Syniti Database Versioning Tool

User Manual

BackOffice Associates, LLC DBA Syniti
info@Syniti.com
www.Syniti.com

Table of Contents

Introduction **1**

Terminology **2**

Components (Folder) 2

Definitions (Folder) 2

Operations (Folder) 2

Operations 3

Export 3

Install 3

Import 3

Update 3

Databases (Folder) 3

Set up a New Extract or Install Package **4**

Example 1: Database Extract 5

Example 2: Custom WebApp Extract 6

Example 3: Content Extract 7

Run the Database Versioning Tool **8**

Command Line 8

Batch File 8

Syniti DBVT Input Files **10**

Operation Files 10

Database Files 11

XML File Structure for use with Baseline or Object Definitions 11

XML File Structure for use with Component Definitions 12

Example 1: Baseline or Objects Definitions 13

Example 2: Component Definitions 14

Definition Files 15

Baseline Definition 15

Example Baseline Definitions 16

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

Objects Definition 17
Component Definition 19

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

Introduction

The Syniti Database Versioning Tool (Syniti DBVT) is an application that was developed by Syniti to extract data and SQL objects from databases and to install them on other SQL Server instances. The extraction process saves the data and SQL scripts into files within a well-organized folder structure. This process enables tools such as GitHub to be used to version manage those files.

This document describes the key concepts required to manage the following scenarios:

1. Databases
    - Extraction of database SQL objects (tables, views, procedures, functions)
    - Extraction of database table data
    - One-time installation of extracted database objects and data
2. Custom DSP WebApps
    - Extraction of custom DSP WebApp SQL objects (tables, views, procedures, functions)
    - Extraction of custom DSP WebApp data
    - Extraction of custom DSP WebApp components (WebApp, Pages, PageColumns, etc) from the CranSoft database
    - One-time installation of extracted database objects and data

IMPORTANT! Full application lifecycle management that includes upgrades and uninstallations is beyond the scope of this document.

3. DSP Application Content
    - Extraction of reusable data (Content) from standard DSP WebApps
    - Installation of extracted reusable data (Content) into other DSP instances

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

Terminology

The following sections outline common terminology used throughout this document. The word "(folder)" appears in titles to indicate that these are folders used with the Syniti DVBT scenarios.

Components (Folder)

A *component* is a collection of application data (table registrations) and/or SQL scripts that define a database schema that is generated when the Syniti DBVT uses the Export operation. The SQL contained within a component belongs to a single application; however, the data can belong to one or more applications. The contents of a component are driven by an associated application/database definition file that is called by an operation file when the Syniti DBVT runs.

Components that relate to application data (table registrations) are logically grouped into sub folders by specifying an instance in the application database XML file. An instance allows variables with specific values to be defined. When the Syniti DBVT runs, the operation files enable the instance of the run to be selected and hence allows specific data records (driven by the instance variable values) to be extracted.

Definitions (Folder)

Definitions are XML files in which the tables/views/procedures and data export instructions are defined. There are three types of Definitions:

- **Baseline Definition (XML file)**—The Baseline Definition is a standard Syniti DBVT definition that is specifically used to extract table schema information and data from a database. Baseline Definitions cannot be used in conjunction with Instances/Instance Variables (specified in Database.xml file)
- **Objects Definition (XML file)**—The Objects Definition is a standard Syniti DBVT definition that is used to extract view, procedure and function schema information from a database. Object Definitions cannot be used in conjunction with Instances/Instance Variables (specified in Database.xml file)
- **Component Definitions (XML file)**—A Component Definition contains a collection of logically grouped tables within a single database where data needs to be extracted. A component definition does not enable any database schema related information to be extracted. A Component Definition can be given any name and can be used in conjunction with Instances/Instance Variables (specified in Database.xml file). This feature enables a single definition to be reused for extracting.

Operations (Folder)

The *Operations* folder contains one or more configuration files that specify what Syniti DBVT operations are run against a specific application or database based on a specific definition and instance. (The instance value is optional for Component Definitions and should not be used for Baseline or Object Definitions since they do not support instances.) A single operation file can contain multiple operations and thus enable many extract or installation tasks to be strung together.

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

Operations

The following key operations are relevant to the scenarios defined in this document:

- Export
- Install
- Import
- Update

Export

The export operation is used to extract SQL schema information (views, stored procedures and functions) and data from a given database.

The export operation can be used with the following types of definitions:

- Baseline Definition
- Objects Definition
- Component Definitions

NOTE: Refer to the Definitions (Folder) section for more information on these definitions.

