TIBCO Data Virtualization Developer Guidelines

<table>
<thead>
<tr>
<th>Project Name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Release</td>
<td>3.0</td>
</tr>
<tr>
<td>Date</td>
<td>April 2018</td>
</tr>
<tr>
<td>Primary Author</td>
<td>Tony Young</td>
</tr>
<tr>
<td>Document Owner</td>
<td>Tony Young</td>
</tr>
<tr>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>Document Location</td>
<td></td>
</tr>
<tr>
<td>Purpose</td>
<td>Offers guidelines for developers using TIBCO Data Virtualization to design, implement and maintain data integration projects.</td>
</tr>
</tbody>
</table>
Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>October 2012</td>
<td>Tony Young</td>
<td>Initial revision</td>
</tr>
<tr>
<td>2.0</td>
<td>July 2015</td>
<td>Matthew Lee</td>
<td>Updated with new product branding</td>
</tr>
<tr>
<td>2.1</td>
<td>October 2016</td>
<td>Matthew Lee</td>
<td>Updated for TDV 7.0.4</td>
</tr>
<tr>
<td>3.0</td>
<td>April 2018</td>
<td>Deane Harding</td>
<td>Updated with TIBCO branding</td>
</tr>
</tbody>
</table>

Approvals
This document requires the following approvals. Signed approval forms are filed in the project files.

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Title</th>
<th>Company</th>
<th>Date of Issue</th>
<th>Version</th>
</tr>
</thead>
</table>

Distribution
This document has been distributed to:

<table>
<thead>
<tr>
<th>Name</th>
<th>Title</th>
<th>Company</th>
<th>Date of Issue</th>
<th>Version</th>
</tr>
</thead>
</table>

Related Documents
This document is related to:

<table>
<thead>
<tr>
<th>Document</th>
<th>File Name</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDV User’s Guide</td>
<td>TIB_tdv_7.0.6_UsersGuide.pdf</td>
<td>TIBCO DV Engineering</td>
</tr>
<tr>
<td>TDV Reference Guide</td>
<td>TIB_tdv_7.0.6_ReferenceGuide.pdf</td>
<td>TIBCO DV Engineering</td>
</tr>
</tbody>
</table>
Copyright Notice
COPYRIGHT© TIBCO Software Inc. This document is unpublished and the foregoing notice is affixed to protect TIBCO Software Inc. in the event of inadvertent publication. All rights reserved. No part of this document may be reproduced in any form, including photocopying or transmission electronically to any computer, without prior written consent of TIBCO Software Inc. The information contained in this document is confidential and proprietary to TIBCO Software Inc. and may not be used or disclosed except as expressly authorized in writing by TIBCO Software Inc. Copyright protection includes material generated from our software programs displayed on the screen, such as icons, screen displays, and the like.

Trademarks
All brand and product names are trademarks or registered trademarks of their respective holders and are hereby acknowledged. Technologies described herein are either covered by existing patents or patent applications are in progress.

Confidentiality
The information in this document is subject to change without notice. This document contains information that is confidential and proprietary to TIBCO Software Inc. and its affiliates and may not be copied, published, or disclosed to others, or used for any purposes other than review, without written authorization of an officer of TIBCO Software Inc. Submission of this document does not represent a commitment to implement any portion of this specification in the products of the submitters.

Content Warranty
The information in this document is subject to change without notice. THIS DOCUMENT IS PROVIDED "AS IS" AND TIBCO MAKES NO WARRANTY, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO ALL WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TIBCO Software Inc. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Export
This document and related technical data, are subject to U.S. export control laws, including without limitation the U.S. Export Administration Act and its associated regulations, and may be subject to export or import regulations of other countries. You agree not to export or re-export this document in any form in violation of the applicable export or import laws of the United States or any foreign jurisdiction.

