Integrations (Direct)

GitLab — Last update: 31 October 2022

Basis Technologies

Table of Contents

1. Document Purpose

Introduction

ActiveControl includes an out-of-the-box Integration Framework which has enabled bi-directional integration capabilities with ITSM products such as JIRA, ServiceNow and HPSM for many years.

More recently, this Integration Framework has evolved to also be used to faciliate Customer requirements to integrate ActiveControl with DevOps and Automated Testing products.

Gitlab Integration as part of a CI/CD pipeline

During early 2018, the ActiveControl Integration Framework capability was extended to include an integration with GitLab, a third-party DevOps product used increasingly within IT organisations to automate aspects of software development building, testing and deployment. This integration was built on the back of a specific requirement for a new Customer wanting to to evolve their continuous integration and continuous delivery (CI/CD) capabilities in delivering SAP change as part of an ActiveControl workflow (along with non-SAP change delivered outside of ActiveControl).

Gitlab integration to trigger Automated Testing

During early 2020, a Jenkins integration (to trigger automated testing in Selenium) was created as a proof of concept by Basis Technologies. Later in the same year, the triggering of automated testing in Tosca via GitLab as part of an ActiveControl workflow was implemented as part of another new Customer requirement. The rest of this Integration Guide summarises this Gitlab integration capabiity. It should be noted that this document (and Basis Technologies integration solution) is testing tool agnostic; ie the same Gitlab integration could potentially be used to trigger automated testing within different 3rd Party testing products and not just Tosca.

Document Audience

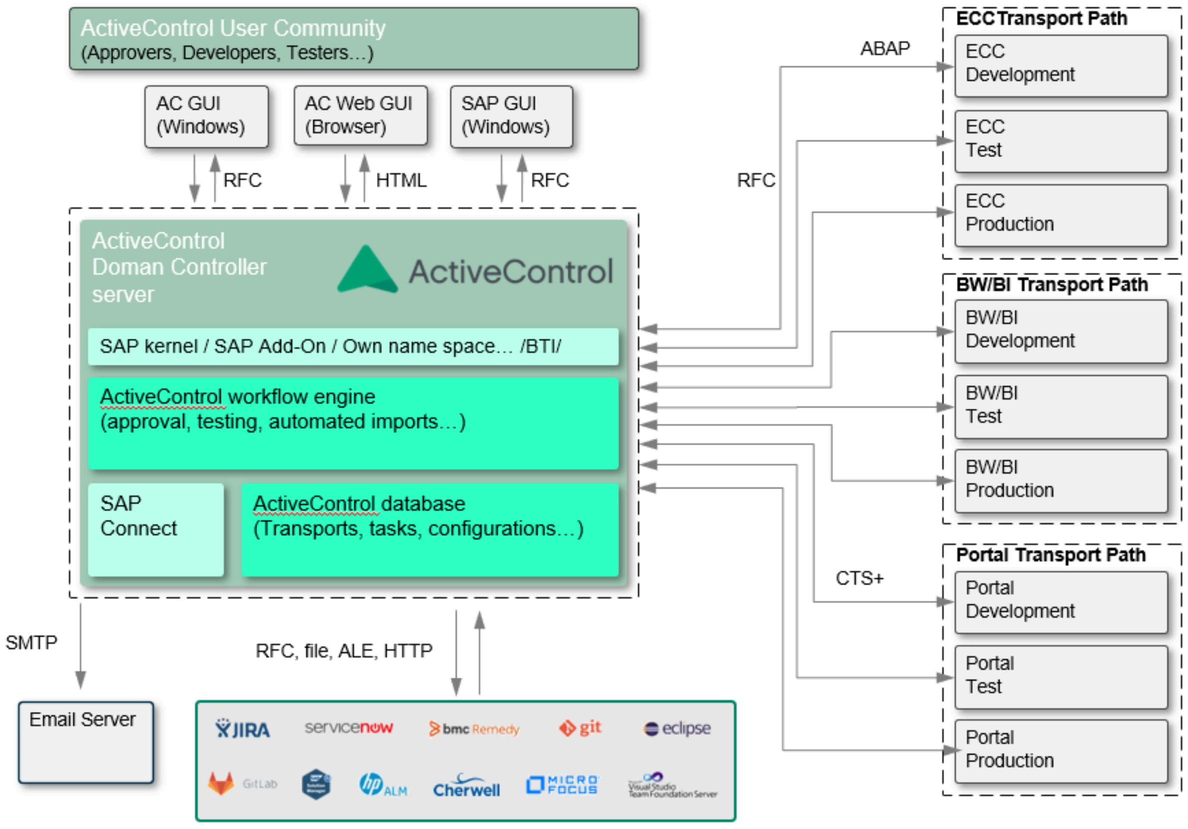
The intended audience for this document are the technical teams looking to implement integration between ActiveControl and GitLab. It does not detail how ActiveControl can be configured to manage the change process and it assumes a reasonable knowledge of standard change processes with SAP, and also an existing working knowledge of ActiveControl administration and configuration.

! The ActiveControl Gitlab integration is currently not an out-of-the-box ‘plug it and play’ Integration in the sense of many of the more established and mature ActiveControl integrations with ITSM tools such as JIRA or ServiceNow. It should be anticipated that some sort of consulting – and possible even development – services will be required to setup the integration, both on the Basis Technologies side and also Customer technology team. Technical expertise on GitLab will also be required on the Customer side for the installation and setup of GitLab. Basis Technologies responsibilities lie in sending the relevant paylod from ActiveControl to Gitlab as an Outbound integration, and in processing the results sent back to ActiveControl as an inbound integration. The Customer technologies team are responsible for processing this information from GitLab out to the automated testing tool , and sending back results (ie pass or fail) to ActiveControl as an inbound integration.

2. ActiveControl Integration Framework

ActiveControl Domain Controller

The architecture of ActiveControl can be broken down into: client software, a controlling SAP system, other participating SAP systems and integration systems. The diagram below illustrates the central role of the controlling SAP system – referred to as the ActiveControl “domain controller”.



The majority of ActiveControl (and Integration) configuration is done within the Domain Controller system. Unless specifically mentioned, all SAP setup detailed in this GitLab Integration Guide is done in the ActiveControl Domain Controller SAP system.

More information on the ActiveControl Domain Controller concept can be found in the main ActiveControl Administrator Guide documentation available at https://docs.basistechnologies.com/.

The Integration Framework Architecture

The ActiveControl Integration Framework is divided between inbound and outbound processes.

- For outbound calls there is a configurable framework that includes data extraction, transformation, mapping and sending routines, alongside error detection, correction and reporting.

- For inbound calls, those made by a third party system into ActiveControl, a number of web services are exposed allowing the external system to manipulate ActiveControl objects. Calls to ActiveControl web services will return appropriate error messages, but expect the calling system to deal with queuing, service levels and retries for failed integration transactions. The inbound solution for the GitLab integration specifically, is detailed later in this Integration Guide.

3. GitLab Integration

Example GitLab integration

The below workflow diagram describes a potential GitLab Integration scenario as part of an overall ActiveControl workflow. A JIRA integration is also depicted in this example.

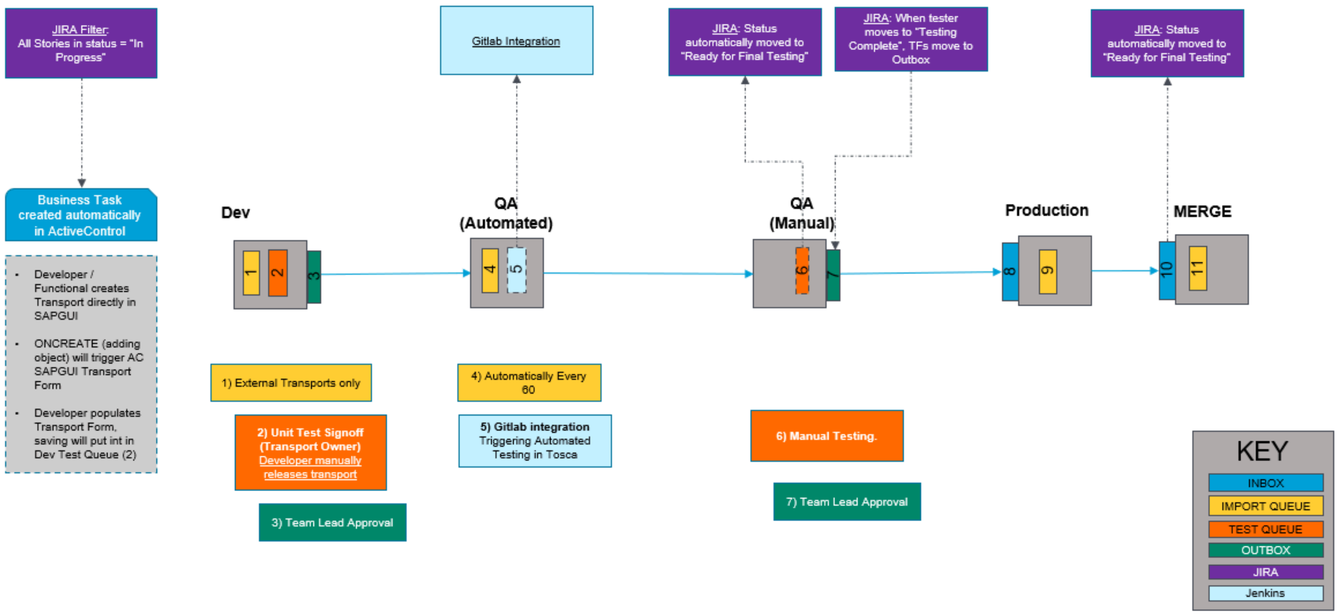

JIRA Filter: All Stories in status = "In Progress"
Gitlab Integration
JIRA: Status automatically moved to "Ready for Final Testing"
JIRA: When tester moves to "Testing Complete", TFs move to Outbox
JIRA: Status automatically moved to "Ready for Final Testing"
Business Task created automatically in ActiveControl
Developer / Functional creates Transport directly in SAPGUI
ONCREATE (adding object) will trigger AC SAPGUI Transport Form
Developer populates Transport Form, saving will put int in Dev Test Queue (2)
Dev
QA (Automated)
QA (Manual)
Production
MERGE
1
2
3
4
5
6
7
8
9
10
11
1) External Transports only
2) Unit Test Signoff (Transport Owner) Developer manually releases transport
3) Team Lead Approval
4) Automatically Every 60
5) Gitlab integration Triggering Automated Testing in Tosca
6) Manual Testing.
7) Team Lead Approval
KEY
INBOX
IMPORT QUEUE
TEST QUEUE
OUTBOX
JIRA
Jenkins

Figure: Potential Gitlab (and JIRA) integration reflected as part of an ActiveControl workflow

Key Points of a bi-directional GitLab integration

Outbound Integration

i) When transports land in a (configurable) Control Point location within the ActiveControl workflow, the outbound integration is triggered.

ii) A payload is sent by ActiveControl to GitLab, containing all relevant field information relating to the associated Business Tasks and Transport Forms. This would likely include Custom Field information, where the Transport Form owner (ie Developer or Functional) or other such SAP resource could record the relevant automated test scripts to be run.

iii) ActiveControl 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 running).

iv) On receiving the payload, Gitlab would process and trigger the relevant automated testing within the

Customers testing tool.

Inbound Integration

i) On completion of automated testing cycle, GitLab would receive pass/fail information back from the automated testing tool, and send this back to ActiveControl.

ii) ActiveControl would process this pass/fail information, and move the associated Business Tasks / Transport Forms forward to the next location in the workflow (in the event of successful automated testing).

iii) ActiveControl would automatically unlock the Import Queue, so that subsequent transports/changes can again be imported to the SAP system.

4. Integration Components

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

1.	Summary	Notes
1	Remote Function Calls	None
2	System Users	Refer to here
3	Number Range	Refer to here
4	Configuration Tables	Refer to here
5	Application Tables	Yes
6	Programs	Refer to here
7	APIs	None required for Gitlab integration
8	User Exit	None required for Gitlab integration
9	Jobs	Yes
10	Error Logging	Refer to here (Outbound logging only)
11	General Configuration	Refer to here
12	Gitlab setup	Refer here

4.1. Pre-Requisites

The following are pre-requisites for setting up the ActiveControl integration with Gitlab.

