



Git Backup Extension
User Guide
Version 2.0.0



Software Change Log	2
Introduction and Installation	2
About the Git Backup Extension	3
Installing the Git Backup Extension	5
Configuration and Usage	6
Configuration	6
Usage	11
Known Limitations	21
Compatibility	22
Document Revision History.....	22

Software Change Log

Version	Release Date	Changes	Contributors
1.0	20/12/2017	Initial Release	Vladimir Rosu Pierre Tessier Dumitru Zanfir
1.1	27/12/2017	Added support for Git Branches and Checkout	Vladimir Rosu
1.2	25/01/2018	UI restyling and UX improvements; Added auto AppKey Creation for Extension Import; Added capability to Export to Source Control entities a project from the Home Mashup; Updated ExportExtensions Extension to version 1.0.19	Gabriel Bucur Vladimir Rosu Pierre Tessier
1.2.1	01/02/2018	Fixed Extension Import bug	Vladimir Rosu
1.3.0	26/04/2018	Added new functionality: Git Status page shows info equivalent to “git status” and formatted diffs per file. Added Utility service: RemoveRemoveEntityHistoryInfo Added DiffViewer widget which pretty prints the file diff.	Vladimir Rosu Moritz von Hasselbach
1.3.1	04/05/2018	Fixed Extension Import bug for ThingWorx 8.2.1 (“universal” attribute is not allowed to appear in element Entities)	Vladimir Rosu
2.0.0	11/01/2019	Major UI restyling and UX improvements. The extension has now a single page that offers access to all the functionality of the extension. Added support for querying and selecting the Bitbucket repositories that a user has access. Updated the ExtensionExportExtension.	Gabriel Bucur Bogdan Mihaiciuc Moritz von Hasselbach Pierre Tessier

Note: Version 2.0.x is a major version and requires removing any existing 1.x version (including existing GitBackup Things)

Introduction and Installation

Extensibility is a core aspect of the architecture and design of ThingWorx. Partners, third parties, and general ThingWorx users can easily add new functionality into the system, seamlessly. Extensions can be in the form of Service (function/method) Libraries, Connector Templates, Widgets, and more. This document provides installation and usage instructions for the Git Backup Extension.

About the Git Backup Extension

Git Backup Extension allows you to backup (push) and/or retrieve (pull) all artifacts related to a ThingWorx application to/from a Git repository. “All artifacts” is defined as Entities (Things, DataTables, etc), Data (the actual rows from a DataTable/Stream/ValueStream) and Extensions (Zip files that contain Widgets/Java-based functionality and more).

The main purpose of the extension is to allow easy replication of ThingWorx artifacts from one ThingWorx instance to another through a Git repository. Another use case is allowing easy access to these artifacts for build systems like Jenkins.

It has been designed to also allow importing that application into a ThingWorx server from the provided Mashups, providing an easier process than the out of the box import system.

This extension utilizes the jGit API. For more information, visit <https://www.eclipse.org/jgit/>

Note 1: The Git Backup Extension uses functionality from the ExportPlatformExt Extension for Importing and Exporting Extensions to the snapshot. The Git Backup Extension can work without this extension installed in the system, but you will not be able to Export/Import Extensions. For ease of use this Extension is included in this package. Documentation for that Extension is not available in this document.

Note 2: We include for convenience a File Repository called GitRepository. You can use this for storing your projects, or you can use any other File Repository of your choosing.

Note 3: We include a `GitBackup.Main.Mashup` that offers a User Interface for interaction with the back-end services. This mashup allows consumption of the services for non-scripted tasks. For automated tasks please use directly in the script the services below.

Note 4: The Extension contains a pack of 5 extensions.

Extension name	Version	Description
DiffViewer	1.0.6	Provides pretty-print of diff output
Autocomplete	1.0.31	Provides the Autocomplete widget
InfotableSelector_Extension	2.0.0	Provides various infotable related services
ExtensionExportExt	1.0.21	Provides the ability to export extensions
GitBackup Extension	2.0.0	The core GitBackup Extension

You may already use versions of some of these extensions in your ThingWorx instance. Ideally you should use the versions embedded in this package, but if you cannot, unpack the GitBackupExtensionPack.zip file, remove the conflicting extension, zip the remaining extensions again and try the import procedure one more time.

The Git Backup Extension offers the capability to create a Git Backup Thing in ThingWorx. The thing houses the configuration information to the Git Backup Thing instance and provides the following services:

Main services:

1. **Push (Message):** This service adds all the modified, removes all the deleted files, creates a new commit with a specific message and pushes this to the remote. The current working directory is

the Repo path folder from the Repository selected in the Configuration tab. This method will also initialize the Git repository if there is none.

2. **Pull (Force):** This service will execute a Pull from the Remote. It will create a local image of the Remote repository. Setting to True the Force parameter will result in a Reset and Pull from the Remote.
3. **GetBranchList():** Returns the list of current branches that belong to a repository.
4. **GetCommitList():** returns the list of commits specific to the current branch OR, if you're in detached head mode, to the initial branch configured in the Configuration section (typically master)
5. **GetCurrentBranch():** returns an infotable with one row and 2 columns: Branch Name or Commit ID (String) and IsDetachedHead(boolean). If you're on the current branch you will see the name of the branch and false

Current Branch Name (Or Commit)	Is HEAD Detached?
master	false

Or the Commit ID and true

Current Branch Name (Or Commit)	Is HEAD Detached?
01f56efe65fe1dc790d3013b80996400664b7c98	true

6. **Checkout(BranchNameOrCommit):** checks out a specific branch or commit. If you're going to a specific commit, then the GetCurrentBranch will also report a detached head and Push will fail.
7. **DeleteLocalBranch():** deletes a local branch that can result from deleting/merging a remote branch.
8. **Status():** retrieves a list similar to the output of the command "git status".

File	Status
StyleDefinitions/DefaultChartStyle11.xml	Modified
ExtensionPackages/StatePanelWidget-extension.xml	Modified
StyleDefinitions/PTC.Core.ChartStyle.xml	Modified
Things/PTC.SCA.Common.AlertNotification.NotificationDeliveryConfiguration.xml	Modified
StyleDefinitions/PTC.Factory.Status.Warning.xml	Modified

9. **GetDiffPerFile(File):** retrieves a diff string representing the diffs for a specific file. The input parameter is in the format delivered by the Status() command.