Install

The install operation is used to insert/update records within a given database table and is compatible with the Component Definition.

Import

The import operation is used to create tables and insert their data, and is compatible with the Baseline Definition.

Update

The update operation is used to run the SQL scripts contained within the Component or Objects sub folders and is compatible with the Objects Definition.

Databases (Folder)

The Database folder contains XML files for each Database that is relevant to what's being extracted. These file define which definitions can be used and allow variable values to be associated with a specific Instance. This instance can be called when running the DVBT allowing specific records to be extracted.

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

Set Up a New Extract or Install Package

When the DSP is installed, the Syniti DBVT is installed into the DSP installation directory. When creating a new Syniti DBVT extract or install package, a series of folders and files are required to use the Syniti DBVT. These need to be either manually created or copied from the DSP installation location.

NOTE: It is important that all the relevant folders and files are copied; otherwise, the Syniti DBVT does not work.

To set up a new extract or install package:

1. Create a folder to represent the database or application that is being extracted.
2. Copy the Web folder from $Install/BOA/DSP into the newly created folder.

   NOTE: $Install is the default file path where DSP was installed; the file path can be changed.

3. Create a folder called “Databases” within the newly created folder.
4. Create a folder called “Apps” within the Databases folder.
5. Copy the Config, Deploy, Logs, Templates and Tools folders from the $Install/BOA/DSP/Databases folder into the newly created Databases folder.

NOTE: At this stage, the newly created folder structure contains all the generic tools and files that the Syniti DBVT requires to extract and install database or application data. However, the newly created folder structure does not include the specific instructions to extract a particular WebApp or set of reusable data (Content).

6. Create Database/Application. Within the Apps folder, the Syniti DBVT requires a very specific structure of folders and associated files to extract data. The Syniti DBVT uses a series of Operation, Database and Definition files to determine what needs to be extracted. The Syniti DBVT then writes the data into folder and files under the Component folder.

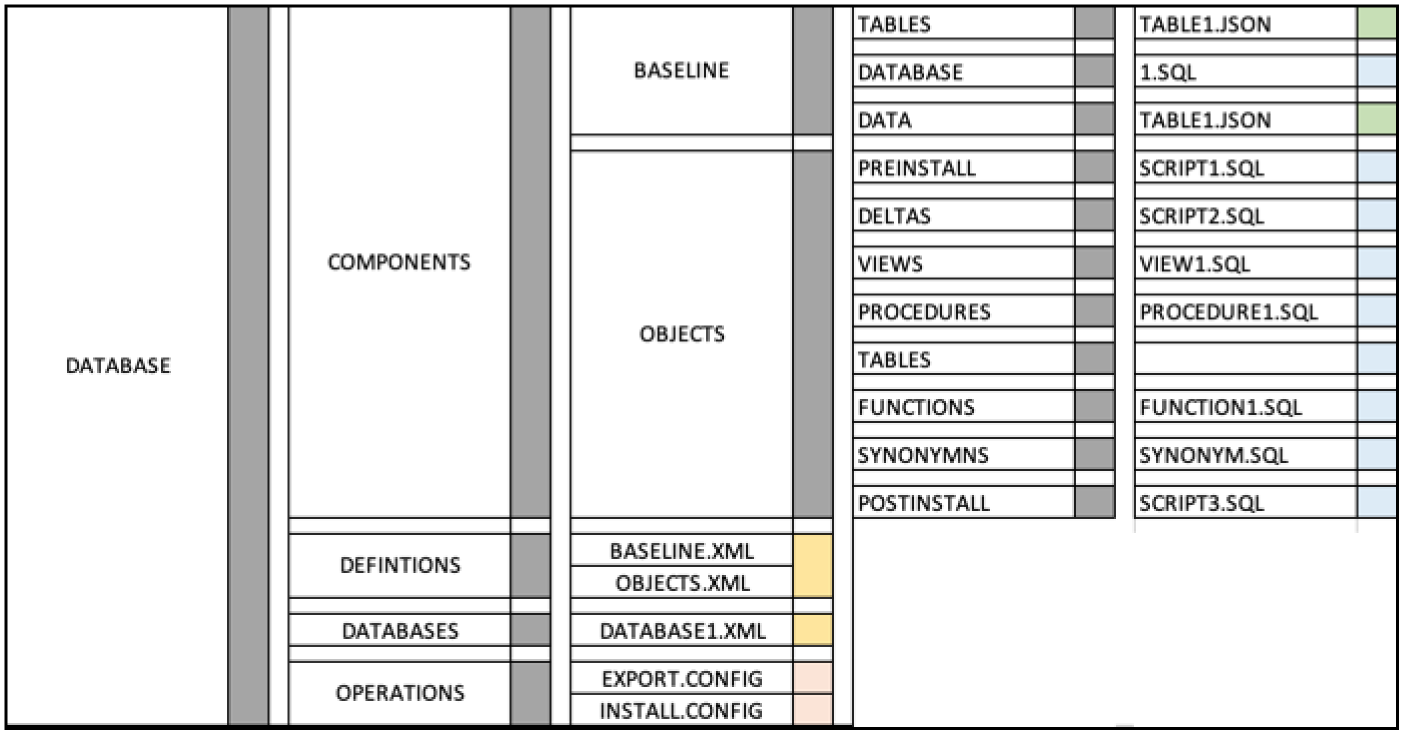
Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