It should be noted that these setup activities are the responsibility of the Customer team; Basis Technologies do not have expertise in GitLab.

1. Install GIT

A pre-requisite for using the ActiveControl/GitLab integration is to have GIT installed within the ActiveControl Domain Controller SAP system.

A good online source of information can be found here.

2. Setup Remote Repository

A Remote Repository is required to be setup within GitLab, for ActiveControl.

This repository should be given a name such as “ActiveControl” or “ActiveControl_ECC” – depending on the exact customer requirements for the Integration.

3. GitLab pipeline

GitLab pipeline should be setup for the ActiveControl remote repository created in the previous step.

4. Clone Remote Repository

The ActiveControl remote repository within Gitlab should be cloned in the local git installation. This will create the folder where ActiveControl will write the outbound information (ie Business Task change file)

h3. 5. Define the scripts used via transaction SM49 in the ActiveControl Domain Controller SAP system

gitswitch.sh – switches branches
gitpush.sh – pushed changes into the remote repository
gitindir.sh – used for troubleshooting

(these will be provided by Basis Technologies)

6. Test Connections

After performing the previous steps to setup Git, perform Add, Commit and Push of initial AC_BT_Change.json file (this will be provided by Basis Technologies).

This will This will test out the connection between the ActiveControl Domain Controller SAP system and GitLab.

4.2. Outbound Integration

4.2.1. SAP User

A SAP user is required to support the Gitlab 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 seperate System user for the Gitlab integration to help differentiate events relating to the Gitlab Integration from other Integrations (eg Jira or ServiceNow), or other events such as Scheduled Imports within ActiveControl.

If creating a seperate User, it should have the same authorisations as the AC_Batch user:

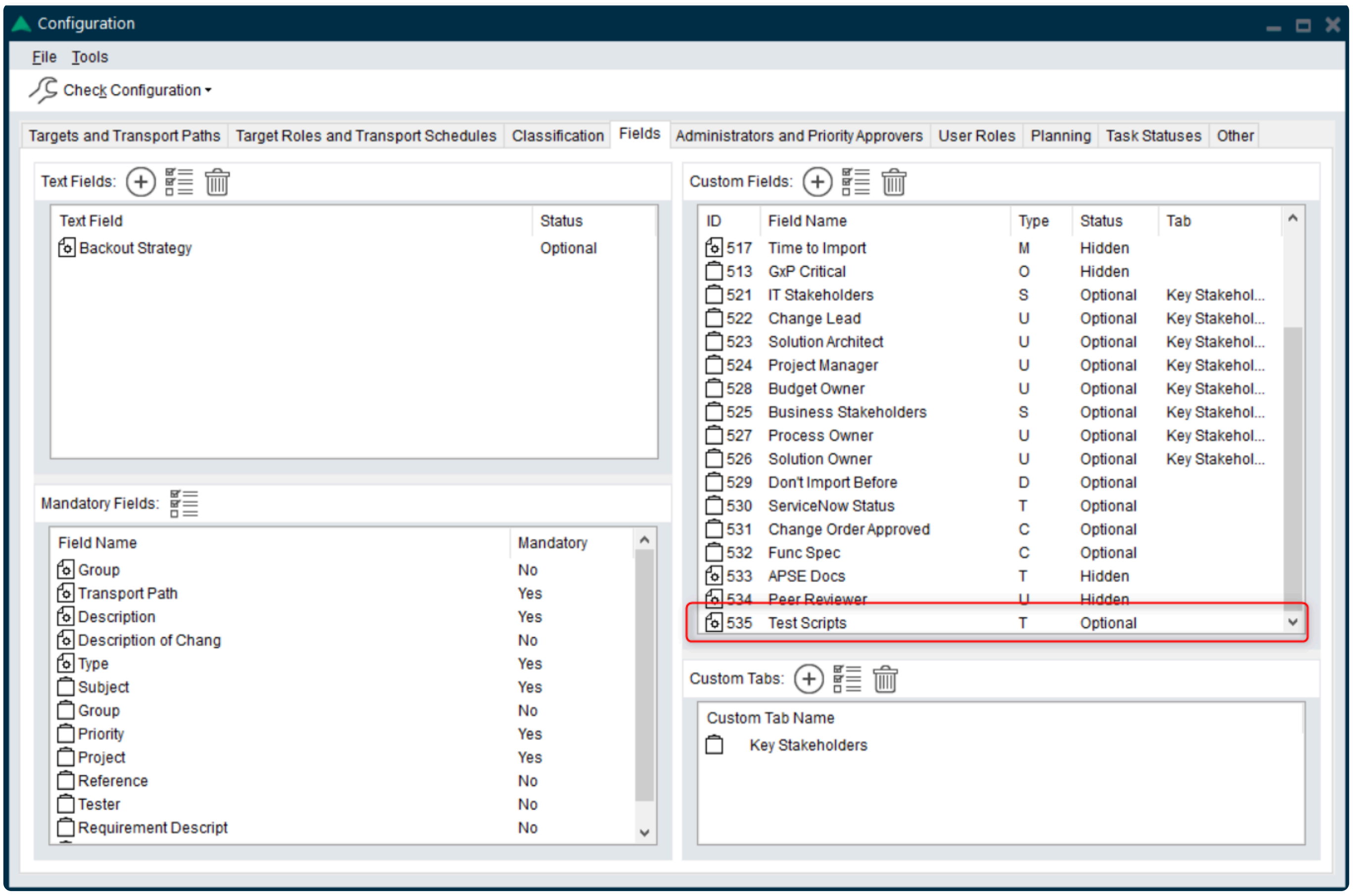
- /BTI/TE:CTS_ADMIN_USER
- /BTI/TE:CTS_RFC
- /BTI/TE:COMP_ADMIN_ROLE

4.2.2. ActiveControl General Configuration

4.2.2.1. Custom Fields

The ActiveControl / Gitlab integration for automated testing relies on GitLab 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.



Text Field	Status
Backout Strategy	Optional

Field Name	Mandatory
Group	No
Transport Path	Yes
Description	Yes
Description of Chang	No
Type	Yes
Subject	Yes
Group	No
Priority	Yes
Project	Yes
Reference	No
Tester	No
Requirement Descript	No

ID	Field Name	Type	Status	Tab
517	Time to Import	M	Hidden	
513	GxP Critical	O	Hidden	
521	IT Stakeholders	S	Optional	Key Stakehol...
522	Change Lead	U	Optional	Key Stakehol...
523	Solution Architect	U	Optional	Key Stakehol...
524	Project Manager	U	Optional	Key Stakehol...
528	Budget Owner	U	Optional	Key Stakehol...
525	Business Stakeholders	S	Optional	Key Stakehol...
527	Process Owner	U	Optional	Key Stakehol...
526	Solution Owner	U	Optional	Key Stakehol...
529	Don't Import Before	D	Optional	
530	ServiceNow Status	T	Optional	
531	Change Order Approved	C	Optional	
532	Func Spec	C	Optional	
533	APSE Docs	T	Hidden	
534	Peer Reviewer	U	Hidden	
535	Test Scripts	T	Optional	

Custom Tab Name
Key Stakeholders

Figure: Custom Field creation in the ActiveControl Windows GUI

The information stored in these custom field(s) information is passed over to GitLab as part of the outbound Integration (along with the Business Task and Transport Form information) – and the Customer Gitlab Administrator would then need to pass this information over from GitLab to the 3rd Party test tool to trigger the associated automated testing.

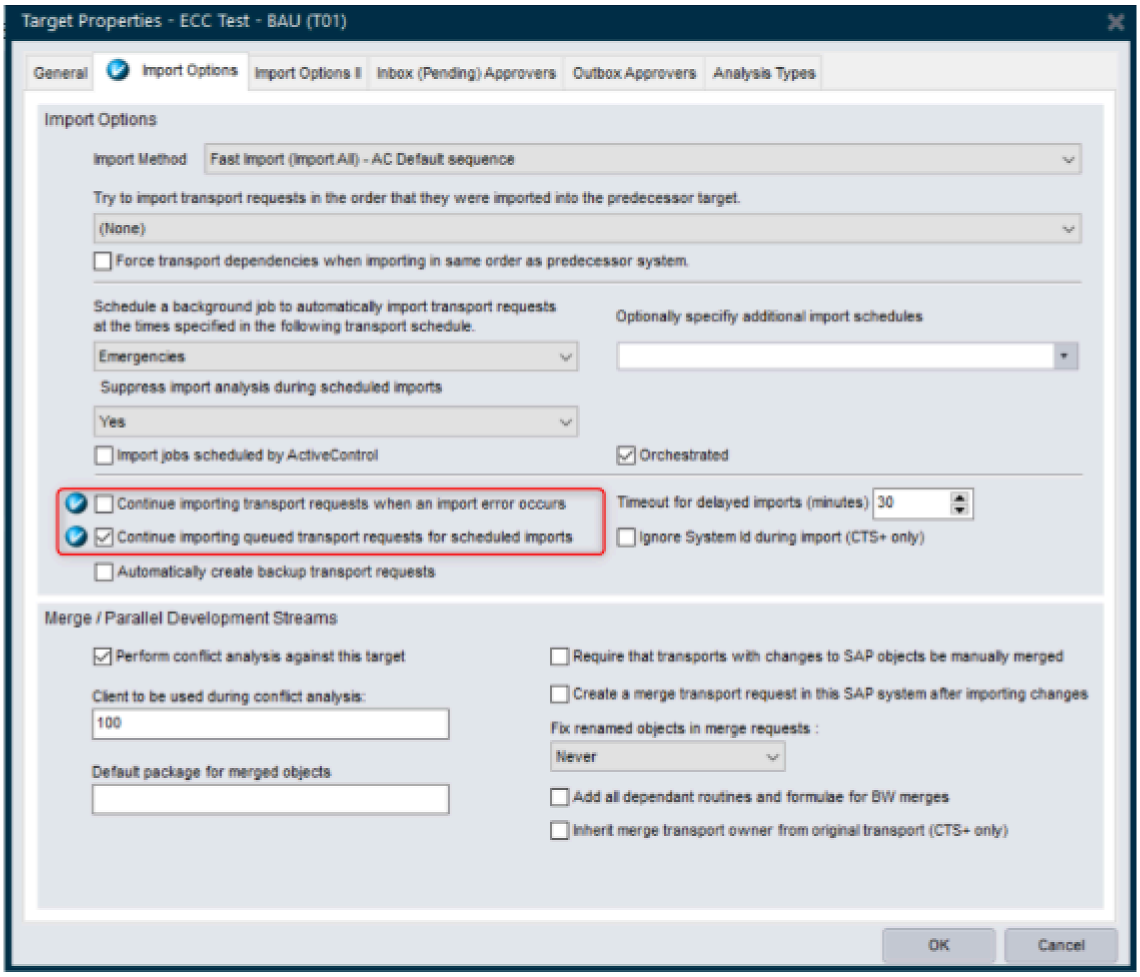
4.2.2.2. Import Options

As the 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)

Figure: Recommended Import target configuration within ActiveControl to support an automated testing Integration.

4.2.3. SAP configuration tables

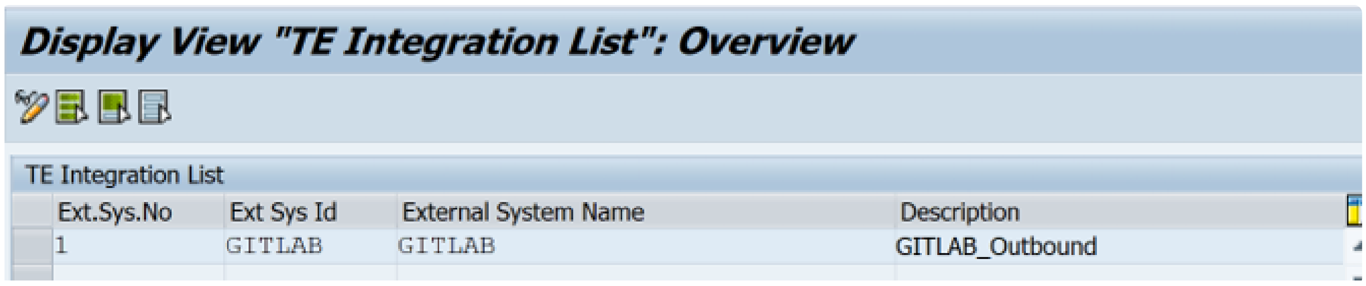
4.2.3.1. /BTI/TE_INT_SYST

Table /BTI/TE_INT_SYST 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 Gitlab Integration.
TASKFIELD_LINK	Not required for Gitlab Integration.
A FORMFIELD_LINK	Not required for Gitlab Integration.
INT_USER	Not required for Gitlab Integration.
INT_PASSWORD	Not required for Gitlab Integration.

