

Agile Requirements Designer Hub Best Practices

Marta Benach

Broadcom Limited

Web: www.broadcom.com

Corporate Headquarters: San Jose, CA

© 2018 by Broadcom All rights reserved.

Revision History

Revision	Date	Change Description
1.0	Nov 8, 2020	Initial draft

Overview

Agile Requirements Designer (ARD) is a collaborative Model Based Testing tool which allows users to create visual representations of application requirements and to generate test cases, test data and test automation scripts in a fast and optimized fashion.

ARD 2.9 introduced the ARD Hub which provides easy collaboration, a web based GUI and optimized flow storage for ARD models.

The ARD Hub optimized the storage of the ARD models, significantly decreasing the time required to access models, and decreasing the size of the model vtf files. There is a 100x performance increase in loading models, saving models and inserting subflows. Additionally, storage has been redesigned so that very large flows, flows with large numbers of subflows, and versions are more efficiently handled and stored.

ARD 3.1 has continued to add significant functionality to the ARD hub including

- Version Compare functionality
- Improved and more granular permissions
- Integration with OpenID
- More granular details about Application links

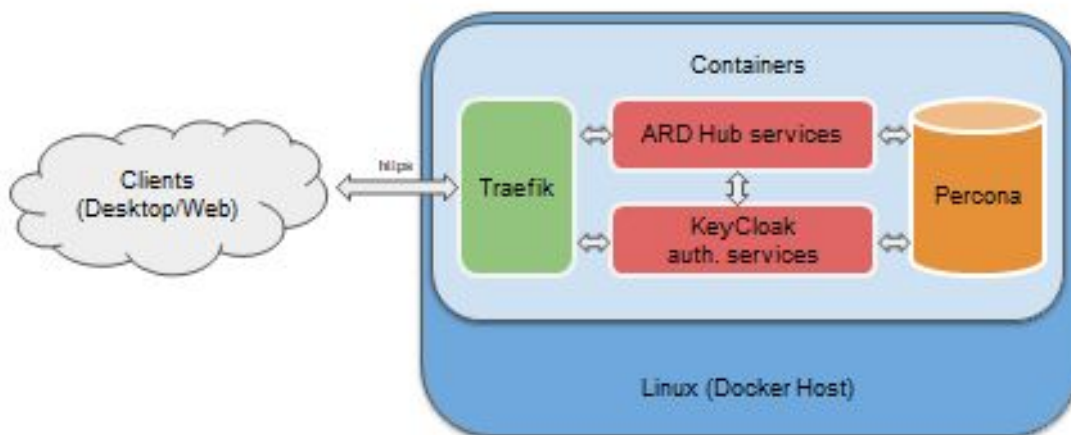
ARD Marquee Features

Feature	ARD 2.9	ARD 2.10	ARD 3.0	ARD 3.1
Hub	x	x	x	x
Projects	x	x	x	x
Versions	x	x	x	x
Active Directory Support	x	x	x	x
Requirements Insights		x	x	x
New licensing model			x	x
Docker			x	x

installation process				
User Permissions			X	X
Flow Migration			X	X
SubFlow Loading and Change Handling			X	X
Multiple Automation Configuration Support				X
Version Compare				X
Optimizer Enhancements				X
qTest Integration				X
Open ID Integration				X
Smart Test Case Description Generation				X