The following table provides a color code for the type of file and whether it is an input or an output file. Use this table to understand the data extract examples.

Color	Type	Syniti DBVT Input/Output
⬛	Folders	
🟨	Xml files	Input
🟧	Config files	Input
🟩	Json files	Output
🟦	SQL scripts	Output

Example 1: Database Extract

The following diagram shows an example folder structure in a scenario where the Syniti DBVT is being used to extract or install a database, its SQL objects and data.



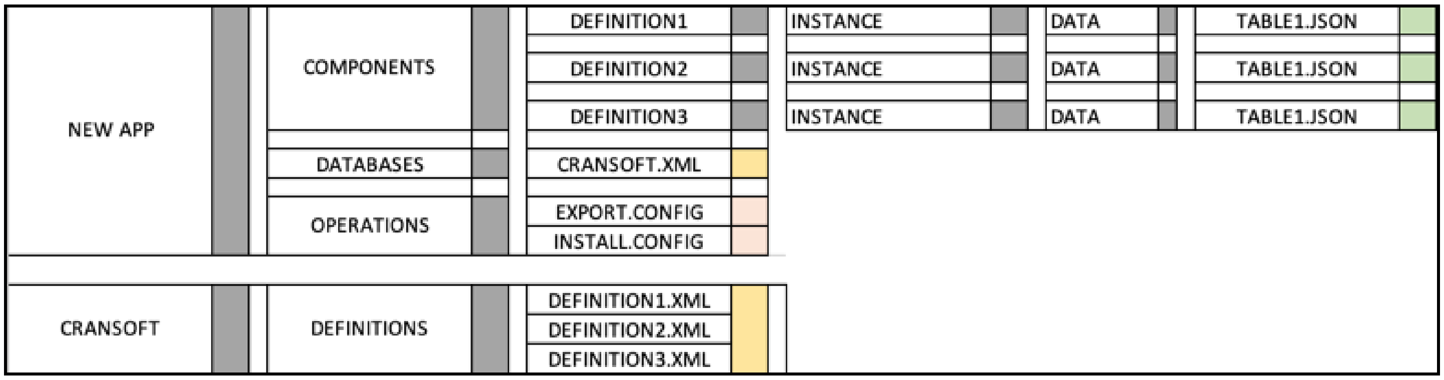
Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

IMPORTANT! In addition to the automatically created scripts, the Syniti DBVT enables the use of manually created scripts to build more complex installation processes.

- **Folder DATABASE, File 1.SQL**—This file instructs the Syniti DBVT to create a new database.
- **Folder PREINSTALL**— This folder contains scripts related to the Objects Definition that are run **before** any view, procedure, function or synonym scripts are run.
- **Folder DELTA**—This folder contains delta scripts related to the Objects Definition that are run **before** any view, procedure, function or synonym scripts are run. Delta scripts enable changes to an application's tables, views or procedures to be delivered as a versioned delta script that will be applied during an install or upgrade. Delta scripts must be versioned with a number followed by the name, e.g., 1.0.1.AlterMyTable. When a delta script is installed, it is registered in the database version table specified in the associated database XML file. This table provides a control of all delta scripts that have been applied.
- **Folder POSTINSTALL**—This folder contains scripts related to the Objects Definition that are run **after** the view, procedure, function or synonym scripts are run.

Example 2: Custom WebApp Extract

The following diagram shows an example folder structure in a scenario where the Syniti DBVT is being used to extract/install a custom WebApp.