Example configuration


Display View "TE Integration List": Overview

TE Integration List

Ext.Sys.No	Ext Sys Id	External System Name	Description
1	GITLAB	GITLAB	GITLAB_Outbound

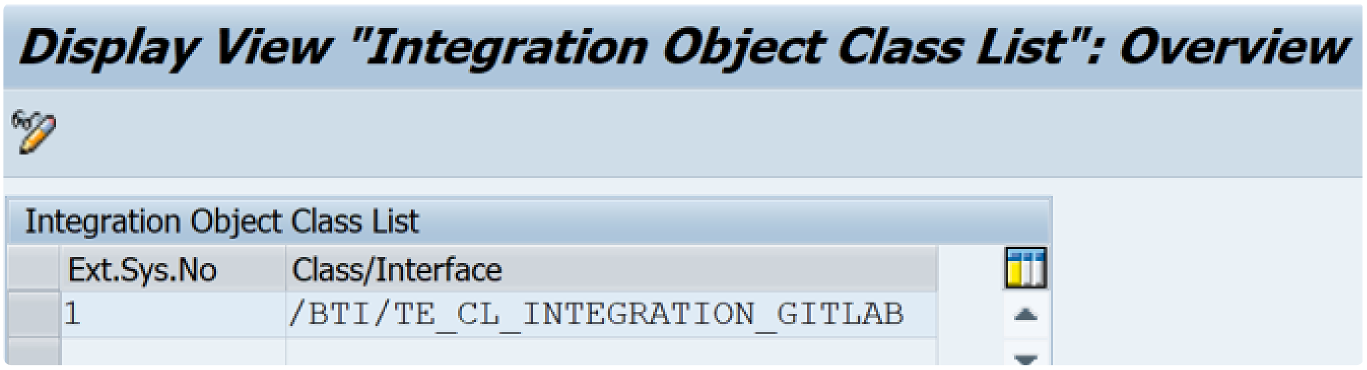
4.2.3.2. /BTI/TE_INT_CLAS

Table /BTI/TE_INT_CLAS 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.

Field	Explanation of Field
EXTSYS_NO	Integration system number (as configured in /BTI/TE_INT_SYST
CLASSNAME	Integration works on the principle of having a class for each external system that we need to integrate with.

Example configuration


Display View "Integration Object Class List": Overview

Integration Object Class List

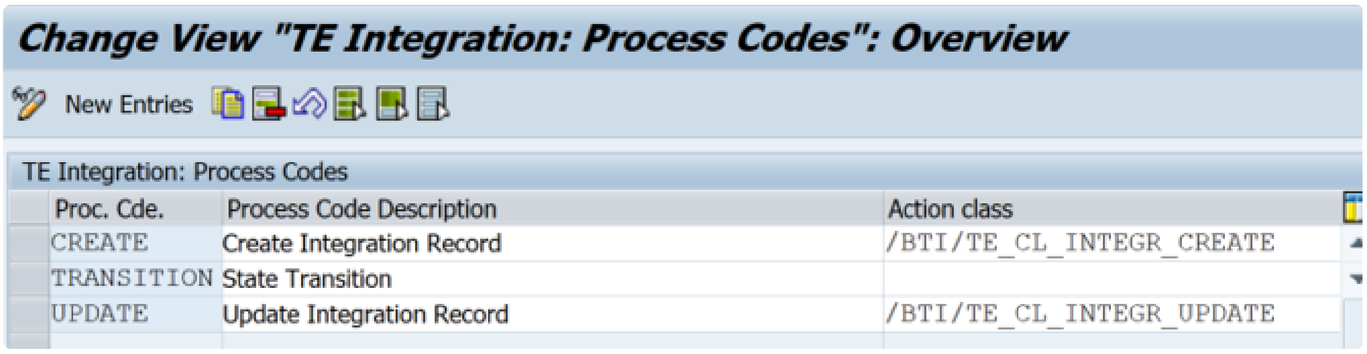
Ext.Sys.No	Class/Interface
1	/BTI/TE_CL_INTEGRATION_GITLAB

4.2.3.3. /BTI/TE_INT_PC

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

Field	Description
PROCESS_CODE	The process codes used by the integration framework to perform some kind of action. The framework gets shipped with two standard process codes CREATE and UPDATE.
CODE_DESCRIPTION	Description of above code.

Example configuration



TE Integration: Process Codes

Proc. Cde.	Process Code Description	Action class
CREATE	Create Integration Record	/BTI/TE_CL_INTEGR_CREATE
TRANSITION	State Transition	
UPDATE	Update Integration Record	/BTI/TE_CL_INTEGR_UPDATE

As part of Gitlab integration, only CREATE process code is used / required.

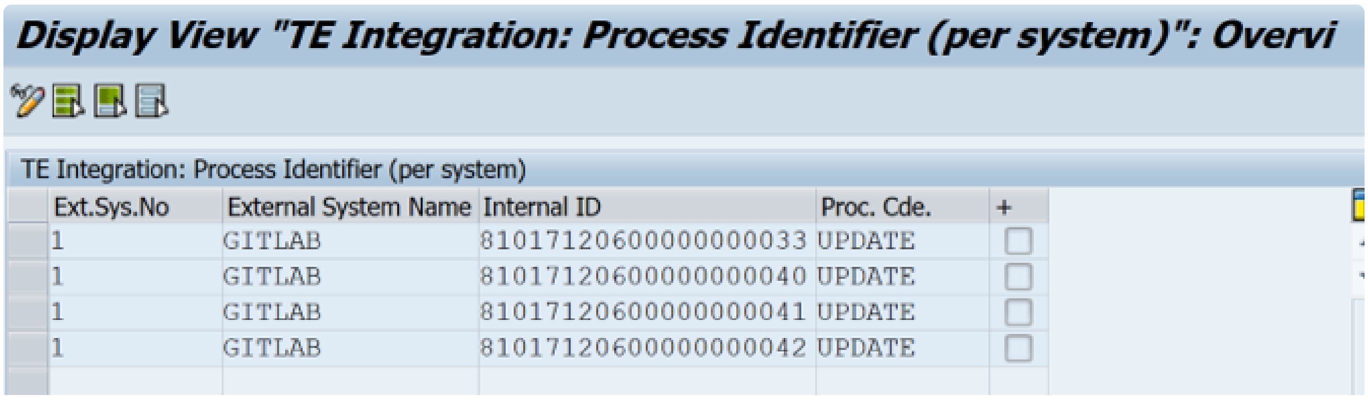
4.2.3.4. /BTI/TE_INT_PROC

Table /BTI/TE_INT_PROC is used within the Integration Framework to define the Process Identifiers that are used within the Integration.

Field	Description
EXTSYS_NO	Integration system number (as configured in /BTI/TE_INT_SYST
EXTSYS_NAME	Full description of external system
IDENTIFIER	This identifier is the crux of the integration framework and denotes a point of integration, more than likely this would be some kind of internal id, in our OOTB example it is a task status. This point of integration is attached to a process code denoted above and this is what would cause an integration to be performed when this identifier is reached.
PROCESS_CODE	The process codes used by the integration framework to perform some kind of action. The framework gets shipped with two standard process codes CREATE and UPDATE.
IGNORE_CHANGES	This flag is set when you wish to ignore previous changes in case the integrated object has skipped through more than one integration point since the integration trigger program was last run.

As part of GitLab 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.

Example configuration


Display View "TE Integration: Process Identifier (per system)": Overvi

TE Integration: Process Identifier (per system)

Ext.Sys.No	External System Name	Internal ID	Proc. Cde.	+
1	GITLAB	8101712060000000033	UPDATE	☐
1	GITLAB	8101712060000000040	UPDATE	☐
1	GITLAB	8101712060000000041	UPDATE	☐
1	GITLAB	8101712060000000042	UPDATE	☐

4.2.3.5. /BTI/TE_INT_MAPP

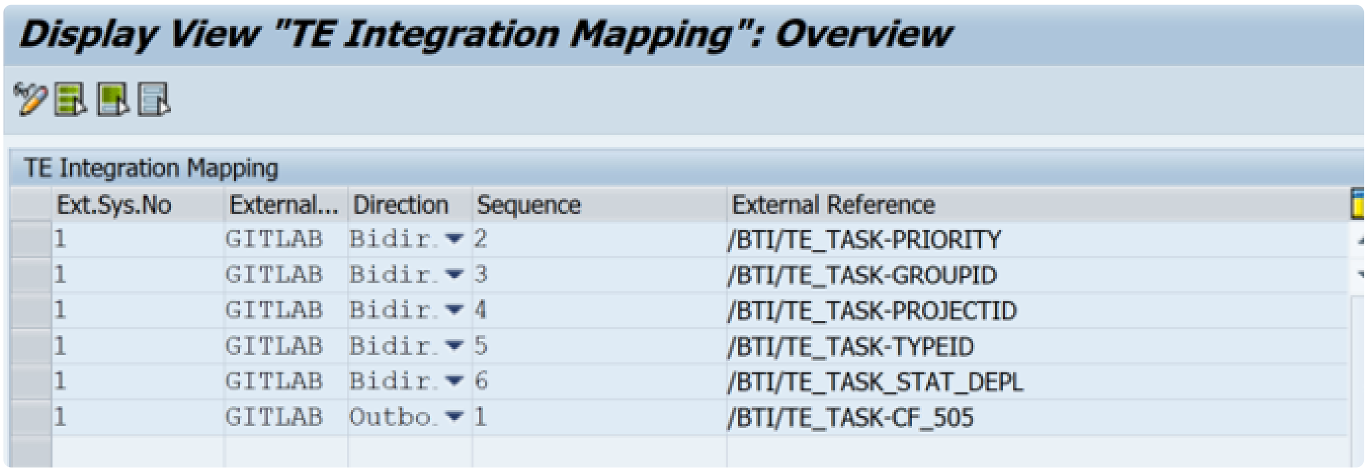
An essential part of the integration framework is mapping ActiveControl fields to the equivalent fields on any external system.

This is achieved using the table '/BTI/TE_INT_MAPP'.

Ideally, this process will need to be undertaken before the framework can be used. For general fields the ActiveControl field should be entered complete with table name into field TEFIELDREF and the external fieldname must be entered in the EXTERNAL_REF field. There is also the functionality to be able to reference any ActiveControl Custom fields the custom field ID's would need to be added to TECUSTFIELD_REF, also multiple line itemed fields are able to be handled here such as text fields. Finally, on the mapping table there is a KEY_FIELD field this is used to hold the external system record key in general use a specific non display custom field on the task would be created for this purpose.

Field	Description
EXTSYS_NO	Integration system number (as configured in /BTI/TE_INT_SYST
EXTSYS_NAME	Full description of external system
TEFIELDREF	This is the AC Field that needs to be mapped to a field on the external system. This table name is required in the field as well. I.e. /BTI/TE_TASK-PRIORITY
EXTERNAL_REF	This is the fieldname that the frameworks calling web service needs to reference to map across the data.
KEY_FIELD	This field is the link between the AC record, in our task record we have set up a custom field which is hidden from view and in here we store the ID of the created record on the integrated system.
TECUSTFLD_REF	ID of AC Custom field to be mapped.
DEFAULT_VAL	Defaulted Value to be mapped over to the integrated system field.

