



Integrations (Direct)

Jenkins — Last update: 31 October 2022

Basis Technologies

Table of Contents

1. Background	1
2. Integration Overview	2
2.1. Functional Solution.....	3
2.2. Technical Solution	6
2.2.1. Outbound Integration	7
2.2.2. Inbound Integration	8
2.2.3. Integration Components	9
3. Integration Setup (SAP)	10
3.1. Remote Function Calls	11
3.2. SAP User	12
3.3. ActiveControl general configuration	13
3.3.1. Custom Fields	14
3.3.2. Import Options	16
3.4. SAP Configuration Tables	17
3.4.1. /BTI/TE_INT_SYST	18
3.4.2. /BTI/TE_INT_CLAS	19
3.4.3. /BTI/TE_INT_MAPP	20
3.4.4. /BTI/TE_INT_PC	21
3.4.5. /BTI/TE_INT_PROC	22
3.4.6. /BTI/TE_TVARV	23
3.5. Programs	24
3.5.1. /BTI/TE_INTEG_TRIGGER	25
3.5.2. /BTI/TE_INTEG_SEND	26
3.6. Scheduled Jobs.....	27
3.7. Error Logging	28
4. Integration Setup (Jenkins).....	29
4.1. Build Pipeline	30
4.2. Jenkins User	31
4.3. URL scripts	32
4.4.	33
5. ActiveControl HTTP API	35
5.1. Get Queue Contents	36
5.2. Lock Queue.....	41
5.3. Unlock Queue	43
5.4. Save Business Task Result.....	45
6. Further Information	48

1. Background

ActiveControl includes an out-of-the-box Integration Framework which enables bi-directional integration capabilities with ITSM products such as JIRA, ServiceNow and HPSM and also more recently with DevOps lifecycle tools such as Gitlab.

During early 2020, this Integration capability has been extended to include an integration with **Jenkins**, another third-party DevOps product used increasingly within IT organisations to automate aspects of software development building, testing and deployment. Basis Technologies believes that this new Jenkins integration will enable SAP customers to further evolve their continuous integration and continuous delivery (CI/CD) capabilities in delivering SAP change as part of an ActiveControl workflow.

The initial use-case for which this Jenkins integration was built by Basis Technologies was to trigger automated testing scripts within tools such as Selenium or Tosca, as part of an ActiveControl workflow, however the reality is that this new integration could potentially be adapted to fulfil numerous other Jenkins-triggered scenarios in the future to help Basis Technologies' customers become more autonomous in their delivery of SAP change.

The rest of this Integration Guide aims to detail the functional and technical solution for the new ActiveControl / Jenkins integration to trigger automated testing.



The Jenkins integration for automated testing is currently not an out-of-the-box 'plug it and play' Integration in the sense of many of the more established ActiveControl integrations such as JIRA and ServiceNow. Setting up this Jenkins Integration in a customer environment will require a combination of ActiveControl, Jenkins and the Automated Testing tool Administrator resources to be involved from the Customer side, in addition to the Basis Technologies team. Some coding will also be required on the customer side, to trigger the automated testing from Jenkins, and then pass back the information to ActiveControl.

2. Integration Overview

2.1. Functional Solution

Example Jenkins integration

The below workflow diagram describes a potential Jenkins Integration scenario as part of an overall ActiveControl workflow.

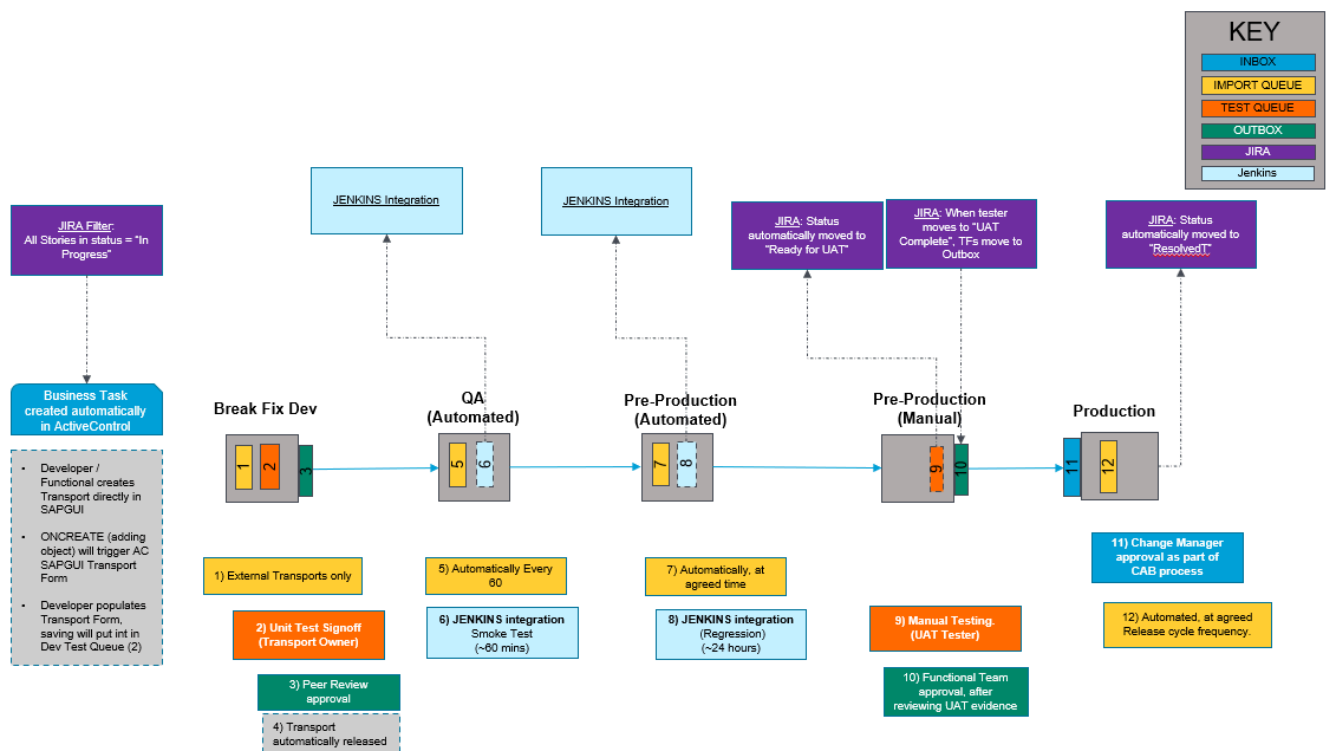


Figure: Jenkins (and JIRA) integration reflected as part of an end-to-end ActiveControl workflow

Key Functional Points of Integration

When transports land in a (configurable) Test Queue of a target within an ActiveControl workflow, the following activities will occur:

1.	Summary
1	Transports will move to the Test Queue of the Target.
2	The Integration will lock the Target Import Queue (unless it is a Virtual Target) so that no subsequent transports are imported.
3	Integration will call the automated test to be performed via Jenkins URL script. (see note (i))

4	Automated Test will run from Jenkins.
5a	<p>If automated tests PASS – then the following will happen:</p> <ul style="list-style-type: none"> i) Integration will add a “Testing Successful” test results entry into the Business Task, with details of the Automated Test that was performed. ii) Integration will approve the Business Task(s), and the underlying Transport Form(s) will move forward from the Test Queue – to the next control point in the workflow. iii) .Integration will unlock the QA Import Queue, so that subsequent transports can be imported into QA again.
5b	<p>If automated tests FAIL, then the following will happen:</p> <ul style="list-style-type: none"> i) Integration will add a “Problem Found” Test Results entry into the Business Task, with details of the Automated Test failure. ii) ActiveControl will send an email notification to the Transport Owner of the failed testing iii) Business Task(s) will remain in the QA Test Queue. iv) Integration will unlock the QA import Queue, so that subsequent transports can be imported again.

Notes

- i) The actual test scripts to be called can be stored in a custom field on the Business Task (or Transport Form) within ActiveControl. All custom field information on the Business Task(s) / Transport Form(s) is passed to Jenkins as part of the Integration, making it possible for Jenkins to thereafter trigger the relevant automated test scripts in other tools such as Selenium or Tosca based on the actual Business Tasks (or Transport Forms) in the Import Queue.
- ii) Given the Integration is triggered when Transport Forms reach Test Queue, it is advisable to configuration ActiveControl so that RC8 transports remain in the Import Queue. Otherwise, the Integration could be triggered on Changes that already have deployment issues.
- iii) It is possible to integrate ActiveControl with multiple third-party tools. In the example workflow, a JIRA integration is also being used, in addition to the Jenkins integration..
- iii) It is possible to trigger multiple scripts via Jenkins from one Test Queue.
- iv) It is possible to trigger different scripts in different Test Queues (ie Smoke Testing in a QA system, and full automated Regression Testing in a Pre-Production system).

