CR2 High Level Design



**Author: Suneel Jalagadugu**

**Version:**

**Status:**

**Kingfisher Group Confidential**

Document location

Ensure that this document is current. Printed documents and locally copied files may become obsolete due to changes to the master document.

The most current version of this document is located ??

Reviewers

|  |  |  |
| --- | --- | --- |
| **Reviewer Name** | **Role** | **Changes marked** |
| Mark Lane | Reviewer & Approver |  |
| Chris Hample | Reviewer |  |
|  |  |  |

Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version Number** | **Who** | **Revision Date** | **Summary of Changes** | **Changes marked** |
| 0.1 | Suneel Jalagadugu | 27/12/2013 | Initial version |  |
| 0.2 | Richard Brooke | 13/01/2014 | JBoss property details added to Appendix A |  |
|  |  |  |  |  |

Distribution

This document has been distributed to:

| **Name** | **Title** |
| --- | --- |
| Darwin | All members of the Darwin Programme |
|  |  |
|  |  |

References

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Title** | **Document Name** | **Date** |
| R | Darwin Technical Architecture Document | Darwin TAD Version 2.0 |  |
| R | Darwin “CRx” High Level Design | Darwin CR2 High level Design Document. |  |
| R | Release & Configuration Management Design Strategy | Config\_Deployment\_Release Strategy |  |
| R | Darwin F2 Deployment Services - HIGH LEVEL INFRASTRUCTURE DESIGN SD05 | Darwin R2 Deployment Services HLID v04 |  |
| 6 |  |  |  |
| R |  |  |  |

**Contents**

[Kingfisher Group Confidential 1](#_Toc375926666)

[Document location 2](#_Toc375926667)

[Revieweres 2](#_Toc375926668)

[Revi History 2](#_Toc375926669)

[References 2](#_Toc375926670)

[1 Introduction 5](#_Toc375926671)

[1.1 Overview of Darwin 5](#_Toc375926672)

[1.2 Introduction to CR166 5](#_Toc375926673)

[1.3 Scope 7](#_Toc375926674)

[1.4 Target Audience 7](#_Toc375926675)

[2 Darwin Deployment System 8](#_Toc375926676)

[2.1 Introduction: 8](#_Toc375926677)

[2.1.1 Design 8](#_Toc375926681)

[2.1.2 DDS Implementation 12](#_Toc375926682)

[2.1.3 Configuration Setup for Apache and JBoss instances 13](#_Toc375926683)

[2.1.4 Build, Release and Deployment of Deployable Artefacts 14](#_Toc375926684)

[3 Apache Configuration Setup 15](#_Toc375926685)

[3.1 Apache Configuration Setup Strategy 15](#_Toc375926686)

[3.1.1 Apache Server– Vanilla Installation 15](#_Toc375926689)

[3.2 Apache Configuration Implementation 16](#_Toc375926690)

[3.2.1 Apache Server Configuration – httpd-ecomm.conf 16](#_Toc375926693)

[3.2.2 Apache Server Configuration – Common configuration 17](#_Toc375926694)

[3.2.3 Apache Server Configuration – ATG-Persona Specific Configuration (VirtualHost) 17](#_Toc375926695)

[3.2.4 Apache Server Configuration – Redirections-Specific Configuration 17](#_Toc375926696)

[3.2.5 DDS Scripts 18](#_Toc375926697)

[4 JBoss Configuration Setup 21](#_Toc375926698)

[4.1 JBoss Configuration Setup Strategy 21](#_Toc375926699)

[4.1.1 JBoss Server– Vanilla Installation 21](#_Toc375926702)

[4.2 JBoss Configuration Implementation 21](#_Toc375926703)

[4.2.1 JBoss Server Configuration – Security 21](#_Toc375926705)

[4.2.2 JBoss Server Configuration – Port Configuration 22](#_Toc375926706)

[4.2.3 JBoss Server Configuration – JVM Route 22](#_Toc375926707)

[4.2.4 JBoss Server Configuration –Timeouts 23](#_Toc375926708)

[4.2.5 DDS Scripts 23](#_Toc375926709)

[5 Database Data Source Configuration Setup 27](#_Toc375926710)

[5.1 Data Source Configuration Setup Strategy 27](#_Toc375926711)

[5.1.1 DDS Scripts 27](#_Toc375926714)

[6 ATG Code Build, Release and Deployment Strategy 28](#_Toc375926715)

[6.1 Introduction 28](#_Toc375926716)

[6.2 Code Compilation 28](#_Toc375926717)

[6.3 Release and Deployment strategy to various environments 28](#_Toc375926718)

[6.4 Miscellaneous activities, such as DB Deploy etc., 28](#_Toc375926719)

[7 Appendix – A : Properties file Arguments Description 29](#_Toc375926720)

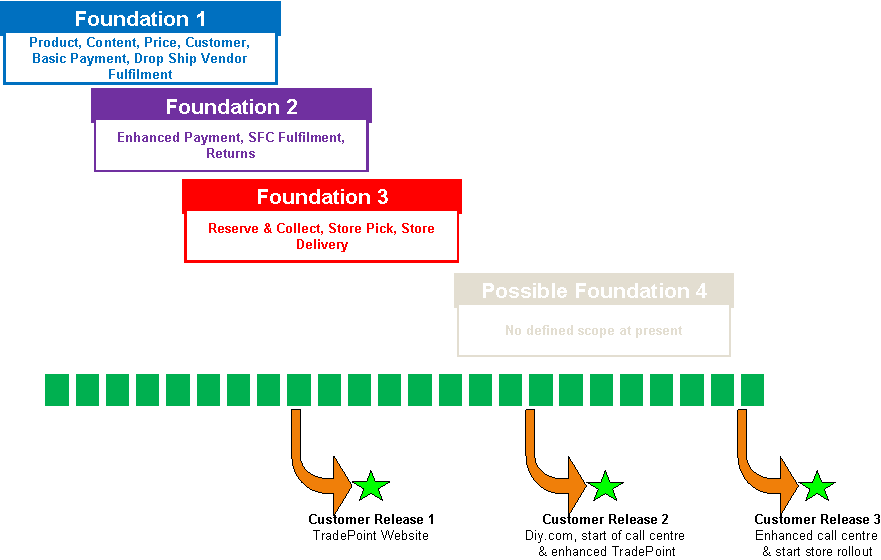
1. Introduction
   1. Overview of Darwin

Darwin is a business and IT transformation programme and the first and most significant building block to deliver the B&Q’s Omni-channel offering. The Darwin programme will deliver a web, call centre, and in-store solution to support the B&Q’s multi-channel offering for both Retail and Trade customers.

The IT solution element of Darwin will:

* Enable the B&Q business proposition;
* Provide an IT solution foundation that has the potential to accelerate the delivery of multi‑channel solutions in other OpCo’s across the Kingfisher Group;
* Address end of life and support issues with existing systems.

