**Algorithms integration (HAI)**

**User Manual**

|  |  |
| --- | --- |
| Document Control: | |
| Project: | HAI: Highly Automated integration |
| Revision: | 0.1 |
| Last Change: | 03-Sep-2015 |
| Confidence Level: | 🞏 Public 🗷 Confidential |

CONFIDENTIAL AND PROPRIETARY PROPERTY OF ADAS SIBIU - ALL RIGHTS RESERVED

|  |  |
| --- | --- |
| Document State: | |
| State: | Draft |
| Author: | Medhat Hussain |
| Reviewed by: | Medhat Hussain |
| Released by: | DD-MMM-YYYY |

Revision History

| Version | Date | Change Description | Responsible | Approver |
| --- | --- | --- | --- | --- |
| 0.1 | 03-Sep-2015 | Initial draft version for review | Medhat Hussain | Medhat Hussain |
| 0.2 | 05-Sep-2015 | Updated | Dixy Abraham | Medhat Hussain |
| 1.0 | 08-Sep-2015 | After revision | Medhat Hussain | Medhat Hussain |
|  |  |  |  |  |
|  |  |  |  |  |

Table of Contents

[1 Introduction 4](#_Toc430857009)

[1.1 Purpose 4](#_Toc430857010)

[1.2 Overview 4](#_Toc430857011)

[2 Prerequisites 4](#_Toc430857012)

[3 Configuration 5](#_Toc430857013)

[3.1.1 User configurations 5](#_Toc430857014)

[3.1.2 Project configurations 6](#_Toc430857015)

[3.1.2.1 Jenkins.xml 6](#_Toc430857016)

[3.1.2.2 make\_all\_tmp.xml 6](#_Toc430857017)

[4 Usage 7](#_Toc430857018)

[5 HAI Automated Activity Workflow 8](#_Toc430857019)

[6 Utilities 10](#_Toc430857020)

[6.1.1 Build 10](#_Toc430857021)

[6.1.2 Inject 10](#_Toc430857022)

[6.2 Advantages 10](#_Toc430857023)

# Introduction

## Purpose

This document contains the guideline for HAI’s users.

## Overview

The HAI is an automated integration procedure used to produce integration checkpoints .The HAI implementation helps minimize dependency of the integration activities on human interaction and reduces the overall time and effort required to create integration checkpoints.

# Prerequisites

* Active Lotus notes client
* Active MKS client.
* Python 2.7
* Python package xlrd

# Configuration

## User configurations

The Conf.ini has multiple “STD\_ON”, “STD\_OFF”,”STD\_AUTO” and “STD\_USER” configurations which allow the user to modify various parameters to work as standard operation.

|  |  |
| --- | --- |
| Operation Parameter | Description |
| STD\_ON | Keeps the operation parameter ON as standard |
| STD\_OFF | Keeps the operation parameter OFF as standard |
| STD\_USER | Prompts the user to perform an operation on the command line |
| STD\_AUTO | The HAI makes a decision based on the operation |

Some sample operation parameter descriptions are shown below:

|  |  |  |
| --- | --- | --- |
| Operation | Parameter | Description |
| [Configuration] | Update\_Cfg\_Files | Updates/check-in the configuration files (\*.xml) |
| Run\_Config | Run the configuration operation |
| [Build] | Rebuild | Run the build operation(-r) |
| Build\_ALL | Build all components regardless how many were changed. |
| [TEST] | Check\_DevPath | Check and verify the development path in MKS and the one in the module overview are the same |
| Check\_Labels | Parse all \*.xml files and verify the labels exist on MKS |
| Remote\_SW\_DevGate | Invoke DevGate test as a part of our integration test. |
| [PostBuild] | Auto\_CP\_Creation | Create the checkpoint automatically |
| [XlsUpdate] | AUTO\_XLS | Automatic update of the xls files |
| [Email] | SEND\_EMAIL | Activate/deactivate the emailing service |
| STEP\_EMAIL\_NOTIFICATION | Step by step notification of the process by email, progress emailing activation |
| Default\_EMAIL | The default mailing address , can also be determined by the commandline parameter *–email=* |

## Project configurations

## Jenkins.xml

Such file contains all configurations relevant to invoke Jenkins jobs, like the DevGate and the Compstate

*For each you need to determine*

* server names,
* job name,
* user name
* Password.

For e.g.

<Job name="Local\_Compstate" protocol ="http" server="lul2yxmg:8888" job="Local\_Smoketest" username="" pwd=""/>

The username=”” and pwd =”” can be modified to include individual user names and passwords.

Such jobs can be switched ON/OFF by the parameters within Conf.ini, kindly refer to 3.1.1

## make\_all\_tmp.xml

Such file contains the dependencies in-between the component, it’s used to determine which components to build based on the new labels delivered.

