# Installing the Sitecore Demandbase module using the Sitecore package

## Acquire the package

The Sitecore Demandbase plugin can be found on the marketplace.

## Install the package

Using preferably a non-production environment, install the module package using Sitecore's Development Tools > Installation Wizard. \* Upload package to sitecore \* Run installation procedure

There are no sitecore items involved in the module package. The entirety of the module is self contained through initialize pipeline procedures. This means if something is accidentally deleted it can be easily repaired with a simple recycle of the app pool.

## Configure the module

Installing the module will place a configuration file in sitecore at App\_Config/Include called SitecoreDemandbase.config. Inside this config is an Xml node to enter your Demandbase api access key.

# Configuring Demandbase

There are a few configuration options available. Many of which will never need to be modified, however it is important to understand what the options do

<?xml version="1.0"?>  
<configuration xmlns:patch="http://www.sitecore.net/xmlconfig/">  
 <sitecore>  
 <!--   
 Uncomment model node for Xdb storage of demandbase data  
 -->  
 <model>  
 <elements>  
 <element interface="SitecoreDemandbase.Data.Interface.IXdbFacetDemandbaseData, SitecoreDemandbase" implementation="SitecoreDemandbase.Data.XdbFacetDemandbaseData, SitecoreDemandbase" />  
 </elements>  
 <entities>  
 <contact>  
 <facets>  
 <facet name="Demandbase Data" contract="SitecoreDemandbase.Data.XdbFacetDemandbaseData, SitecoreDemandbase" />  
 </facets>  
 </contact>  
 </entities>  
 </model>  
 <pipelines>  
 <httpRequestBegin>  
 <processor patch:after="\*[@type='Sitecore.Pipelines.HttpRequest.EnsureServerUrl, Sitecore.Kernel']" type="SitecoreDemandbase.Pipeline.HttpRequestBegin.ValidateUser, SitecoreDemandbase" />  
 </httpRequestBegin>  
 <initialize>  
 <processor type="SitecoreDemandbase.Pipeline.Initialize.InitializeDemandbase, SitecoreDemandbase" >  
 <param name="restApi">http://api.demandbase.com/api/v2/ip.json</param>  
 <param name="key">Enter\_your\_demandbase\_key\_here</param>  
 <param name="demandbaseIp">4.16.87.224</param>  
 <!--   
 Uncomment following for session storage of demandbase data  
 -->  
 <!--<UserService type="SitecoreDemandbase.Data.SessionUserData, SitecoreDemandbase">  
 <Timeout>500</Timeout>  
 </UserService>-->  
 <!--   
 Uncomment following for Xdb storage of demandbase data  
 -->  
 <UserService type="SitecoreDemandbase.Data.XdbUserData, SitecoreDemandbase">  
 <Timeout>500</Timeout>  
 </UserService>  
 <!--  
 You can remove the following two XML nodes to manually manage Demandbase Rules. They are located here /sitecore/system/Settings/Rules/Definitions/Demandbase  
 -->  
 <!-- this is where the Demandbase attributes are defined, omitted for space concerns -->  
 </processor>  
 </initialize>  
 </pipelines>  
 <commands>  
 <command name="demandbase:mockip" type="SitecoreDemandbase.Commands.MockIp, SitecoreDemandbase" />  
 </commands>  
 </sitecore>  
</configuration>

## Misc. Settings

Params \* RestApi - The version of the rest api from Demandbase being used to gather data on the user. \* key - The Demandbase api access key **Very important to set this up right away**. \* demandbaseIp - The IP address to Demandbase, this is utilized to dynamically determine account settings.

## Utilizing XDB (default)

There are two optional configuration nodes that are enabled by default to utilize XDB as a storage medium for Demandbase data. \* Uncomment Model node at the top of the file. \* Comment out UserService node of type SitecoreDemandbase.Data.SessionUserData \* Uncomment UserService node of type SitecoreDemandbase.Data.XdbUserData.