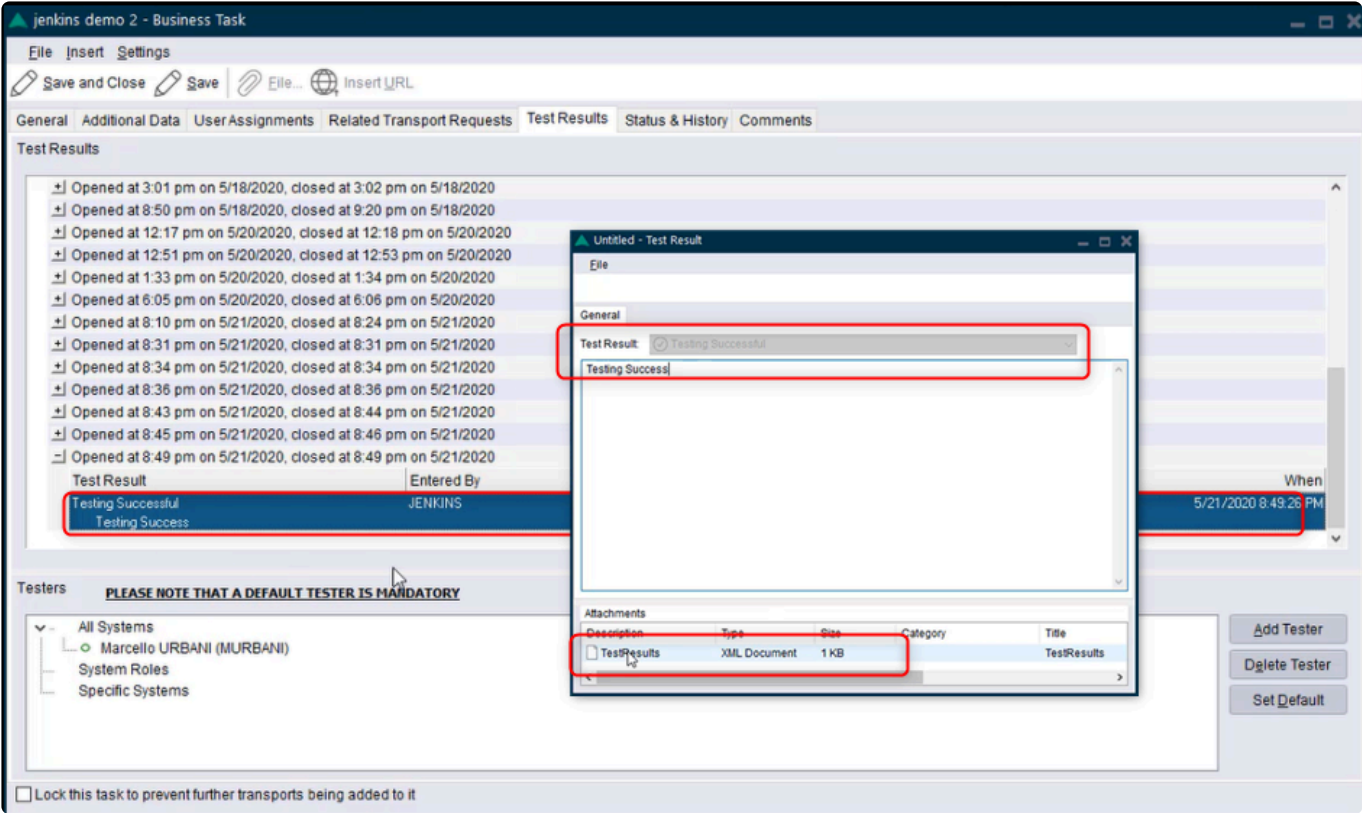


Figure: Example of an automated Test Signoff triggered via Jenkins integration. Depending on the requirement, attachments or URL links can be passed back to ActiveControl and appended to the Business Task

2.2. Technical Solution

The ActiveControl Jenkins integration includes both Outbound and Inbound integrations .

2.2.1. Outbound Integration

When transports land in a (configurable) Control Point location, the Jenkins automated testing is triggered via a URL send call.

An XML payload contains all information relating to the associated Business Tasks and Transport Forms, including all Custom Field information.

The Integration can also automatically lock the Import Queue at this point (since in most circumstances, a customer would not want further transports to be imported whilst an automated testing cycle is occurring).

2.2.2. Inbound Integration

When the automated testing cycle completes, Jenkins will report back to ActiveControl whether the testing was a Pass or Fail.

Based on that feedback, the Integration will then perform subsequent actions to add a Test Results entry onto the Business Tasks associated with the Transport Forms in the Import Queue, and if Pass, also move the Transport Forms out of the current Control Point location, to the next one.

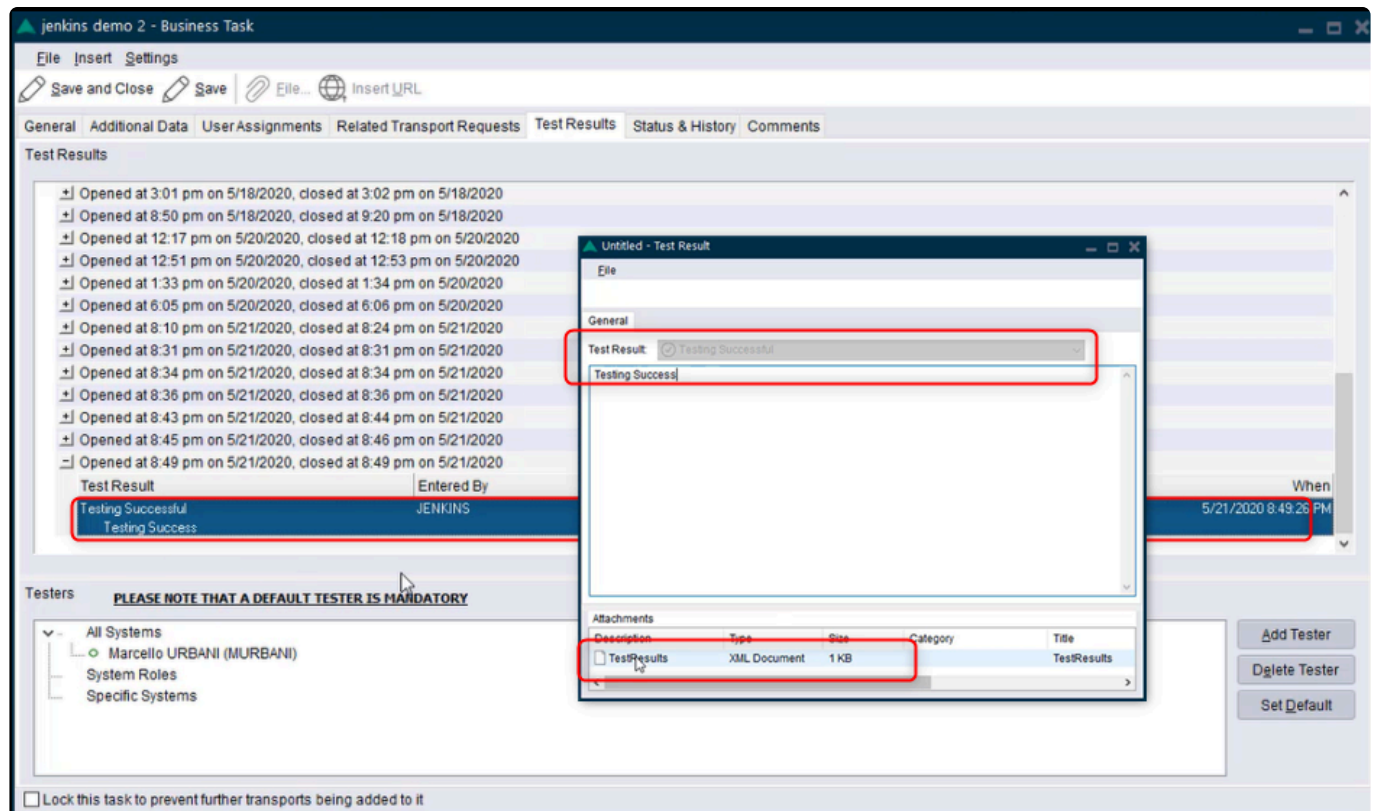


Figure: Example of an automated Test Signoff triggered via Jenkins integration. Depending on the requirement, attachments or URL links can be passed back to ActiveControl and appended to the Business Task

2.2.3. Integration Components

The following components form part of the ActiveControl / Jenkins Integration solution.

1.	Summary	Notes
1	Remote Function Calls	Yes
2	System Users	Yes. Both SAP user and Jenkins user
3	Number Range	None required for Jenkins Integration
4	Configuration Tables	Yes
5	Application Tables	Yes
6	Programs	Yes
7	APIs	None required for Jenkins integration
8	User Exit	None required for Jenkins integration
9	Jobs	Yes
10	Error Logging	Yes
11	General Configuration	Yes
12	Jenkins setup	

Associated Files

1.	File	Details
1	D00K904069	The Jenkins integration described in this Integration Guide is included as part of ActiveControl 8.3. This transport can be applied from ActiveControl 8.2 onwards (in the AC Domain Controller).
2	Pipeline.groovy	Groovy file for Jenkins pipeline
3	README.MD	Contains Developer level information relating to the outbound/inbound calls.

3. Integration Setup (SAP)

3.1. Remote Function Calls

An “HTTP Connection to External Server” RFC Destination is required in the ActiveControl Domain Controller to communicate with Jenkins.

This is configured via the usual way in SM59. The RFC requires an [integration user](#) to be created in Jenkins.

The figure displays two screenshots of the SAP SM59 configuration interface for an RFC Destination named **JENKINS_AWS**.