The program will be delivered through a combination of Foundation releases and iterative development of customer releases which is in ATG.



F1 and CR1 have delivered the necessary solution and infrastructure to cover the fully transactional trade-point website.F2 has delivered the services required for diy.com. CR2 , F3/CR3 is under development and due to be delivered.

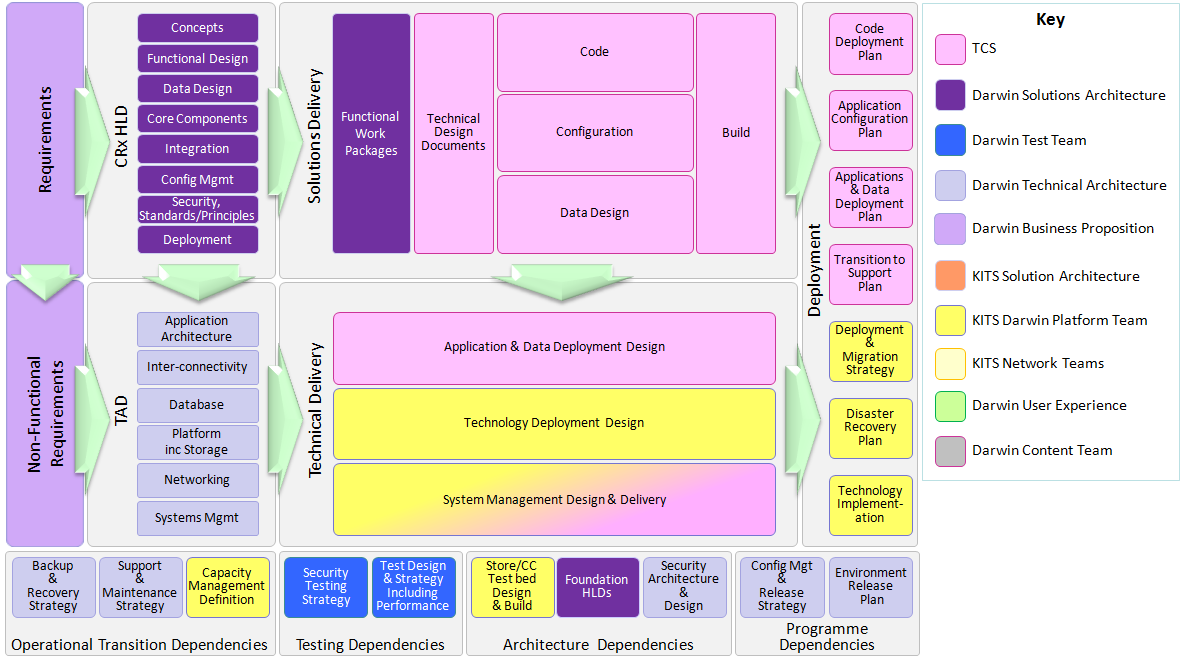
* 1. Introduction to CR166

To meet the non-functional requirements, increased volumes of CR2 and CR3, and lessons learned from CR1 have recommended for revised infrastructure design. The proposed solution is a “Change Request” submitted to CRB having key features as below

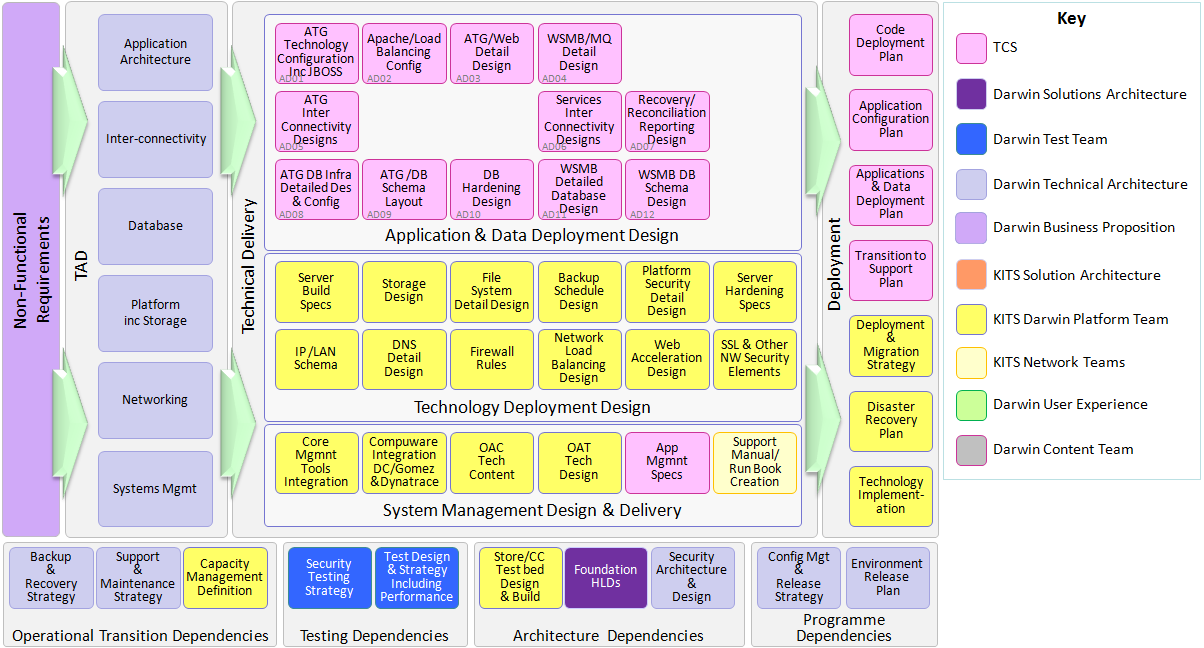
1. Site will operate ATG instances across two datacentres with active-active traffic management
2. Introducing Application Delivery controllers for Production and Non-Production DMZ’s in order to manage the traffic routing to required domain names and services
3. The key services are defined as diy.com, trade-point.co.uk, stores/contact-centres which are defined as service pools at ADC.
4. The above services will be served from silo based design. Each silo is a one to one mapping of a web server and an application server. Refer to Technical Architecture document ( TAD Version 2)
5. Upgrading ATG from ATG10.0.3 to ATG10.1.2 for CR2 and ATG 10.2 for CR3.
6. As a result of ATG upgrade and per TAD design, re-platform the Darwin TEST and Production environment estate.
7. Installing 3rd party software’s such as Jboss and Apache in a supportable and maintainable format.
8. Transition of existing production estate to new platform and decommissioning the existing.

Below figures shows the High level over view of the solution design and Technical architecture document

**Fig 1.2a Solution Design overview.**



**Fig 1.2b Technical Architecture Documentation overview.**



* 1. Scope

This document covers the holistic view of Design and Implementation for Build & Deploy, Release and Configuration management for Darwin’s CR2 and CR3 releases around ATG, Apache web server and JBoss application server configurations and managing the environments. This implementation is based on the approved design, solution and Infrastructure design strategy documents for CR2 and CR3.