Hub Architecture

ARD 3.1 Architecture - Docker

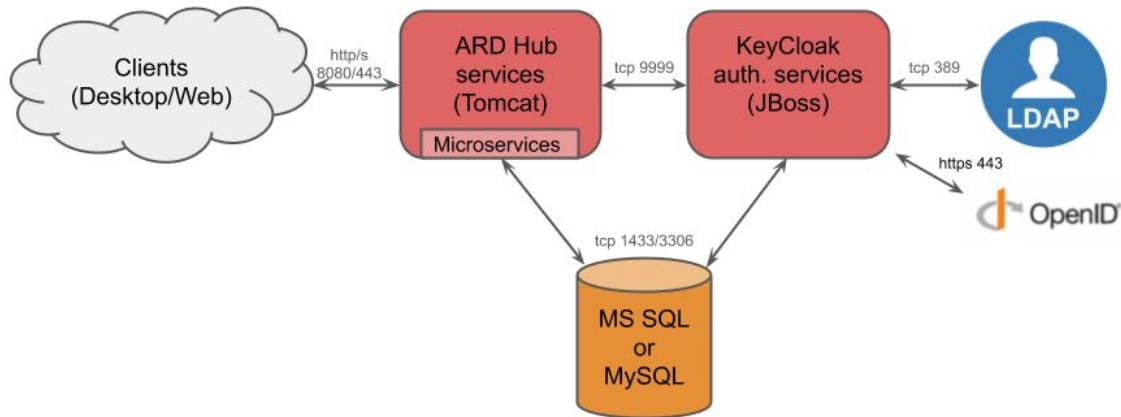


There are multiple docker containers that comprise the ARD Hub:

- Ard/Hub container, which runs tomcat and the ARD Hub war files.
- jBoss/Keycloak - which runs keycloak and provides identity management services
- jBoss/Keycloak/Aux - works with jBoss/Keycloak container
- Percona Database 5.6 - open source database
- Traefik - load balancer and reverse proxy which is only utilized if the Corporate Proxy is enabled

Non Docker:

ARD 3.1 Architecture - Non Docker



If you choose to do a manual install, all of the required software must be manually installed, including OpenAdopt JDK, Tomcat 9, Keycloak 5 and an MSSQL or MySQL database and appropriate database driver.

Hub Installation

The recommended method of deploying the ARD hub is via Docker. There are multiple docker containers that comprise the ARD Hub. They are easily deployed by running the `ard.sh` script. This script is very flexible and makes it easy to deploy the hub with multiple configuration options including doing an offline install from a private repo, setting up SSL, configuring a proxy or changing the default port. The documentation for installing ARD Hub with Docker can be found here:

<https://techdocs.broadcom.com/us/en/ca-enterprise-software/devops/agile-requirements-designer/3-1/installing/install-ard-hub/install-ard-hub-using-docker.html>

Non Docker Manual installation on Windows and Linux are also supported. Details for manual installation can be found here:

<https://techdocs.broadcom.com/us/en/ca-enterprise-software/devops/agile-requirements-designer/3-1/installing/install-ard-hub/install-ard-hub-manually.html>

Administering/ Maintaining the Hub

It is important to define Application owners and ARD System Administrators for the ARD Hub. Application owner and Application Sys Administrator can be the same person/team but it is important to note the different responsibilities. Application owners will have overall responsibility for maintaining and administering Agile Requirements Designer Hub and the ARD Desktop Client, including managing Desktop client distribution. Application System Administrators are responsible for managing the application's day to day activities including onboarding users, and managing projects. Application System Administrators are responsible for working with Application owners for initial installation and upgrades, and configuring LDAP/OpenID.

Determining the appropriate process for delivering the desktop client is also an important consideration. ARD Desktop client distribution can be done via a packaging app for automatic delivery to practitioners. It is important to factor in the time required for packaging when planning upgrades to the hub or desktop client. If automatic delivery cannot be done, it is best to create a shared location for practitioners to access the Desktop Client application and instructions on how and when to install or upgrade.

Application owners should ensure that the ARD Hub and all of its components, ie databases are properly monitored and backed up. At a minimum, the servers should be monitored for free disk space, CPU usage, memory usage and network connectivity. At a minimum, daily backups should be taken of the ARD Hub Flowstorage and Keycloak databases and a proper restoration procedure should be defined and documented.

It is a best practice to create both a test and a production instance of the ARD hub. The test instance will be used for testing out new versions, upgrades and fixes before they are deployed to the production environment.

Test and production systems should be as similar as possible including using the same database types, application configurations and servers types (same OS).

Upgrading to ARD 3.1

To upgrade ARD Hub 3.0 to 3.1,

- Upgrade ARD Hub to 3.1

- Upgrade ARD Studio 3.1

To upgrade ARD Hub 2.10 and earlier to 3.1,

- Upgrade ARD Studio to 3.0,
- Upgrade ARD Hub to 3.1,
- Upgrade ARD Studio to 3.1

Third Party Requirements:

ARD 2.9	ARD 2.10	ARD 3.0	ARD 3.1
Tomcat 8	Tomcat 8	Tomcat 9	Tomcat 9
Java 8.5	Java 8.5	Open JDK 11	Open JDK 11
-	-	Keycloak 5	Keycloak 5

For the upgrade it is best practice to upgrade the test environment and perform baseline testing in this environment before upgrading the production environment.