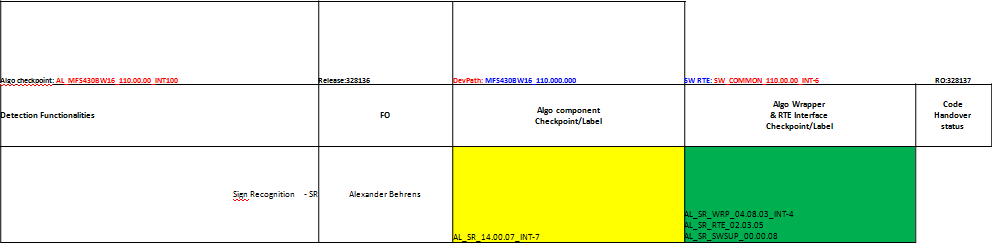
# Usage

1. **Update the Module Overview Excel Sheet**

In accordance with the normal integration procedure, one has to update the Module Overview excel sheet with the following points.

1. The checkpoint labels to be added
2. The name of the checkpoint
3. RO number
4. RTE update if any
5. Release Number

A sample module overview update is shown below:



*SW RTE*

*Realization Order*

*Release Number*

*Component Name*

*Checkpoint Name*

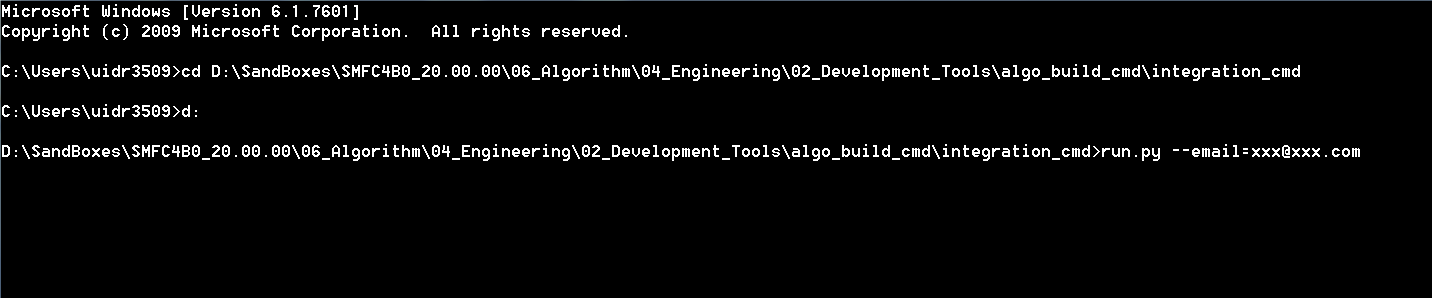
After updating the excel sheet **check in** the Excel sheet into the MKS client.

1. **Invoking the HAI**
2. The HAI can be invoked from the following folder:

**06\_Algorithm\04\_Engineering\02\_Development\_Tools\algo\_build\_cmd\integration\_cmd,** using the command line. Use the run.py command to run the HAI operation.

1. Entering your email will help the HAI provide you with real time results of the operation. Enter your email in the following format: --email = [xxx@xxxx.com](mailto:xxx@xxxx.com). Multiple emails can be entered by using the comma (,) operator.

A snapshot of the operation is given below:



# HAI Automated Activity Workflow

After invoking the HAI as mentioned in section 1.5, the following tasks are carried out:

1. Verification of the presence of the updated component checkpoints labels in the MKS is done. (**Optional**: Verification of all the components present in the xml files is also possible) If the components are not present an error report is generated.
2. If the checkpoint labels are present, updating and configuration of the new component checkpoint labels into the respective sandbox in MKS is done by the HAI.
3. Resynchronization of the ***04\_Engineering*** folder for build operation
4. Building operation of the checkpoints is carried out for the new component label checkpoints. In case of software RTE or CCT update the system will carry out a build of the entire components. Error log files are generated in case of errors, else the HAI is continued.
5. All the xml files are checked into the MKS server.
6. The *dlls*, *sdls* and *libs* generated during the build operation are checked into the deliverables folder.
7. An MTS test and DevGate test is carried out in the Jenkins server after checking in the dlls, sdls and libs into the deliverables folder. Any errors produced during the testing phase will stop the HAI operation
8. The checkpoint is created and *02\_System* folderupdated.

***1.6.1 Operation Flowchart***

# Utilities

## Build

The Build command is used to build one or more components for one or more filters command as follows:

***Build.py -c em,pc,ld -f ecu,sim***

In case of multiple components separate the components with the comma operator **(,)** and in case of a needed rebuild add **–r** to the end of the command.

## Inject

The Inject command is used to inject one or more components’ labels into the current sandbox, the object is to be able testing certain checkpoint before modifying it on the server

Usage:

***Inject.py -l AL\_PC\_01.08.05\_INT-5,*** ***AL\_CB\_03.11.02\_INT3***

You do not need to determine the component’s name, which will be detected automatically, such feature can be used be the integrators or by the components’ teams before the delivery of their checkpoints.

It causes no modifications on the server.

## Advantages

1. Reduce dependency on human interaction (integrator) for integration process.
2. Increase productivity by reducing time taken for integration For eg, manual integration takes 2 hours at the minimum which can be reduced drastically using automated integration.
3. Avoiding errors introduced from the integrator side(Human errors)