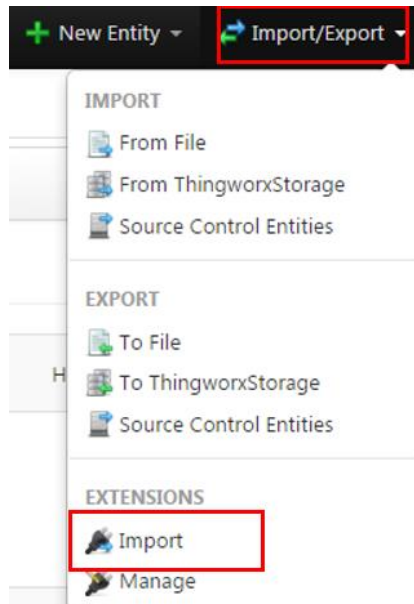
Helper services:

1. **DeleteLocalRepoContent:** This service deletes the specific local folder corresponding to this Repo. It has the same functionality as the DeleteFolder service from the File Repository, but it is added here in order to bypass the need to switch to the Repository Thing and call that function.
2. **GetConfigurationTableValue:** This service returns a value of a configuration table parameter for mashup use.
3. **GetFilteredDirectoryListing:** This service returns the directory structure from the folder specified in the Configuration tab.
4. **RemoveEntityHistoryInfo:** this service removes the ConfigurationChanges and the lastModifiedDate from each of the XML files exported by ThingWorx. It is not currently used in

the UI, but is useful for automation in older versions of ThingWorx when the system modifies the lastModificationDate when you save an entity, but you changed nothing.

Installing the Git Backup Extension

1. From a web browser, launch ThingWorx.
2. Log into ThingWorx as an administrator.
3. Go to **Import/Export > Import**.



4. Click Choose File and select GitBackupExtensionPack.zip
5. Click **Import**.

Note: If an **Import Successful** message does not display, contact your ThingWorx System Administrator.

6. Click **Yes** to refresh Composer after importing the final extension.

Note:

Import Extensions

Choose File GitBackupExtensionPack.zip

Close Validate **Import**

Refresh Composer?

Extensions often include widgets. You will only be able to see the widgets if you refresh Composer. Refresh now?

Yes No

7. Confirm that the Extension has been imported properly. Check the Application Log for potential problems.

Configuration and Usage

Usage of the Git Backup Extension requires creation and configuration of a Git Thing based on the GitBackupTemplate in ThingWorx. Starting version 2.0.0, the preferred option to do all the operations is via the newly provided UI, accessible via the [GitBackup.Main.Mashup](#). The previous system that used the Home Mashup for the GitBackup Things is no longer used.

Configuration

1. Access the GitBackup.Main.Mashup

Example URL:

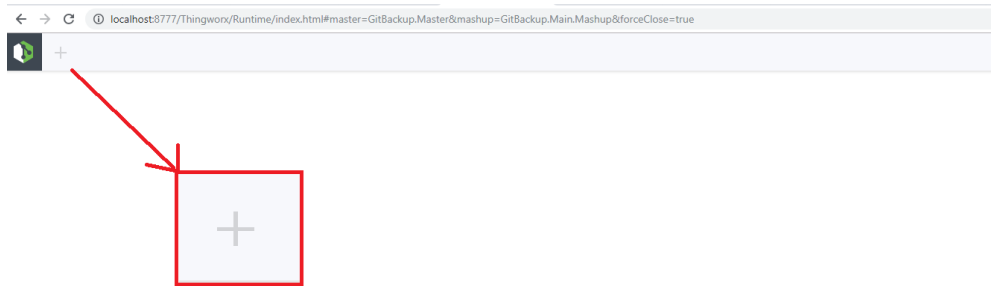
<https://localhost:8777/Thingworx/Runtime/index.html#master=GitBackup.Master&mashup=GitBackup.Main.Mashup>

[replace the localhost and port with your respective parameters]

In the following screens you will need to set several configuration parameters. They are described below.

Configuration Field Name	Type	Description
User	STRING	The Git repository username.
Password	PASSWORD	The Git repository password.
Commit Username	STRING	The Git username that will be used for commit purposes.
Commit Email	STRING	The Git email that will be used for commit purposes.
Git Repo URL	STRING	Git Repository URL Example URLs: For Bitbucket Online: https://bitbucket.org/vrosu/integritytest.git or https://vrosu@bitbucket.org/vrosu/testrepo.git For Other Git Repo types: HelloWorld/_git/HelloWorld">https://vrosu@dev.azure.com/vrosu/HelloWorld/_git/HelloWorld http://roicentersvn.ptcnet.ptc.com/vrosu/GitBackupExtension.git
File Repository	STRING	Selected File Repository where you will store the selected Git repository. For convenience the Extension already provides a GitRepository that you can use without creating a new File Repository. You can use the same FileRepository for multiple GitBackup Things, you just need to modify the File Repository Path.
File Repository Path	STRING	The path from within the File Repository where your repository will be created and stored.
Initial Branch	STRING	The branch that will be used to get the commit list if you're on detached head.

2. Click on the Plus button



No Git Backup Thing defined. Use the "+" button to define a new repository.

3. The New Repo window appears. This is a wizard-type process with 2 screens. There are 2 options available in this wizard: BitBucket or Other. If you choose Bitbucket, you will benefit from **user/password verification** and **automatic Git URL completion** during the add process.

A screenshot of a 'New Repo' wizard form. At the top, it says 'New Repo'. Below that is a progress indicator with two steps: '1 Login' and '2 Repo settings'. The '1 Login' step is currently active. The form contains the following fields and options:

- 'Git thing name' with a text input field and a help icon (?)
- 'Git server' with two radio button options: 'BitBucket' (selected) and 'Other'
- 'Git username' with a text input field
- 'Git account password' with a text input field

At the bottom right of the form, there are two buttons: 'Cancel' and 'Next'.

3.1. **The Bitbucket option (default).** Complete the Git Thing Name (it can be any valid Thing Name in the platform), Git User and Git Password. Pressing the Enter in the Git Account Password will allow you to go to the next screen.

New Repo - TestGitThing

1 2
Login **Repo settings**

Git thing name
 ?

Git server
 BitBucket Other

Git username

Git account password

Note: In case the Bitbucket user and password are invalid, the system will display the following message:

Git username

Git account password

Username or password is invalid. Try again! Continue anyway

You have a choice to continue, or to provide the correct information. This check is executed only in case of an online BitBucket repository.

You will arrive in the second wizard screen. Fill in the required details: Committer Name, Commit e-mail, GitRepo URL, File Repository Path and the Initial Branch. When filling the GitRepo URL you will see search results as you type, based on the BitBucket repositories you have access to:

New Repo - TestGitThing

✓ Login 2 Repo settings

Committer Name

Commit e-mail

Git repo URL

File repository

File repository path

Initial branch

Git repo URL

- https://vrosu@bitbucket.org/vrosu/testrepo.git
- https://vrosu@bitbucket.org/vrosu/test5.git
- https://vrosu@bitbucket.org/thingworxtechsales/autowaretest.git
- https://vrosu@bitbucket.org/vrosu/thingworxtest.git
- https://vrosu@bitbucket.org/rotwx/android-ramp-up-test.git
- https://vrosu@bitbucket.org/vrosu/integritytest.git

Click Add.

Click with the mouse or select with the keyboard the needed repository

3.2. **The Other Option.** Complete the Git Thing Name (it can be any valid Thing Name in the platform), Git User and Git Password. Pressing the Enter in the Git Account Password will allow you to go to the next screen.

New Repo - AzureGitRepo

1 2
Login Repo settings

Git thing name
 ?