Before beginning the upgrade, a database back-up should be created. It is also a best practice to create a back-up of the ARD directory.

Follow the upgrade procedure as described in the documentation.

<https://techdocs.broadcom.com/us/en/ca-enterprise-software/devops/agile-requirements-designer/3-1/installing/install-ard-hub/upgrade-or-reconfigure-ard-hub.html>

Migration

The ARD Hub introduced a fundamental change in the method models are stored, because of this it is necessary to use the migration process on existing models when moving to the hub.

The change in how ARD handles model storage has the benefit of significantly improving performance when opening, saving, updating and closing models. It also has reduced the size footprint of the model files significantly. The current sizing is that each flow will roughly consume 1MB of disk space.

Previous to the introduction of the ARD Hub, models could be stored in 3 different ways. They could be stored locally, in shared files or in the Test Data Manager (TDM) repo. Migration is the process for moving existing Models into the ARD Hub for the first time. Models can be migrated from the TDM repo, local or shared files. Migration is a one time cost and is NOT required for any NEW models created in the new version. For example, new models can be saved directly into the ARD Hub using the save feature.

It is a best practice to develop a migration plan when deploying the ARD hub for the first time. The plan should include the following information

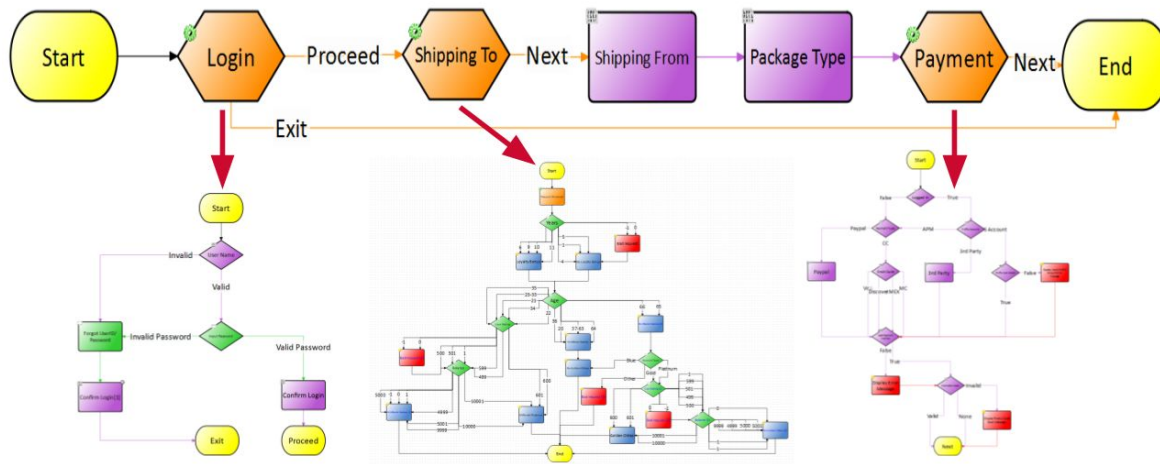
- Who owns the migration process
- Understanding how many flows will be migrated
- Identifying the most critical flows to migrate and the order in which to migrate them
- Creating an appropriate project structure in the ARD Hub (more on this below)
- Verifying that all of the flows are accessible to the ARD Server (particularly flows that are stored locally or in shared drives.)
 - Verify that both Parent and Child flows are accessible to the ARD Hub Server
 - For example, verify that paths to subflows aren't on local drives that the ARD hub server cannot access

There are some important considerations about the migration process. The biggest difference in ARD is how the ARD hub manages flows and subflows. In particular, parent flows now reference subflows by GUID instead of including the whole subflow as part of the parent flow file. This works to substantially reduce the size of the parent model and provides the performance improvements in opening, saving and closing models. During migration, the subflows of a parent flow are automatically migrated as well and the subflow references are automatically updated.