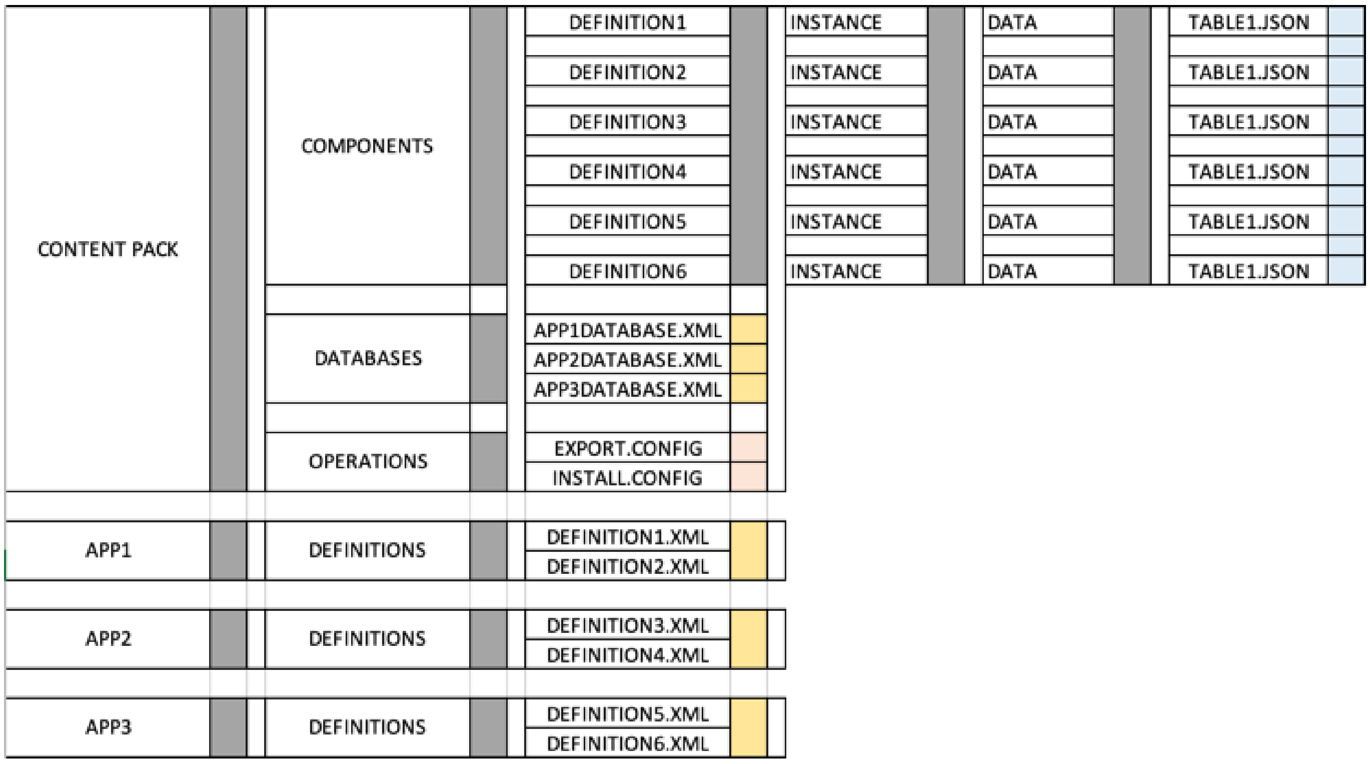
IMPORTANT! The important feature of this scenario is that the Export/Install operations defined within the NEW APP folder call definitions within the CranSoft folder and save the associated data from tables, such as WebApp, WebAppPage, WebAppPageColumn within the Components folder under the NEW APP folder.

TIP! A custom DSP WebApp is comprised of BOTH Database and WebApp table registrations; therefore, when creating a 'Content Pack' for a custom application, the APP folder must include the files and folders from Example 1 and 2.

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

Example 3: Content Extract

The following diagram shows an example folder structure in a scenario where the Syniti DBVT is used to extract/install reusable ‘Content’ from one or more DSP WebApps.



IMPORTANT! The important feature of this scenario is that the Export/Install operations defined within the CONTENT PACK folder call definitions within folders related to other DSP WebApps (APP1, APP2, APP2). The Syniti DBVT saves the associated data into the Components folder under the CONTENT PACK folder. This feature enables a single Content Pack to include data from across several different DSP applications. For example, a Data Migration ‘Content Pack’ is comprised of data from the following DSP WebApps:

- CranSoft
- Console
- cMap
- Cranport
- DSW

TIP! In the case of a Content Pack such as a Migration Pack that includes Application Table registrations and specific databases (Source, Target and dsw<WAVE>), folders for the Content Pack, Applications and Databases can all be included under the same App folder.