## Utilizing SessionUserData

To enable Session for environments without XDB \* Comment out the Model node at the top of file. \* Comment out UserService node with the type SitecoreDemandbase.Data.XdbUserData. \* Uncomment out UserService node with the type SitecoreDemandbase.Data.SessionUserData.

# Demandbase Data in sitecore

This document will go over the details of how Demandbase data is integrated into Sitecore as well as how you can have access to this data as part of a users session.

## Demandbase context

Installing the module comes with a context singleton object.

DemandbaseContext.Attributes; //A list of demandbase attributes   
 DemandbaseContext.AccountWatch; //A list of watch list attributes  
 DemandbaseContext.DemandbaseIp; //IP address for demandbase  
 DemandbaseContext.Key; //Demandbase access key  
 DemandbaseContext.Levels; //List of headquarter heirarchy levels  
 DemandbaseContext.NoHq; //Boolean for if no hierarchy levels exist  
 DemandbaseContext.RestApi; //Rest api to get user attributes  
  
 DemandbaseContext.User.GetFullObject(); //Get a dynamic object representing the entirety of the Demandbase payload  
 DemandbaseContext.User.GetValue<T>(AttributeId); //Get the value of a particular demandbase attribute without a headquarter heirarchy  
 DemandbaseContext.User.GetSecondTeirValue<T>(HqLevel, AttributeId); //Get the value of a particular demandbase attribute with a headquarter heirarchy

This singleton can be utilized anywhere inside the context of a web request.

## Attribute definitions

The definition for the available Demandbase attributes are set up in the Demandbase.config file.

<attribute customizable="0">  
 <type>string</type>  
 <name>Audience</name>  
 <id>audience</id>  
 <values>  
 <value>Enterprise Business</value>  
 <value>Mid-Market Business</value>  
 <value>SMB</value>  
 <value>Government</value>  
 <value>Education</value>  
 <value>Hospitality</value>  
 <value>Residential</value>  
 <value>Wireless</value>  
 <value>Bot</value>  
 <value>SOHO</value>  
 <value>Obscured</value>  
 </values>  
 </attribute>

Using the attribute definitions defined within this configuration file the system automatically generates rules for Sitecore personalization. If there are values in the configuration file that you would never care about, you can feel free to remove them to clean up the authoring experience.

Node definitions: \* values tag - Set of default values, creates a picklist for authors to choose particular values when setting up rules. \* id tag - Attribute id from Demandbase. \* name tag - User friendly name for the Demandbase attribute. \* type tag - Data type of attribute value *int, string, or bool*. \* Customizable attribute - If this is set to 1 the attribute will be included in the free form value entry rules where you define a particular string value for an attribute. If it is set to 0 and the attribute has default values, it will be excluded from the free form rule.

# Configuring Analytics

Sitecore out of the box does not provide an easy way to track metrics on personalization value.  
Part of this module is a system to generate a metatag on a page that's dynamically generated to match the personalization context. This metatag is then usable by any tag management tool such as Google Tag Manager to track value of visits based on the personalization state.

## Personalization Context

A static class provides a method of outputting a dynamic metatag to represent a state of personalization. Here is an example usage in a cshtml context.

@Html.Raw(PersonalizationContext.GenerateMetaTags())

Which will then generate a meta tag like the following

<meta name="sitecorePersonalization" content="B2B,High Traffic">

In the above tag, this is a representation of a page that has two personalization blocks that evaluated to true, "B2B" and "High Traffic". These names are assigned when the user is setting up a personalization state for a component.