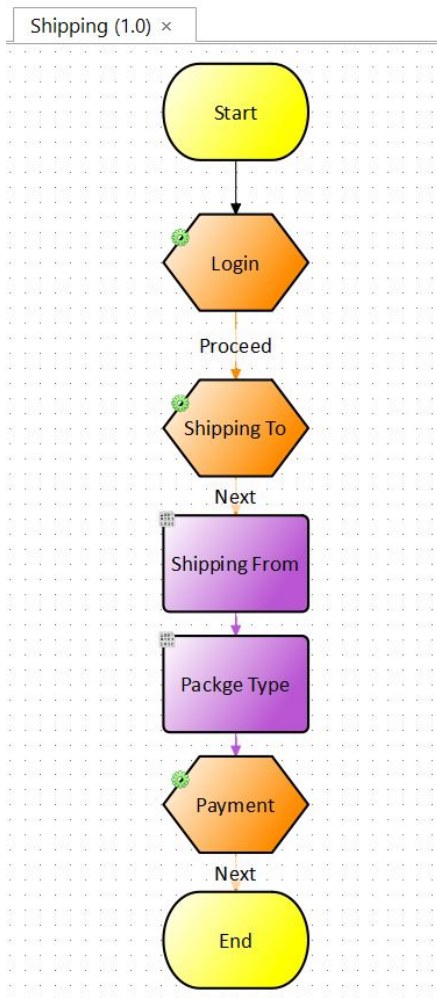
In the following picture and model, the Shipping Model is the parent and it has 3 children or subflows, Login, Shipping To and Payment. When the Shipping model is migrated, it's 3 children will also be migrated and in the process, the application links to the subflows for all three children will be updated to reflect the ARD Hub.

Scalable Modeling Solution

Efficient representation of complex flows via abstraction through *subflows*




6 | Broadcom Proprietary and Confidential. Copyright © 2018 Broadcom. All Rights Reserved. The term "Broadcom" refers to Broadcom Inc. and/or its subsidiaries.



Broadcom Proprietary and Confidential. © 2018 Broadcom. All rights reserved.

You can verify the application links have been updated by

- Opening the parent flow
- Right clicking on the subflow
- Selecting edit properties
- Selecting Application Links
- Verify that the Subflow Details show the path to the ARD Hub.

 Login

General

Subflow Decision

Subflow Parameters

Output Details

Test Data

Automation

Make System Data

Find System Data

Images

Custom Fields



Properties





Stored Paths


Application Links

Requirements IDs

Add links to external applications (e.g. HP ALM)

	Application	Object Type	Details	Browse	Edit	Reload	View
1	 Subflow	Flow	Subflow: "Login" From: "Shipping/1.0/Login" ARD Hub Version: "1.0" Path Type: "Test Cases"				



It isn't necessary to do this verification for every flow, but it is a good sanity test at the beginning of the migration process. Once the Shipping flow and its subflows have been migrated, they will show up in the ARD Hub Connector panel, in the project and version specified during migration.

Connectors Dock

Shipping

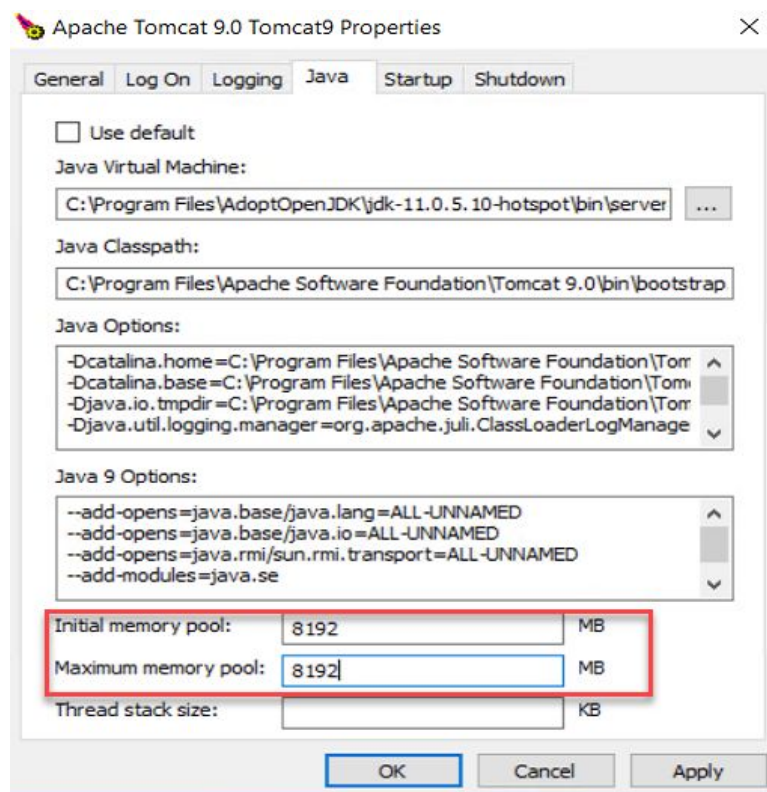
ARD Hub: <http://lvntest006633.bpc.broadcom.net:8080>