This design strategy proposed in this document is in accordance and compliance with CR166, in reference to the TAD 2.0 and CRx High Level Design documents. The document mainly aiming on the key factors of the Darwin program delivery which are

* Robust and Stable
* Supportable
* Secure
* Scalable
* Performant

This document details about the low level How-To’s on the Build, Release and Deploy and this can also be used as a Run book. But this document doesn’t cover the details of Code merger, code branching into github.com, release management activities.

* 1. Target Audience

The Target Audience for this document is very broad. It is aimed at:

Those that wish to gain a high-level understanding of Darwin, particularly from an ATG perspective:

* Project / Programme Management
* Architects
* Lead Designers and Developers
* Test Teams
* Build Engineer’s and Deployment Team
* Infrastructure / Network Team
* Operational Support
* Maintenance and Support teams

1. Darwin Deployment System
   1. Introduction:

To ease the Infrastructure Support from Operations perspective, DDS is implemented to separate Application server and Web Server Configuration setup from Build, Release and Deployment process. As part of the DDS deliverables different teams , including Platform management team, KITS Operations team, Build Release team, Darwin Architecture Team, mandate below inputs.

* + DDS will be implemented by using simple Shell scripting and Basic Java programming, and would restrict to use advance and complex solutions such as YAML, Capistrano, Ruby etc.,
  + DDS should use placeholders in form of properties files, named after each ATG Persona, and these properties files should be easily manageable.
  + DDS shall refer to the properties files for configuration and deployment for respective environments.
  + DDS code should also be versioned and maintained in [www.github.com](http://www.github.com) under the Repository name KITS\_Continuous\_Integration.
  + DDS should be run as user “ecommadm”.
  + All users, who use DDS, should have and use their own user credentials to the DDS servers along with “sudo su – ecommadm” permission.

2. 1. 1. Design

**Outline**:

* All common Apache and JBoss configuration values, such as hostnames, port #s etc., should be stored in respective properties files under /app/ecomm/DDS/atg/installScripts/properties and names after each ATG-Persona.
* Environment specific Apache and JBoss configuration values should be prepended with environment name such as PVT, CC, CCS etc.,
* All parameters used in the properties files are explained in Appendix-A.
* DDS is broadly categorised into Setup scripts, Configuration Update scripts, Deployment Scripts and Operation Scripts.



**Directory Structure Layout**:

The Darwin Deployment system implementation , there is a configurable coupling of the directory structure, which should be configured in environment.sh.

DDS base code should be executed under the directory, /app/ecomm/DDS/atg. The same is configured as automationHome in environment.sh.

Below is the DDS automationHome layout and its description.

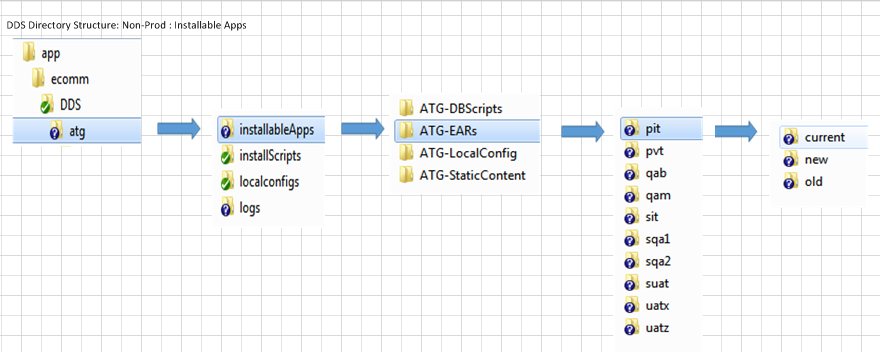
installScripts: This folder is the home directory for all DDS implementation code, such as configuration scripts, setup scripts, deployment scripts etc., and is again divided as java, properties, sctripts, templates, scripts and menus. This directory also contains environment.sh which supplies environment related arguments.

installableApps: This folder contains the deployable artefacts categorised as ATG-EARs for serving ATG Persona code (EAR files), ATG-StaticContent for serving Web Server Static content (zip format), ATG-LocalConfig for serving ATG Local configuration (zip format) and ATG-DBScripts for serving SQL queries to DB Deploy tool. All these directories have sub-directories into layers/environments and each environment again have three directories, new for the deployable artefacts to be deployed, current for the deployable artefacts which are currently deploying and or deployed, old for the deployable artefacts which are backups of previous releases and can be used for roll backs.

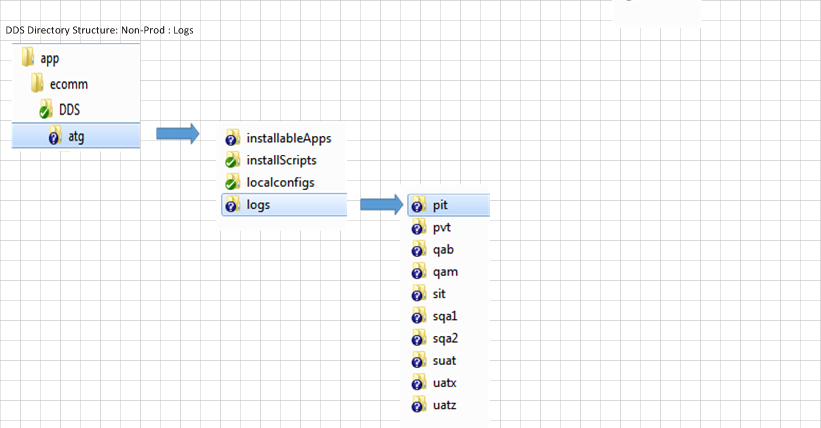
logs: This folder contains log files, which have the DDS scripts output. These logs will be used for troubleshooting and auditing puposes.

installScripts directory has same layout on both production and non-production DDS servers, but installableApps and logs directories contain only the environment which each DDS server is intended to serve, i.e., Production DDS contain CC, CCS, PFD, PFT and PROD and Non-production for the rest.