For example, to set up the values in this meta tag make sure you note the process for setting up component personalization \* Choose a component to personalize \* Open personalization form \* Add a personalization state \* Give this state a name **this is the field that ends up in the meta tag** \* Add rules

**Note that this meta tag isn't unique to Demandbase rules, but applies to all personalization rules in sitecore**

# Injecting Demandbase attributes into rich text

This Step is optional, it allows you to inject whatever Demandbase attributes are associated with the current user into rich text. Through integration with another 3rd party module Token Manager

## Token Manager

The Sitecore [Token manager](https://github.com/JeffDarchuk/SCTokenManager) is a Sitecore Module that allows content authors to inject dynamic or managed content into a rich text field. This is particularly useful for Demandbase as it allows us to inject Demandbase data into our content.

## Installing Token Manager using Sitecore Package

Token Manager is available to install via Sitecore package available with Demandabase, that you can Install following these steps: Using preferably a non-production environment, install the module package using Sitecore's Development Tools > Installation Wizard. \* Upload package to sitecore \* Run installation procedure for

**Note: For Sitecore 9.0 install this way.**

## Installing Token Manager using Nuget

Token Manager is available to install via [nuget](https://www.nuget.org/packages/TokenManager) using visual studio or the package manager console command: Install-Package Token Manager. Install into your web solution and make sure the dll is in the bin folder and configuration file (tokens.config) is under App\_Config/Include.

**Note: For Sitecore 9.0 install using the package provided with Demandbase following above steps.**

## Integrating with Demandbase

There is nothing to do here, Token manager will detect the token definitions which are contained within the Demandbase module and wire up the tokens for use automatically. So simply install Token Manager and you're done.

## Injecting Content

After installing there will be a new button in the rich text editor that looks like an orange lightning bold. If you click this button you will be shown a form to fill out for which attribute you would like to inject onto the page. During edit mode this will show up as a blue highlighted placeholder reporting what attribute will be returned. Once outside of edit mode it will be resolved into a value for the attribute based on the user viewing the page.

# Utilizing personalization through a CDN

## Understanding CDNs

What it boils down to is a CDN caches the HTML output and whenever anyone requests a page it simply returns a flat HTML version of your page.

This is amazingly powerful when it comes to getting fast response times, however it's almost completely incompatible with any form of dynamic content with a few exceptions.

## Pass through

CDNs will give the ability to white list pages that will always retrieve content from the source, in our case Sitecore. This requires foresight from content authors as well as cooperation from the team managing the CDN to white list any pages that are going to be personalized before the personalization is live.

It's very important that you make sure the white list of a page is intact before setting up personalization, otherwise when the CDN refreshes the page cache for a particular user it faces the risk of caching personalized content. At which point you could be delivering content for a specific target audience to everyone.

## JavaScript

While significantly more complex, you can utilize JavaScript to dynamically load personalized content.

Sitecore doesn't provide a solution for this out of the box, but if needed the steps to accomplish this would be.

\* Create an endpoint for personalization

\* white list this endpoint to pass through the CDN

\* Patch into Sitecore's conditional rendering engine to place an identifier instead of resolving the personalized placeholder

\* On page load detect these identifiers and send them to your endpoint including the page guid and placeholder attempting to render

\* Your endpoint will take the page id, as well as placeholder and return personalized html

\* Your JavaScript will then replace the identifiers with the HTML returned from the endpoint

## Recommendation

due to the simplicity, we would recommend the pass through of personalized pages approach.

# Testing content

There are a few things to keep in mind while testing content that is personalized by Demandbase attributes

## Testing attribute-based personalization

There is a widget included in preview mode to simulate a particular ip. With this we can render the page as it would be seen from a particular consumers ip.

This works by setting a cookie in your browser, once you have the cookie set you will impersonate that ip until you erase it from the widget or clear your cookies.

However an important note is that in order to test personalization you need to be in normal rendering mode.

You can enter normal mode by changing the sc\_mode query string parameter to sc\_mode=normal.

# More info

You can check Demandbase Sitecore installation and usage at:

<https://support.demandbase.com/hc/en-us/articles/218313803-Demandbase-Sitecore-Installation-and-Usage>