Git server
 BitBucket Other

Git username

Git account password

You will arrive in the second wizard screen. Fill in the required details: Committer Name, Commit e-mail, GitRepo URL, File Repository Path and the Initial Branch:

New Repo - AzureGitRepo

✓ 2
Login Repo settings

Committer Name

Commit e-mail

Git repo URL

File repository

File repository path

Initial branch

Click Add.

Note: This functionality was tested with the Azure DevOps Git repository, using alternate credentials.

Note that in this variant there is no user/password check or autocomplete available for the Git URL.

Usage

The description of the services is presented in the About section.

In this section we will present 3 usecases:

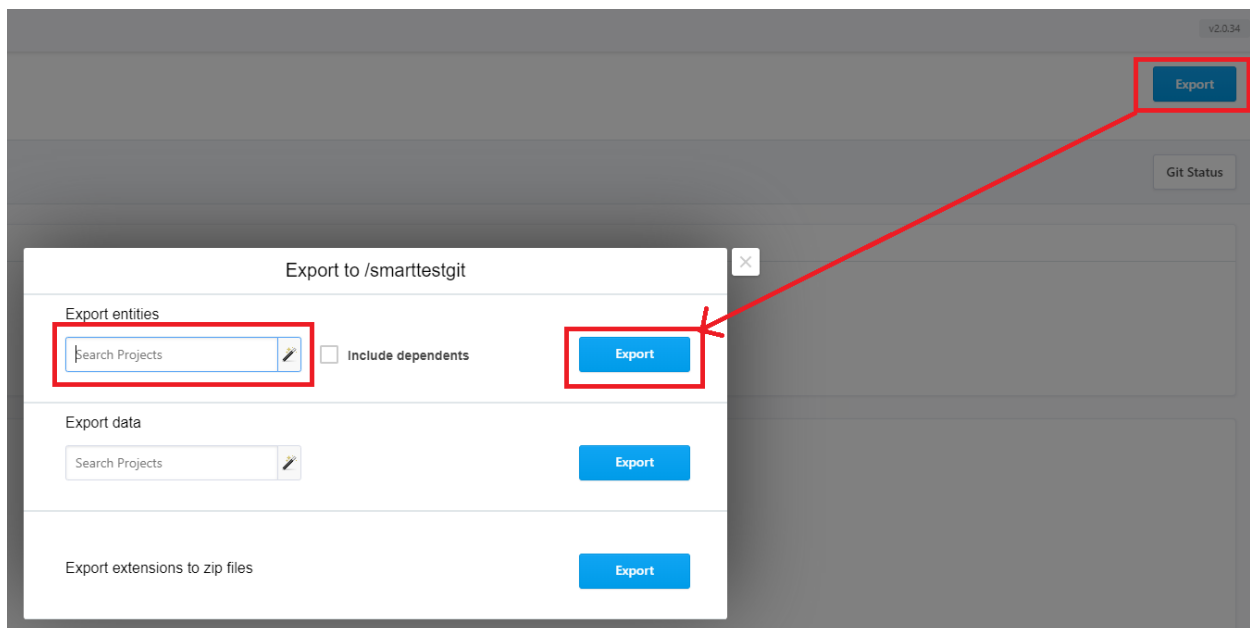
Usecase 1: Working on a new ThingWorx application which was not previously stored in a Git repository

This usecase assumes that you started development of a new ThingWorx app that you would like to store in Git.

Prerequisites: create a new Git repository in a system of your choosing (eg: Bitbucket). Create a new GitBackup Thing for this project using the [GitBackup.Main.Mashup](#) and configure it as per the Configuration section above.

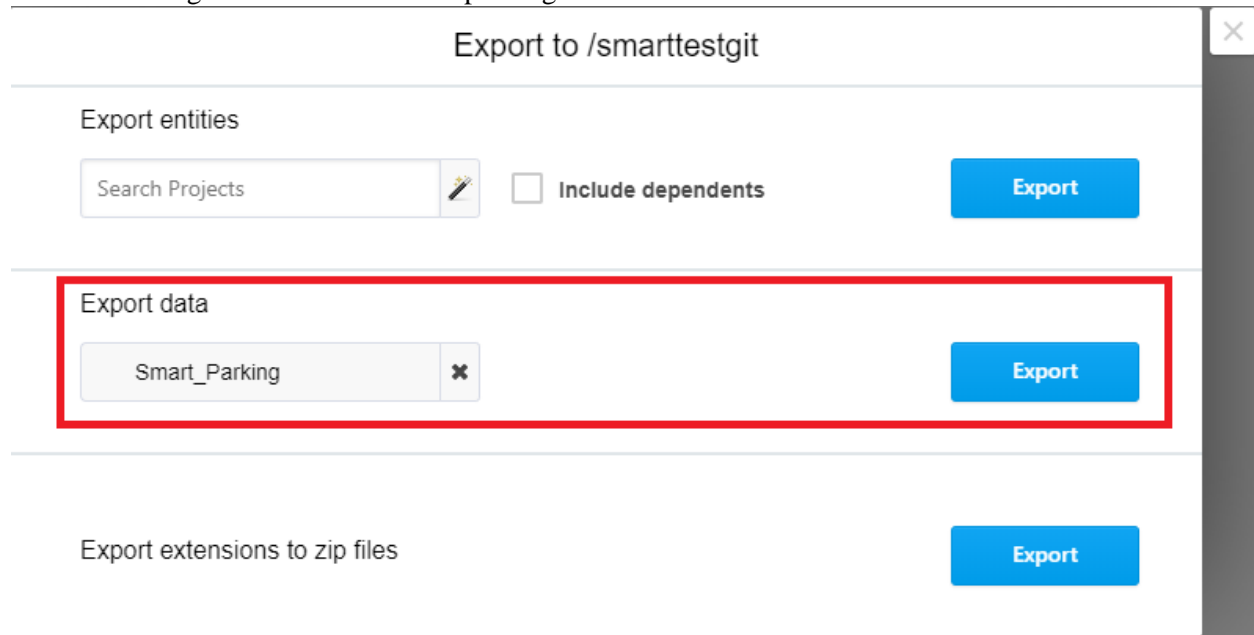
Step 1: Export to Source Control Entities your ThingWorx application. This functionality is embedded in the [GitBackup.Main.Mashup](#). Click on the Export button and then on the Export button from the Entities section.

This process will export all the Entities in the folder from the repository that were configured for this GitBackup Thing.



Step 2(optional): Export to File Data. This step is optional, meaning that if your solution does not use DataTables/Streams/Blogs/Wikis/ValueStreams you don't need to perform this operation. This functionality is embedded in the [GitBackup.Main.Mashup](#). Click on the Export button from the Entities Export data section.

This will export all the data that belongs to the Data type entities above in a folder from the repository that were configured for this GitBackup Thing.



Step3 (optional): Export Extensions. This functionality depends on the ExportPlatformExt_Extension. It is optional.

This will export all the extensions that this server contains in a specific folder called "Extensions" in the folder from the repository that was configured for this GitBackup Thing.