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

Run the Database Versioning Tool

The Syniti DBVT can be run via the command line or by using a batch file. As a minimum, it is recommended that a batch file is created to enable users to easily install packages.

Command Line

From the Command line, users must navigate to the Database/Tools folder and then run the following:

```
Tools\RunOperation [Application / Database / Content Pack] [Operation File] -verbose
```

For example, **Tools\RunOperation S4FINANCE_MIGRATION EXPORT -verbose**

Batch File

A batch file can be created to enable users to extract/install using the Syniti DBVT. The format of the batch file is:

```
@echo off

pushd %~dp0

setlocal

title [Insert Name of Package]
```

THIS CODE BLOCK CAN BE REPEATED TO PROCESS MULTIPLE CONFIG IN DIFFERENT FOLDERS

```
echo [Insert Name of the Step]
## Navigate to relative path of the Tools folder ##
e.g., call CD..
call Tools\RunOperation [Application / Database / Content Pack] [Operation File]
e.g., Tools\RunOperation S4FINANCE_MIGRATION EXPORT
if "%ERRORLEVEL%" NEQ "0" (
      echo ERROR: [Error Message]
      goto Failure
)
```

THIS CODE BLOCK CAN BE REPEATED TO PROCESS MULTIPLE CONFIGS IN DIFFERENT FOLDERS

```
goto Success

:Success

call :Cleanup

goto EOF
```

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

```
:Failure

call :Cleanup

exit /B 1

:Cleanup

endlocal

popd

if "%1" == "--nopause" goto EOF

pause

goto EOF

:EOF
```

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

Syniti DBVT Input Files

The Syniti DBVT uses several input files to drive the extraction of data and SQL scripts from a DSP instance. These files are located in the following folders:

- Operation
- Database
- Definitions

Operation Files

The structure of an Operation file is as follows:

Application/Database [TAB] Operation [TAB] Definition [TAB] Instance (Optional)

Example 1

```
CranSoft	export	Content.DataSource	SecurityMigrationWebApp
CranSoft	export	Content.WebApp	SecurityMigrationWebApp
CranSoft	export	Content.Catalog	SecurityMigrationWebApp
```

Example 2

```
CranSoft	install	Content.DataSource	SecurityMigrationWebApp
CranSoft	install	Content.Catalog	SecurityMigrationWebApp
CranSoft	install	Content.WebApp	SecurityMigrationWebApp
```

Example 3

```
CentralSecurityMigrationUtility	Export	Baseline
CentralSecurityMigrationUtility	Export	Objects
```

Example 4

```
CentralSecurityMigrationUtility	Import	Baseline
CentralSecurityMigrationUtility	Update	Objects
```

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

Database Files

Database files are XML files that have different formats depending on whether they are being used in conjunction with Baseline or Object Definitions, or Component Definitions. Importantly, Baseline and Objects Definitions cannot be used in conjunction with Instances and Instance Variables.

XML File Structure for use with Baseline or Object Definitions

```
Tag: Datasource
      Attribute Name:

      Tag: Setup
            Attribute CommandTimeout: This is the SQL command timeout when executing a
            component. It overrides the default setting of 1800 seconds. If a component
            install was expected to take longer than 1800 seconds for example, use the
            setting to set a higher timeout.
            Attribute databaseVersionLogTableName: This is the name of table in which the
            list of executed delta scripts that have been applied is stored. If the table
            does not exist, it is created.

            Tag: Components

                  Tag Component
                        Attribute Type
```

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

XML File Structure for use with Component Definitions

```
Tag: Datasource
     Attribute Name:

     Tag: Setup
          Attribute CommandTimeout: This is the SQL command timeout when executing a
          component. It overrides the default setting of 1800 seconds. If a component
          install was expected to take longer than 1800 seconds, for example, use the
          setting to set a higher timeout.
          Attribute databaseVersionLogTableName: This is the name of table in which the
          list of executed delta scripts that have been applied is stored. If the table
          does not exist, it is created.

          Tag: Components

               Tag Component
                    Attribute Type

                    Tag: Instances

                         Tag; Instance

                              Tag: Variables

                                   Tag: Variable
                                        Attribute Name
                                        Attribute Value
```

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

Example 1: Baseline or Objects Definitions

This database XML file enables the use of Baseline and Objects Definitions. A table called ztDatabaseVersionLog is created in the CentralSecurityMigrationUtility database and any delta scripts that are executed as part of the installation process are registered in this table.

```
<datasource name="CentralSecurityMigrationUtility">
      <setup databaseVersionLogTableName="ztDatabaseVersionLog" />
            <components>
                  <component type="baseline" >
                  </component>

                  <component type="objects" >
                  </component>

            </components>
</datasource>
```