Example Configuration



Display View "TE Integration Mapping": Overview

TE Integration Mapping

Ext.Sys.No	External...	Direction	Sequence	External Reference
1	GITLAB	Bidir.	2	/BTI/TE_TASK-PRIORITY
1	GITLAB	Bidir.	3	/BTI/TE_TASK-GROUPID
1	GITLAB	Bidir.	4	/BTI/TE_TASK-PROJECTID
1	GITLAB	Bidir.	5	/BTI/TE_TASK-TYPEID
1	GITLAB	Bidir.	6	/BTI/TE_TASK_STAT_DEPL
1	GITLAB	Outbo.	1	/BTI/TE_TASK-CF_505

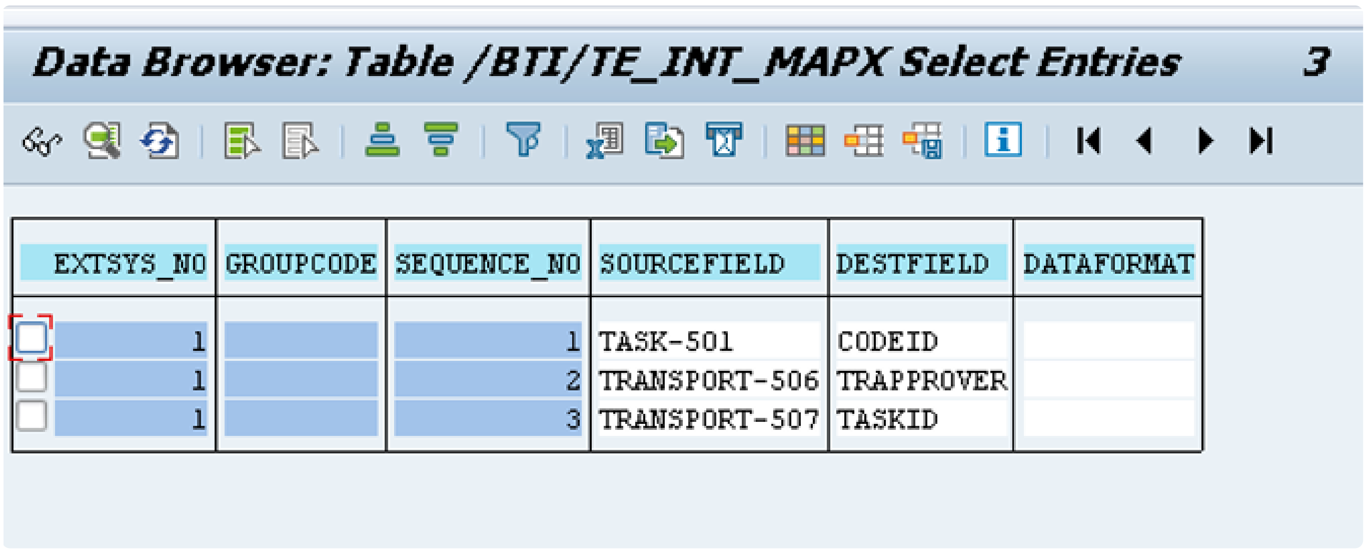
4.2.3.6. /BTI/TE_INT_MAPX

Fields to be mapped for some json fields have to be set in table /BTI/TE_INT_MAPX in the ActiveControl Domain Controller SAP system

/BTI/TE_INT_MAPX can be used to send over Custom Field numbers to Gitlab, which may be needed for processing on the receiving end, as part of CI/CD pipeline or automated testing activities.

Field	Description
EXTSYS_NO	Integration system number (as configured in /BTI/TE_INT_SYST
GROUPCODE	
SEQUENCE_NO	
SOURCEFIELD	ActiveControl field number (from ActiveControl Windows GUI
DESTFIELD	Destination field, name of the field on the receiving end. For the Gitlab integration – the following are the available DESTFIELD values: ‘ReleaseNum’ ‘CodeID’ ‘Commit_Approval’ ‘Commit_Approver’ ‘Unit_Test_Count’ ‘Unit_Test_Pass’ ‘Unit_Test_Fail’ ‘Transports’ ‘TRnum’ ‘TRtype’ ‘TRowner’ ‘TRapprover’ ‘taskID’
DATAFORMAT	

h3 Example Configuration


Data Browser: Table /BTI/TE_INT_MAPX Select Entries 3

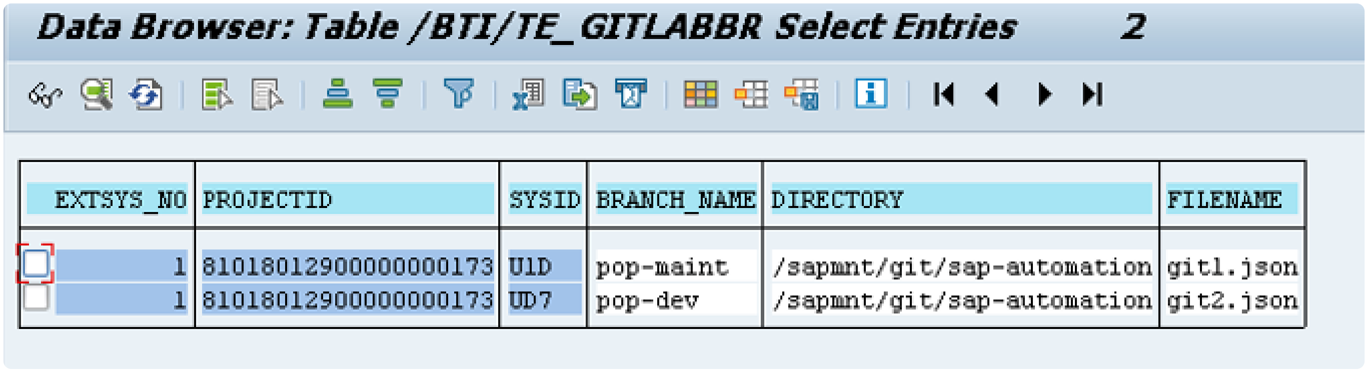
EXTSYS_NO	GROUPCODE	SEQUENCE_NO	SOURCEFIELD	DESTFIELD	DATAFORMAT
1		1	TASK-501	CODEID	
1		2	TRANSPORT-506	TRAPPROVER	
1		3	TRANSPORT-507	TASKID	

4.2.3.7. /BTI/TE_GITLABBR

GitLab Branch Repositories are configured in table /BTI/TE_GITLABBR

Field	Description
EXTSYS_NO	Integration System Number [as defined in /BTI/TE_INT_SYST]
ProjectID	Is the ActiveControl project ID assigned to the Business Task
SYSID	is the origin of the transports (At this stage we assume all the transports in a task come from the same source)
BRANCH_NAME	is the branch to commit to
Directory	is the repository location
FILENAME	is the name of the json file that will be created by the integration

Example Configuration

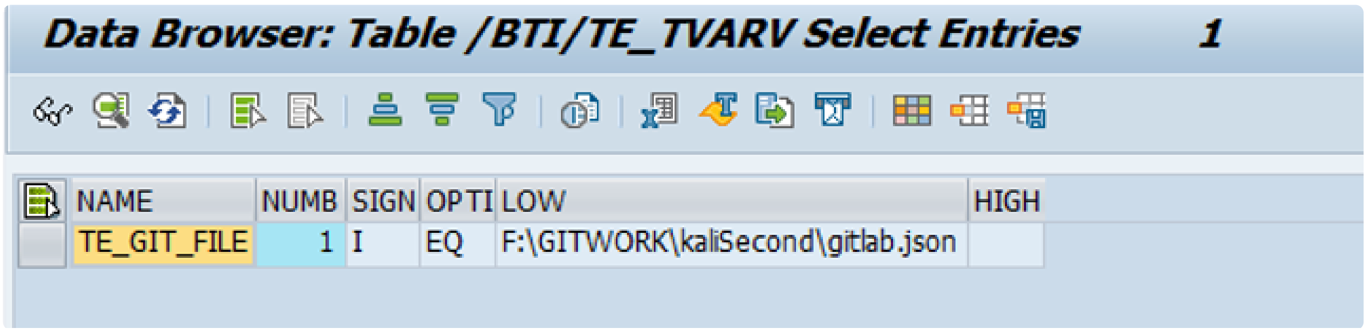


Data Browser: Table /BTI/TE_GITLABBR Select Entries 2

EXTSYS_NO	PROJECTID	SYSID	BRANCH_NAME	DIRECTORY	FILENAME
1	81018012900000000173	U1D	pop-maint	/sapmnt/git/sap-automation	git1.json
1	81018012900000000173	UD7	pop-dev	/sapmnt/git/sap-automation	git2.json

4.2.3.8. /BTI/TE_TVARV

Clone or pull the SAP GitLab Repository to the SAP application server and configure the file path(the file containing peer review information to be pushed in gitlab repo) in table /BTI/TE_TVARV with variant variable TE_GIT_FILE


Data Browser: Table /BTI/TE_TVARV Select Entries 1

NAME	NUMB	SIGN	OPTI	LOW	HIGH
TE_GIT_FILE	1	I	EQ	F:\GITWORK\kaliSecond\gitlab.json	

The batch template attached for windows/Linux that contains commands to call git to push file into gitlab repository has to be uploaded into SAP application server based on the OS of machine after updating the drive information in the template.

Push2Git_Windows.bat Push2Git_Linux.sh

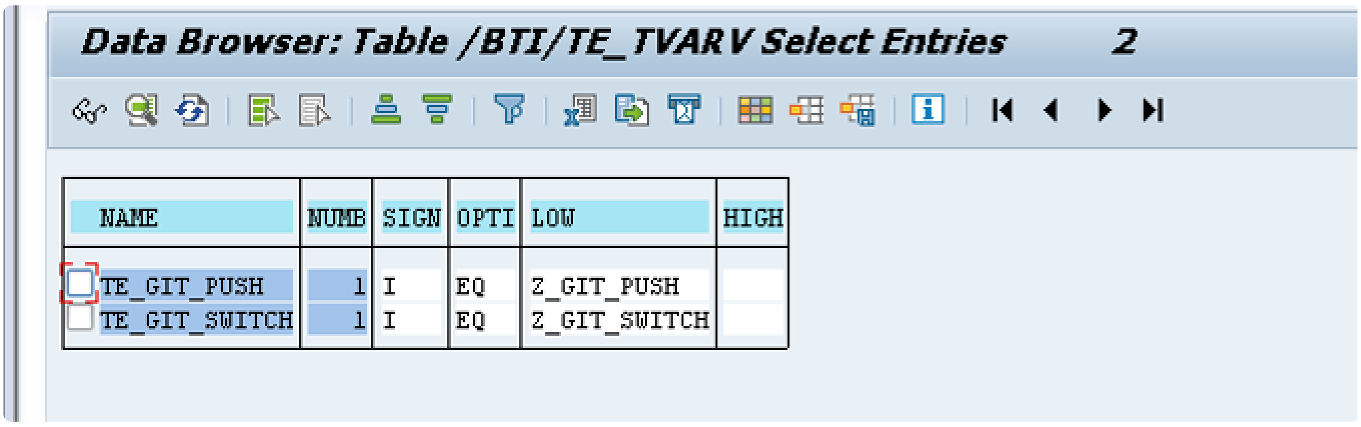
Two separate scripts will be used to switch branch and commit, identified by 2 keys in /BTI/TE_TVARV, These refer to OS commands defined with transaction SM69.Syntax is as follows:

Z_GIT_PUSH
Z_GIT_SWITCH

Sample scripts will be provided separately.

The batch file in the directory can be run by an external commands created in transaction SM69.

