ActionVector IoT

THE SMART IOT PLATFORM

INSTALLATION AND SETUP

Table of Contents

Introduction

This document describes the ActionVector IoT Platform Installation and Setup details. This is a work-in-progress document.

Pre-Requisites

- `Linux: Ubuntu 15.04 or higher / Fedora 14 (Laughlin) or higher
- PostGreSQL: postgres 7.4 or higher
- Tomcat: apache-tomcat 7.0.34 or higher
- Java: JDK 7 or higher
- NetBeans IDE 7.4 or higher

Optional:

Docker 17.09 or higher

If installed, one can see the Framework Demo with a single click.

Jenkins 2.46.3 or higher

If installed, framework build and deploy can be automated with a single click.

Configurations

- Add below mapping in /etc/hosts file:
  - merit.actionvector.com 127.0.0.1
- Change the tomcat port to 8090 in tomcat/conf/server.xml/
  - <Connector port=”8090” protocol=”HTTP/1.1” …>
- Create postgres database “avsa” and import DB schema using the below command:
  - su – postgres
  - createdb avsa
  - psql avsa < avsa-.dmp
  - Add below entry in postgres pg_hba.conf file:
    host all all 0.0.0.0/0 md5
  - Restart postgres:
    service postgres restart

Code Download

The complete framework source code is available at the git URL:
https://github.com/MeritSystemsPvtLtd/ActionVector_IoT_Platform

Click on Clone/Download and download it to USER_HOME/GIT-AVSA directory

Build

Each Framework is a NetBeans Project. Any code change can be done using Netbeans IDE. The entire solution can be built in a single step or selective frameworks can be built individually.

Reference Solution Build:

The complete reference solution can be built in a single step. Here are the steps to build:

1. Open the NetBeans Project corresponding to the framework.
2. Do the code changes and save.
3. Open the terminal and cd user_home/GIT-AVSA/scripts
4. Execute the solution build script:
   sh IoTReferenceSolutionBuild.sh --<solution> --all //command line parameters are in angular brackets
   <solution> is IT/Solar

IoT Platform Build:

The IoT Platform alone can be built in a single step. Here are the steps to build:

1. Open the NetBeans Project corresponding to the framework.
2. Do the code changes and save.
3. Open the terminal and cd user_home/GIT-AVSA/scripts
4. Execute the platform build script:
   sh IoTPlatformBuild.sh --all //command line parameters are in angular brackets
   <solution> is IT/Solar

Framework Build:

Any framework across IoTPlatform/IoTDomainSpecific can be built individually with a variation in the command line parameter as mentioned in the below section:

1. Open the NetBeans Project corresponding to the framework.
2. Do the code changes and save.
3. Open the terminal and cd user_home/GIT-AVSA/scripts
4. Execute the framework build script:
   sh IoTFrameworkBuild.sh --<solution> --<frameworkname> //command line parameters are in angular brackets
   <solution> is IT/Solar
   < frameworkname > is the name of the framework for which code change is done. Below is the list of command line argument corresponding to each framework:

IoT Common Applications and Service Frameworks:

Framework	Source Path	command line argument
Database Connection	ICASF/CommonLibraries	DatabaseConn

		ection**(Not Listed in API doc)**
Resource Configuration GUI Application	ICASF/ IOT_Applications/Admin_GUI_Application	SA-DeskAdminTool
CEP Event Processor	ICASF/ IOT_Applications/CEP_Event_Processor_Framework	CEPEventProcessor**(Not Listed in API doc)**
Failed Events Uploader	ICASF/ IOT_Applications/FailedEventsUploader/Failed-Event	Failed-EventsUploader
Alerts and Notification API	ICASF /IOT_Gateway/Alerts_and_Notifications_Framework/EMailSMSUtility	EMailSMSUtility
End User API	ICASF/ IOT_Server/End_User_API/ConsumerAPI	ConsumerAPI
Failed Events Dispatcher	ICASF /IOT_Gateway/Gateway_Event_Dispatcher_Framework/Failed_Events_Dispatch_Framework/FailedEventsDispatcher	FailedEventsDispatcher
Gateway Event Dispatcher (HTTP)	ICASF /IOT_Gateway/Gateway_Event_Dispatcher_Framework/HTTP-Dispatcher/HTTPEventDispatcher	HTTPEventDispatcher
Gateway Event Dispatcher (TCP)	ICASF /IOT_Gateway/Gateway_Event_Dispatcher_Framework/TCP-Dispatcher/TCPEventDispatcher	TCPEventDispatcher
Device Configuration APIs	ICASF /IOT_Server/Device_Management_Framework/Device_Management_API/DeviceManagment	DeviceManagment
Event Logger Framework	ICASF /IOT_Server/Event_Logger_Framework/source/ETLAdapter	ETLAdapter
License Key Framework	ICASF /IOT_Server/License_Framework/LicenseKeyGenerator	LicenseKeyGenerator
Event Receiver Mapper	ICASF /IOT_Server/Multi-point_Event_Router_Framework/Event_Receiver_Mapper/EventReceiverMapper	EventReceiverMapper
Timeline Aggregator Framework	ICASF /IOT_Server/Performance_Analytics_and_Reports_Framework/Timeline_Aggregator_Framework/source/TimeLineAggregator	TimeLineAggregator
Timeline Consumed Generator Framework	ICASF /IOT_Server/Performance_Analytics_and_Reports_Framework/Timeline_Consumed_Generator/source/TimeLineConsumedGenerator	TimeLineConsumedGenerator
Reports Generator Framework	ICASF /IOT_Server/Performance_Analytics_and_Reports_Framewor	ReportsGenerator

	k/XLS_Reports_Generator/source/ReportsGenerator	
Chart Data Generator Framework	ICASF /IOT_Server/Chart_Data_Generator_Framework/source/DashBoard-JSONgenerator-ArbitraryDate	JSONgenerator
CEP Framework	ICASF /IOT_Server/Complex_Event_Processor_Framework/ComplexEventProcessingService/source/sadeskCEP2.0	CEPEngine
TCP Event Listener Service	ICASF /IOT_Server/Multi-point_Event_Router_Framework/TCP_Event_Receiver/source/TCPEventReceiver	TCPEventReceiver
Chart Data Generator Framework for IT Monitoring Solution	ICASF /IOT_Server/Chart_Data_Generator_Framework/IT/source/DashBoard-JSONgenerator	IT-JSONgenerator