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

Example 2: Component Definitions

This database XML file enables the use of the following Component Definitions that belong to the CranSoft WebApp:

- Content.DataSource
- Content.WebApp
- Content.Catalog

These definitions use an Instance called 'SecuirtyMigrationWebApp' to define some variables with associated values. When the Syniti DVBT is run, the required Instance can be selected to allow the defined variable and their values to be passed to the associated Component Definition, allowing specific records to be extracted from the tables in the definition files.
A table called DatabaseVersionLog is created in the CranSoft database (if not already existing) and any delta scripts that are executed as part of the installation process is registered in this table.

```
<datasource name="CranSoft">
      <setup commandTimeout="2000" databaseVersionLogTableName="DatabaseVersionLog" />
      <components>

            <component type="Content.DataSource">
                  <instances>
                        <instance name="SecurityMigrationWebApp">
                              <variables>
                                    <variable name="DataSourceID"
                                                value="'1D4BCE74-7EA2-435E-859C-
0C9F00A74897'"
                                                />
                              </variables>
                        </instance>
                  </instances>
            </component>

            <component type="Content.WebApp">
                  <instances>
                        <instance name="SecurityMigrationWebApp">
                              <variables>
                                    <variable name="WebAppID"
                                                value="'AD4E10B1-693F-4A1A-98B4-
43A5EBF921A3'"
                                                />
                                    <variable name="Namespace" value="Central Security
Migration Utility" />
                              </variables>
                        </instance>
                  </instances>
            </component>
            <component type="Content.Catalog">
                  <instances>
```

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

```
                        <instance name="SecurityMigrationWebApp">
                              <variables>
                                    <variable name="CatalogID"
                                                value="'f3483da4-6fab-4f7b-9df4-
0b7eef6419e1'"/>
                              </variables>
                        </instance>
                  </instances>
            </component>
            <component type="SiteContainerLinks">
                  <instances>
                        <instance name="SecurityMigrationWebApp">
                              <variables>
                                    <variable name="Namespace" value="Central Security
Migration Utility" />
                              </variables>
                        </instance>
                  </instances>
            </component>
      </components>
</datasource>
```

Definition Files

Baseline Definition

The Baseline Definition XML files use the following tags and associated attributes to perform the task of extracting table schema information into SQL scripts and table data in JSON files.

```
Tag: Settings
      Attribute excludeTableSchema: If set to true, table SQL definition is not included.
      By default, the table schema is included

      Tag: Tables

            Tag: Pattern
                  Attribute Expression: Enter the name of table to be extracted or
                  excluded from extraction. This attribute permits the use of wildcards
                  select all (*) or a selection of tables that contain a common word
                  (web*)
                  Attribute Include: If set to false, table(s) entered in Expression are
                  excluded.
                  Attribute WhereClause: Enter a valid SQL statement to select records to
                  extract. Alternatively, enter 1=1 to extract all records or 1=0 to
                  extract no records.

                  Tag: IncludeColumns (by default, the data in all columns is included)
```

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

```
            Tag: Column
                  Insert Column Name

      Tag ExcludeColumns (used to exclude data in specific column e.g.,
      passwords, username)

            Tag: Columns
                  Insert Column Name
```

Example Baseline Definitions

The following are two examples of Baseline Definition XML files that support common table extract scenarios.

Example 1

This definition creates SQL scripts for all tables (excludeTableSchema="false", pattern expression="*") in the specified database. Before installing the tables, all tables in the database are dropped (Settings dropObjects="true"). The data from all the tables are extracted/installed; however, the data contained within any columns called 'Password' are excluded.

```
<Config name="Baseline">
      <Settings excludeTableSchema="false"/>
      <Tables>
            <Pattern expression="*" include="true" whereClause="1=1">
                        <IncludeColumns>
                        <Column>*</column>
                  </IncludeColumns>
                  <ExcludeColumns>
                        <Column>Password</column>
                  </ExcludeColumns>
                  </Pattern>
      </Tables>
</Config>
```

