BAM 12.1.3 Project Artifact File Management Practices

Overview

This document informally describes best practices for handling and managing BAM file system artifacts for Data Objects, Projects, etc. Every project and development team has different needs and operating procedures, so use this document as a guideline to design your workflow.

Managing BAM Artifacts

A BAM Composer project has many artifacts associated -- its Data Objects, Enterprise Message Sources, Queries, Business Views, KPIs, Parameters and Dashboards. Managing these artifacts correctly in your source control will help you maintain consistency in future updates to your project.

When designing your source control layout, you want to separate artifacts into the following general categories:

- Shared Artifacts
  - Data Objects – data objects shared by multiple projects
    - Physical Data Objects
    - Logical Data Objects (these usually join multiple Physical Data Objects)
  - EMS – Enterprise Message Sources shared by multiple projects
- Project specific artifacts
  - Data objects – data objects used by only one project
    - Physical Data Objects
    - Logical Data Objects (these usually join multiple Physical Data Objects)
  - Project – the actual project itself

Configuring BAMCommand

1. Edit $MW_HOME/soa/bam/bin/BAMCommandConfig.xml and set the existing properties as well as add the following properties

   a. <username>username</username>
   b. <password>password</password>
   c. <dbpassword>dbpassword</dbpassword>

Using BAMCommand To Export Your Project

When you use BAMCommand to export a project, the project will contain the project artifacts as well as referenced data objects. When designing your source control, you need to be aware that the project export zip will contain data objects, and must plan to extract them into a separate zip, or delete them if you are exported data objects explicitly).

Below is an abbreviated list of artifacts contained within the zip file exported by bamcommand.

- dataobject/

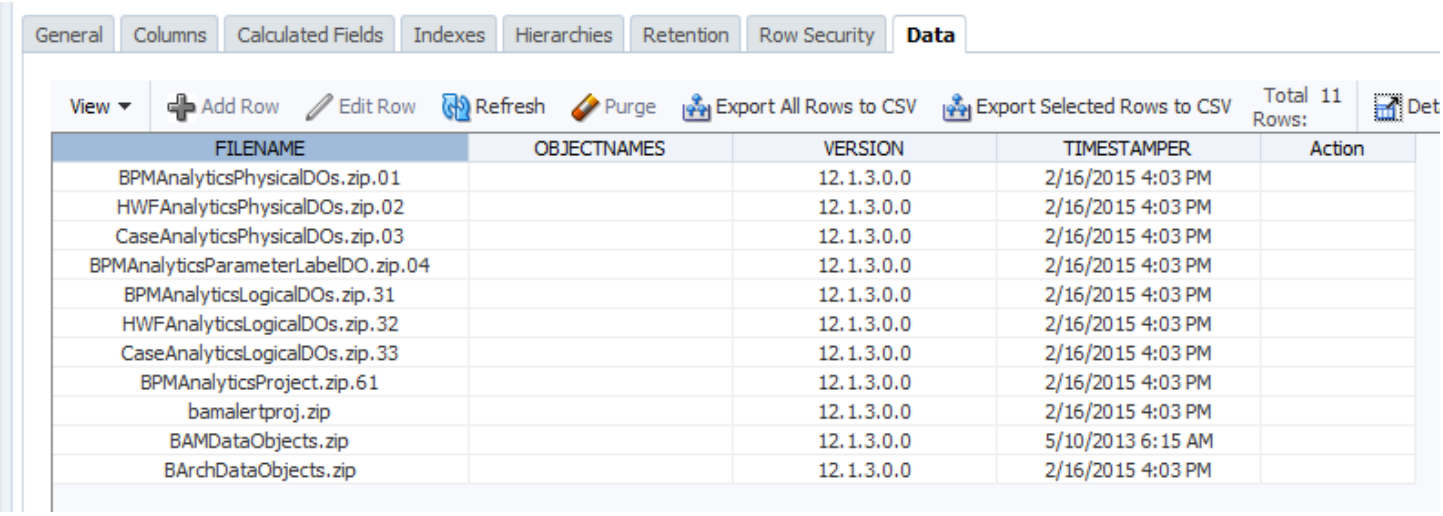
    - DATAOBJECT_NAME/
        - DataObject.xml – dataobject defintiion
        - data.csv – dataobject contents
- ems/
    - EMS_NAME/*
- project/
    - PROJECT_NAME/
        - parameter/*
        - businessquery/*
        - view/*
        - kpi/*
        - dashboard/*
        - alert/*

Below is a list of commonly used BAMCommand parameters and their result:

`-cmd export -type project -name "project name" -file export.zip`	Project export
`-cmd export -type project -regex "project name" -file export.zip`	Project export based on regular expression
`-cmd export ….  -contents 0`	Project export without data object contents
`-cmd export -type dataobject -name "DO_NAME" -contents 0`	Dataobject export without contents
`-cmd export -type ems -name "EMS_NAME"`	EMS export
`-cmd import -file import.zip -mode overwrite`	Import, overwriting existing artifacts
`-cmd import -file import.zip -mode update`	Import, keeping existing artifacts and adding new ones

Using BAM Preseeding to Import Your BAM Artifacts

On every startup, the BAM server will compare the file list at $MW_HOME/soa/bam/preseeding with the contents of dataobject oracle/bam/internal/PreseedingFileHistory. See below:



FILENAME	OBJECTNAMES	VERSION	TIMESTAMPER	Action
BPMAnalyticsPhysicalDOs.zip.01		12.1.3.0.0	2/16/2015 4:03 PM	
HWFAnalyticsPhysicalDOs.zip.02		12.1.3.0.0	2/16/2015 4:03 PM	
CaseAnalyticsPhysicalDOs.zip.03		12.1.3.0.0	2/16/2015 4:03 PM	
BPMAnalyticsParameterLabelDO.zip.04		12.1.3.0.0	2/16/2015 4:03 PM	
BPMAnalyticsLogicalDOs.zip.31		12.1.3.0.0	2/16/2015 4:03 PM	
HWFAnalyticsLogicalDOs.zip.32		12.1.3.0.0	2/16/2015 4:03 PM	
CaseAnalyticsLogicalDOs.zip.33		12.1.3.0.0	2/16/2015 4:03 PM	
BPMAnalyticsProject.zip.61		12.1.3.0.0	2/16/2015 4:03 PM	
bamalertproj.zip		12.1.3.0.0	2/16/2015 4:03 PM	
BAMDataObjects.zip		12.1.3.0.0	5/10/2013 6:15 AM	
BArchDataObjects.zip		12.1.3.0.0	2/16/2015 4:03 PM	

If a file does not appear in this list, it will be automatically imported. The import order is determined by the numeric suffix. Files with lower numbers will be imported first. Files without any numeric suffix will be imported afterwards, but in no guaranteed order. **Numeric suffixes 0-200 are reserved for BAM internal use only.**

Splitting Your Export ZIP (Process Analytics Example)

The export zip can be split apart and re-imported separately. For example, you can separate dataobject and project directory into two zip files, and import them. Import will fail if dependent artifacts are missing, so zips must be imported in the correct order. You can see the Process Analytics zip structure in the above screen shots. We split the export zip into separate project categories using a shell script that includes a manifest. Note that gaps in the numeric ordering – this is to allow for future expansion. The following table includes notes on why certain files are split. (*Note: BPMAnalytics is Process Analytics*).

BPMAnalyticsPhysicalDOs.zip.01	Contains physical DOs for Process Analytics. Shared with all Logical DOs.
HWFAnalyticsPhysicalDOs.zip.02	Contains physical DOs for HWF. Used only by HWF Logical DOs.
CaseAnalyticsPhysicalDOs.zip.03	Contains physical DOs for Case. Used only by Case Logical DOs.
BPMAnalyticsParameterLabelDO.zip.04	Contains Process Analytics specific DO and data.
BPMAnalyticsLogicalDOs.zip.31	Contains LogicalDOs for Process Analytics. Calculated fields can be edited here without locking other projects.
HWFAnalyticsLogicalDOs.zip.32	Contains LogicalDOs for HWF. Shared with Process Analytics and HWF projects.
CaseAnalyticsLogicalDOs.zip.33	Contains LogicalDOs for Case. Used by Process Analytics and Case projects
BPMAnalyticsProject.zip.61	Contains Process Analytics Project artifacts.
HWFAnalyticsProject.zip.62	*Contains HWF project (not included in 12.1.3)*
CaseAnalyticsProject.zip.63	*Contains Case project (not included in 12.1.3)*
bamalertproj.zip	Contains standard seeded alert for Process Analytics, and import order doesn't matter. Separated from project to more easily change alert definition.
BAMDataObjects.zip	contains standard BAM DOs (import order doesn't matter)
BArchDataObjects.zip	contains standard Business Architecture DOs (import order doesn't matter)

Versioning

We do not provide any versioning info within the zip file itself. However, here are some options for embedding versioning:

1. Including versioning info within zip file name. This also has the advantage of showing in the PreseedingFileHistory the version of the file that was preseeded. The older versions should be removed from preseeding directory, otherwise all versions will be preseeded.

2. Include zip file comments. These comments can be set on a per-file basis, so this may be a good way to give more fine grained versioning and commenting ability.

Best Practices When Working With Split Export ZIPs

This approach requires development to be aware of the interdependencies between zip files. Because you are working with zip files you cannot rely on diffs. Source controlling individual files from the export zip is not practical, due to number of interdependencies.

- One person per project. If you need multiple people to work on the same project, we recommend setting up a shared server. Once work is completed, one designated developer can do the source control. You will want to keep detailed source control comments.

- If needed, you can merge two zip files. Make sure there is no overlapping artifacts. You can use any zip utility to combine them.

- Updating calculated field will require you to also save and update your project queries and views.

- Adding physical data object column may require you to add the column in the corresponding logical DO, so you will have to also source control those as well. If there are no calculated field or query changes, you do not need to update project zip.

Splitting Project Directory Into Separate ZIPs

We not NOT recommend splitting the project level zip into query, view, and dashboard artifacts at the directory level. There are too many interdependencies between these artifacts that would require an unmanageable level of coordination between developers.

Handling Regressions

Sometimes, a project is source controlled with a bad artifact mistakenly included that causes a regression from a previous checked in project version. You can import an older project version with “-mode overwrite” to rollback on your server instance.

To fix this in source control, you should use a known good copy to transfer the ‘good’ artifacts into your latest project zip. After re-importing, re-saving, and re-testing, you can check in the new fixed project.