Left Screenshot (Logon Procedure Tab):

- Connection Test:** Icon for testing the connection.
- RFC Destination:** JENKINS_AWS
- Connection Type:** G HTTP Connection to External Serv
- Description:**
 - Description 1: jenkins on AWS
 - Description 2:
 - Description 3:
- Administration / Technical Settings / Logon & Security / Special Options:**
 - Logon Procedure:**
 - Logon with User:**
 - ☐ Do Not Use a User
 - ☒ Basic Authentication
 - User:** activeControl
 - PW Status:** leaved
 - Logon with Ticket:**
 - ☒ Do Not Send Logon Ticket
 - ☐ Send Logon Ticket Without Ref. to a Target System
 - ☐ Send Assertion Ticket for Dedicated Target System
 - System ID:** Client
 - Security Options:**
 - Status of Secure Protocol:**
 - ☒ Inactive
 - ☐ Active
 - SSL Certificate:** DEFAULT SSL Client (Standard) Cert. List
 - Authorization for Destination:**

Right Screenshot (Target System Settings and HTTP Proxy Options Tabs):

- Target System Settings:**
 - Target Host:** 10.0.1.119
 - Service No.:** 8080
 - Path Prefix:**
- HTTP Proxy Options:**
 - Global Configuration:**
 - Proxy Host:**
 - Proxy Service:**
 - Proxy User:**
 - Proxy PW Status:** is initial

Figure: A 'G' Type RFC Destination is required to communicate from SAP to Jenkins as part of Outbound integration.



The connectivity between Jenkins and SAP is absolutely key pre-requisite to a working Integration. Basis Technologies strongly recommend our Customers to review this topic as early on in the process for setting up the Integration as possible, to avoid delays caused by firewall or other network-level issues. Communication issues between the SAP system used as the ActiveControl Domain Controller and the 3rd Party system is the 'Number 1' challenge in setting up all ActiveControl integrations, and this Jenkins integration is no exception.

3.2. SAP User

A SAP user is required to support the Jenkins Integration.

The Integration could potentially use the existing AC_BATCH user, however given that this User is what the Test Results will be signed off against as part of the Integration – most customers might prefer to use a separate System user for the Jenkins integration to help differentiate events relating to the Jenkins Integration from other Integrations or other events such as Scheduled Imports within ActiveControl.

If creating a separate User, it should have the same authorisations as the AC_Batch user.

The screenshot shows the 'Display Users' form in SAP. The user 'JENKINS' is selected. The form includes tabs for Documentation, Address, Logon Data, SNC, Defaults, Parameters, Roles, Profiles, Groups, Personalization, and Lic. Data. The 'Person' section contains fields for Title, Last name (JENKINS), First name, Academic Title, Full Name (JENKINS), and Language. The 'Work Center' section includes Function, Department, Room Number, Floor, and Building code. The 'Communication' section has fields for Telephone, Mobile Phone, Fax, E-Mail Address, and Method. The 'Company' section shows 'Basis Technologies / Sheen Road 23 / GB TW9 1BN Richmond'.

The screenshot shows the 'Display Users' form with the 'Roles' tab selected. It displays a table of role assignments for the user 'JENKINS'.

Status	Role	T	Start Date	End Date	Short Role Description	Indi.
	/BT/TE:CTS_ADMIN_USER		05.10.2017	31.12.9999	Transport Express Admin role	
	/BT/TE:STD_ADMIN_AUTHS		05.10.2017	31.12.9999	Transport Express Standard Administration Auths	
	/BT/TE:STD_ADMIN_ROLE		05.10.2017	31.12.9999	Transport Express Standard Administrator Role	
	/BT/TE:STD_BASIS_AUTHS		05.10.2017	31.12.9999	Transport Express Standard Basis Role	
	/BT/TE:STD_DEVELOPER_AUTHS		05.10.2017	31.12.9999	Transport Express Standard Developer Auths	
	/BT/TE:STD_PLANNER_AUTHS		05.10.2017	31.12.9999	Transport Express Standard Planner Auths	
	/BT/TE:STD_TEAM_LEAD_AUTHS		05.10.2017	31.12.9999	Transport Express Standard Team Lead Authorisa	
	/BT/TE:STD_VIEW_AUTHS		05.10.2017	31.12.9999	Transport Express Standard View Auths	
	/BTR/MDR:REP_ADMIN		05.10.2017	31.12.9999	MDR Reporting Admin User	
	ZGF_STD_VIEW_AUTHS		05.10.2017	31.12.9999	Transport Express Standard View Auths	

The credentials of this SAP system user need to be stored in Jenkins to be referred in the scripts:

The screenshot shows the Jenkins 'Credentials' page. A credential is listed with the following details:

T	P	Store	Domain	ID
		Jenkins	(global)	acd_jenkins

The credential is of type 'Username with password' and is labeled 'jenkins/***** (User to connect to ACD)'.

3.3. ActiveControl general configuration

3.3.1. Custom Fields

The ActiveControl / Jenkins integration for automated testing relies on Jenkins being able to tell the testing tool what automated tests need to be performed.

In most scenarios, this is best done by the user (ie a Developer or Tester) manually indicating on the the Business Task (or Transport Form) what automated testing scripts should be performed against the particular Change or Transport. This is achieved in the current Integration through the use of custom field(s) on the Business Task or Transport Form (or both depending on the exact Customer requirement); these Custom Fields are configured via the Windows GUI configuration [Fields] tab in the usual way.

The information stored in these custom field(s) information is passed over to Jenkins as part of the outbound Integration (along with the Business Task and Transport Form information) – and the Customer Jenkins Administrator would then need to pass this information over from Jenkins to the test automation tool to trigger the relevant testing.

The screenshot shows a web application window titled "jenkins demo 2 - Business Task". The interface includes a menu bar with "File", "Insert", and "Settings". Below the menu bar are icons for "Save and Close", "Save", "File...", and "Insert URL". The main content area has several tabs: "General", "Additional Data", "User Assignments", "Related Transport Requests", "Test Results", "Status & History", and "Comments". The "General" tab is active, showing fields for "Identification" (Subject: jenkins demo 2, Reference: jd2, JIRA Change: BAU) and "Classification" (Priority: Normal, Type: Change Request, Group: UK). Below these are "Deployment Status" (smote test completed) and "Planning Status". A "Requirements Description" field is present. A red box highlights the "Automated Tests" field, which contains a list of test scripts: SeleniumScript1, SeleniumScript2, SeleniumScript3, and SeleniumScript4. At the bottom, there is a checkbox labeled "Lock this task to prevent further transports being added to it".

Figure: Example of test script information being stored in a Business Task custom field. This data is then passed over to Jenkins, for triggering the corresponding Testing scripts.



If multiple automated test cycles are required as part of an ActiveControl workflow (eg triggering Smoke Tests as part of a QA system and triggering Regression Tests as part

of a Pre-Production system), then the recommendation would be to have multiple Custom Fields. Again, it would be the responsibility of the Customer's Jenkins and Test Tool to call the relevant scripts and triggering of scripts via Jenkins.



All custom fields are sent across as part of the XML – ie there is no need to define specific Custom Fields as part of the integration configuration.

3.3.2. Import Options

Given Outbound integration is triggered by transports being imported into a system and landing in the Test Queue, it is advisable to DISABLE the following target configuration option

Continue importing transport requests when an import error occurs

It is also recommended to ENABLE the following target configuration option:

Continue importing queued transport requests for scheduled imports.

Target Properties - ECC Test - BAU (T01)

General Import Options Import Options II Inbox (Pending) Approvers Outbox Approvers Analysis Types

Import Options

Import Method: Fast Import (Import All) - AC Default sequence

Try to import transport requests in the order that they were imported into the predecessor target.
(None)

☐ Force transport dependencies when importing in same order as predecessor system.

Schedule a background job to automatically import transport requests at the times specified in the following transport schedule.
Emergencies

Optionally specify additional import schedules

Suppress import analysis during scheduled imports
Yes

☐ Import jobs scheduled by ActiveControl ☒ Orchestrated

☒ Continue importing transport requests when an import error occurs
☒ Continue importing queued transport requests for scheduled imports
☐ Automatically create backup transport requests

Timeout for delayed imports (minutes): 30

☐ Ignore System id during import (CTS+ only)

Merge / Parallel Development Streams

☒ Perform conflict analysis against this target
Client to be used during conflict analysis:
100
Default package for merged objects

☐ Require that transports with changes to SAP objects be manually merged
☐ Create a merge transport request in this SAP system after importing changes
Fix renamed objects in merge requests :
Never
☐ Add all dependant routines and formulae for BW merges
☐ Inherit merge transport owner from original transport (CTS+ only)

OK Cancel

3.4. SAP Configuration Tables

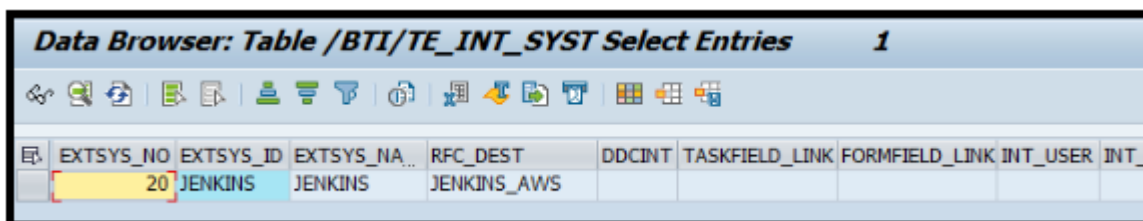
This section details the configuration tables as part of ActiveControl / Jenkins integration.

3.4.1. /BTI/TE_INT_SYST

This configuration table is used to specify the integrations that are running, and also some key information relating to the integration. It is possible to run multiple Integrations as part one ActiveControl implementation.