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

Example 2

This definition creates SQL scripts for all tables in the specified database. Before installing the tables, all tables in the database are dropped (Settings dropObjects="true"). The data from tables Orders, Customer, BankAccounts are not extracted/installed (whereClause="1=0"). The data from all other tables are extracted/installed (whereClause="1=1"); however, the data contained within any columns called ‘Password’ are excluded.

```
<Config name="Baseline">
      <Settings excludeTableSchema="false"/>
      <Tables>
            <Pattern expression="Orders" include="true"  whereClause="1=0" />
            <Pattern expression="Customer" include="true"  whereClause="1=0" />
            <Pattern expression="BankAccounts" include="true"  whereClause="1=0" />
            <Pattern expression="*" include="true"  whereClause="1=1">
                        <IncludeColumns>
                        <Column>*</column>
                  </IncludeColumns>
                  <ExcludeColumns>
                        <Column>Password</column>
                  </ExcludeColumns>
                  <Pattern/>
      </Tables>
</Config>
```

Objects Definition

The Objects Definition XML file uses the following tags and associated attributes to perform the task of extracting database view, procedure and function schema information into SQL scripts files.

```
Tag: Settings
      Attribute dropObjects: If set to true, all views/procedures/functions in database
      are dropped.

      Tag: Views

            Tag: View
                  Attribute Expression: Enter the name of the view to be extracted or
                  excluded from extraction. This attribute permits the use of wildcards
                  select all (*) or a selection of views that contain a common word
                  (e.g., web*)
                  Attribute excludeFromScript: If set to true, the view(s) entered in
                  Expression are excluded.
                  Attribute stubObject: If set to true, the Syniti DBVT creates a dummy
                  script that has the same structure as the original object, but without
                  referencing dependent objects. This attribute is used when there are
                  complex dependencies that cause scripts to fail. A delta script can
                  subsequently be used to update the objects with the correct definition.
```

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

```
Tag: Procedures

        Tag: Procedure
            Attribute Expression: Enter the name of the procedure to be extracted
            or excluded from extraction. This attribute permits the use of
            wildcards select all (*) or a selection of procedures that contain a
            common word (e.g., web*)
            Attribute excludeFromScript:  If set to true, the procedures(s) entered
            in Expression are excluded.

Tag: Functions

        Tag: Function
            Attribute Expression: Enter the name of the function to be extracted or
            excluded from extraction. This attribute permits the use of wildcards
            select all (*) or a selection of function that contain a common word
            (e.g., web*)
            Attribute excludeFromScript:  If set to true, the function(s) entered
            in Expression are excluded.
```

Example 1

```
<Config name="Objects">
      <Settings dropObjects="true"/>
      <Views>
            <View expression="test*" excludeFromScript="true" />
            <View expression="*" excludeFromScript="false" />
      </Views>
      <Procedures>
            <Procedure expression="test*" excludeFromScript="true" />
            <Procedure expression="*" excludeFromScript="false" />
      </Procedures>
      <Functions>
            <Function expression="test*" excludeFromScript="true" />
            <Function expression="*" excludeFromScript="false" />
      </Functions>
</Config>
```

Example 2

```
<Config name="Objects">
      <Settings dropObjects="true"/>
      <Views>
            <View expression="test*" excludeFromScript="true" />
            <View expression="dynamic*" excludeFromScript="false" stubObject=”true’/>
            <View expression="*" excludeFromScript="false" />
      </Views>
      <Procedures>
            <Procedure expression="test*" excludeFromScript="true" />
```

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

```
            <Procedure expression="*" excludeFromScript="false" />
      </Procedures>
      <Functions>
            <Function expression="test*" excludeFromScript="true" />
            <Function expression="*" excludeFromScript="false" />
      </Functions>
</Config>
```

Component Definition

Component Definition XML files use the following tags and associated attributes to perform the task of extracting table data from an application database. Component Definitions have some additional capability (compared to the Baseline Definition) that enable sophisticated queries to be used to export only specific records from a database.

```
Tag: Settings
      Attribute excludeTableSchema: If set to true, table SQL definition is not included.
      By default, the table schema is included

      Tag: Parameters
            Tag: Parameter
                  Insert Parameter Name: Insert the name of parameters (Database.xml
                  Instance Variables) that are passed from the associated database.xml
                  file when the Syniti DBVT is run with a specific instance. This enables
                  a single definition to be reused for multiple purposes.

      Tag: Variables
            Tag: Variable
                  Attribute Name: A variable can be used within queries or data exports.
                  Attribute Value: A variable can be given a value so that the same value
                  is consistently used.

      Tag: Queries
            Tag: Query
                  Attribute Name: Provide a name for the result set of the query. This
                  results set can be referenced by using @QueryName@ when selecting data
                  to export.
                  Insert Query: Insert a valid query that selects the relevant record
                  set. The query can include variables passed in via the definition or
                  parameters whose values are set within the associated database.xml file
                  and selected when the Syniti DBVT export is run with a particular
                  instance by using #Variable or Parameter#

      Tag: Tables

            Tag: Pattern
                  Attribute Expression: Enter the name of table name to be extracted or
                  excluded from extraction. This attribute permits the use of wildcards
```