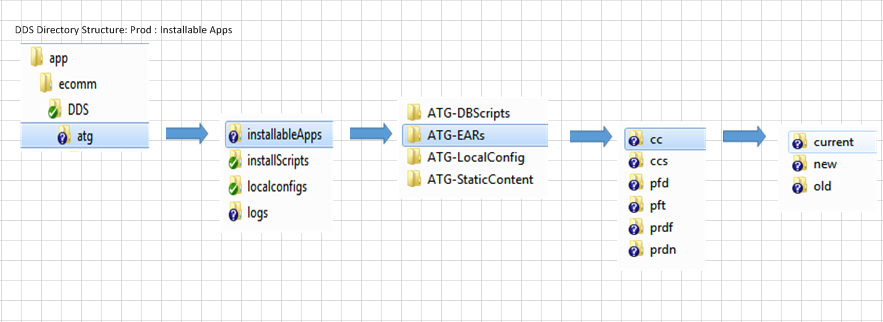
DDS Directory Layout : Non-Production : installableApps



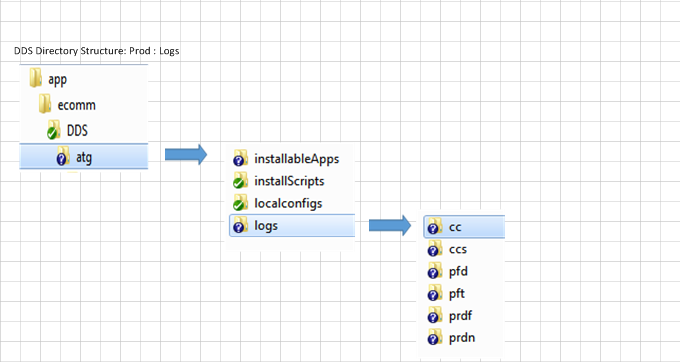
DDS Directory Layout : Non-Production : logs



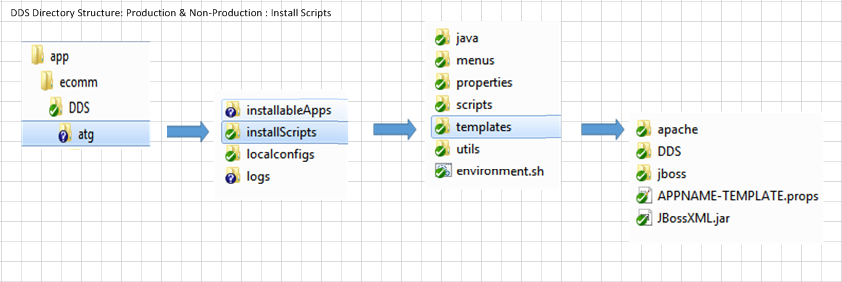
DDS Directory Layout : Production : installableApps



DDS Directory Layout : Production : logs



DDS Directory Layout : Production and Non-Production : installScripts



Below is description for each sub-folder and file under installScripts.

* environment.sh: This file contains all DDS configuration parameters and is called by all relevant scripts.
* java: This folder contains Java source code for modifying JBoss configuration XML files.
* menus: (Deprecated) This folder contains all menu-based scripts which can be used by Deployment team. But as the Deployment team is Jenkins as an interface to communicate with DDS, this folder is deprecated.
* properties: This folder contains ATG-Persona properties files and each file supplies configuration data for the scripts’ placeholders.
* scripts: This folder contains all configuration, setup, deployment, operation scripts. All these scripts are self-explanatory.
* templates: This folder contains template files for configuring Apache and JBoss instances.
* temp: This folder acts as a working directory to create/update the configuration data on DDS server.
* utils: This folder contains the common shell scripts, which are used by the scripts.
  + 1. DDS Implementation

Below are the steps to configure an environment from vanilla implementation.

* Check whether below Pre-configuration tasks are completed.
  + Request Network team to create DNS Aliases for Jboss and Apache hosts.
  + Request Network team to update firewall rules to provide connectivity between web server and app server, if any.
  + Request Database Connection details and also credentials from DBA Team.
  + Verify whether NAS shares are correctly mounted on respective Apache and JBoss servers

|  |  |
| --- | --- |
| NAS Share | Server Type |
| /app/ecomm/docroot | Web servers, Auxiliary Server (web) |
| /app/ecomm/ATGConfigFileSystem | Storefront, Agent, Staging, BCC, Fulfillment, SLM (web & agent), Auxiliary (web & agent) |
| /app/ecomm/ATGSearchDeployShare | Search servers, BCC |

* Prepare/Update environment related configuration values in respective ATG-Persona properties files. Below are the properties file names for ATG Personas.

|  |  |  |
| --- | --- | --- |
| ATG-Persona Name | Properties File Name | Comments |
| Storefront | storefront.props | Customer-facing (web) |
| Agent | agent.props | Customer-facing (agent) |
| Publishing | publishing.props | BCC |
| Staging | staging.props | Staging &/ Preview server |
| Fulfillment | fulfilment.props | OMS (Order status updates etc.,) |
| Auxiliary Storefront | auxstorefront.props | Schedulers (Batch jobs for web) |
| Auxiliary Agent | auxagent.props | Schedulers (Batch jobs for agent) |
| Lock Server Storefront | lockstorefront.props | SLM (web) |
| Lock Server Agent | lockagent.props | SLM (agent) |

* Setup SSH keys for ecommadm from respective DDS server to the environment Apache and JBoss hosts.
* Setup Apache instance(s) for required ATG-Personas (Storefront, Agent, Staging and Agent). Setup JBoss instances for all ATG Personas.
* Setup Search servers, including indexing server.
* Build and deploy the deployable artefacts (ATG Code for each persona in form of EAR files, ATG Local config in form of ZIP files, Static Content in form of ZIP files, DB Scripts) through Jenkins.
  + 1. Configuration Setup for Apache and JBoss instances

As shown in blow diagram the the configuration setup for Apache and JBoss instances for various environments should only be done from either DDS1 server (lnxs0587) for Production and Production-Supporting environments or DDS2 server (lnxs0586) for all development and testing environments.



* + 1. Build, Release and Deployment of Deployable Artefacts

Build and Release management team initiates the build and deployment process with the help if Jenkins Master. Below diagram depicts the Build and Deployment process.



1. Apache Configuration Setup
   1. Apache Configuration Setup Strategy

Apache configuration has been separated to and maintained in small configuration files.

* Worker Multi-Processing Module should be used instead of Prefork MPM.
* The configuration data should be divided into small sections of Configuration files, such as Initialization, Logging, Compression, Security etc.,
* A NAS share should be mounted on all Apache servers to serve HTDOCS.
* The HTDOCS content will use a switchable link and is copied only once per environment.
* The Deployment server should prepare the Configuration data and then push it to target server.
* The common logs, access.log and error.log should be stored under /app/ecomm/web/logs and ATG-Persona specific logs, ${APP}-access.log, ${APP}-error.log and ${APP}-redirect.log should be stored under /app/ecomm/web/logs/ECommApps
* The Configuration data should be sectioned into a common configuration data, which is used by all ATG-Personas and should be served from /app/ecomm/web/conf/common, if that Apache instance is hosting more than two ATG-Personas, and ATG-Persona specific Configuration data, such as VirtualHost configuration, and should be served from /app/ecomm/web/conf/ECommApps. If ATG-Persona has redirects, then it should be served from /app/ecomm/web/conf/Redirects

1. 1. 1. Apache Server– Vanilla Installation

The vanilla installation of Apache is outlined below.