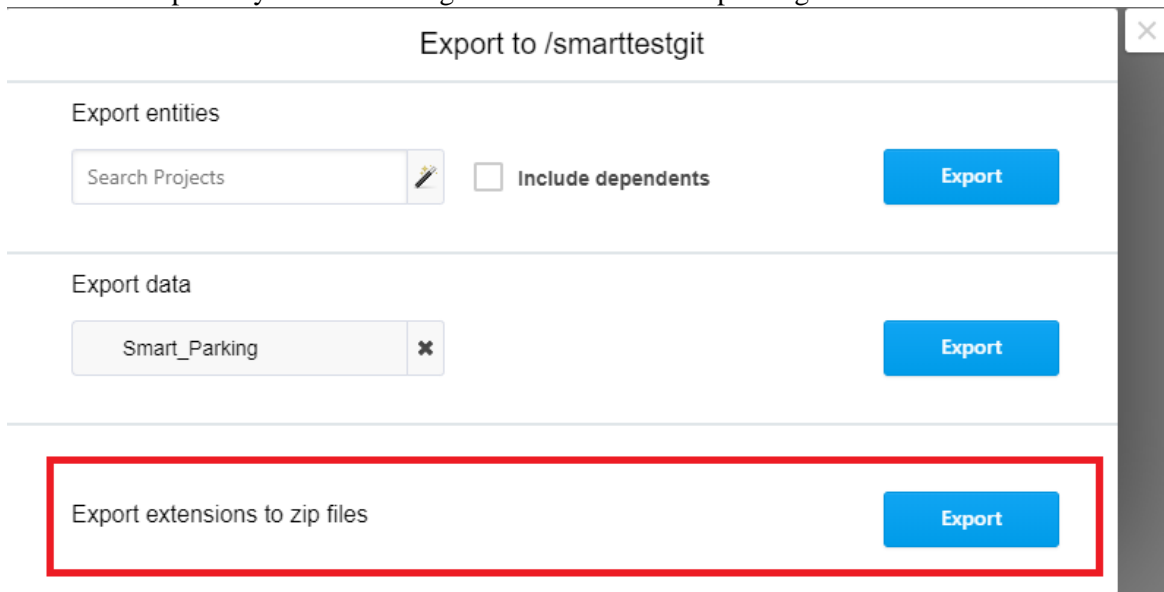
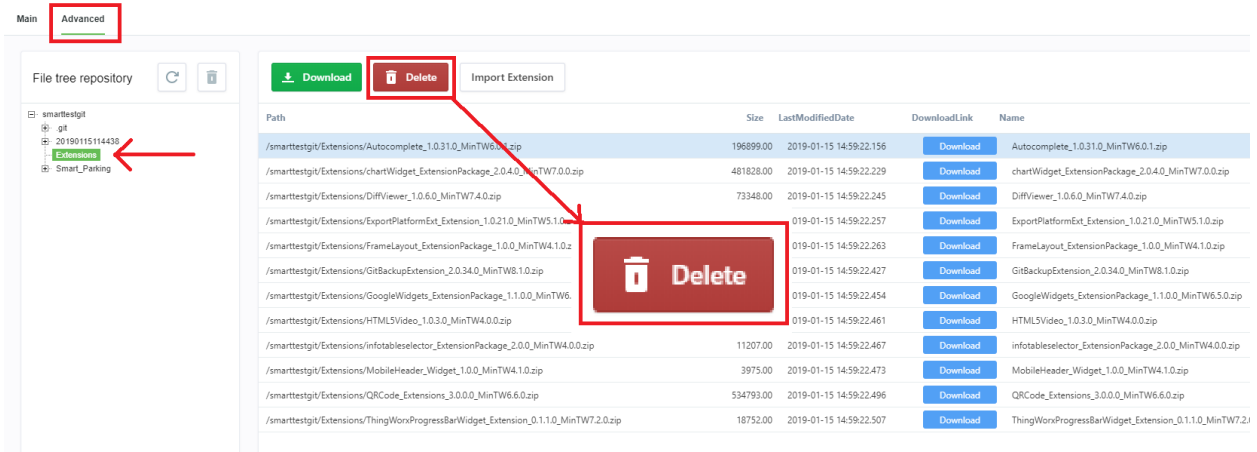


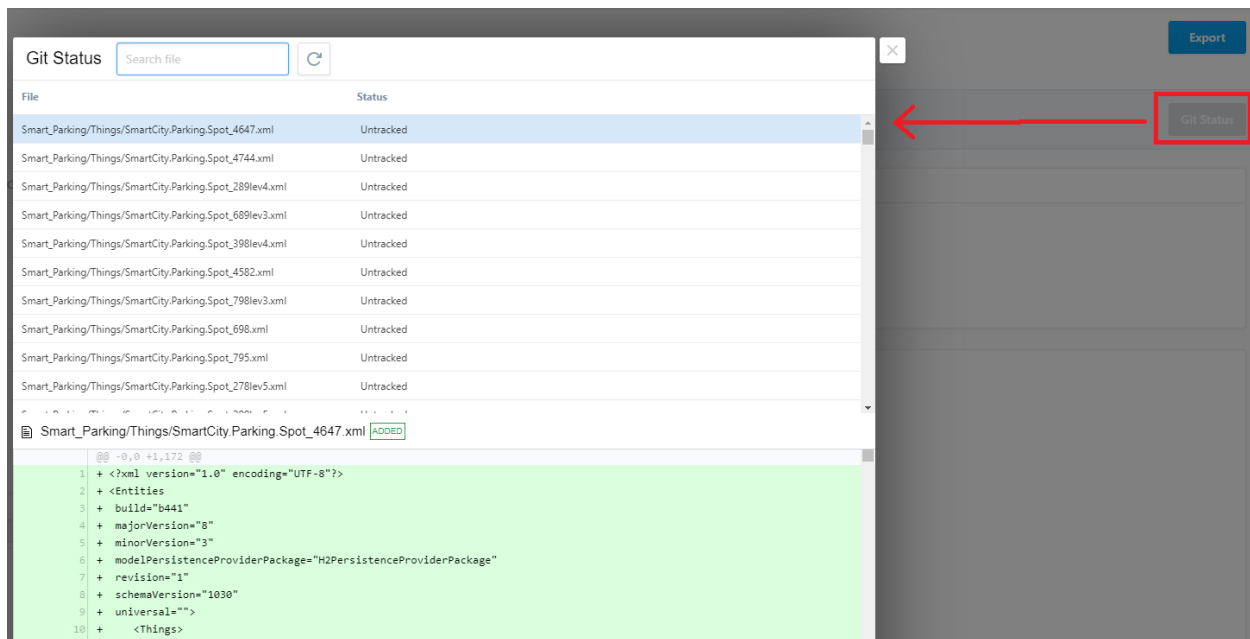
Fig 2. Exporting Extensions by using the Home Mashup

Step 4 (optional): Cleanup your Extensions. The Extension export process at Step 3 dumps all the Extensions from the system. If your application only requires a small subset of that it is recommended to go to the Extensions folder and remove the Extensions which are not needed. Use the Advanced tab from the GitBackup Main Mashup.