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

```
select all (*) or a selection of tables that contain a common word
(web*)
Attribute Include: If set to false, table(s) entered in Expression are
excluded.
Attribute WhereClause: Enter a valid SQL statement to select records to
extract. Alternatively, enter 1=1 to extract all records or 1=0 to
extract no records.

Tag: IncludeColumns (by default, the data in all columns is included)

      Tag: Column
            Insert Column Name

Tag ExcludeColumns (used to exclude data in specific column e.g.,
passwords, username)

      Tag: Columns
            Insert Column Name
```

Example 1

In the following example there is a definition called Reuse.WebApp. This definition inherits a parameter WebAppID from the associated database.xml file when the Syniti DBVT is run for a specific instance. The definition includes a variable called MenuID that has a value assigned within the definition. The parameter #WebAppID# is used in a query called 'PageIDsToKeep' to create a record set of all the PageIDs that are related to WebAppID parameter value that are passed in at runtime. The values obtained from the PageIDsToKeep query are used in another query called 'PageEventRuleIDsToKeep' to create a record set of all the associated page event rules. The variables and queries are then used (Pattern expression) in the where clauses to export the required table records.

```
<config name="Reuse.WebApp">
      <settings excludeTableSchema="true">
            <parameters>
                  <parameter>WebAppID</parameter>
            </parameters>
            <variables>
                  <variable name="MenuID" value="'d79eef0e-5507-473e-a984-ee013de3d2ab'"
/>
            </variables>
            <queries>
                  <query name="PageIDsToKeep">
                        SELECT PageID FROM Page WHERE WebAppID = #WebAppID#'
                  </query>
                  <query name="PageEventRuleIDsToKeep">
                        SELECT PageEventRuleID FROM PageEventRule WHERE PageID IN
(@PageIDsToKeep@)
                  </query>
            </queries>
            <pattern expression="Menu" include="true" whereClause="MenuID = #MenuID#" />
```

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

```
            <pattern expression="Page" include="true" whereClause="PageID IN
(@PageIDsToKeep@)">
                  <excludeColumns>
                        <column>ExcelHelpTextAsComment</column>
                  </excludeColumns>
            </pattern>
            <pattern expression="PageColumn" include="true" whereClause="PageID IN
(@PageIDsToKeep@)">
                  <excludeColumns>
                        <column>LinkToLayered</column>
                  </excludeColumns>
            </pattern>
            <pattern expression="PageEvent"  include="true" whereClause="PageID IN
(@PageIDsToKeep@)" />
            <pattern expression="PageEventParameter"     include="true"
whereClause="PageID IN (@PageIDsToKeep@)" />
            <pattern expression="PageEventRule"          include="true"
whereClause="PageID IN (@PageIDsToKeep@)" />
            <pattern expression="PageEventRuleLink"      include="true"
            whereClause="PageEventRuleID IN (@PageEventRuleIDsToKeep@)" />
            <pattern expression="PageEventRuleParameter" include="true"
            whereClause="PageEventRuleID IN (@PageEventRuleIDsToKeep@)" />
            <pattern expression="PageEventValidation"    include="true"
whereClause="PageID IN (@PageIDsToKeep@)" />
      </tables>
</config>
```

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.

The following are examples of database.xml and operation.config files that pass WebAppID '9ac95b56-f805-4fc4-9d9b-590b05cd0e7c' into the definition above when the Syniti DBVT is run with the Operation file.

Example: Operation.Config File

```
CranSoft     Export       Reuse.WebApp       WebAppExport
```

Example: Database.xml File

```
<datasource name="CranSoft">
      <setup commandTimeout="300" externalConfigType="CranSoftInstallation"
                  databaseVersionLogTableName="DatabaseVersionLog" />
      <components>
            <component type="Reuse.WebApp">
                  <instances>
                        <instance name="WebAppExport">
                              <variables>
                                    <variable name="WebAppID" value="'9ac95b56-f805-4fc4-
9d9b-590b05cd0e7c'"/>
                              </variables>
                        </instance>
                  </instances>
            </component>
      </components>
</datasource>
```

Last updated: 1/27/2020

Copyright © 2020 BackOffice Associates, LLC dba Syniti and/or its affiliates. All rights reserved. This document contains confidential and proprietary information and reproduction is prohibited unless authorized by BackOffice Associates. Other names appearing on the Site may be trademarks of their respective owners.