* Uncommented the line in “#HTTPD=/usr/sbin/httpd.worker” in /etc/sysconfig/httpd.
* Updated /usr/sbin/apachectl from “HTTPD='/usr/sbin/httpd'” to “HTTPD='/usr/sbin/httpd.worker'”
* The /etc/httpd/conf/httpd.conf has following line only, “Include /app/ecomm/web/conf/httpd-ecomm.conf”
* The permission of the /etc/httpd/conf/httpd.conf has configured with 770 with ownership as apcadm:apache
* The conf and htdocs directories are created under/app/ecomm/web with permissions ecommadm:apache and ownership 770
* The /etc/mime.types is copied to /etc/httpd/conf/mime.types
* Unlink “logs” under “/etc/httpd” and link the same to “/app/ecomm/web/logs”
  1. Apache Configuration Implementation

As mentioned above, all the Configuration data is separated into small sections and prepared on the Deployment server and then pushed to the Target server. Below diagram depicts the configuration data preparation and push.



* 2. 1. Apache Server Configuration – httpd-ecomm.conf

The httpd-ecomm.conf file should only include all common configuration files, ATG-Persona specific configuration files and Redirection configuration files, if any.

Below is the sample configuration sample from PVT environment Storefront Apache instance.

*[ecommadm@lnxs0455 conf]$ cat httpd-ecomm.conf*

*Include /app/ecomm/web/conf/common/common-httpd-general.conf*

*Include /app/ecomm/web/conf/common/common-httpd-initialise.conf*

*Include /app/ecomm/web/conf/common/common-httpd-modules.conf*

*Include /app/ecomm/web/conf/common/common-httpd-log.conf*

*Include /app/ecomm/web/conf/common/common-httpd-security.conf*

*Include /app/ecomm/web/conf/common/common-httpd-caching.conf*

*Include /app/ecomm/web/conf/common/common-httpd-compression.conf*

*Include /app/ecomm/web/conf/ECommApps/storefront-vhosts.conf*

*Include /app/ecomm/web/conf/Redirects/storefront-redirects.conf*

*Include /app/ecomm/web/conf/ECommApps/agent-vhosts.conf*

* + 1. Apache Server Configuration – Common configuration

All Common configuration data is served from /app/ecomm/web/conf/common folder.

* common-httpd-general.conf: This file contains Apache core related Configuration Data, such as ServerRoot, ServerName, Listen, User, Group, Directory, Location
* common-httpd-initialise.conf: This file contains PIDFile, LockFile, Worker MPM settings
* common-httpd-modules.conf: This file contains LoadModule directive details.
* common-httpd-log.conf: This file set Apache logging level and log file details.
* common-httpd-security.conf: This file alters cookie values, controls HTTP response header.
* common-httpd-caching.conf: This file sets HTTP Response header with cache expiry times for different file types. Currently all are commented and can be configured during Performance tests. This is an optional configuration and is configured as a Boolean in properties files.
* common-httpd-compression.conf: This file sets compression types for various file types. This is an optional configuration and is configured as a Boolean in properties files.
  + 1. Apache Server Configuration – ATG-Persona Specific Configuration (VirtualHost)

The ATG-Specific Configuration data contains VirtualHost configuration and its directives.

* + 1. Apache Server Configuration – Redirections-Specific Configuration

This file contains all redirects for Customer-facing Storefront (Web). This is an optional configuration and is configured as a Boolean in properties files.

* + 1. DDS Scripts

The DDS scripts can categorised into three types, Setup scripts, Configuration Update scripts and Operational Scripts.

* + - 1. **Adding New Silos with Setup Scripts**

By using the override mechanism described in 7.1 it is possible to add new silos via the setup scripts to an existing environment. Without this mechanism, service would be put at risk by the setup scripts since they apply to all known servers by default.

* + - 1. **Setup Scripts:**

The main setup script, callSetup-Apache-Instance.sh, calls many other scripts to configure sectioned configuration data. Below diagram depicts the functional calls from the setup scripts.



* callSetup-Apache-Instance.sh: This scripts takes two arguments, layer and properties file name, and prepares the Apache configuration in a temporary folder and then pushes it to the target server.

USAGE:

*callSetup-Apache-Instance.sh $LAYER $PROPS*

* callPreapareApache-Config.sh: This scripts prepares the Apache configuration in a temporary folder, under name of the apache host name.

USAGE:

*callPreapareApache-Config.sh* *$LAYER $PROPS*

* callPushApache-Config.sh: This script pushes the Apache configuration to the target server and deletes the temporary folder in Deployment server.

USAGE:

*callPushApache-Config.sh $LAYER $PROPS*

* prepareApacheEnv.sh: This scripts creates temporary folder structure in Deployment server and also creates startup scripts for Apache

USAGE:

./*prepareApacheEnv.sh* *$APACHE-HOST-NAME $APACHE-HOME-DIR $ATG-PERSONA-NAME*

* prepareApacheCommonConfig.sh: This script prepares common configuration for the Apache. It also verifies whether the Apache instance is already configured on the target server or not. If the Apache instance is already configured, then it copies the httpd-ecomm.conf locally onto Deployment server and updates it, if needed. If the Apache instance doesn’t exists on the targets servers, it creates new httpd-ecomm.conf and other common configuration data. This script also verifies, if there is any environment specific common configuration in the template folder. If so, then it copies only that environment specific common configuration data to the temporary folders.

USAGE:

*./prepareApacheCommonConfig.sh $LAYER $APACHE-HOST-NAME $APACHE-PORT $APACHE-HOME-DIR $USESCACHING $USESCOMPRESSION*

* prepareApacheAppConfig.sh: This script prepares Application specific Virtual host configuration for the Apache. It also verifies whether the Apache instance is already configured on the target server or not. If the Apache instance is already configured, then it copies the httpd-ecomm.conf locally onto Deployment server and updates it, if needed. This script also verifies, if there is any environment specific common configuration in the template folder. If so, then it copies only that environment specific common configuration data to the temporary folders.

USAGE:

*./prepareApacheAppConfig.sh* *$LAYER $APACHE-HOST-NAME $APACHE-HOME-DIR* $APACHE-LISTEN-SERVER-NAME $APACHE-LISTEN-SERVER-PORT $*ATG-PERSONA-NAME* $JBOSS-LISTEN-SERVER $JBOSS-LISTEN-ADDRESS "$VIRTUAL-HOST-NAMES" "$ATG-PERSONA-ALIASES"

* + - 1. **Configuration Update Scripts**

Below two scripts should be used to update the Apache configuration, one for the common configuration and another for any ATG-Persona VirtualHost configuration.

* callUpdate-Apache-CommonConfig.sh: This script updates Apache common configuration data and then pushes the changes to the target host. This script also verifies for environment specific common configuration, if any, in the template directory. If so, it copies the data to working directory and processes further.