gittest-smartparking.git

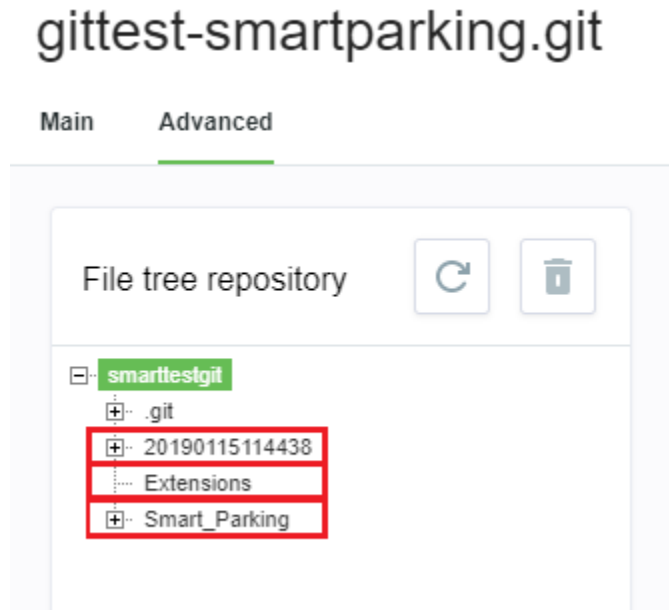


Step 5: Check the Git status (optional). Press the Git Status button in order to understand what you are pushing, what files changed, and to see the file diffs. The output of this screen is similar to the “git status” command. You can use it from the Main tab of the GitBackup Main Mashup.



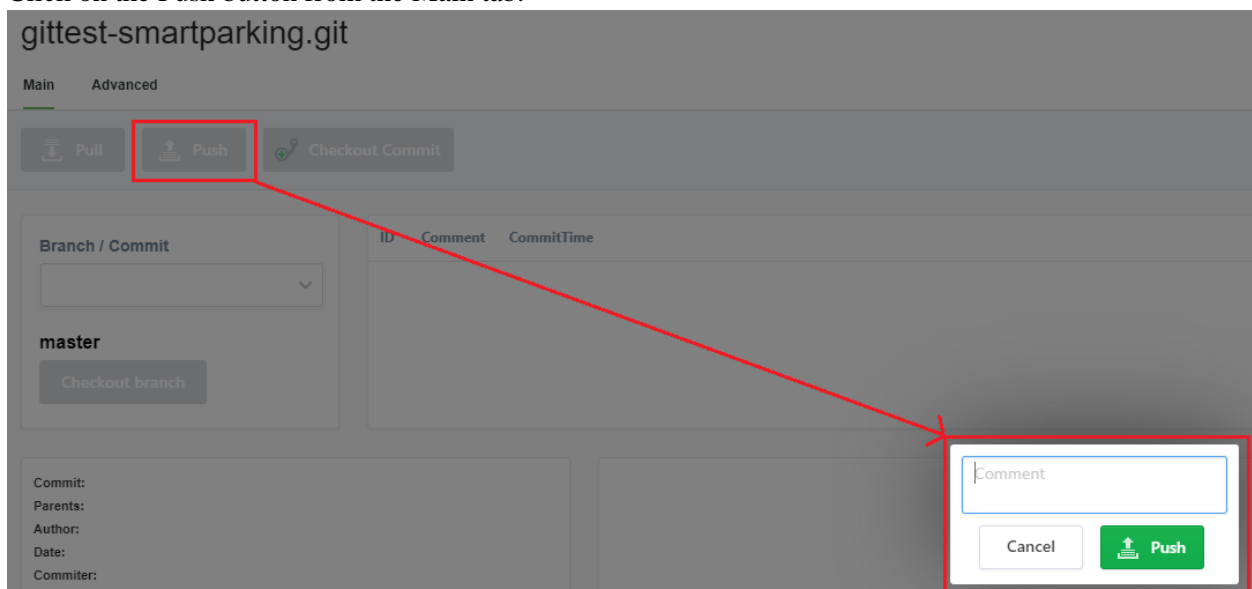
Step 6: Push your project to the Git remote. This will push all the contents of the folder selected in the configuration section of the GitBackup Thing to the remote git repository.

Checkpoint: If you created an application that uses entities, data and custom extensions, you should see three folders in the Advanced tab: the first is named like the DateTime and it contains the data, then a folder named Extensions and a third one named like the Project that you exported, containing the entities. You might not have the DateTime or the Extensions folder, if you did not export them, but the Project folder is mandatory.



If you have more than a Date Time folder, delete the oldest one and keep only the most recent one.

Click on the Push button from the Main tab.



Provide a commit message in the popup then click Push. A progress window will appear while the push will be in progress. You can check the push was successful in your git browser (Fig. 5).



Executing Push to Git...

Git Push progress window

ID	Comment	CommitTime
923eaa4e3b9ab569e1d795a3bd578f36c71b6b2a	First Push	2019-01-15 15:01:56

Remote Git repository commit history

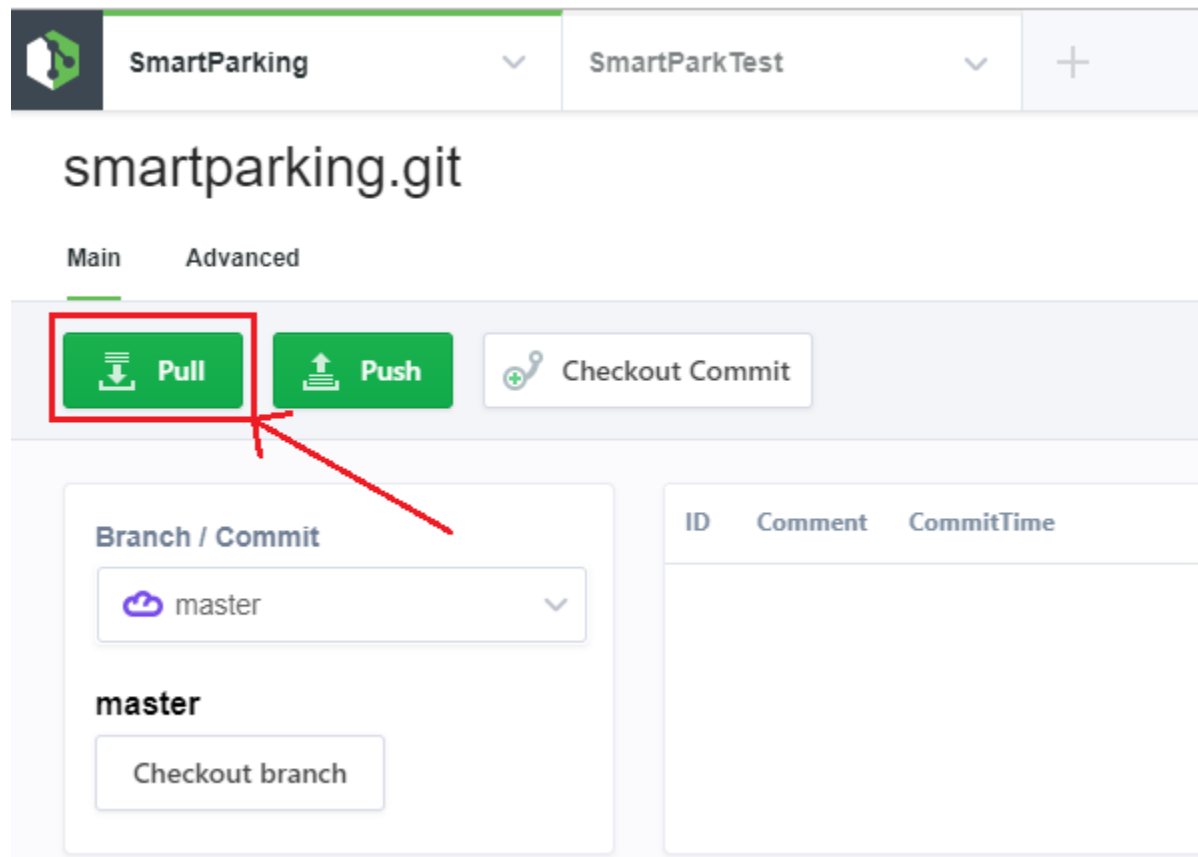
Usecase 2: Working on a ThingWorx application that is already stored in a Git repository