- Shipping (1)
 - 1.0 (4)
 - Login
 - Payment
 - Shipping
 - Shipping To

When models are migrated, the latest version of the model is migrated. No previous versions will be migrated. In the case where a model has a link to an earlier version, the best practice is to save the model under a new name perhaps with the original name and the version number in the name and migrate it.

Migration requires an increase in the Tomcat Java Heap Size to at least 8GB. This memory increase is only required during migration, and can be reduced after migration is complete. 8 GB should be sufficient for the migration process, however, if there are flows with multiple nested subflows, it may be required to increase the memory even further. For example, extremely complex models with multiple nested children like a parent flow that includes child, grandchild and great grandchild subflows may require up to 12 GB of memory.

Setting the memory for the migration is accomplished in different ways depending on the installation method used for the ARD hub. The ARD Hub docker installation has a flag, `--jvm-heap-max-size` which can be used to set the memory. For ARD Hubs that are manually installed on windows it is a best practice to install Tomcat as a service and use the Configure Tomcat utility to set the memory as shown below.

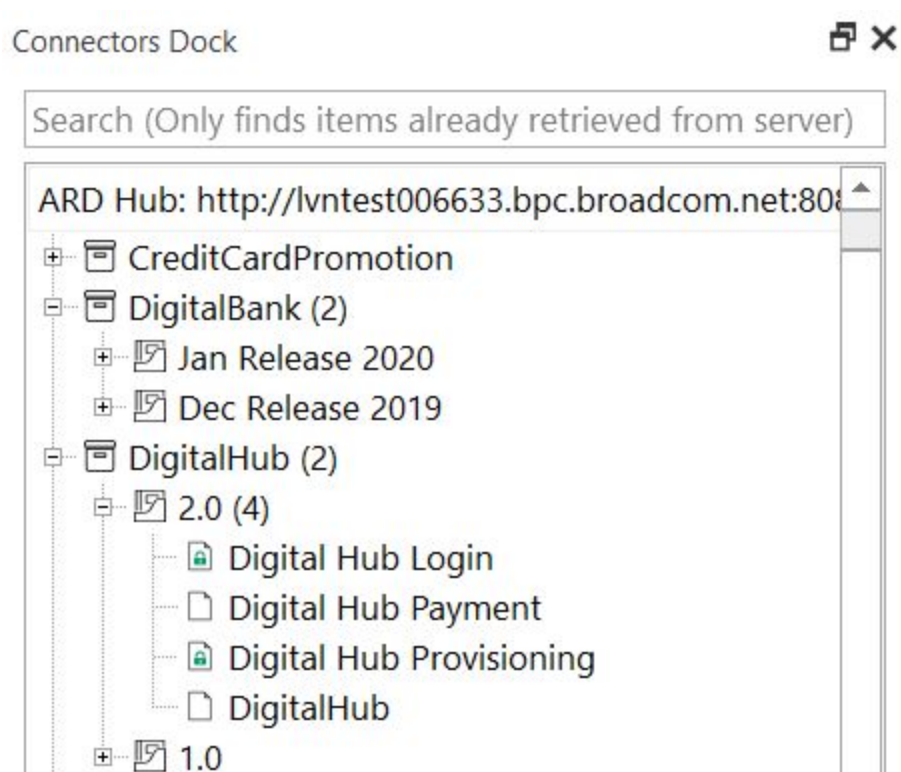


In order to complete the migration plan, it is recommended to understand and organize the hub which is covered in the next section.

Organizing the Hub

Projects are the foundation of the Hub organizational structure. Projects should contain models that are related. For example, Projects can map to Applications, Releases or Teams or Business logic/requirements or a combination of those listed. Projects have versions and thus models that should be versioned together, should be in the same project. When a project is created, the initial version is created as well. Version names are also flexible and can be version numbers or releases, for example.