Example Configuration

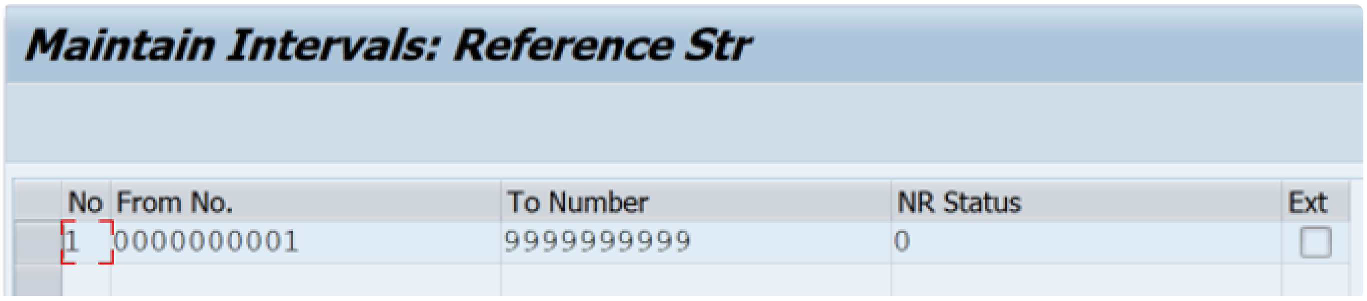

Data Browser: Table /BTI/TE_TVARV Select Entries 2

NAME	NUMB	SIGN	OPTI	LOW	HIGH
TE_GIT_PUSH	1	I	EQ	Z_GIT_PUSH	
TE_GIT_SWITCH	1	I	EQ	Z_GIT_SWITCH	

4.2.3.9. /BTI/TE_RF

Number range for object /BTI/TE_RF needs to be setup in the TE Domain Controller via SNRO for the Integration Framework to operate.

Example configuration



Maintain Intervals: Reference Str

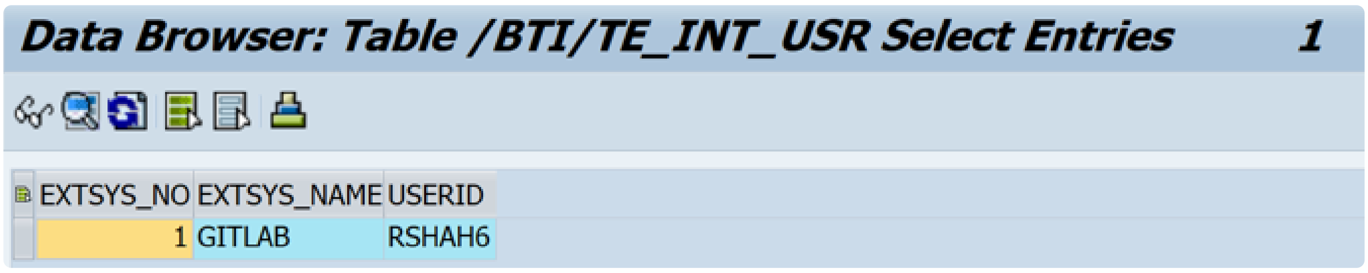
No	From No.	To Number	NR Status	Ext
1	0000000001	9999999999	0	☐

4.2.3.10. /BTI/TE_INT_USR

Within the ActiveControl integration framework, it is possible to set up ‘Notification Users’ per external system that can be notified when an integration message has gone into an error status.

This is run through the Email Notification Engine and the table that needs to be maintained is ‘/BTI/TE_INT_USR’.

Example configuration


Data Browser: Table /BTI/TE_INT_USR Select Entries 1

EXTSYS_NO	EXTSYS_NAME	USERID
1	GITLAB	RSHAH6

%(color-gray)Figure: Username of desired recipient of Integration notifications should be maintained in /BTI/TE_INT_USR

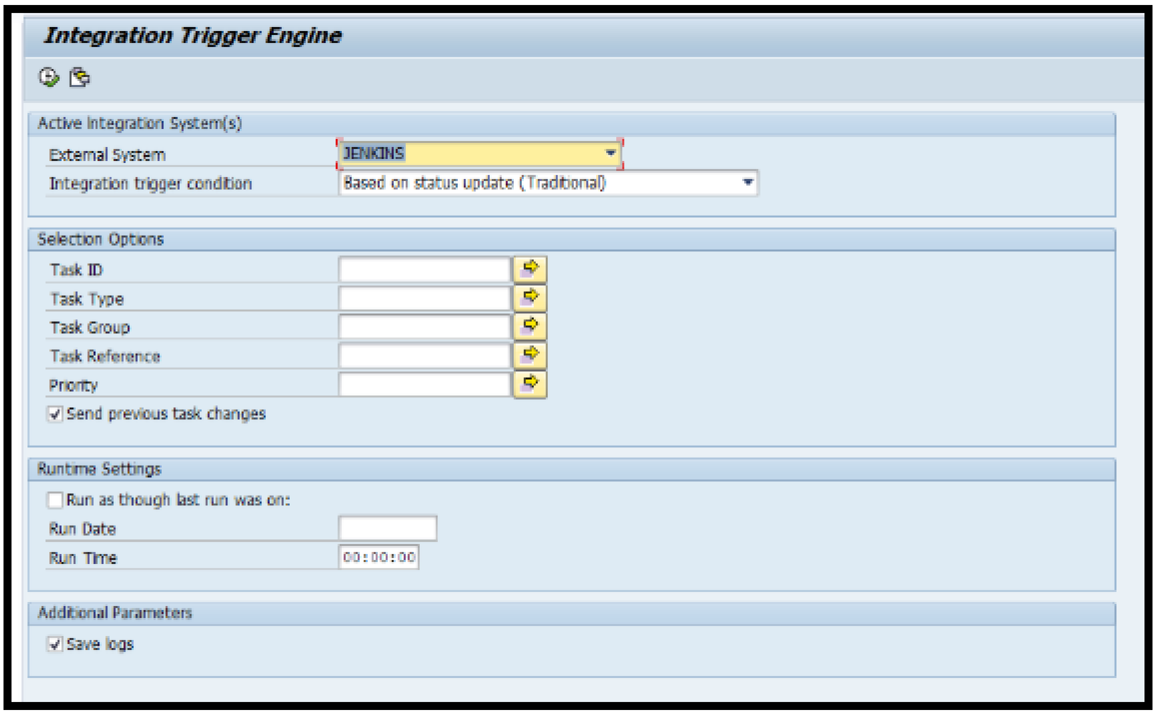
4.2.4. SAP Programs

4.2.4.1. /BTI/TE_INTEG_TRIGGER

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

This program would be scheduled as a variant to run every 5-10 minutes, to push the latest payload out to Gitlab.

Example Variant



Integration Trigger Engine

Active Integration System(s)

External System	JENKINS
Integration trigger condition	Based on status update (Traditional)

Selection Options

Task ID	
Task Type	
Task Group	
Task Reference	
Priority	

[✓] Send previous task changes

Runtime Settings

[ ] Run as though last run was on:

Run Date	
Run Time	00:00:00

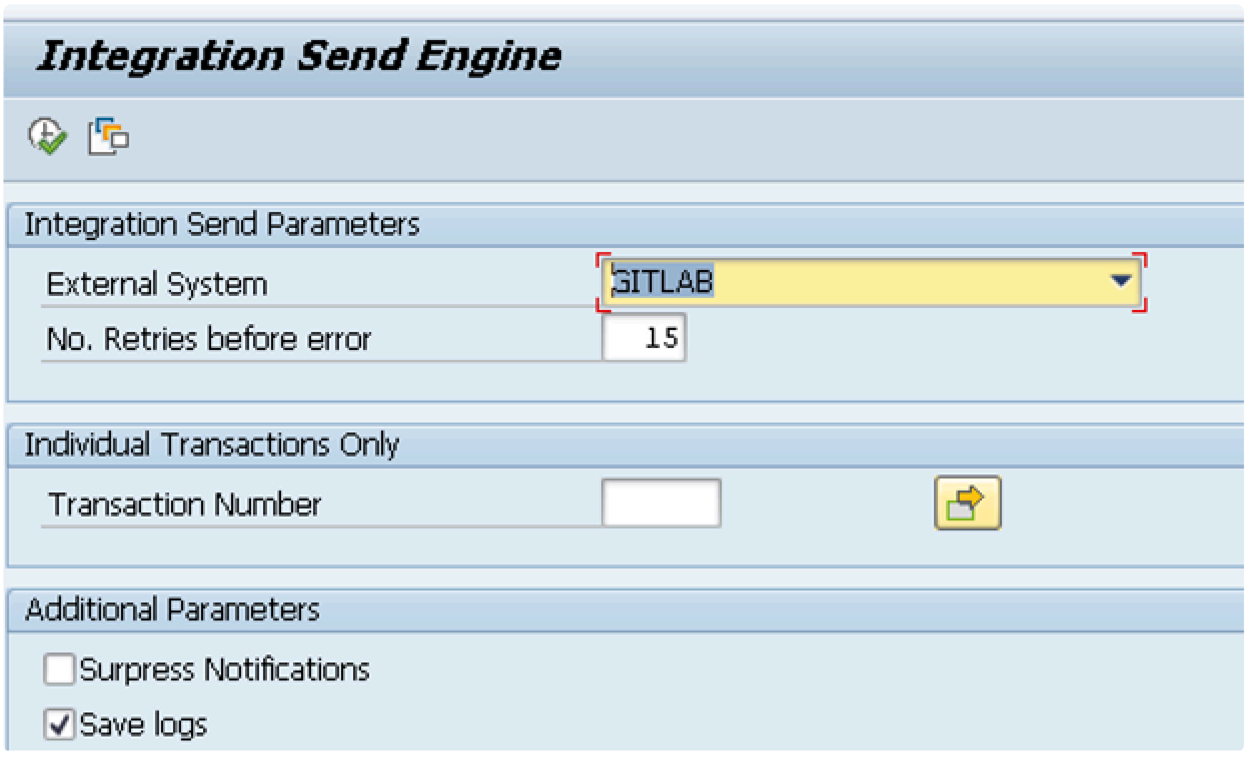
Additional Parameters

[✓] Save logs

4.2.4.2. /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.

Example variant



Integration Send Engine

Integration Send Parameters

External System	GITLAB
No. Retries before error	15

Individual Transactions Only

Transaction Number	

Additional Parameters

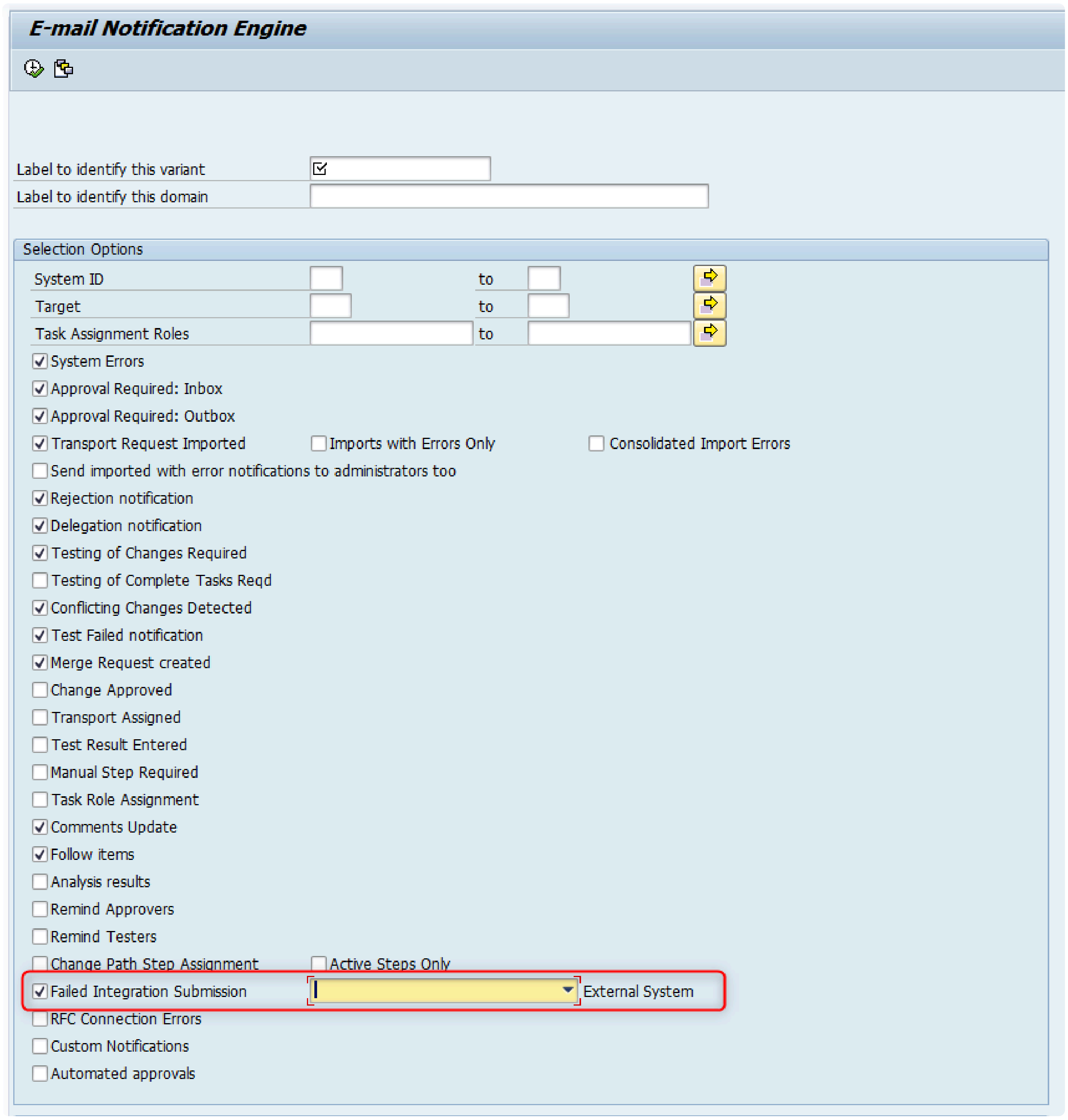
- [ ] Surpress Notifications
- [x] Save logs

4.2.4.3. /BTI/TE_RNOTIFICATION_ENGINE

ActiveControl includes a standard Notification Engine to notify the appropriate stakeholders at the appropriate time in the workflow.