USAGE:

*callUpdate-Apache-CommonConfig.sh $LAYER $PROPS*

* callUpdate-Apache-AppConfig.sh: This script updates Apache Application specific configuration data, VirtualHost, and then pushes the changes to the target host. This script also verifies for environment specific common configuration, if any, in the template directory. If so, it copies the data to working directory and processes further.

USAGE:

*callUpdate-Apache-AppConfig.sh $LAYER $PROPS*

* + - 1. **Operational Scripts**

Below scripts should be used majorly by Operations team, i.e., restarts, static content deployment.

* callStop-Apache-Servers.sh: This script forcefully stops all Apache instances per environment.

USAGE:

*./callStop-Apache-Servers.sh $LAYER $PROPS*

* callStart-Apache-Servers.sh: This script starts all Apache instances per environment sequentially,

USAGE:

*./callStart-Apache-Servers.sh $LAYER $PROPS*

* callRestart-Apache-Servers.sh: This script gracefully restarts all Apache instances per environment.

USAGE:

*./callRestart-Apache-Servers.sh $LAYER $PROPS*

* callDeploy-HTDOCS.sh: This script deploys static content, which is in form of a ZIP file, to Apache servers. This script can also roll back the static content to its previous version. This script uses switching mechanism to copy the static content, aka copy the content to passive directory and relink the htdocs to passive directory, making it active. If there are more than one Apache instance, this script will copy the static content on first execution and for all other executions, it simply relinks the htdocs directory to the passive directory.

USAGE:

*./callDeploy-HTDOCS.sh $LAYER $PROPS $BACKOUT $RESTART*

* callBackupApacheConfig.sh: This script is used to archive all Apache configuration data as a tar file. This script can be used before applying any update changes or static content deployments.

USAGE:

*callBackupApacheConfig.sh $LAYER $PROPS*

* callRestoreApacheConfig.sh: This script is used to restore Apache configuration data, which is archived earlier.

USAGE:

*callRestoreApacheConfig.sh $LAYER $PROPS*

JBoss Configuration Setup

* 1. JBoss Configuration Setup Strategy
* JBoss server template, default, is used to configure all ATG-Persona specific JBoss server instances.
* Custom Port configuration is used for all ATG-Personas

1. 1. 1. JBoss Server– Vanilla Installation

Below is the JBoss vanilla installation outline.

* Directory permissions of **/app/ecomm/server/** should be770.
* Directory permissions of **/var/lib/jbossas, /var/lib/jbossas/bin**, **/var/lib/jbossas/bin/run.sh**, **/var/lib/jbossas/bin/shutdown.sh** should be750.
* Create kitsaa group with GID 570
* All the sub-directories and files under **/app/ecomm/server/\*** should be 770.
* Directory permission of **/etc/jbossas/** should be **770** and Owner of **/etc/jbossas** is **jbossadm:jboss**
* Except **“default” directory,** all other directories have to be renamed witha preceding **“old-“** in **/app/ecomm/server** and **/etc/jbossas.**
* The log link under directory in **/app/ecomm/server/default** should be deleted
* Configure the links, /usr/bin/java -> /etc/alternatives/java -> /usr/java/default/bin/java
* Configure the links, /usr/java/default -> /usr/java/latest -> /usr/java/jdk1.6.0\_45
  1. JBoss Configuration Implementation
  2. 1. JBoss Server Configuration – Security

No file in JBoss Configuration data should contain plain user credentials. The passwords are encrypted by using JBoss out-of-the-box encryption algorithm, org.jboss.resource.security.SecureIdentityLoginModule. The same algorithm is used to encrypt DB credentials too.

* + 1. JBoss Server Configuration – Port Configuration

All ATG-Persona specific JBoss server instances should be using custom JBoss port configuration with different offsets as mentioned below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **JBoss Configuration** | | | | | |
|  | **JBoss Ports Set Name** | **JBoss Ports Offset** | **HTTP Port** | **HTTPS Port** | **AJP Port** | **Remote JMX** |
| **Server Persona** |  |  |  |  |  |  |
| Storefront | ports-storefront | 0 | 8080 | 8443 | 8009 | 22222 |
| Publishing | ports-publishing | -10 | 8070 | 8433 | 7999 | 22212 |
| Staging | ports-staging | -20 | 8060 | 8423 | 7989 | 22202 |
| Fufillment | ports-fulfillment | -40 | 8040 | 8403 | 7969 | 22182 |
| Agent | ports-agent | -50 | 8030 | 8393 | 7959 | 22172 |
| Lock\_Storefront | ports-lockstorefront | -60 | 8020 | 8383 | 7949 | 22162 |
| Lock \_Agent | ports-lockagent | -70 | 8010 | 8373 | 7939 | 22152 |
| Auxilliary Store | ports-auxstorefront | -30 | 8050 | 8413 | 7979 | 22192 |
| Auxilliary Agent | ports-auxagent | 10 | 8090 | 8453 | 8019 | 22232 |

* + 1. JBoss Server Configuration – JVM Route

In order to identify the JBoss server instance, each ATG-Persona specific JBoss server.xml is updated with unique jvmRoute strings.

|  |  |
| --- | --- |
| ATG-Persona | jvmRoute String |
| Storefront | str |
| Publishing | pub |
| Staging | stg |
| Fulfillment | fft |
| Agent | agt |
| Lock-Storefront | lck-str |
| Lock-Agent | lck-agt |
| Aux-Storefront | auxstr |
| Aux-Agent | auxagt |

* + 1. JBoss Server Configuration –Timeouts

This feature has to be set accordingly with Performance test results and should be updated in specific configuration data.

Following can be updated by using DDS scripts.

|  |  |
| --- | --- |
| Parameter | Configuration Item |
| AJP Protocol Connection Timeout | server.xml |
| HTTP Protocol Connection Timeout | server.xml |
| Session Timeout | web.xml |
| Transaction Timeout | transaction-jboss-beans.xml |
| AJP Protocol Connector Max Threads | server.xml |
| HTTP Protocol Connector Max Threads | server.xml |

* + 1. DDS Scripts

The DDS scripts for JBoss setup can categorised into three types, Setup scripts, Configuration Update scripts and Operational Scripts.

* + - 1. **Setup Scripts:**

The main setup script, callSetup-JBoss-Instance.sh, calls many other scripts to configure sectioned configuration data. Below diagram depicts the functional calls from the setup scripts.



* callSetup-JBoss-Instance.sh: This script is used to create and configure the JBoss instance for ATG Personas. This script first verifies whether the JBoss instance does exist or not. If so, it deletes all content under ATG-Persona home, except ATG-DATA and recreates the configuration data.