Field	Explanation
EXTSYS_NO	Integration System Number, this is a unique numerical identifier of the system to integrate with (as it is possible to integrate with multiple systems)
EXTSYS_ID	External System ID
EXTSYS_NAME	Name of External System
RFC_DEST	Name of the RFC Destination used for the Integration.
DDCINT	Not required for Jenkins Integration.
TASKFIELD_LINK	Not required for Jenkins Integration.
A FORMFIELD_LINK	Not required for Jenkins Integration.
INT_USER	Not required for Jenkins Integration.
INT_PASSWORD	Not required for Jenkins Integration.

Example configuration



EXTSYS_NO	EXTSYS_ID	EXTSYS_NAME	RFC_DEST	DDCINT	TASKFIELD_LINK	FORMFIELD_LINK	INT_USER	INT_PASSWORD
20	JENKINS	JENKINS	JENKINS_AWS					

Figure: Example configuration of the Jenkins integration



EXTSYS_NO does not need to be 20 as show in the Example configuration screenshot. It can be 01, or 02 – or whatever else is the lowest number not already configured in /BTI/TE_INT_SYST. Note that this Number is used in some of the other configuration tables.

3.4.2. /BTI/TE_INT_CLAS

This table is used to define Integration(s) and their corresponding Class; the classes are the bulk of the integration processing is done. The ActiveControl integration works on the principle of having a class for each external system that we need to integrate with.

Data Browser: Table /BTI/TE_INT_CLAS Select Entries

	EXTSYS_NO	CLASSNAME
	20	/BTI/TE_INTEGRATION_CI

3.4.3. /BTI/TE_INT_MAPP

This table is used to define the fields that form part of the Jenkins integration. For the current Jenkins integration, Group and Priority information is sent across as part of the Integration.

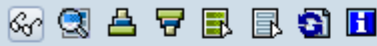
Data Browser: Table /BTI/TE_INT_MAPP Select Entries 2						
EXTSYS_NO	EXTSYS_NAME	DIRECTION	SEQUENCE_NO	TEFIELDREF	EXTERNAL_REF	KEY_F
20	JENKINS	O	1	/BTI/TE_TASK-GROUPID	/BTI/TE_TASK-GROUPID	
20	JENKINS	O	2	/BTI/TE_TASK-PRIORITY	/BTI/TE_TASK-PRIORITY	

3.4.4. /BTI/TE_INT_PC

Table /BTI/TE_INT_PC details the process codes that are available as part of the Integration Integration Framework.

As part of Jenkins integration, only CREATE process code is used.

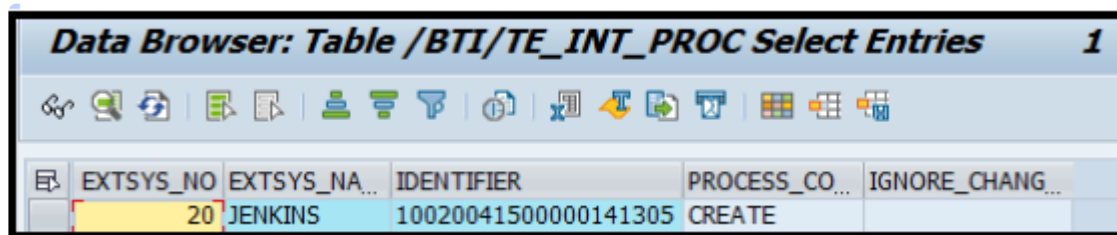
This table comes pre-configured as part of ActiveControl, and so should not need to be updated as part of the Jenkins integration

Data Browser: Table /BTI/TE_INT_PC Select Entries			4
			
Table: /BTI/TE_INT_PC			
Displayed Fields: 3 of 3		Fixed Columns: [1]	List Width 0250
PROCESS_CODE	CODE_DESCRIPTION	INBOUND ACTION CLASS	
<input type="checkbox"/> CREATE	Create Integration Record	/BTI/TE_CL_INTEGR_CREATE	
<input type="checkbox"/> TESTRES	Enter test results	/BTI/TE_CL_INTEGR_TESTRES	
<input type="checkbox"/> TRANSITION	State transition		
<input type="checkbox"/> UPDATE	Update Integration Record	/BTI/TE_CL_INTEGR_UPDATE	

3.4.5. /BTI/TE_INT_PROC

This table is used within the Integration Framework to define the Process Identifiers that are used within the Integration.

As part of Jenkins integration, only CREATE process code is used. However it is a slightly unique use of CREATE, as in most other existing Integrations such as with ServiceNow and JIRA, the CREATE is used to create a Business Task within ActiveControl.



The screenshot shows a 'Data Browser' window titled 'Table /BTI/TE_INT_PROC Select Entries 1'. It contains a table with the following data:

EXTSYS_NO	EXTSYS_NA...	IDENTIFIER	PROCESS_CO...	IGNORE_CHANG...
20	JENKINS	10020041500000141305	CREATE	

Figure: Only the CREATE proces code is used as part of the Jenkins integration

Notes

1. EXTSYS_NO needs to match the Number configured in /BTI/TE_INT_SYST.
2. The Identifier is the Deployment Status GUID from /BTI/TE_TASKSTAT at which point the Integration is being triggered. For the triggering of automated testing, it will be Deployment Status of the Test Queue at which point Jenkins should be triggered.

3.4.6. /BTI/TE_TVAREV

Table /BTI/TE_TVAREV is used as part of the Jenkins integration – to configure the trigger URL used as part of the Outbound integration.

Data Browser: Table /BTI/TE_TVAREV Select Entries 1						
NAME	NUMB	SIGN	OPTI	LOW	HIGH	
CI_INTEGRATION_URI_20	1			/job/activeControlDemo/build?token=fromAc		

The entry needs to reference the jenkins trigger URL, which depends on the script name and token:

Jenkins > activeControlDemo >

General

Build Triggers

Advanced P

☐ Disable this project

☐ Quiet period

☒ Trigger builds remotely (e.g., from scripts)

Authentication Token

fromAc

Use the following U

Optionally append &

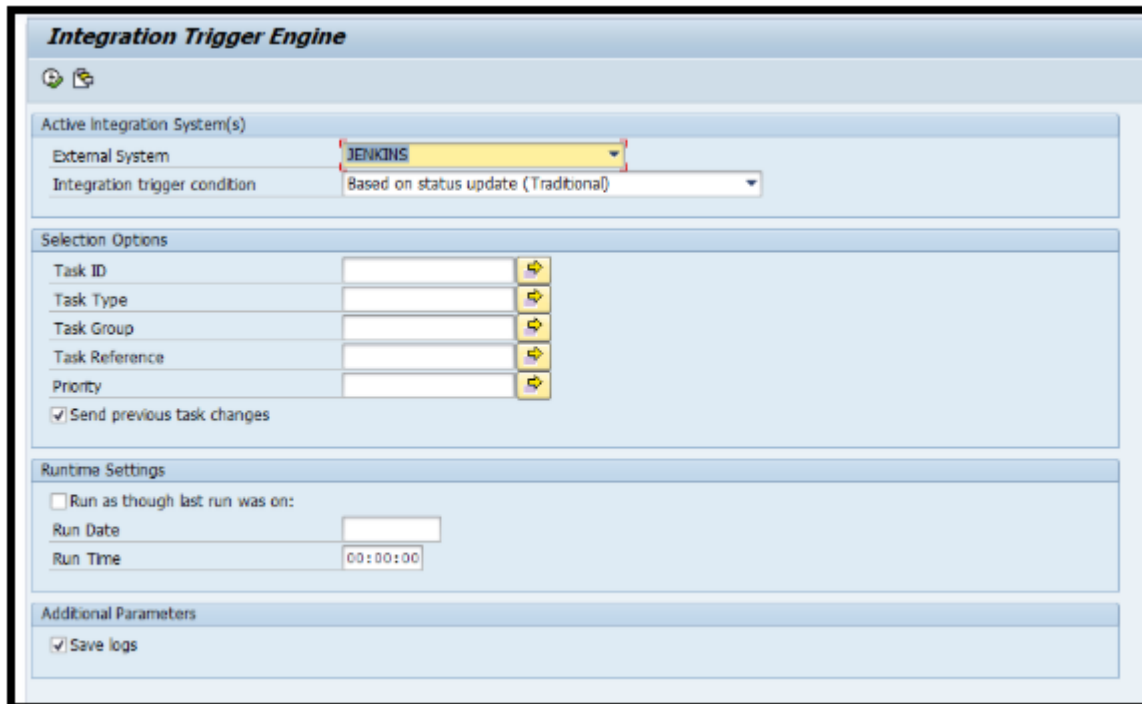
! Note that there is a 50 character limit that can entered as the script token.

3.5. Programs

3.5.1. /BTI/TE_INTEG_TRIGGER

The /BTI/TE_INTEG_TRIGGER trigger program is used as part of the Jenkins Integration to select the appropriate ActiveControl records to push out to Jenkins.