These notifications are switched on/off via program /BTI/TE_RNOTIFICATION_ENGINE in the Domain Controller, and scheduled to run as a variant every 5-10 minutes.

As part of the Gitlab integration, the ‘Failed Integration Submission’ notification type should be switched on. If this is done, then all users configured in /BTI/TE_INT_USR will receive notifications of any failed integrations. (outbound integration only)



E-mail Notification Engine

Label to identify this variant	☑		
Label to identify this domain			

Selection Options

System ID		to	
Target		to	
Task Assignment Roles		to	

- ☑ System Errors
- ☑ Approval Required: Inbox
- ☑ Approval Required: Outbox
- ☑ Transport Request Imported ☐ Imports with Errors Only ☐ Consolidated Import Errors
- ☐ Send imported with error notifications to administrators too
- ☑ Rejection notification
- ☑ Delegation notification
- ☑ Testing of Changes Required
- ☐ Testing of Complete Tasks Reqd
- ☑ Conflicting Changes Detected
- ☑ Test Failed notification
- ☑ Merge Request created
- ☐ Change Approved
- ☐ Transport Assigned
- ☐ Test Result Entered
- ☐ Manual Step Required
- ☐ Task Role Assignment
- ☑ Comments Update
- ☑ Follow items
- ☐ Analysis results
- ☐ Remind Approvers
- ☐ Remind Testers
- ☐ Change Path Step Assignment ☐ Active Steps Only
- ☑ Failed Integration Submission [ ] External System
- ☐ RFC Connection Errors
- ☐ Custom Notifications
- ☐ Automated approvals

Figure: Standard Notification Engine integration type is used to alert defined users of any Integration errors during the Outbound integration

4.2.5. Integration Setup (GitLab)

The ActiveControl / GitLab integration outbound integration works by ActiveControl calling GitLab

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

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

Number	Event
1	ActiveControl 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 from ActiveControl to GitLab
3	GitLab initiating the required Tests within the automated testing tool.
4	GitLab receiving PASS or FAIL test result information from the automated testing tool, and sending this back to ActiveControl.
5	ActiveControl 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.

4.2.5.1. GitLab User

A system user is required in GitLab to support the integration

The nature of this User will depend on the requirements and final Integration solution.

4.3. Inbound Integration

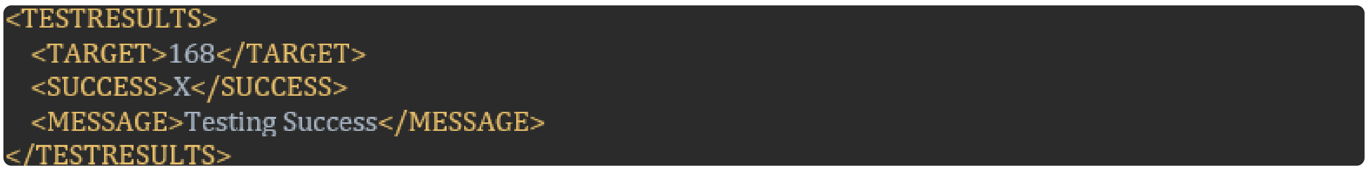
When the automated testing cycle completes, Gitlab will need to report back to ActiveControl whether the testing was a Pass or Fail.

Typically the following activities will happen as part of Gitlab automated testing Inbound integration:

1. GitLab pipeline gathers a single pass/fail result from all of the Tosca Execution Lists from all of the transport forms.
2. GitLab pipeline sends HTTP Post call to the ActiveControl server. a. https://docs.basistechnologies.com/integration-administration-guides/0.7/en/topic/save-business-task-result b. queryresultsfile – see information below c. attachement – this can be a .url file that contains the url to the detailed Tosca results described in step 2.

queryresultsfile example

Following screenshot shows an example of the queryresultsfile form data parameter. This can be string data and not an actual file.



```
<TESTRESULTS>
  <TARGET>168</TARGET>
  <SUCCESS>X</SUCCESS>
  <MESSAGE>Testing Success</MESSAGE>
</TESTRESULTS>
```

– can be found in /BTI/TE_TARG table in the ActiveControl server.
– X is the standard true value in SAP. If there was a failure, leave blank.
– The overall status message. This is saved in the Business Task Test Result.