This use case assumes that there is an existing ThingWorx app which is stored in a Git repository that you would like to download locally to work on it.

Prerequisites: have the URL of a Git repository and HTTP access credentials. Create a new GitBackup Thing using the GitBackup.Main.Mashup.

Step 1: Pull the project locally from the repository.

Click the Pull button from the Main tab of the GitBackup.Main.Mashup.

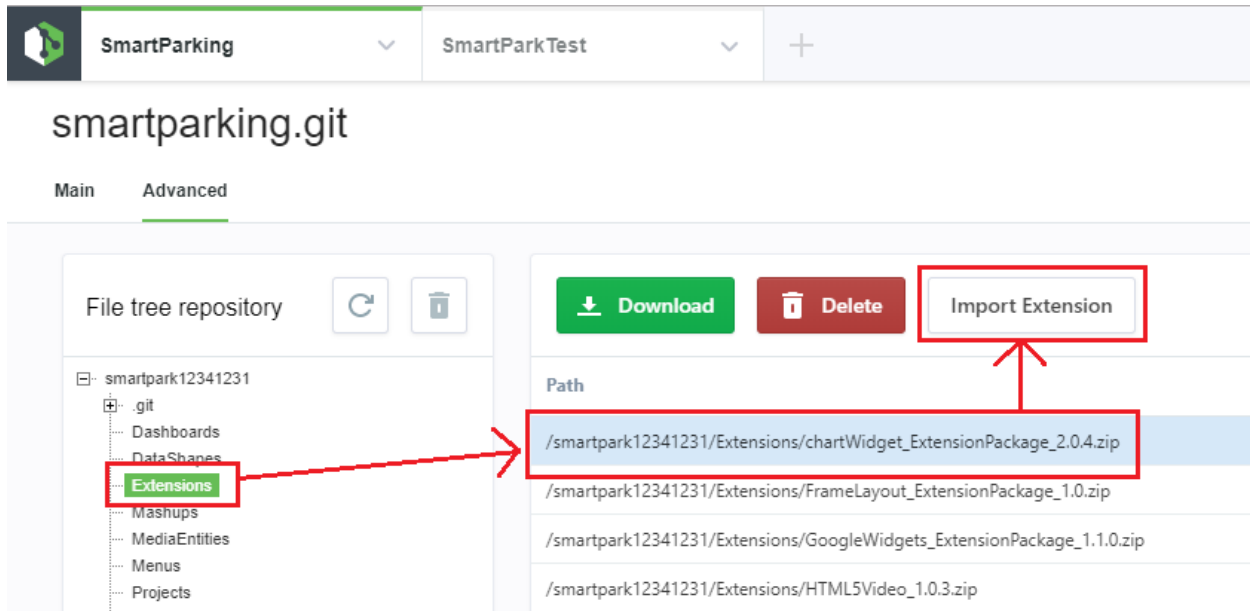


Step 2. Importing Extensions (optional): if your project contains extensions that are not included in your ThingWorx instance, it is **mandatory** to first install the extensions, the first step of a standard import process.

This functionality is offered by the ExportPlatformExt Extension.

This makes it easier to install many individual extensions without leaving the mashup. Error or success messages will be displayed in a grid that is displayed in the lower part of the display.

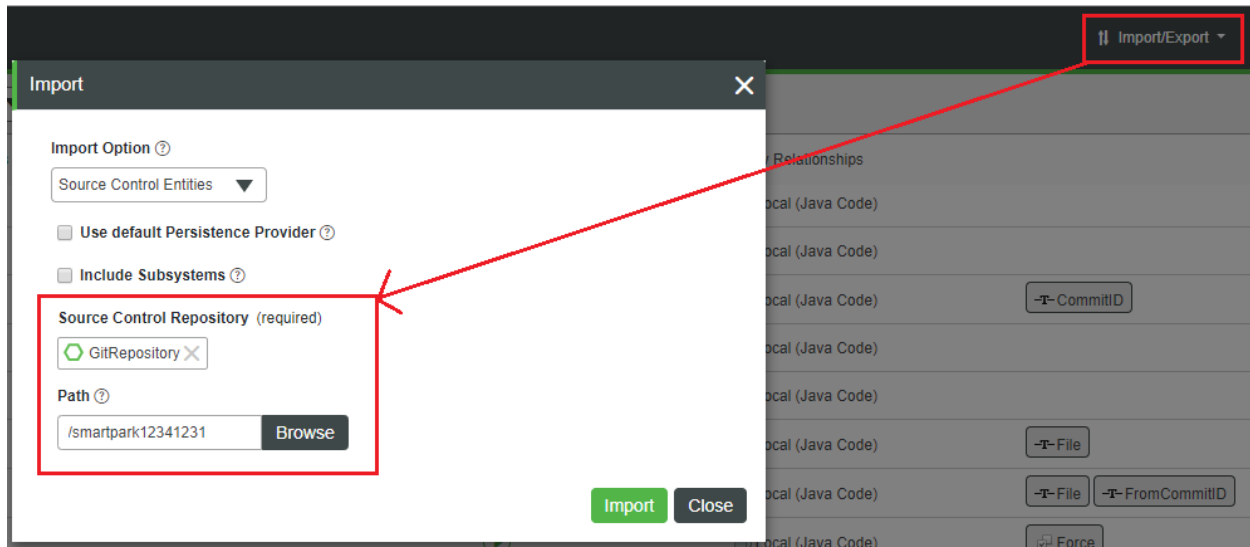
- Select the Extensions folder in the Advanced tab (**mandatory step; the Import functionality will not be visible in the interface without doing this step**)
- Select the Extension
- Click on the “Import Extension” button.
- The result of the import will be visible in a panel in the lower part of the screen.