A Variant should be saved as per screenshot:



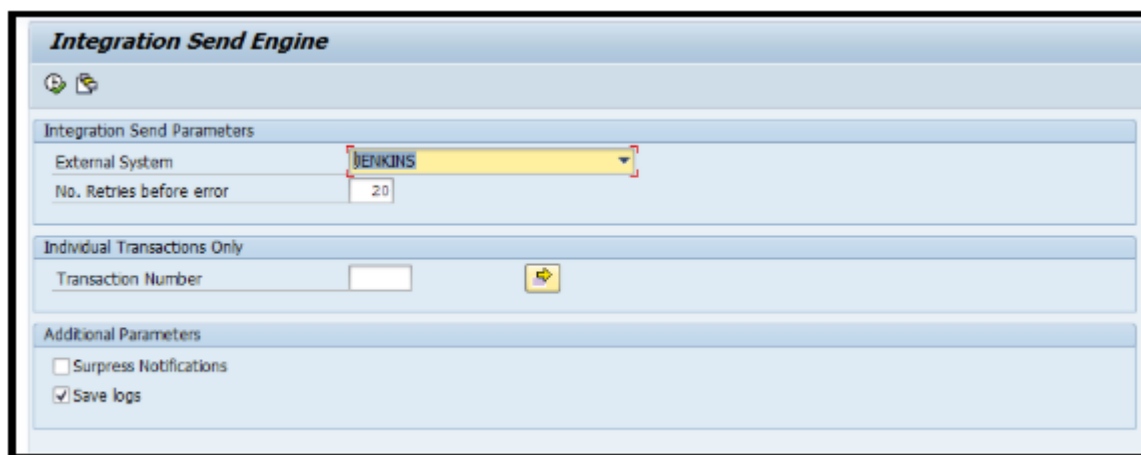
The screenshot displays the 'Integration Trigger Engine' configuration window. It is organized into several sections:

- Active Integration System(s):** Contains two dropdown menus. The 'External System' is set to 'JENKINS', and the 'Integration trigger condition' is set to 'Based on status update (Traditional)'.
- Selection Options:** Includes five input fields for 'Task ID', 'Task Type', 'Task Group', 'Task Reference', and 'Priority', each with a yellow arrow icon to its right. Below these fields is a checked checkbox labeled 'Send previous task changes'.
- Runtime Settings:** Features a checkbox 'Run as though last run was on:' which is currently unchecked. Below it are input fields for 'Run Date' and 'Run Time' (set to '00:00:00').
- Additional Parameters:** Contains a checked checkbox labeled 'Save logs'.

3.5.2. /BTI/TE_INTEG_SEND

The /BTI/TE_INTEG_SEND send program is used to pick up the mapped transactions and send them out to the configured external systems. It retrieves the required records and then uses the configured send methods for each particular integration scenario to actually push the data out to the receiving systems.

A Variant should be saved as per screenshot.



The screenshot shows the 'Integration Send Engine' configuration window. It has a title bar with a clock icon and a help icon. The window is divided into three main sections:

- Integration Send Parameters:** Contains a dropdown menu for 'External System' with 'JENKINS' selected, and a text box for 'No. Retries before error' with the value '20'.
- Individual Transactions Only:** Contains a text box for 'Transaction Number' and a yellow button with a right-pointing arrow.
- Additional Parameters:** Contains two checkboxes: 'Surpress Notifications' (unchecked) and 'Save logs' (checked).

3.6. Scheduled Jobs

The two aforementioned Trigger and Send Program variants should be scheduled to run in the Domain Controller system as a scheduled job via SM37.

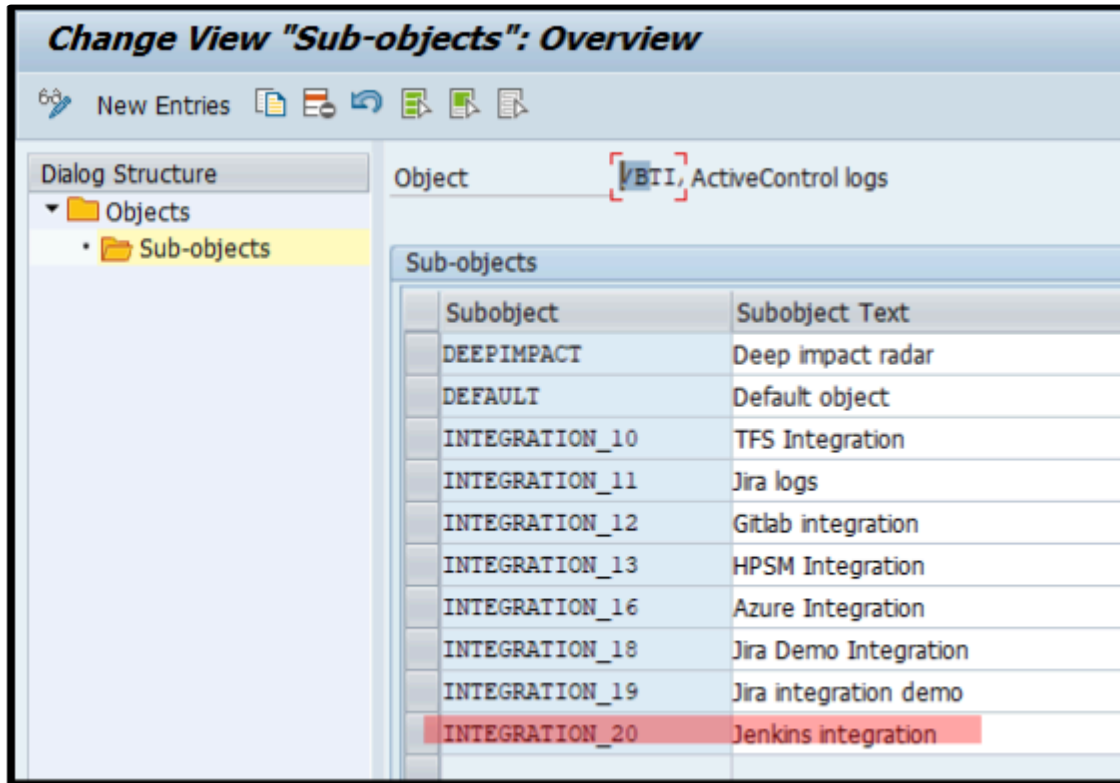
Every 2 minutes should suffice, with the Trigger program as the first step, and the Send program as the second step.

The AC_Batch user can be used to schedule this Job.

3.7. Error Logging

Standard SAP logging is possible as part of the Jenkins integration. The prerequisite for this is that the subobject of /BTI/TE is created via transaction SLG0 in the Domain Controller.

Note that INTEGRATION_NN, the NN should be the EXTSYST_NO as defined in /BTI/TE_INT_SYST.



Transaction SLG1 can be used within ActiveControl Domain Controller system to view the integration Logging output as part of the Outbound integration.

Nothing is logged in SLG1 as part of Inbound integration.

4. Integration Setup (Jenkins)

4.1. Build Pipeline

The ActiveControl / Jenkins integration outbound integration works by ActiveControl calling Jenkins pipeline build.

Customer Jenkins Administrators would need to setup this Jenkins pipeline for the purpose of the Integration with ActiveControl. This would be used to as part of the ActiveControl outbound integration, to perform the relevant Jenkins-side action ie triggering automated testing scripts within tools such as Selenium or Tosca. They would also be used to trigger events in ActiveControl as part of Inbound Integration.

The likely events as part of outbound and inbound integration are as follows:

Number	Event
1	Locking the Import Queue (to prevent the import of subsequent transports whilst the Integration is in progress.
2	Passing contents of the Test Queue / Required Tests
3	Initiate the required Tests
4	Sending PASS or FAIL test result information back to ActiveControl.
5	Unlocking the Import Queue at the end of Inbound integration.
6	Uploading Test results to the Business Task, and moving the associated Transport Forms forward to the next location in the workflow.

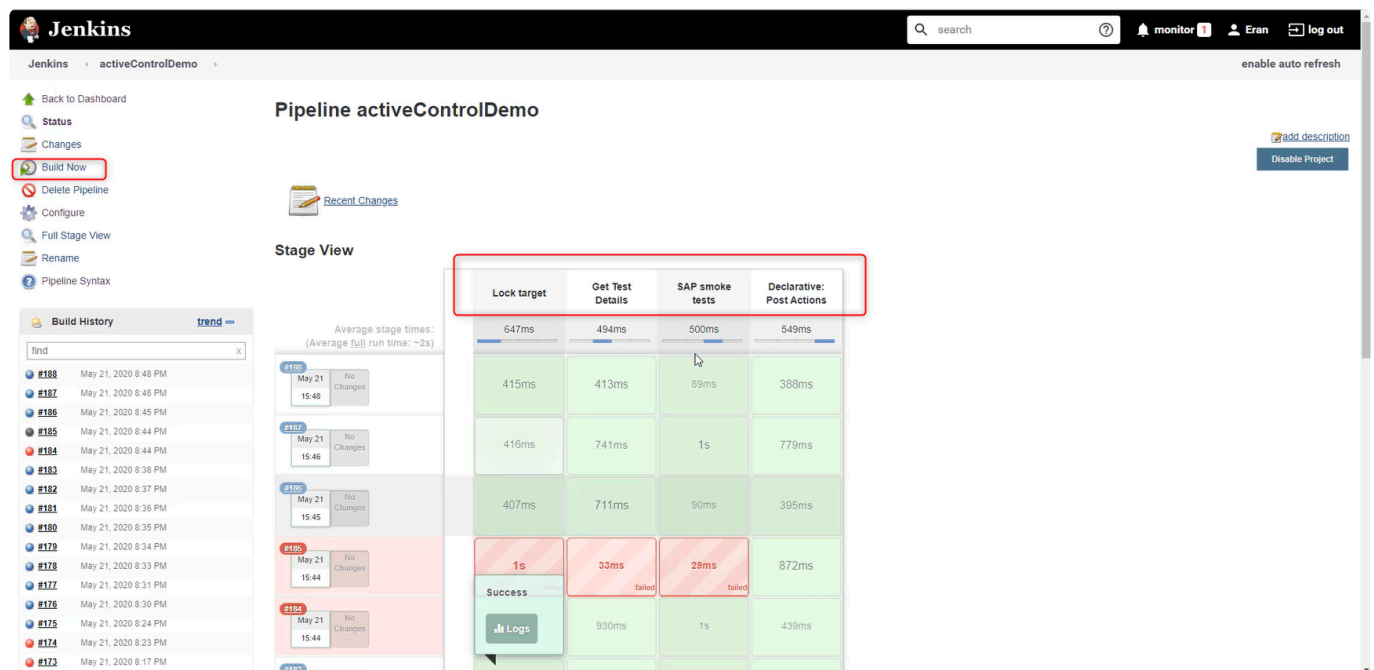


Figure: Example of Jenkins Pipeline to support automate testing.

