you must restart the service for the changes to take effect.

6. Troubleshooting

This section can help you resolve the most common problems you might encounter with Universal Tracker.

6.1. Status page

Both collection and processing services include a status page that shows the status for the service and its connections. You can access the status pages at the URL `<instanceName>/Status`. For example, if you installed the collection service with the default *instancename*, the URL for the collection status page is `Sitecore.Tracking.Collection.Service/status`.

To make sure that connections to SQL and xConnect are set properly, navigate to both collection and processing `/Status` pages. Every indicator on each status page should be green.

By default, the status pages are only available from the local machine. If you need to access the status pages from a remote machine, navigate to the `website\sitecore \Sitecore.Tracking.Plugin.Status\Config` folder, open the `config.xml` file, and set the *LocalAccessOnly* property to *false*.

6.2. xConnect connection failure

If you experience an xConnect connection failure:

- Make sure your thumbprint in the connection string is in uppercase letters. For example: 3D703B5198D6D3CEE1D0C1B1BC9ECB6D34989BA4.
- Make sure that the client authentication certificate (under the local machine's Personal store) has read permissions for the IIS_IUSR group and the NETWORK SERVICE group.
- For Azure deployments, make sure that application settings have *WEBSITE_LOAD_CERTIFICATES* set to *.

On Azure deployments, you can see the installed certificates using PowerShell. For example:

1. Open Azure Portal, go to **Advanced tools**, click **Go**.
2. Click **Debug console, PowerShell**.
3. Run the command `cd cert:\currentuser\my`.
4. Run the command `dir`.

6.3. xConnect connection is present but interaction is not sent

If you encounter this problem:

- Make sure that the interaction contains a configured channel.
- Check if the log files contain the following exception:

```
Sitecore.xConnect.XdbCollectionUnavailableException:
An error occurred while sending the request.
---> Sitecore.Xdb.Common.Web.ConnectionTimeoutException: A task was canceled.
---> System.Threading.Tasks.TaskCanceledException: A task was canceled.
```

  If they do contain this exception, check your batch size and *Aggregation.xConnectClientTimeout* settings. Sending a large amount of data requests to xConnect can result in failure due to a timeout, in which case, you must increase the *Aggregation.xConnectClientTimeout* setting.

6.4. Service fails to start

The services require that the `license.xml` file is present in the `<websitename>/sitecoreruntime` folder. Normally, the install scripts automatically copy the `license.xml` file to this folder. If the service fails to start, check that the license file is present.

6.5. Known Issues

6.5.1. Request failure

Submitting a large batch of interactions with a large amount of events in each interaction in one single request can result in failure. This is due to the limitation for the request length set in the configuration of the xConnect Collection service.

You might be experiencing this issue, if the log files on the UT processing service contain the following exception:

```
Sitecore.xConnect.XdbCollectionUnavailableException: The HTTP response was not successful:
InternalServerError
 at Sitecore.xConnect.Client.WebApi.ConfigurationWebApiClient.<Refresh>d__4.MoveNext()
```

and the log files on xConnect collection service contain an exception similar to the following:

```
Microsoft.OData.ODataException: The maximum number of bytes allowed to be read from the stream has
been exceeded.
After the last read operation, a total of 1055946 bytes has been read from the stream;
however, a maximum of 1048576 bytes is allowed.
   at
Microsoft.OData.MessageStreamWrapper.MessageStreamWrappingStream.IncreaseTotalBytesRead(Int32
bytesRead)
```

There are two possible ways to fix the issue:

- Decrease the batch size value in the configuration.
- Increase the value of the *maxRequestLength* and *maxAllowedContentLength* settings in the `Web.config` file of the xConnect Collection service.

For additional information about the *maxRequestLength* and *maxAllowedContentLength* settings please refer to the following articles:

https://docs.microsoft.com/en-us/previous-versions/dotnet/netframework-1.1/e1f13641(v=vs.71)

https://docs.microsoft.com/en-us/iis/configuration/system.webserver/security/requestfiltering/requestlimits/

6.5.2. The service instance shuts down after timeout period

For Azure deployments, consider implementing a job schedulera job scheduler.

For on-premise deployments:

1. In IIS, navigate to the *Application Pools*, **select the app pool for the processing service, select Advanced settings**.
2. Set *Start Mode* to *AlwaysRunning*.
3. Set *Idle Time-out* to *0*.
4. Recycle the app pool.