Installation of Extension via the Home Mashup interface

Step. 3. Importing the Entities

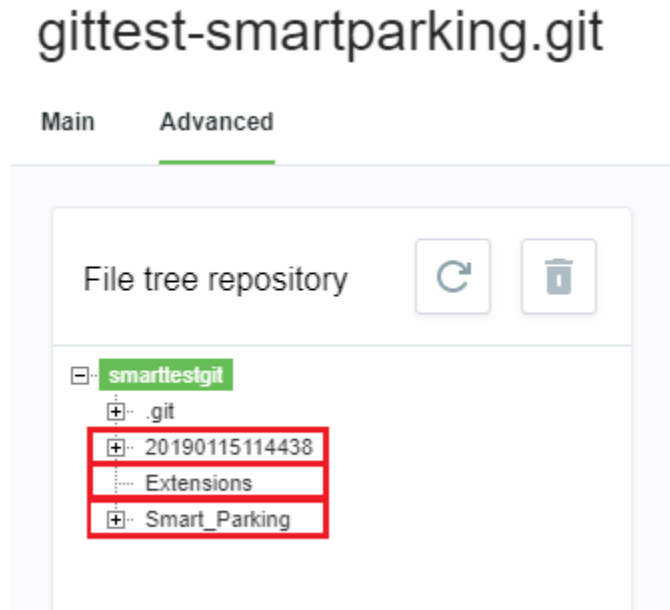
This is achieved through the standard ThingWorx functionality Import / Source Control Entities. Manually select the correct Repository and path where the project was pulled.



Importing project entities

Step 4. Import Data (optional). If the project is using any kind of data which needs to be stored in Streams/ValueStreams/DataTables/Blogs/Wikis you will need to import it. This is only possible if the data was exported and pushed in the first phase.

It is easy to detect if the project contains such an export. Check if in the Advanced tab you see a folder with a name like a DateTime stamp.



Data Export is present

If yes, then proceed to load the data using the standard ThingWorx User Interface. Note: there is a known bug in ThingWorx 8.3 Next Gen Composer (NGC) that does not allow using it to import data from imported Git Repositories.

Import From File

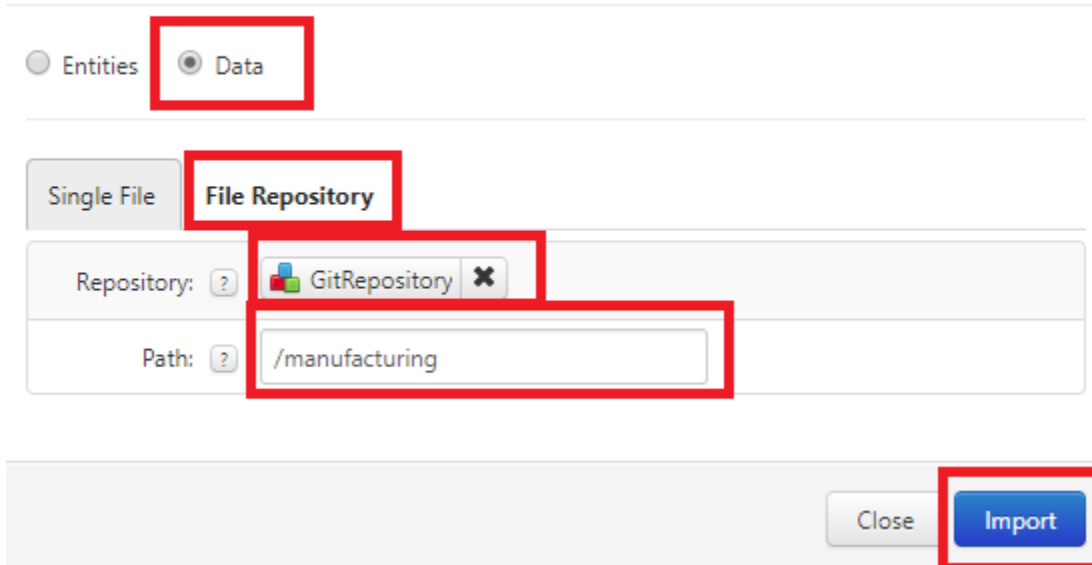
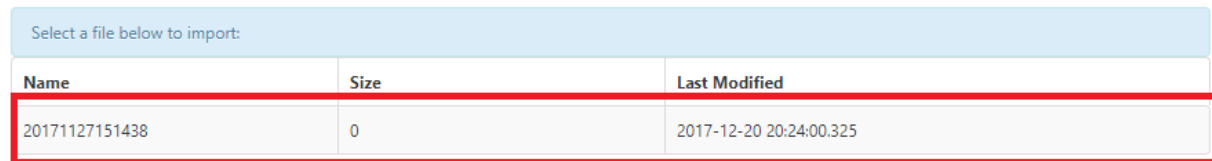


Fig. 12. Import Data using the standard ThingWorx User Interface

Make sure to manually select the proper File Repository and path. If everything was done correctly and there is a Data Export present, you will see the following screen. Click Import to initialize the Data import.

Import from File Repository



Name	Size	Last Modified
20171127151438	0	2017-12-20 20:24:00.325

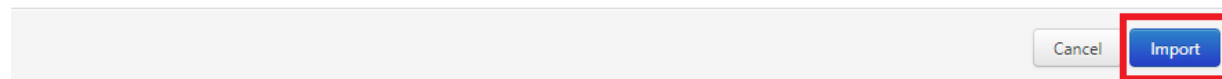


Fig. 13. Import Data

This is the end of the importing part of Usecase 2. After you have provided your own modifications you can just follow the process from Usecase 1 to push your modifications back to the repository.

Usecase 3: Switching branches or commits

This is an advanced usecase and deals with checking out a specific branch or commit ID from the history of commits.

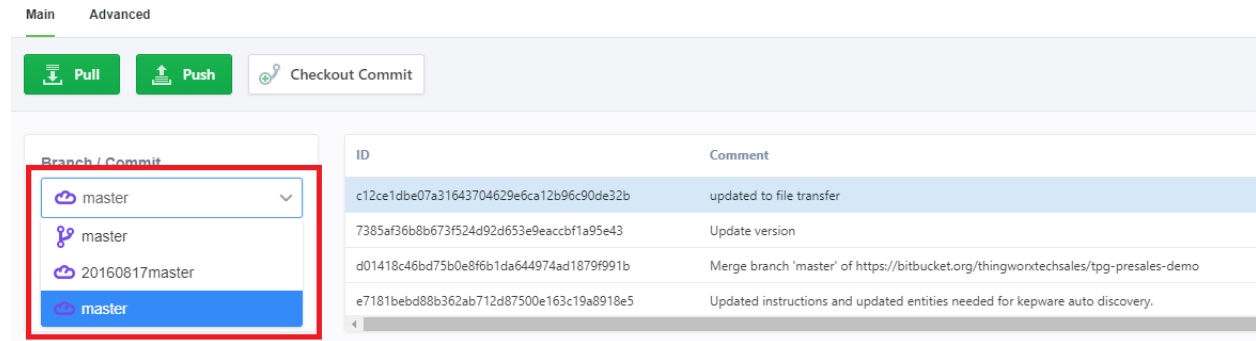
The purpose of the Checkout is to:

- switch to a specific branch to allow importing it and continue pushing to that branch
- switch to a specific commit in the history of that branch so you can import old artifacts (Entities, Data, Extensions). You won't be able to push if you have checked out a specific commit. The Push button is disabled in this case. Calling directly the Push service will result in an error.