4.2. Jenkins User

The integration requires a user in Jenkins that has authorisation to trigger jobs, with an authorization token defined.

User/group	Overall	Credentials		Agent			Job			Run	View	SCM	Lockable Resources										
	Administer	Create	Delete	Update Manage Domains	View	Build	Configure	Cancel	Discover	Create	Read	Move	Workspace	Delete	Configure	Update	Create	Delete	Read	Tag	Preserve	Unlock	View
Anonymous Users																							
Authenticated Users																							
ActiveControl trigger																							
Marcello Urbani																							

This user needs a token that will be stored in the SAP RFC Destination as the password.

Jenkins

Jenkins

ActiveControl trigger

People

Status

Builds

Configure

My Views

Credentials

Full Name

ActiveControl trigger

Description

API Token

Current token(s)

ACD

Created 5 day(s) ago

4.3. URL scripts

Please refer to README file provided with the Integration for some developer Curl examples.

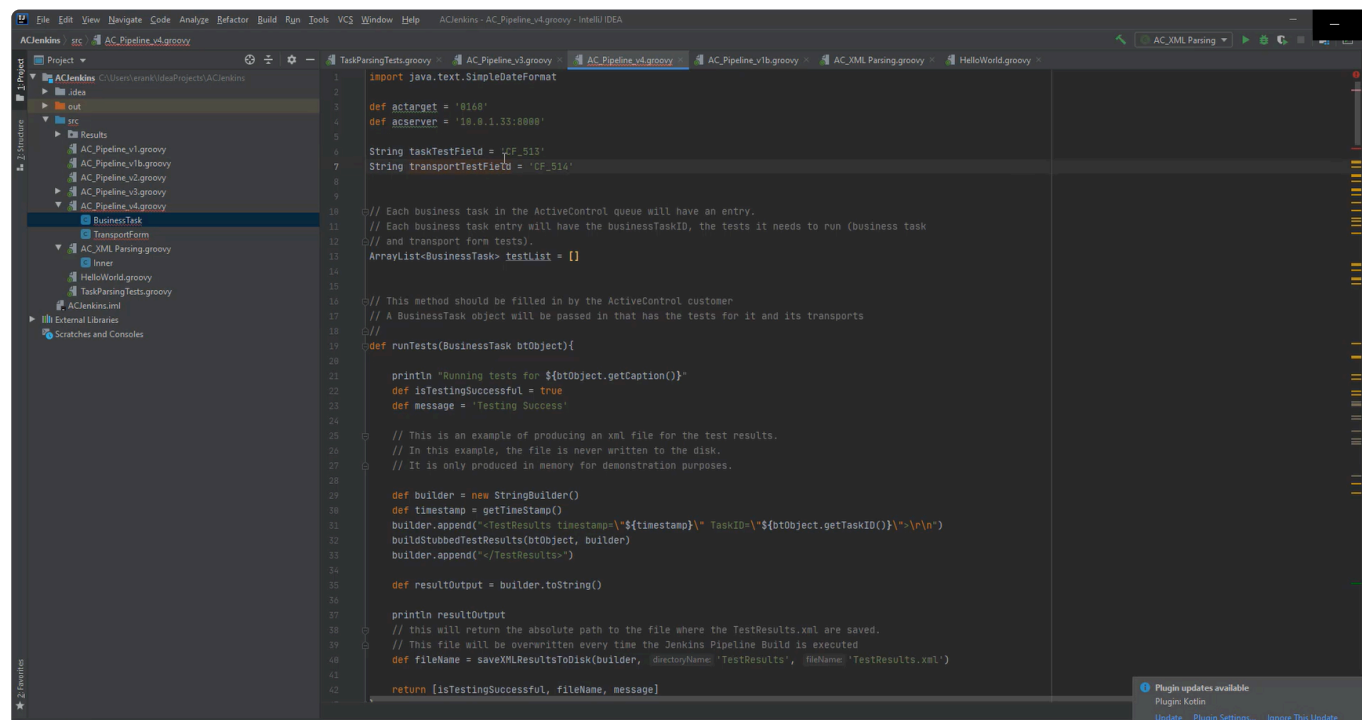


Figure: Example of Groovy script used as part of the Integration.

4.4.

1. Basic jenkins installation for testing of ActiveControl

AC will trigger jenkins calling an URL, like:

```
```bash
```

1. AC will not actually use CURL...  
 export triggeruser=acuser  
 export triggerToken=secret # this is a crypto key generated by Jenkins  
 export jenkinsServer=myjenkinsserverurl  
 export jobname=activeControlDemo  
 export AcToken=sentFromActiveControl

```
curl "https://$triggeruser:$triggerToken@$jenkinsServer/job/$jobname/build?token=$AcToken"
```
```

Jenkins will trigger actions in AC in a similar way

```
```bash
```

1. These usually go in some secret place inside jenkins  
 export acuser=acuser  
 export acpass=secret  
 export acserver=te.basistechnologies.net:8000

1. lock a target

```
curl "http://$acuser:$acpass@$acserver/bti/te_web_services?action=LOCK_TARGET&TARGETID=0001"
```

1. unlock a target  
 curl "http://\$acuser:\$acpass@\$acserver/bti/te\_web\_services?action=UNLOCK\_TARGET&TARGETID=0001"

1. read contents of a test queue

```
curl "http://$acuser:$acpass@$acserver/bti/te_web_services?action=QUEUE_CONTENTS&TARGETID=0001&LOCATION=T"
```

1. pass tests in a target  
 curl -X POST -d '0001XPassed' "http://\$acuser:\$acpass@\$acserver/bti/te\_web\_services?action=TARGET\_TESTRESULTS"

1. fail tests in a target  
 curl -X POST -d '0001Failed' "http://\$acuser:\$acpass@\$acserver/bti/te\_web\_services?action=TARGET\_TESTRESULTS"

```
te_web_services?action=TARGET_TESTRESULTS"
```

```
...
```

Sample response from lock/unlock/test results:

```
```XML
```

```
<?xml version="1.0" encoding="utf-8"?>
```

```
EErrorMethod ENTER_TEST_RESULTS requires a x_task or xt_task parameter
```

```
...
```

Sample read queue response:

```
```XML
```

```
<?xml version="1.0" encoding="utf-8"?>
```

```
10020041500000141320jd1jenkins demo 1 ACDK908578ACDK90861810020041600000141745jd4jd4
```

```
ACDK908608SInformationSuccess
```

```
...
```

## 5. ActiveControl HTTP API

---

## 5.1. Get Queue Contents

### HTTP GET

Example URL:

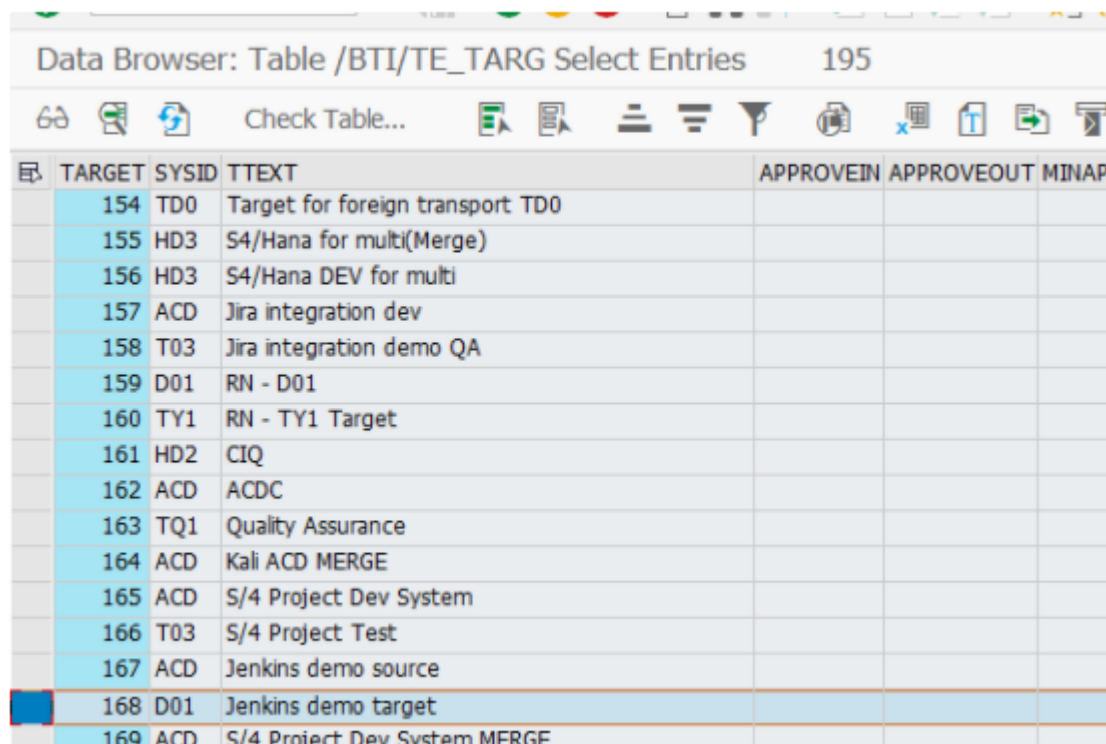
<http://te.basistechnologies.net:8000/bti/>

[te\\_web\\_services?action=QUEUE\\_CONTENTS&TARGETID=168&LOCATION=T](http://te.basistechnologies.net:8000/bti/te_web_services?action=QUEUE_CONTENTS&TARGETID=168&LOCATION=T)

Authentication – Basic (ActiveControl user credentials)

### Request Parameters

Name	Value	Description
action	QUEUE_CONTENTS	The get queue content action
TARGETID	168	Look in the /BTI/TE_TARG table for the target id. See example below.
LOCATION	T	The queue location in the target system. I – Inbox Q – Import Queue T – Test queue O – Outbox



Data Browser: Table /BTI/TE\_TARG Select Entries 195

TARGET	SYSID	TTEXT	APPROVEIN	APPROVEOUT	MINAP
154	TD0	Target for foreign transport TD0			
155	HD3	S4/Hana for multi(Merge)			
156	HD3	S4/Hana DEV for multi			
157	ACD	Jira integration dev			
158	T03	Jira integration demo QA			
159	D01	RN - D01			
160	TY1	RN - TY1 Target			
161	HD2	CIQ			
162	ACD	ACDC			
163	TQ1	Quality Assurance			
164	ACD	Kali ACD MERGE			
165	ACD	S/4 Project Dev System			
166	T03	S/4 Project Test			
167	ACD	Jenkins demo source			
168	D01	Jenkins demo target			
169	ACD	S/4 Project Dev System MERGE			

## Response

### XML Payload