USAGE:

*callSetup-JBoss-Instance.sh $LAYER $PROPS*

* callPrepareJBossEnv.sh: This script prepares ATG-Persona JBoss instance.
  + copies the default template to ATG-Persona name server instance
  + creates log link under ATG-Persona instance pointing to /app/ecomm/logs/$ATG-PERSONA-NAME
  + creates ATG\_DATA folder under ATG-Persona instance
  + creates bin folder under ATG-Persona instance.
  + creates a new folder, old-deploy, and moves the JBoss services which are not-required as ATG Best practices.
  + Copies all lib files under $TEMPLATE-HOME/jboss/lib to lib directory under ATG-Persona instance.

USAGE:

*callPrepareJBossEnv.sh $LAYER $PROPS*

* callUpdate-JBoss-Credentials.sh: This script is used to encrypt the JBoss console user credentials, in the file jmx-console-users.properties, by using org.jboss.resource.security.SecureIdentityLoginModule algorithm provided by JBoss.

USAGE:

*callUpdate-JBoss-Credentials.sh $LAYER $PROPS*

* callUpdate-JBoss-Ports.sh: This script is used to update the port configuration as defined above.

USAGE:

*callUpdate-JBoss-Ports.sh $LAYER $PROPS*

* callUpdate-JBoss-StartUp-Scripts.sh: This script creates/modifies ATG-Persona instance start and stop scripts. This also configures JVM arguments and classpath.

USAGE:

*callUpdate-JBoss-StartUp-Scripts.sh $LAYER $PROPS*

* callUpdate-JBoss-Startup-Scripts-Wrappers.sh: This script is used to create/modify ATG-Persona specific Start and Stop scripts to run by user “ecommadm”.

USAGE:

*callUpdate-JBoss-Startup-Scripts-Wrappers.sh $LAYER $PROPS*

* callUpdate-JBoss-JVMRoute.sh: This script is used to update the server.xml file with ATG-Persona specific JVM Route element.

USAGE:

*callUpdate-JBoss-JVMRoute.sh $LAYER $PROPS*

* callUpdate-JBoss-AJP-Connection-Timeout.sh: This script updates the connectionTimeout attribute value for AJP Protocol in deploy/jbossweb.sar/server.xml file. The default time out is set to 20000ms.

USAGE:

*callUpdate-JBoss-AJP-Connection-Timeout.sh $LAYER $PROPS*

* callUpdate-JBoss-HTTP-Connection-Timeout.sh: This script updates the connectionTimeout attribute value for HTTP Protocol in deploy/jbossweb.sar/server.xml file. The default time out is set to 20000ms.

USAGE:

*callUpdate-JBoss-HTTP-Connection-Timeout.sh $LAYER $PROPS*

* callUpdate-JBoss-Session-Timeout.sh: This script updates the session-timeout element value in deployers/jbossweb.deployer/web.xml file. The default time out is set to 30 mins.

USAGE:

*callUpdate-JBoss-Session-Timeout.sh $LAYER $PROPS*

* callUpdate-JBoss-Transaction-Timeout.sh: This script updates the transaction-timeout element value in deploy/transaction-jboss-beans.xml file. The default time out is set to 300 sec.

USAGE:

*callUpdate-JBoss-Transaction-Timeout.sh $LAYER $PROPS*

* callUpdate-JBoss-AJP-Connector-MaxThreads.sh: This script updates the maxThreads attribute value for AJP Protocol in deploy/jbossweb.sar/server.xml file. The default time out is set to 200.

USAGE:

*callUpdate-JBoss-AJP-Connector-MaxThreads.sh $LAYER $PROPS*

* callUpdate-JBoss-HTTP-Connector-MaxThreads.sh: This script updates the maxThreads attribute value for AJP Protocol in deploy/jbossweb.sar/server.xml file. The default time out is set to 200.

USAGE:

*callUpdate-JBoss-HTTP-Connector-MaxThreads.sh $LAYER $PROPS*

* + - 1. **Configuration Scripts:**

All of the above Update scripts can be run independently to update respective JBoss configuration date. Along with the above, DDS also has below configuration update scripts.

* callUpdate-JBoss-Custom-Lib.sh: This script is used to copy all the custom libraries stored under $TEMPLATE-HOME/jboss/lib on DDS server to ATG-Persona instance lib folder.

USAGE:

*callUpdate-JBoss-Custom-Lib.sh $LAYER $PROPS*

* + - 1. **Operational Scripts:**

Below scripts can be used by Operations team.

* callFlush-JBoss-Temp.sh: This script is used to delete all content under tmp and work directories for ATG-Persona instance.

USAGE:

callFlush-JBoss-Temp.sh $LAYER $PROPS

* callStop-JBoss-Servers.sh: This script is used to stop JBoss instances. This script will wait for 5 mins and will be checking if the JBoss process stops gracefully. If not, then it kills the JBoss process forcefully. This script can also stop the JBoss processes in parallel.

USAGE:

*callStop-JBoss-Servers.sh $LAYER $PROPS*

* callStop-JBoss-Servers-SILO.sh: This script is used to stop a single JBoss instances. This script will wait for 5 mins and will be checking if the JBoss process stops gracefully.

USAGE:

*callStop-JBoss-Servers-SILO.sh $LAYER $PROPS $SILO-NAME*

* callStart-JBoss-Servers.sh: This script is used to start JBoss instances. Once the start command is issued on the JBoss instance host, this script will also tests whether the application is returning a HTTP response code 200 and also the test pettern which is configured in the properties file. This script can also start the JBoss processes in parallel.

USAGE:

*callStart-JBoss-Servers.sh $LAYER $PROPS*

* callStart-JBoss-Servers-SILO.sh: This script is used to start a single JBoss instances. Once the start command is issued on the JBoss instance host, this script will also tests whether the application is returning a HTTP response code 200 and also the test pettern which is configured in the properties file.

USAGE:

*callStart-JBoss-Servers-SILO.sh $LAYER $PROPS $SILO-NAME*

* callBackupJBossConfig.sh: This script is used to backup all JBoss configuration and prepare a tar file on the JBoss instance host.

USAGE:

*callBackupJBossConfig.sh $LAYER $PROPS*

* callRestoreJBossConfig.sh: This script is used to restore the JBoss configuration from the recent backed-up tar file and replace all current content.

USAGE:

*callRestoreJBossConfig.sh $LAYER $PROPS*

* callDeploy-EARs.sh: This script deploys EAR file to JBoss instances. This script can also roll back the EAR file to its previous version.

USAGE:

*callDeploy-EARs.sh $LAYER $PROPS $BACKOUT*

* callDeploy-ATG-LocalConfig.sh: This script deploys ATG localconfig, which is in form of a ZIP file, to JBoss instances. This script can also roll back the ATG localconfig to its previous version.

USAGE:

*callDeploy-ATG-LocalConfig.sh $LAYER $PROPS $BACKOUT*

* callDeploy-EARs-SILOs.sh: This script deploys EAR file to JBoss instances. This script can also roll back the EAR file to its previous version.