In the below screenshot, there are 3 projects shown. DigitalHub and Digital bank are applications and CreditCardPromotion is a specific business requirement.



The project structure is flexible and there is no one best practice on how to structure projects in the Hub. This will largely depend on the types of models being created and their logical grouping.

It is important to have an appropriate naming convention for projects, versions, sub folders and models. This will help with organization and the collaboration amongst varying teams and team members.

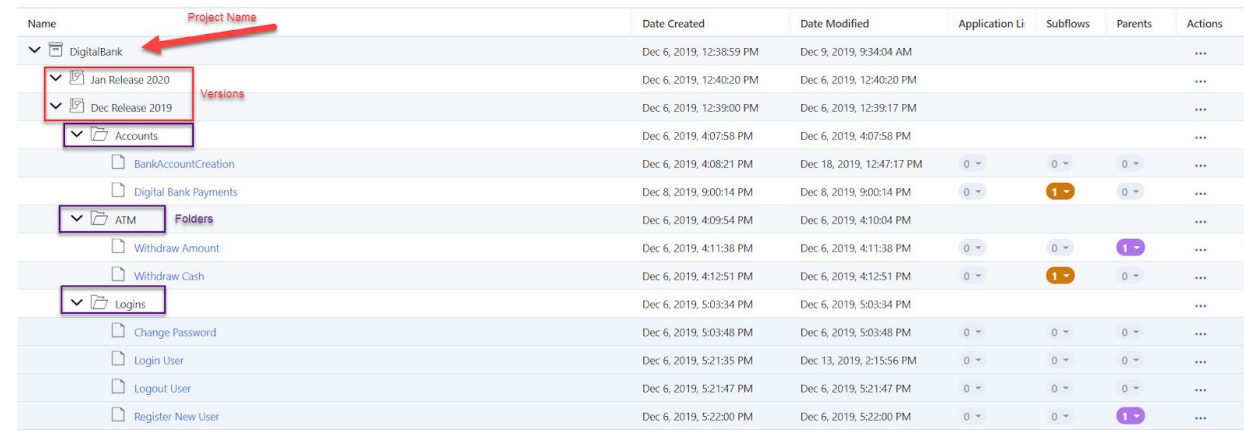
Projects are created by System Administrators and ARD System Administrators have permissions to create new projects in the hub. System Administrators should also assign project owners to the projects. The role of the project owner is to manage user and user access to the project. Project owners are also responsible for creating versions, deleting versions or renaming them. A Project owner should be someone who understands the application or project in depth, understands the release cadence and understands the teams and stakeholders who are involved. Project owners should be available to project practitioners to discuss any issues that arise and they should work with System Administrators in the event of any issues with the application or its data.

It is important to understand how and when to create new versions. New versions can be created

- To correspond to a new release, for example
- Or a major change in functionality

When a new version is created all of the contents of the existing version are duplicated into the new version. These new versions are independent of the original versions and can thus be worked on completely independently and simultaneously. For example, a parent model in version 1 will link to a version 1 child flow. Once the new version, 2, is created the version 2 parent flow will have application links to the version 2 child flow and changes made in version 2 will not be in version 1 and vice versa. It is possible to link models across versions - so a parent in version 1 can link to a child in version 2 but there should be a good reason for doing this. Typically it is better to keep versions intact and internally linked.

In the below screenshot, the DigitalBank project has 2 versions. Dec Release 2019 and Jan Release 2020. It also has subfolders created underneath to keep things organized. There are 3 subfolders, Accounts, ATM, Logins which help identify and catalog models in the hub according to functionality. This is just one example of how to use version names and subfolders to organize the hub data. Another common practice is to create versions according to release numbers for applications, or according to workstreams.