```
<?xml version="1.0" encoding="utf-8"?>
<asx:abap version="1.0" xmlns:asx="http://www.sap.com/abapxml">
 <asx:values>
 <QUEUE>
 <item> <!-- Zero to many Business Tasks -->
 <ID>10020041500000141320</ID>
 <CAPTION>jenkins demo 1</CAPTION>
 <REFERENCE>jd1</REFERENCE>
 <GROUPID>10017100600000000012</GROUPID>
 <TYPEID>10017100600000000017</TYPEID>
 <TESTERID>MURBANI</TESTERID>
 <PRIORITY>2</PRIORITY>
 <PROJECTID>10017100600000000022</PROJECTID>
 <LOCKED/>
 <PATH>00</PATH>
 <STAT_DEPL>10020041500000141305</STAT_DEPL>
 <STAT_PLAN/>
 <STAT_DEPL_MAN/>
 <STAT_PLAN_MAN/>

 <OWNER/>
 <TEXT/>
 <CF_508/>
 <CF_512/>
 <CF_513>Business task test1</CF_513>
 <TRANSPORTS>
 <item> <!-- Zero to many Transport Forms -->
 <TRKORR>ACDK908578</TRKORR>
 <REQTEXT/>
 <FORMDESCRIPTION/>
 <GROUPID>10017100600000000008</GROUPID>
 <TYPEID>10017100600000000020</TYPEID>
 <TRFUNCTION/>
 <TRCATEGORY/>
 <TRSTATUS/>
 <CLIENT/>
 <REQUESTOR>MURBANI</REQUESTOR>
 <REQNAME/>
 <REQDATE>20200415</REQDATE>
 <REQTIME>104145</REQTIME>
 <RELDATE/>
 <RELTIME/>
 <PATH>55</PATH>
```

```

<PATH>55</PATH>
<HASFORM/>
<OWNER/>
<COMPLETED/>
<CLIDEP/>
<QUEUED/>
<HIDDEN/>
<LOCKED/>
<CONFLICTS/>
<REFERENCE/>
<UMODES/>
<NOEXPLOGS/>
<EXPORTSTAT/>
<TSTIMPSTAT/>
<LOCSTATUS/>
<MANDT/>
<RETURNCODE/>
<TARSYSTEM/>
<CHANGETYPE/>
<MANUAL_STEP_STATUS/>
<PREV_IMP_RESULT/>
<CONFLICT_STAT>0</CONFLICT_STAT>
<SCI_STAT>0</SCI_STAT>
<RISK_STAT>0</RISK_STAT>
<NUM_OBJECTS>0000000000</NUM_OBJECTS>
<NUM_KEYS>0000000000</NUM_KEYS>
<CONF_STAT_RUNID/>
<SCI_STAT_RUNID/>

```

```

<RISK_STAT_RUNID/>
<REJECTED/>
<AUTO_APPR_STATUS/>
<CF_500>HAMPTON HILL</CF_500>
<CF_501>N</CF_501>
<CF_502/>
<CF_503/>
<CF_504>N</CF_504>
<CF_505>N</CF_505>
<CF_506>N</CF_506>
<CF_507>N</CF_507>
<CF_509/>
<CF_510/>
<CF_511/>
<CF_514>Test for 908578</CF_514>
</item>
</TRANSPORTS>

```



```

 </TRANSPORTS>
 </item>
</QUEUE>
<BTI_ERROR_MSG>
 <BTIEM_MSGTYP>S</BTIEM_MSGTYP>
 <BTIEM_TITLE>Information</BTIEM_TITLE>
 <BTIEM_MESSAGE>Success</BTIEM_MESSAGE>
 <BTIEM_OVERRIDES/>
 <BTIEM_EXCEPTION/>
 <BTIEM_GUID/>
 <BTIEM_IN_PROGRESS/>
</BTI_ERROR_MSG>
</asx:values>
</asx:abap>

```

## Postman Example

GET [http://te.basistechnologies.net:8000/bti/te\\_web\\_services?action=QUEUE\\_CONTENTS&TARGETID=168&LOCATION=T](http://te.basistechnologies.net:8000/bti/te_web_services?action=QUEUE_CONTENTS&TARGETID=168&LOCATION=T)

Params **Authorization** Headers (9) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	action	QUEUE_CONTENTS	Get Queue Contents action
<input checked="" type="checkbox"/>	TARGETID	168	The ID of the target system
<input checked="" type="checkbox"/>	LOCATION	T	The queue location in the target system.
	Key	Value	Description

Body Cookies (2) Headers (7) Test Results

Pretty Raw Preview Visualize XML

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <asx:abap version="1.0" xmlns:asx="http://www.sap.com/abapxml">
3 <asx:values>
4 <QUEUE/>
5 <BTI_ERROR_MSG>
6 <BTIEM_MSGTYP>S</BTIEM_MSGTYP>
7 <BTIEM_TITLE>Information</BTIEM_TITLE>
8 <BTIEM_MESSAGE>Success</BTIEM_MESSAGE>
9 <BTIEM_OVERRIDES/>
10 <BTIEM_EXCEPTION/>
11 <BTIEM_GUID/>
12 <BTIEM_IN_PROGRESS/>
13 </BTI_ERROR_MSG>
14 </asx:values>
15 </asx:abap>

```

GET [http://te.basistechnologies.net:8000/bti/te\\_web\\_services?action=QUEUE\\_CONTENTS&TARGETID=0168&LOCATION=T](http://te.basistechnologies.net:8000/bti/te_web_services?action=QUEUE_CONTENTS&TARGETID=0168&LOCATION=T)

GET ▼ http://te.basistechnologies.net:8000/bti/te\_web\_services?action=QUEUE\_CONTENTS&TARGETID=0168&LOCATION=I

Params ● Authorization Headers (9) Body Pre-request Script Tests Settings

Headers 🔍 Hide auto-generated headers

	KEY	VALUE
<input checked="" type="checkbox"/>	Authorization ⓘ	Basic Z [REDACTED]
<input checked="" type="checkbox"/>	Cookie ⓘ	sap-usercontext=sap-client=100; SAP_5
<input checked="" type="checkbox"/>	Cache-Control ⓘ	no-cache
<input checked="" type="checkbox"/>	Postman-Token ⓘ	<calculated when request is sent>
<input checked="" type="checkbox"/>	Host ⓘ	<calculated when request is sent>
<input checked="" type="checkbox"/>	User-Agent ⓘ	PostmanRuntime/7.26.5
<input checked="" type="checkbox"/>	Accept ⓘ	*/*
<input checked="" type="checkbox"/>	Accept-Encoding ⓘ	gzip, deflate, br
<input checked="" type="checkbox"/>	Connection ⓘ	keep-alive
	Host	te.basistechnologies.net

# 5.2. Lock Queue

## HTTP GET

Example URL:

*http://te.basistechnologies.net:8000/bti/te\_web\_services?action=LOCK\_TARGET&TARGETID=168*

Authentication – Basic (ActiveControl user credentials)

Note – This locks only the import queue for the target system.

## Request Parameters

Name	Value	Description
action	LOCK_TARGET	Locks the target system import queue.
TARGETID	168	Look in the /BTI/TE_TARG table for the target id.