USAGE:

*callDeploy-EARs-SILOs.sh $LAYER $PROPS $BACKOUT $SILO\_NAME*

* callDeploy-ATG-LocalConfig-SILOs.sh: This script deploys ATG localconfig, which is in form of a ZIP file, to JBoss instances. This script can also roll back the ATG localconfig to its previous version.

USAGE:

*callDeploy-ATG-LocalConfig-SILO.sh $LAYER $PROPS $BACKOUT $SILO-NAME*

Database Data Source Configuration Setup

* 1. Data Source Configuration Setup Strategy
* All Database user credentials for the JBoss data source should be encrypted by and no plain password should be stored in either of the JBoss configuration data.

1. 1. 1. DDS Scripts

Below DDS scripts should be used to create/modify JBoss data source details.

* + - 1. **Data Source Setup Scripts:**
* callUpdate-Application-JDBC-DataSource.sh: This script is used to create/update the atg-ds.xml file with relevant JDBC parameters, such as JNDI Name, Min pool size, Max pool size, Blocking timeout, Idle timeout, URL, SecurityDomain.

USAGE:

*callUpdate-Application-JDBC-DataSource.sh $LAYER $PROPS*

* callUpdate-Application-JDBC-Credentials.sh: This script is used to update /conf/login-config.xml with encrypted DB user credentials.

USAGE:

*callUpdate-Application-JDBC-Credentials.sh $LAYER $PROPS*

ATG Code Build, Release and Deployment Strategy

* 1. Introduction
  2. Code Compilation
  3. Release and Deployment strategy to various environments
  4. Miscellaneous activities, such as DB Deploy etc.,

Appendix – A : Properties file Arguments Description

* 1. Overriding properties

Sometimes it is useful to override the values in the properties files temporarily. e.g. when deploying to a subset of the silos within a large persona, e.g. when adding a new silo to a production layer and wanting to run the 'setup' scripts without disturbing the running system.

This can be done using the following syntax:

export ${PERSONA}\_${LAYER}\_${PROPERTY\_NAME\_WITH\_UNDERSCORES\_FOR\_DOTS}=${VALUE}

e.g. storefront\_prod\_Apache\_Htdocs\_UsesLocationSwitching=N

e.g. agent\_prod\_JBoss\_JBossListenServers="atg-pvt-ndc-app02 atg-pvt-ndc-app04"

The ../utils/getValues.sh, will look for an environment variable with the above format and use it in preference to the values in the properties files.

See the Jenkins job: Test\_Silos\_pvt\_storefront for an example.

* 1. Apache Settings:

|  |  |  |
| --- | --- | --- |
| Property Name | Value | Description |
| UsesWebServer | Y or N | This property represents whether the ATG\_Persona uses a Web server instance or not. Default is N |
| UsesWSCaching | Y or N | This property represents whether the Web server can be configured to cache or not. Default is N |
| UsesWSCompression | Y or N | This property represents whether the Web server can be configured to compress HTTP response or not. Default is N |
| HaveRedirections | Y or N | This property represents whether the Web server can be configured to use for redirections or not. Default is N |
| Apache.HttpDir | /app/ecomm/web | This property is used for configure the Apache Home Directory |
| Apache.ApacheCtl | /app/ecomm/web/bin/apachectl | This property is used to configure Apache web instance apachectl path |
| Apache.HostNames | HTTPHost1 HTTPHost2 | Apache Instance hosting Hostnames. If there are multiple Apache hosts they should be separated by a space.  e.g., atg-sqa-ndc-web01. |
| Apache.Ports | HTTPPort1 HTTPPort2 | Apache Listen port for the main instance. If there are multiple Apache instances, they should be separated by a space. Default value is 50080 |
| Apache.HttpListenServers | HTTPListenServer1 HTTPListenServer2 | Apache Instance hosting Hostnames. If there are multiple Apache hosts they should be separated by a space.  e.g., atg-sqa-ndc-web01. |
| Apache.HttpListenPorts | HTTPListenPort1 HTTPListenPort2 | Apache Listen port for each ATG-Persona. If there are multiple Apache instances, they should be separated by a space. Below are the default values for each ATG-Persona.  Storefront : 9000  Agent : 9001  Staging : 9002  Publishing : 9001 |
| Apache.VirtualHosts | VIRTUALHOST1:HTTPListenPort1 VIRTUALHOST2:HTTPListenPort2 | This property is used to configure the VirtualHost information for the ATG-Personas |
| Apache.Aliases | tp-app diy-app | This property is used to configure the aliases for ATG- Persona URLs. |

* + 1. JBoss Settings:

|  |  |  |
| --- | --- | --- |
| Property Name | Value | Description |
| JBoss.AJP.ConnectionTimeout | 20000 | The number of milliseconds this Connector will wait, after accepting a connection, for the request URI line to be presented. The default value is infinite (i.e. no timeout).  e.g. 600000 (600 seconds or 10 minutes) for pvt |
| JBoss.AJP.ConnectorMaxThreads |  |  |
| JBoss.ATGArguments |  |  |
| JBoss.DynaTraceServer |  |  |
| JBoss.EARFileName |  |  |
| JBoss.GCArguments |  |  |
| JBoss.HeapArguments |  |  |
| JBoss.HTTP.ConnectionTimeout |  |  |
| JBoss.HTTP.ConnectorMaxThreads |  |  |
| JBoss.HTTP-Port |  |  |
| JBoss.IsDynaTraceEnabled |  |  |
| JBoss.IsParallelRestart |  |  |
| JBoss.JBossListenPorts |  |  |
| JBoss.JBossListenServers | JBOSSLISTENSERVER1 JBOSSLISTENSERVER2 | This property is used to configure JBoss host names.  e.g., atg-sqa-ndc-app01 |
|  |  |  |
| JBoss.JDBC.blocking-timeout-millis |  |  |
| JBoss.JDBC.idle-timeout-minutes |  |  |
| JBoss.JDBC.jndi-name |  |  |
| JBoss.JDBC.max-pool-size |  |  |
| JBoss.JDBC.min-pool-size |  |  |
| JBoss.JDBC.Passwords |  |  |
| JBoss.JDBC.security-domain |  |  |
| JBoss.JDBC.URL |  |  |
| JBoss.JDBC.Users |  |  |
| JBoss.JNDI-RMI-Port |  |  |
| JBoss.JVMArguments |  |  |
| JBoss.JVMRoute |  |  |
| JBossListenPorts |  |  |
| JBoss.LocalConfigFileName |  |  |
| JBoss.Password |  |  |
| JBoss.RunAsUser |  |  |
| JBoss.Services2BRemoved |  |  |
| JBoss.SessionTimeout |  |  |
| JBoss.ShutdownScript |  |  |
| JBoss.StartupScript |  |  |
| JBoss.TestMatchText |  |  |
| JBoss.TestURI |  |  |
| JBoss.TransactionTimeout |  |  |
| JBoss.UserArguments |  |  |
| JBoss.UserName |  |  |