4.4. ActiveControl HTTP API

4.4.1. Get Queue Contents

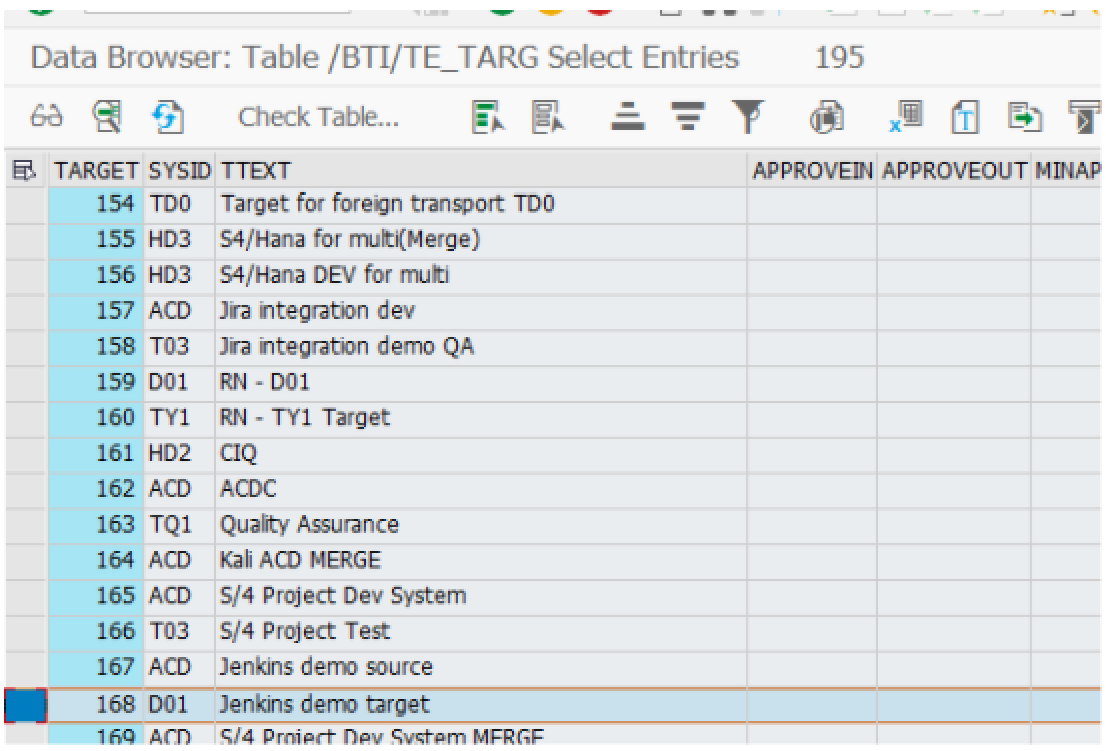
HTTP GET

Example URL:
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

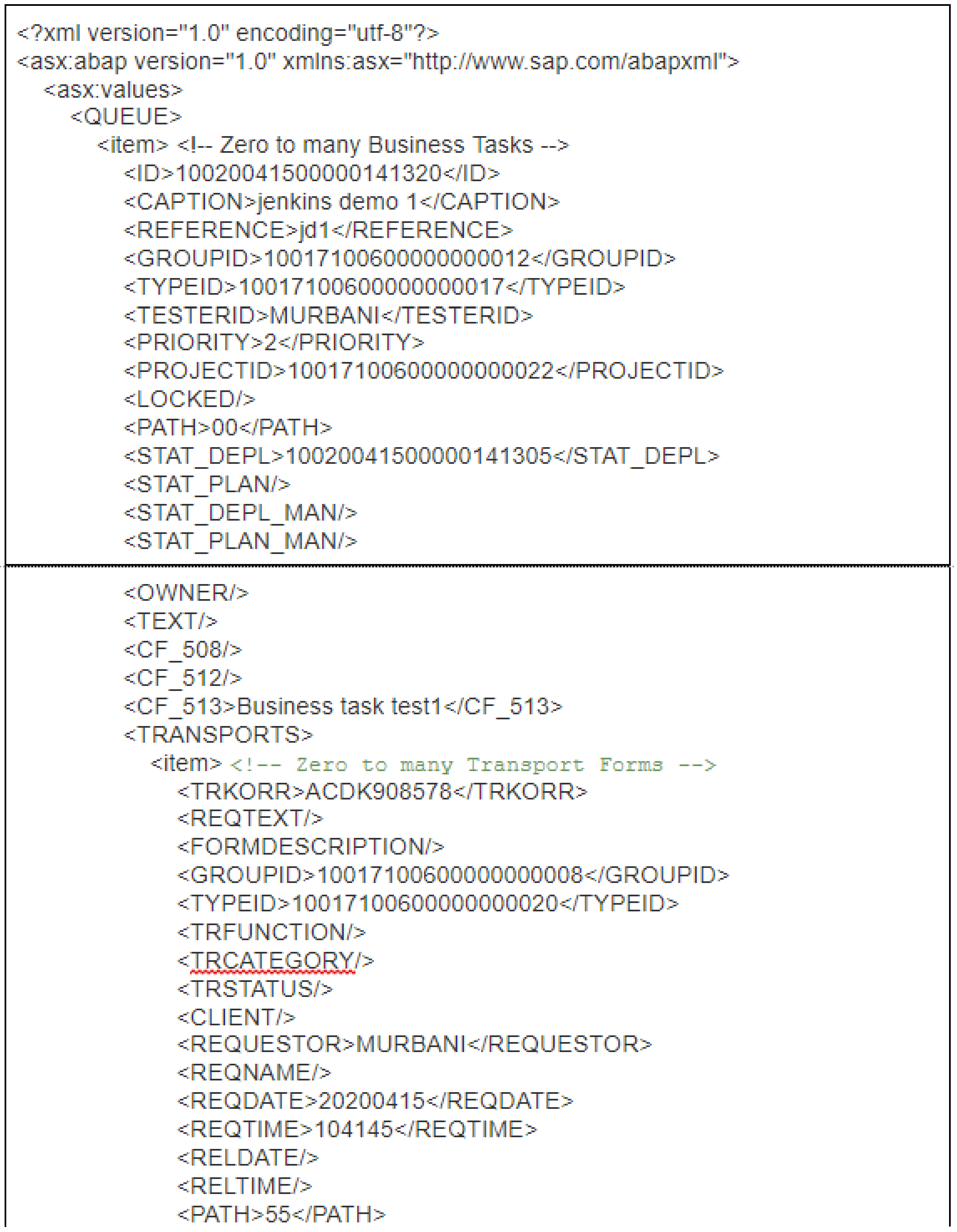


Data Browser: Table /BTI/TE_TARG Select Entries 195

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

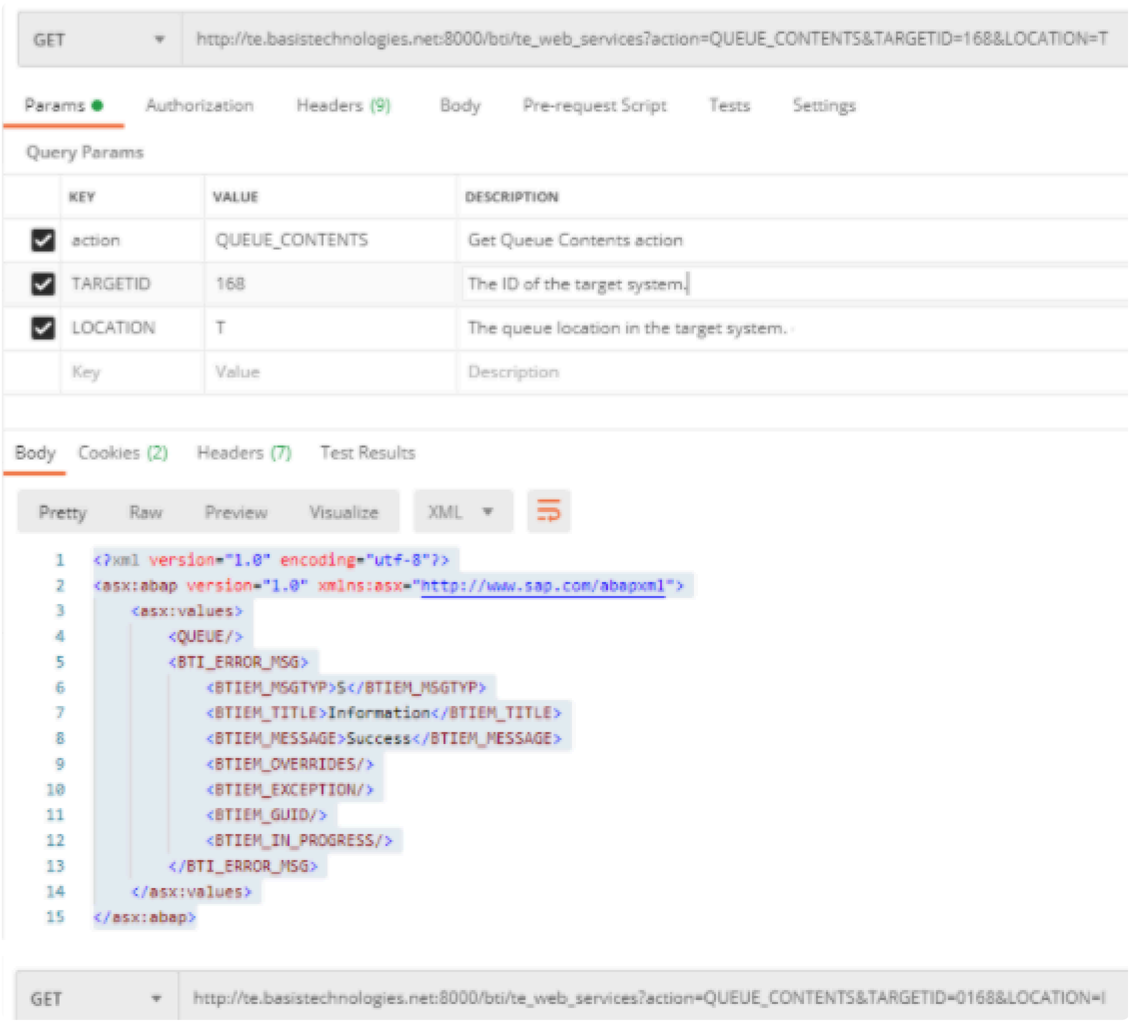
XML Payload



```xml
<?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

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

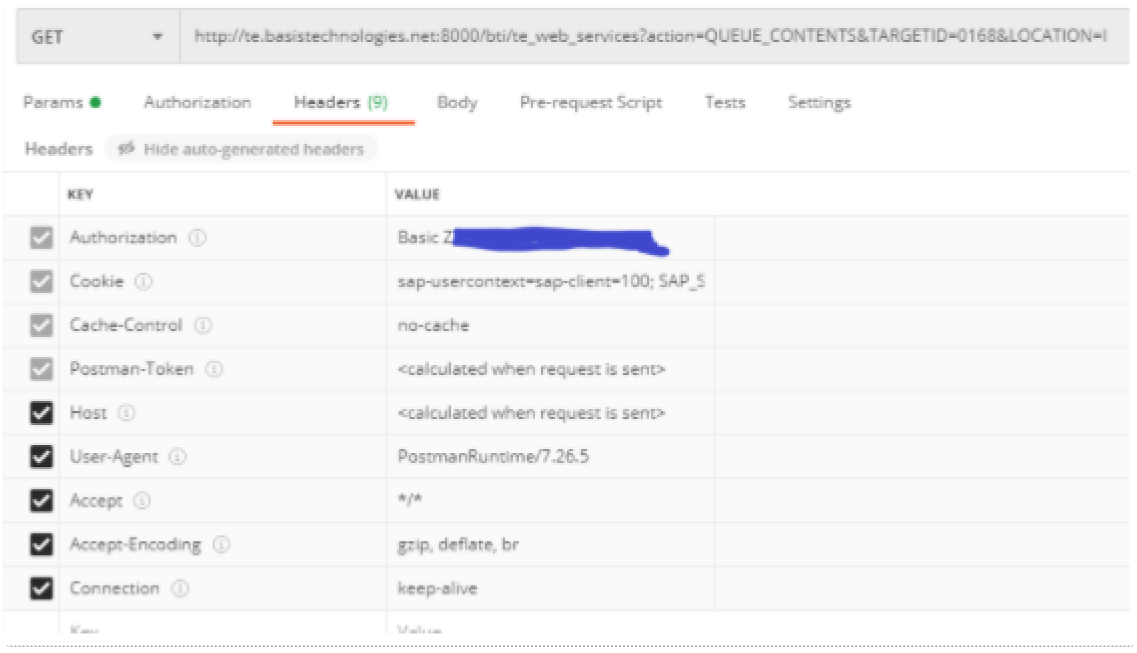
Query Params

	KEY	VALUE	DESCRIPTION
✓	action	QUEUE_CONTENTS	Get Queue Contents action
✓	TARGETID	168	The ID of the target system.
✓	LOCATION	T	The queue location in the target system.
	Key	Value	Description

Body Cookies (2) Headers (7) Test Results
Pretty Raw Preview Visualize XML

```
<?xml version="1.0" encoding="utf-8"?>
<asx:abap version="1.0" xmlns:asx="http://www.sap.com/abapxml">
    <asx:values>
        <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>
```

GET http://te.basistechnologies.net:8000/bti/te_web_services?action=QUEUE_CONTENTS&TARGETID=0168&LOCATION=I



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
✓	Authorization	Basic Z[redacted]
✓	Cookie	sap-usercontext=sap-client=100; SAP_S
✓	Cache-Control	no-cache
✓	Postman-Token	<calculated when request is sent>
✓	Host	<calculated when request is sent>
✓	User-Agent	PostmanRuntime/7.26.5
✓	Accept	*/*
✓	Accept-Encoding	gzip, deflate, br
✓	Connection	keep-alive
	Key	Value

4.4.2. Lock Queue

HTTP GET

Example URL:
http://te.basistechnologies.net:8000/bti/te_web_services?action=LOCK_TARGET&TARGETID=168

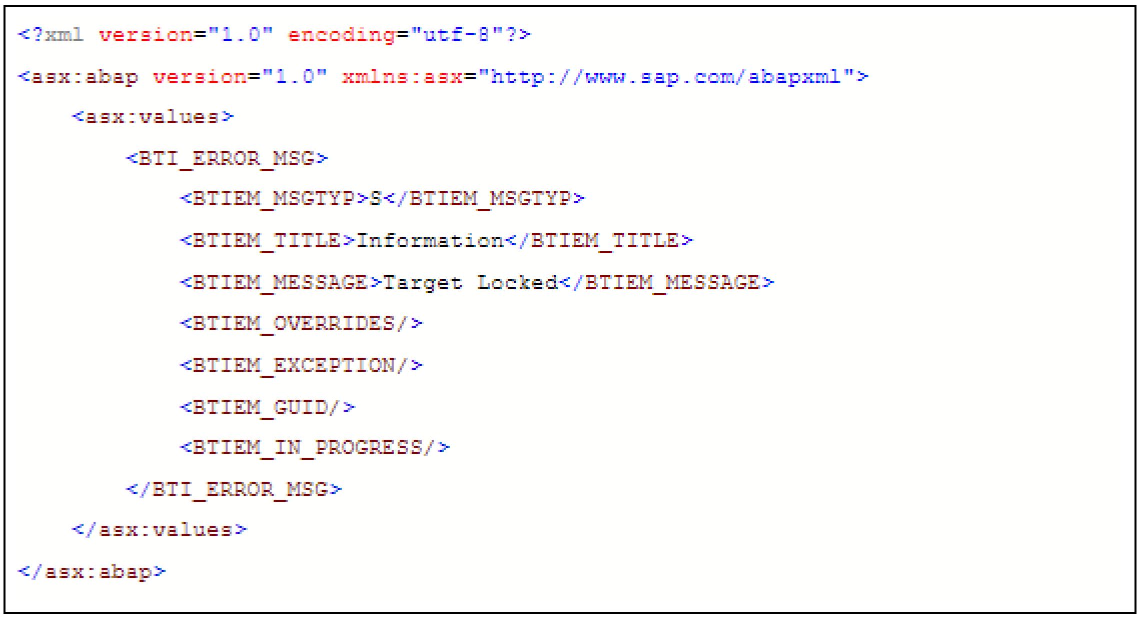
Authentication – Basic (ActiveControl user credentials)

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

Request Parameters

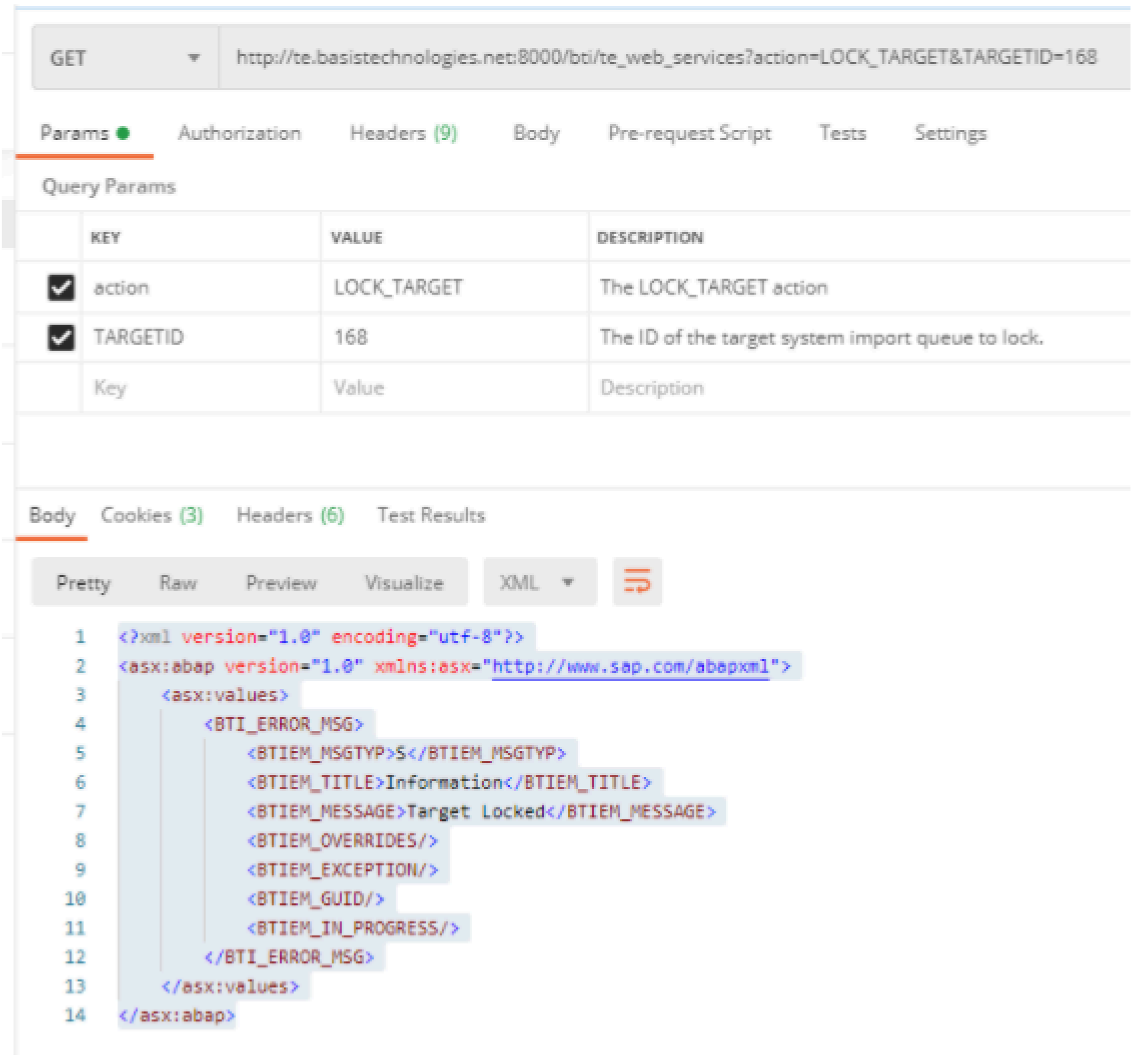
Name	Value	Description
action	LOCK_TARGET	Locks the target system import queue.
TARGETID	168	Look in the /BTI/TE_TARG table for the target id.

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



http://te.basistechnologies.net:8000/bti/te_web_services?action=LOCK_TARGET&TARGETID=168

Query Params

	KEY	VALUE	DESCRIPTION
✓	action	LOCK_TARGET	The LOCK_TARGET action
✓	TARGETID	168	The ID of the target system import queue to lock.
	Key	Value	Description