## Response

```
<?xml version="1.0" encoding="utf-8"?>
<asx:abap version="1.0" xmlns:asx="http://www.sap.com/abapxml">
 <asx:values>
 <BTI_ERROR_MSG>
 <BTIEM_MSGTYP>S</BTIEM_MSGTYP>
 <BTIEM_TITLE>Information</BTIEM_TITLE>
 <BTIEM_MESSAGE>Target Locked</BTIEM_MESSAGE>
 <BTIEM_OVERRIDES/>
 <BTIEM_EXCEPTION/>
 <BTIEM_GUID/>
 <BTIEM_IN_PROGRESS/>
 </BTI_ERROR_MSG>
 </asx:values>
</asx:abap>
```

## Postman Example

The screenshot displays a Postman interface for a GET request. The URL is `http://te.basistechnologies.net:8000/bti/te_web_services?action=LOCK_TARGET&TARGETID=168`. The 'Params' tab is active, showing a table of query parameters.

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	action	LOCK_TARGET	The LOCK_TARGET action
<input checked="" type="checkbox"/>	TARGETID	168	The ID of the target system import queue to lock.
	Key	Value	Description

Below the parameters, the 'Body' tab is selected, showing the XML response in 'Pretty' format. The XML content is as follows:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <asx:abap version="1.0" xmlns:asx="http://www.sap.com/abapxml">
3 <asx:values>
4 <BTI_ERROR_MSG>
5 <BTIEM_MSGTYP>S</BTIEM_MSGTYP>
6 <BTIEM_TITLE>Information</BTIEM_TITLE>
7 <BTIEM_MESSAGE>Target Locked</BTIEM_MESSAGE>
8 <BTIEM_OVERRIDES/>
9 <BTIEM_EXCEPTION/>
10 <BTIEM_GUID/>
11 <BTIEM_IN_PROGRESS/>
12 </BTI_ERROR_MSG>
13 </asx:values>
14 </asx:abap>
```

## 5.3. Unlock Queue

### HTTP GET

Example URL:

*http://te.basistechnologies.net:8000/bti/te\_web\_services?action=UNLOCK\_TARGET&TARGETID=168*

Authentication – Basic (ActiveControl user credentials)

Note – This unlocks only the import queue for the target system.

### Request Parameters

Name	Value	Description
action	UNLOCK_TARGET	Unlocks the target system import queue.
TARGETID	168	Look in the /BTI/TE_TARG table for the target id.

### Response

```
<?xml version="1.0" encoding="utf-8"?>
<asx:abap version="1.0" xmlns:asx="http://www.sap.com/abapxml">
 <asx:values>
 <BTI_ERROR_MSG>
 <BTIEM_MSGTYP>S</BTIEM_MSGTYP>
 <BTIEM_TITLE>Information</BTIEM_TITLE>
 <BTIEM_MESSAGE>Target Unocked</BTIEM_MESSAGE>
 <BTIEM_OVERRIDES/>
 <BTIEM_EXCEPTION/>
 <BTIEM_GUID/>
 <BTIEM_IN_PROGRESS/>
 </BTI_ERROR_MSG>
 </asx:values>
</asx:abap>
```

Note – the BTIEM\_MESSAGE value 'Target Unocked' is misspelled. A defect has been submitted to fix the spelling in a future release.

## Postman Example


GET [http://te.basistechnologies.net:8000/bti/te\\_web\\_services?action=UNLOCK\\_TARGET&TARGETID=168](http://te.basistechnologies.net:8000/bti/te_web_services?action=UNLOCK_TARGET&TARGETID=168)

Params **Authorization** Headers (9) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	action	UNLOCK_TARGET	The UNLOCK_TARGET action
<input checked="" type="checkbox"/>	TARGETID	168	The ID of the target system import queue to lock.
	Key	Value	Description

Body Cookies (3) Headers (4) Test Results

Pretty Raw Preview Visualize XML 

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <asx:abap version="1.0" xmlns:asx="http://www.sap.com/abapxml">
3 <asx:values>
4 <BTI_ERROR_MSG>
5 <BTIEM_MSGTYP>S</BTIEM_MSGTYP>
6 <BTIEM_TITLE>Information</BTIEM_TITLE>
7 <BTIEM_MESSAGE>Target Unocked</BTIEM_MESSAGE>
8 <BTIEM_OVERRIDES/>
9 <BTIEM_EXCEPTION/>
10 <BTIEM_GUID/>
11 <BTIEM_IN_PROGRESS/>
12 </BTI_ERROR_MSG>
13 </asx:values>
14 </asx:abap>
```

## 5.4. Save Business Task Result

HTTP POST

Example URL:

`http://te.basistechnologies.net:8000/bti/`

`te_web_services?action=TARGET_TESTRESULTS&TASKID=10020041500000141320`

Authentication – Basic (ActiveControl user credentials)

### Request Parameters

Name	Value	Description
action	TARGET_TESTRESULTS	The TARGET_TESTRESULTS action
TASKID	10020041500000141320	The ActiveControl Business Task ID

### Response

```
<?xml version="1.0" encoding="utf-8"?>
<asx:abap version="1.0" xmlns:asx="http://www.sap.com/abapxml">
 <asx:values>
 <BTI_ERROR_MSG>
 <BTIEM_MSGTYP>S</BTIEM_MSGTYP>
 <BTIEM_TITLE>Information</BTIEM_TITLE>
 <BTIEM_MESSAGE></BTIEM_MESSAGE>
 <BTIEM_OVERRIDES/>
 <BTIEM_EXCEPTION/>
 <BTIEM_GUID/>
 <BTIEM_IN_PROGRESS/>
 </BTI_ERROR_MSG>
 </asx:values>
</asx:abap>
```

## Error response example

```
<?xml version="1.0" encoding="utf-8"?>
<asx:abap version="1.0" xmlns:asx="http://www.sap.com/abapxml">
 <asx:values>
 <BTI_ERROR_MSG>
 <BTIEM_MSGTYP>E</BTIEM_MSGTYP>
 <BTIEM_TITLE>Error</BTIEM_TITLE>
 <BTIEM_MESSAGE>No task available to record results to</BTIEM_MESSAGE>
 <BTIEM_OVERRIDES/>
 <BTIEM_EXCEPTION/>
 <BTIEM_GUID/>

 <BTIEM_IN_PROGRESS/>
 </BTI_ERROR_MSG>
 </asx:values>
</asx:abap>
```



## Postman Example

POST ▼ [http://te.basistechnologies.net:8000/bti/te\\_web\\_services?action=TARGET\\_TESTRESULTS&TASKID=10020041500000141320](http://te.basistechnologies.net:8000/bti/te_web_services?action=TARGET_TESTRESULTS&TASKID=10020041500000141320)

Params ● Authorization Headers (11) Body ● Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	action	TARGET_TESTRESULTS	The TARGET_TESTRESULTS action
<input checked="" type="checkbox"/>	TASKID	10020041500000141320	The ActiveControl Business Task ID
	Key	Value	Description

POST ▼ [http://te.basistechnologies.net:8000/bti/te\\_web\\_services?action=TARGET\\_TESTRESULTS&TASKID=10020041500000141320](http://te.basistechnologies.net:8000/bti/te_web_services?action=TARGET_TESTRESULTS&TASKID=10020041500000141320)

Params ● Authorization Headers (11) Body ● Pre-request Script Tests Settings

☐ none
 ☒ form-data
 ☐ x-www-form-urlencoded
 ☐ raw
 ☐ binary
 ☐ GraphQL

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	queueresultsfile	<TESTRESULTS> < ...	This specifies the target ID and if the test was successful
<input checked="" type="checkbox"/>	attachment	Select Files	Optional file attachment for the actual results
	Key	Value	Description

POST ▼ [http://te.basistechnologies.net:8000/bti/te\\_web\\_services?action=TARGET\\_TESTRESULTS&TASKID=10020041500000141320](http://te.basistechnologies.net:8000/bti/te_web_services?action=TARGET_TESTRESULTS&TASKID=10020041500000141320)

Params ● Authorization Headers (11) Body ● Pre-request Script Tests Settings

Headers 🔍 Hide auto-generated headers

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	Authorization <span>①</span>	Basic	
<input checked="" type="checkbox"/>	Cookie <span>①</span>	sap-usercontext=sap-client=1	
<input checked="" type="checkbox"/>	Cache-Control <span>①</span>	no-cache	
<input checked="" type="checkbox"/>	Postman-Token <span>①</span>	<calculated when request is s	
<input checked="" type="checkbox"/>	Content-Type <span>①</span>	multipart/form-data; boundar	
<input checked="" type="checkbox"/>	Content-Length <span>①</span>	<calculated when request is s	
<input checked="" type="checkbox"/>	Host <span>①</span>	<calculated when request is s	
<input checked="" type="checkbox"/>	User-Agent <span>①</span>	PostmanRuntime/7.26.5	
<input checked="" type="checkbox"/>	Accept <span>①</span>	*/*	
<input checked="" type="checkbox"/>	Accept-Encoding <span>①</span>	gzip, deflate, br	
<input checked="" type="checkbox"/>	Connection <span>①</span>	keep-alive	
	Key	Value	Description

## 6. Further Information

---

This integration is available from ActiveControl 8.3 onwards.

As with all ActiveControl integrations, Basis Technologies would strongly recommend some formal consulting services support in the implementation of the Jenkins integration, so that we can help you with the design and setup of the solution, and also advise you if any of your Requirements need development outside of the current out-of-the-box Integration offering.

If you are interested in finding out more about the ActiveControl / Jenkins capability, please reach out to your Basis Technologies Account Manager and they can help to arrange a demo.