Name	Date Created	Date Modified	Application Li	Subflows	Parents	Actions
▼ DigitalBank	Dec 6, 2019, 12:38:59 PM	Dec 9, 2019, 9:34:04 AM				...
▼ Jan Release 2020	Dec 6, 2019, 12:40:20 PM	Dec 6, 2019, 12:40:20 PM				...
▼ Dec Release 2019	Dec 6, 2019, 12:39:00 PM	Dec 6, 2019, 12:39:17 PM				...
▼ Accounts	Dec 6, 2019, 4:07:58 PM	Dec 6, 2019, 4:07:58 PM				...
BankAccountCreation	Dec 6, 2019, 4:08:21 PM	Dec 18, 2019, 12:47:17 PM	0 -	0 -	0 -	...
Digital Bank Payments	Dec 8, 2019, 9:00:14 PM	Dec 8, 2019, 9:00:14 PM	0 -	1 -	0 -	...
▼ ATM	Dec 6, 2019, 4:09:54 PM	Dec 6, 2019, 4:10:04 PM				...
Withdraw Amount	Dec 6, 2019, 4:11:38 PM	Dec 6, 2019, 4:11:38 PM	0 -	0 -	1 -	...
Withdraw Cash	Dec 6, 2019, 4:12:51 PM	Dec 6, 2019, 4:12:51 PM	0 -	1 -	0 -	...
▼ Logins	Dec 6, 2019, 5:03:34 PM	Dec 6, 2019, 5:03:34 PM				...
Change Password	Dec 6, 2019, 5:03:48 PM	Dec 6, 2019, 5:03:48 PM	0 -	0 -	0 -	...
Login User	Dec 6, 2019, 5:21:35 PM	Dec 13, 2019, 2:15:56 PM	0 -	0 -	0 -	...
Logout User	Dec 6, 2019, 5:21:47 PM	Dec 6, 2019, 5:21:47 PM	0 -	0 -	0 -	...
Register New User	Dec 6, 2019, 5:22:00 PM	Dec 6, 2019, 5:22:00 PM	0 -	0 -	1 -	...

It is a best practice to set proper permissions for the projects in the ARD Hub. Permissions allow users to control access to sensitive information and data in the application. They also prevent unauthorized people from creating, deleting or editing projects, versions and models within the application. There are 2 types of project permissions, public and private. Public projects are accessible to any ARD practitioner in the system. Public projects are good for example or sample models that are made available for reference. It is conceivable to create public projects which contain models and modelling standards for practitioners to reference. Outside of these projects, it is best practice to make projects private and explicitly assign appropriate access to team members and stakeholders.

It is best practice to assign the least amount of permissions to a user as is necessary to accomplish their work. For example, ARD users who do not create or edit models but do view them for collaboration should be given viewer permissions. This is read-only access that allows users to view the model but not make any changes.

ARD users that are responsible for building, editing and creating models should be given editor privileges within their project. This allows them to create new models, and make changes to existing models. They also have the ability to create, edit and delete folders within a project.

The below screenshot gives detailed information on the permissions available in the ARD Hub and more information about permissions can be found in the documentation.

<https://techdocs.broadcom.com/us/en/ca-enterprise-software/devops/agile-requirements-designer/3-1/requirements-insight/manage-ard-users.html>

Permissions

	System Administrator	Project Admin	System User	Project Editor	Project Viewer
Permissions	-				
Users					
Add/Remove System Administrators	✓				
Manage User Access to Projects	✓	✓			
Projects					
View Projects	✓	✓		✓	✓
Create Projects	✓				
Rename Projects	✓				
Delete Projects	✓				
Versions					
View Versions	✓	✓		✓	✓
Create Versions	✓	✓			
Rename Versions	✓	✓			
Delete Versions	✓	✓			
Update Versions' Subflow Link	✓	✓			
Folders					
View Folders	✓			✓	✓
Create Folders	✓	✓		✓	
Rename Folders	✓	✓		✓	
Move Folders	✓	✓		✓	
Delete Folders	✓	✓		✓	
Flows					
View Flows	✓	✓		✓	✓
Create Flows	✓	✓		✓	
Edit Flows	✓	✓		✓	
Lock Flows / Unlock Own Flows	✓	✓		✓	
Unlock Any Flows	✓	✓			
Rename Flows	✓	✓		✓	
Move Flows	✓	✓		✓	
Delete Flows	✓	✓		✓	
Migrate Flows	✓	✓		✓	
Generate Flow Image	✓	✓		✓	

Conclusion

The ARD Hub brings better management including optimized storage, increased collaboration and model optimization to ARD practitioners. This document details the best practices for installing and running the ARD Hub in order to receive the maximum benefit from the hub. Adherence to the practices outlined in this document will ensure a smooth experience with the ARD Hub.