```
<?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>
```

4.4.3. Unlock Queue

HTTP GET

Example URL:
http://te.basistechnologies.net:8000/bti/te_web_services?action=UNLOCK_TARGET&TARGETID=168

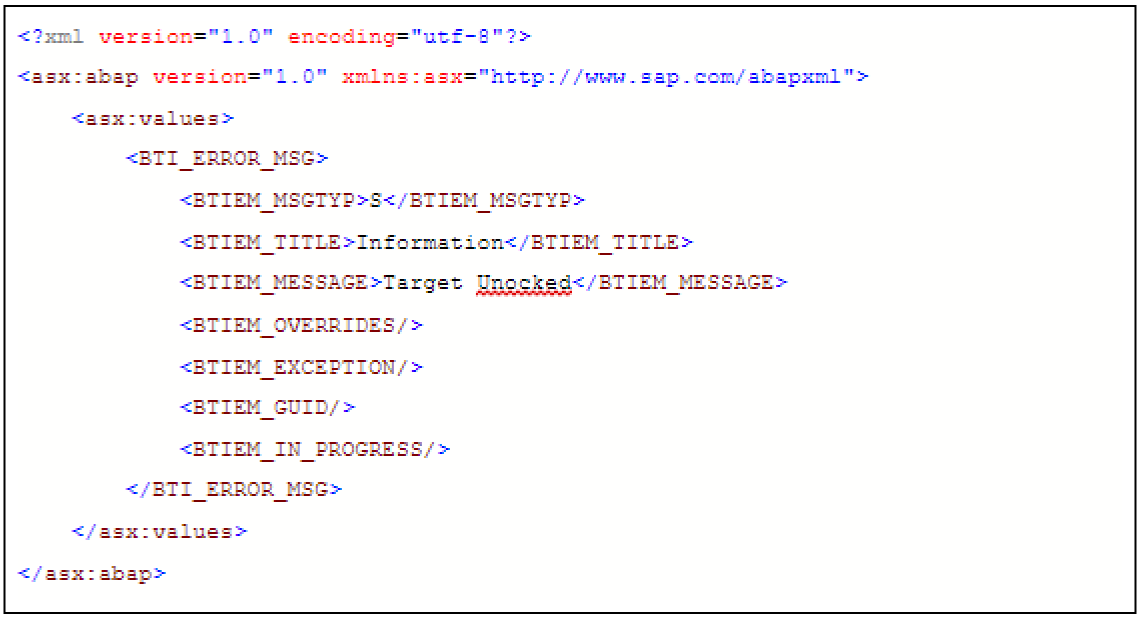
Authentication – Basic (ActiveControl user credentials)

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.

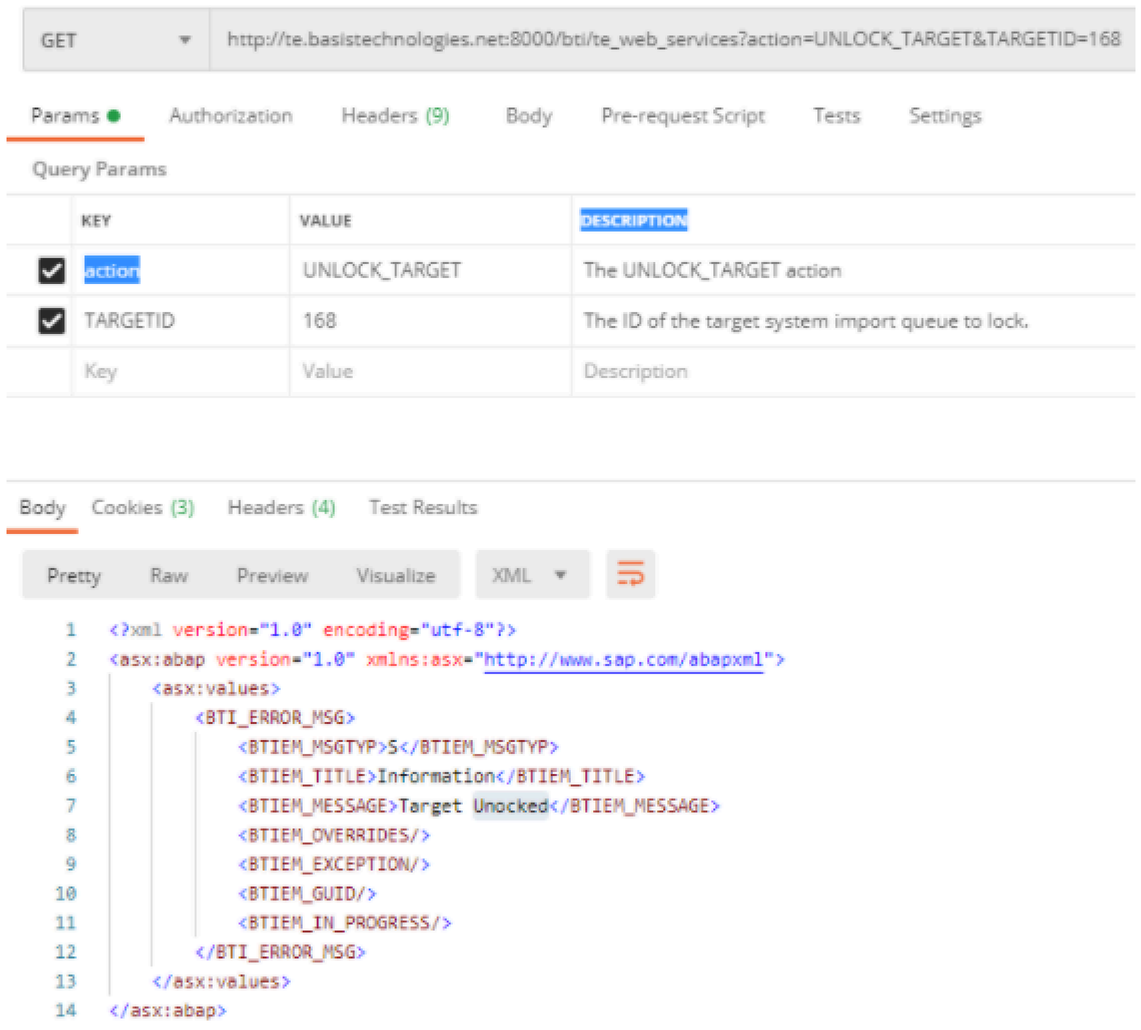
Response



```
<?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



http://te.basistechnologies.net:8000/bti/te_web_services?action=UNLOCK_TARGET&TARGETID=168

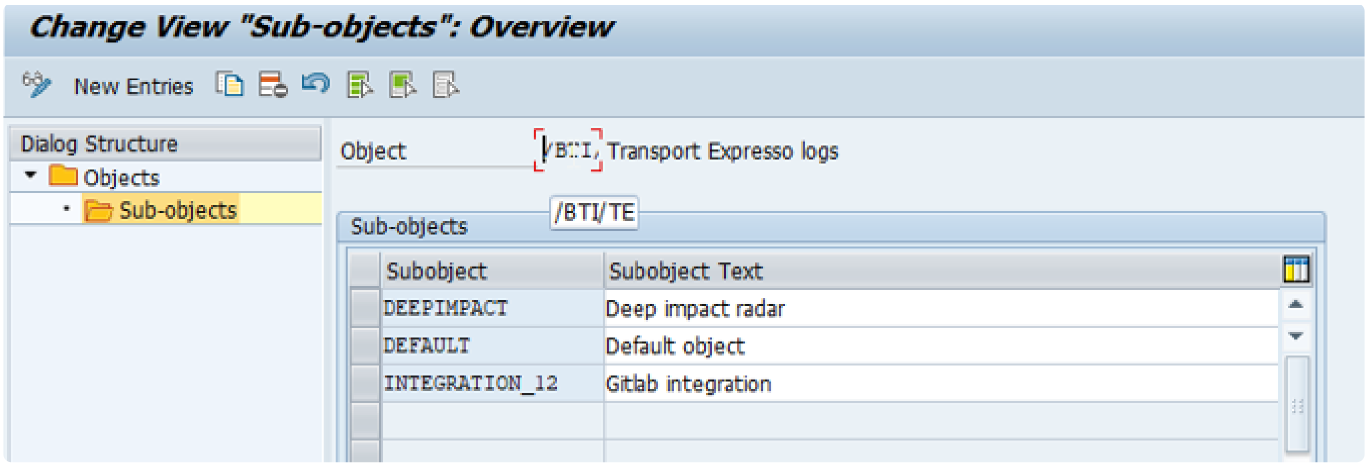
	KEY	VALUE	DESCRIPTION
✓	action	UNLOCK_TARGET	The UNLOCK_TARGET action
✓	TARGETID	168	The ID of the target system import queue to lock.
	Key	Value	Description

```
<?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>
```

4.5. Error Logging

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

Example configuration



Object /BTI/ Transport Expresso logs

Sub-objects /BTI/TE

Subobject	Subobject Text
DEEPIMPACT	Deep impact radar
DEFAULT	Default object
INTEGRATION_12	Gitlab integration

For INTEGRATION_NN, the NN should be the EXTSYST_NO as defined in table /BTI/TE_INT_SYST.

Once the above has been done, tTransaction 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.

5. Further Information

As detailed earlier in this Integration Guide, the ActiveControl / Gitlab integration is not an out-of-the-box 'plug it and play' Integration in the sense of many of the more established and mature ActiveControl integrations such as JIRA and ServiceNow. It should be anticipated that some chargeable consulting – and possible also development – services will be required to setup the integration, both on the Basis Technologies side and also Customer technology team.

If you are interested in finding out more about the ActiveControl / GitLab integration capability to trigger automated testing (or indeed any other purpose) – please reach out to your Basis Technologies Acount Manager.