Usecase 3.1 Checking out a specific branch

Step 1: Having Usecase 2 performed, select a Branch in the Main tab and press **Checkout Branch**.

It does not matter if you are selecting a remote or local branch. All local branches are remote tracking branches. If you're clicking on a remote branch that does not have a local branch, the extension will auto create a local branch which tracks the remote one.



Select a branch and press Checkout Branch.

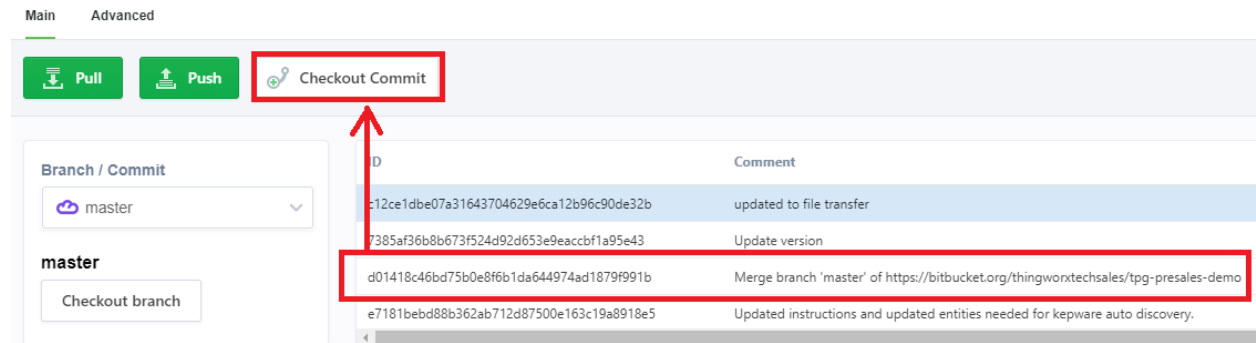
Step 2: You can now use the previous import procedure from Usecase2.

Note: the extension does not create remote branches. You need to have a remote branch already created in your Git repository.

Usecase 3.1 Checking out a specific commit

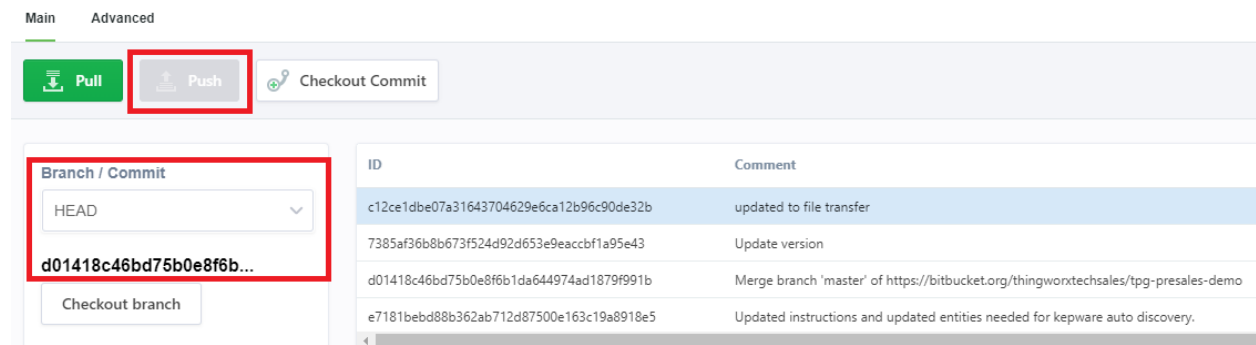
Step 1: Having Usecase 2 performed, in the Main tab select a Commit and press **Checkout Commit**.

The list of commits is the one available for the current branch if you're on branch, or for the initial branch if your HEAD is detached.



The system will display in the left side of the screen the current commit ID:

In this mode, you won't be able to push anything to the branch since your head is detached, and in the interface the Push button is disabled to reflect this.



Step. 2 You can now use the previous import procedure from Usecase2.

Note: whenever you're checking out a commit, you won't be able to push. In the application log will appear a message "nothing to push" characteristic to such situations. You need to checkout a branch to push.

Checkout a branch to be able to push data back in that branch.

Note: importing an application might involve other operations besides importing extensions/entities and data to make it work as it should.

Example of such operations are:

- setting each subsystem's setting
- importing FileRepository contents
- adding additional Thing Shapes to the Session in the UserManagementSubsystem
- setting collection permissions (see the following support articles for more details: <https://www.ptc.com/en/support/article?n=CS199173> and <https://www.ptc.com/en/support/article?n=CS236842>)
- making sure that the UserExtensions ThingShape contains the needed properties. Usually importing it overrides it, but this might be an issue in case of multiple projects.

Known Limitations

The GitBackup Extension only supports Git repositories which use Basic Authentication (User and Password). No SSH support is planned, but source code is available for modifications.
For any Git commands other than the provided Push, Pull and Checkout you must use locally the Git client. This Extension is not intended to provide a full replacement for a Git client.
The extension is designed around exporting Projects. It can be used with manual exports that don't have a Project assigned, but certain steps in this guide might not apply. You should not mix exports of entities based on a Project with exports based on a tag and no project.
Scenarios involving multiple users editing at the same time have not been tested.
If you're trying to Push and the Remote contains a more recent Commit, Push will fail. Please check the Application Log to see the detailed error. You might need to do a Pull, import the changes then Export and Push.
In case there is an error in the Zip Ext process the GitBackup.Main.Mashup will not display any error messages. Please check the Application Log.
The Export to Source Control Entities will fail if in the platform-settings.json you have a relative path defined for your Storage section. Please replace the relative path with an absolute path and restart the ThingWorx server.
The Import extension functionality will fail if the appKey expired. Please check if the AppKey is still valid if the report which appears after clicking on the Import Extension button is empty.
If a remote branch was closed (for example by merging into master), it is required to manually run "git remote prune origin".
The extension does not create a remote branch. You need to have a branch created in your remote.
You can't push new changes if you have checked out a specific commit (meaning if you're on the "Detached HEAD" state). You will receive a "nothing to push" error. When using the GitBackup.Main.Mashup the PUSH button is disabled in such a situation.

Compatibility

This extension was tested for compatibility with the following ThingWorx Platform version(s) and Operating System(s):

ThingWorx Platform Version	ThingWorx 8.3.1, ThingWorx 8.2.3
OS	Windows 10 64 bit

Document Revision History

Revision Date	Version	Description of Change
December 20, 2017	1.0	Initial version
December 27, 2017	1.1	Added documentation for Branch and Checkout related functionality
January 25, 2018	1.2	Updated software changelog and known limitations
February 1 st , 2018	1.2.1	Updated software changelog
April 26 th , 2018	1.3.0	Added documentation for the Git Status functionality. Updated software changelog.
04 May 2018	1.3.1	Updated software changelog
11 January 2019	2.0.0	Updated software changelog, updated guide with v. 2.0.0 UI, added contributor list to the changelog