IoT DAS Frameworks IT Infra Monitoring and Analytics:

Framework	Source Path	command line argument
DashBoard	IoT-DAS-Frameworks-IT-Infra-Monitoring-and-Analytics/IOT_Applications/Performance_Dashboard_Application_Reference_App/source/SA-DashBoard	SA_DashBoard
Ganglia Data Collection Framework	IoT-DAS-Frameworks-IT-Infra-Monitoring-and-Analytics/IOT_Gateway/Ganglia_Data_Collector_Framework/source/GangliaDataCollector	GangliaDataCollector
Ganglia Schema Mapper	IoT-DAS-Frameworks-IT-Infra-Monitoring-and-Analytics /IOT_Gateway/Ganglia_Event_Schema_Mapper_Framework/GangliaEventSchemaMapper	GangliaEventSchemaMapper
Ganglia Protocol Handler Framework	IoT-DAS-Frameworks-IT-Infra-Monitoring-and-Analytics / /IOT_Gateway/Ganglia_Protocol_Handler_Framework/GangliaProtocolHandler	GangliaProtocolHandler
JVM Data Collection Framework	IoT-DAS-Frameworks-IT-Infra-Monitoring-and-Analytics / IOT_Gateway/JVM_Data_Collection_Framework/source/JVMDataCollector	JVMDataCollector

IoT Domain Applications and Services Framework SolarDeviceDomain:

Framework	Source Path	command line argument
DashBoard	IoT-DAS-Frameworks-Solar-Plant-Monitoring-and-Analytics /IOT_Applications/Performance_Dashboard_Application_Reference_App/source/SA-DashBoard-ArbitraryDate	SA_DashBoard
Fronius Data Collection Framework	IoT-DAS-Frameworks-Solar-Plant-Monitoring-and-Analytics /IOT_Gateway/Fronius_Data_Collector_Framework/source/FroniusAdapter	FroniusAdapter
ModBus Data Collection Framework	IoT-DAS-Frameworks-Solar-Plant-Monitoring-and-Analytics /IOT_Gateway/Modbus_Data_Collector_Framework/source/	ModBusAdapter

	ModBusAdapter	
Fronius Protocol Handler Framework	IoT-DAS-Frameworks-Solar-Plant-Monitoring-and-Analytics /IOT_Gateway/Fronius_Protocol_Handler_Framework/FroniusConnector	FroniusConnector
ModBus Protocol Handler Framework	IoT-DAS-Frameworks-Solar-Plant-Monitoring-and-Analytics /IOT_Gateway/Modbus_Protocol_Handler_Framework/ModBusConnector	ModBusConnector
Report Mailer Framework	IoT-DAS-Frameworks-Solar-Plant-Monitoring-and-Analytics /IOT_Server/Daily_Reports_Dispatch_Framework/source/XLSXReportsMailer	XLSXReportsMailer
NOTE: Missing	**Timeline consumed generator, timeline aggregator, reports generator, derived metrics generator**	

Configure

Any configuration changes should be done in the respective frameworks configuration files.

Deployment

The entire reference solution can be deployed in a single step as shown below:

Reference Solution Deployment:

Here are the steps to deploy the solution after code changes:

1. Open the terminal and cd user_home/GIT-AVSA/scripts
   Execute the solution stop script:
   sh IoTFramework_IT_Monitoring.sh --stop //for IT
   sh IoTFramework_Solar_Monitoring.sh --stop //for Solar

2. Follow the Build Step mentioned before.

3. Execute the solution start script:
   sh IoTFramework_IT_Monitoring.sh --start //for IT
   sh IoTFramework_Solar_Monitoring.sh --start //for Solar

IoT Platform Deployment:

Here are the steps to deploy the entire IoT Platform:

1. Open the terminal and cd user_home/GIT-AVSA/scripts
2. Execute the command:
   sh IoTPlatformDeploy.sh --stop
3. Follow the Build Step mentioned before.
4. Execute the command:
   sh IoTPlatformDeploy.sh --start

Framework Deployment :

Any changes in any of the frameworks can be deployed individually.

Here are the steps to deploy any framework after code changes:

1. Open the terminal and cd user_home/GIT-AVSA/scripts
   Execute the framework stop script:
   sh IoTFramework_IT_Monitoring.sh –stop --<frameworkname> //for IT
   sh IoTFramework_Solar_Monitoring.sh --stop --< frameworkname > //for Solar

2. Do the code changes and follow the Build Step mentioned before.

3. Execute the framework start script:
   sh IoTFramework_IT_Monitoring.sh --start --< frameworkname > //for IT
   sh IoTFramework_Solar_Monitoring.sh --start --< frameworkname > //for Solar

   < frameworkname > is a command line argument as described in detail in Build section.

Custom Solution Deployment :

One should refer to the reference domain framework to implement and deploy a custom solution.

Alternatively, Jenkins script is available which does the framework build and deploy in just one single Jenkins script.