For more information, please contact:

TIBCO Software Inc.
3303 Hillview Avenue
Palo Alto, CA 94304
USA
## Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>7</td>
</tr>
<tr>
<td>1.1</td>
<td>Purpose</td>
<td>7</td>
</tr>
<tr>
<td>1.2</td>
<td>Audience</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>Resource Selection</td>
<td>8</td>
</tr>
<tr>
<td>2.1</td>
<td>Data Sources</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Definition Sets</td>
<td>9</td>
</tr>
<tr>
<td>2.2.1</td>
<td>SQL Definition Set</td>
<td>9</td>
</tr>
<tr>
<td>2.2.2</td>
<td>XML Definition Set</td>
<td>10</td>
</tr>
<tr>
<td>2.2.3</td>
<td>WSDL Definition Set</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>Views</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>Procedures</td>
<td>11</td>
</tr>
<tr>
<td>2.4.1</td>
<td>SQL Script</td>
<td>11</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Parameterized Queries</td>
<td>11</td>
</tr>
<tr>
<td>2.4.3</td>
<td>Packaged Queries</td>
<td>12</td>
</tr>
<tr>
<td>2.4.4</td>
<td>XQuery Procedure</td>
<td>13</td>
</tr>
<tr>
<td>2.5</td>
<td>Triggers</td>
<td>13</td>
</tr>
<tr>
<td>2.6</td>
<td>Custom Java Procedures</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>Annotation Standards</td>
<td>15</td>
</tr>
<tr>
<td>3.1</td>
<td>Description</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>Input Parameters</td>
<td>16</td>
</tr>
<tr>
<td>3.3</td>
<td>Output Parameters</td>
<td>16</td>
</tr>
<tr>
<td>3.4</td>
<td>Exceptions</td>
<td>16</td>
</tr>
<tr>
<td>3.5</td>
<td>Version History</td>
<td>16</td>
</tr>
<tr>
<td>3.6</td>
<td>Example</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>Code Formatting</td>
<td>18</td>
</tr>
<tr>
<td>4.1</td>
<td>Indentation</td>
<td>18</td>
</tr>
<tr>
<td>4.2</td>
<td>Capitalization</td>
<td>18</td>
</tr>
<tr>
<td>4.3</td>
<td>Resource, Column, Variable and Parameter Naming</td>
<td>19</td>
</tr>
<tr>
<td>4.4</td>
<td>Inline Comments and Code Blocking</td>
<td>19</td>
</tr>
<tr>
<td>4.5</td>
<td>Formatted Example</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>Help!</td>
<td>21</td>
</tr>
<tr>
<td>5.1</td>
<td>Function Support</td>
<td>21</td>
</tr>
<tr>
<td>5.2</td>
<td>Privilege Issues</td>
<td>21</td>
</tr>
<tr>
<td>5.3</td>
<td>Missing Data Source Resources</td>
<td>21</td>
</tr>
<tr>
<td>5.4</td>
<td>Cache Configuration and Refresh Issues</td>
<td>21</td>
</tr>
</tbody>
</table>
5.5 Layering .......................................................................................................................... 22
5.6 Who to Call Next .............................................................................................................. 22
5.7 TIBCO Data Virtualization Community ............................................................................. 22
5.8 When to Call TIBCO Support ......................................................................................... 22
5.9 When to Call TIBCO Professional Services ..................................................................... 23
6 Code Review Checklist ........................................................................................................ 24
7 Conclusion .......................................................................................................................... 25
1 Introduction

1.1 Purpose
This document outlines guidelines for developers using TIBCO Data Virtualization (TDV) to design, implement and maintain data integration projects. Topics covered include:

- **Resource Selection** – When to use a given resource type such as Views, SQL Scripts, etc.
- **Annotation Standards** – What detail should be included in annotation panel entries, including examples
- **Code Formatting** – Formatting code for easy reading, naming objects, including examples

1.2 Audience
This document is intended to provide guidance to the following users:

- TIBCO Professional Services
- TDV Architects
- TDV Developers
2 Resource Selection

It is important to choose the right TDV resource type for the right job. The following sections outline the different types of TDV resources, and when it is appropriate to use each one.

2.1 Data Sources

The TIBCO Data Virtualization Platform integrates your data sources into a single virtual data layer that can be used to model rich information resources. You can add many types of data sources to the TDV platform using TDV Studio, defining connection profiles, specific source capabilities, and detailed source introspection metadata. TDV views and procedures built on these physical sources can take advantage of capabilities unique to each data source, allowing the integration of disparate parts into a single virtual data layer.

Adding a data source means creating a metadata representation of the underlying native data source in the TDV repository. It does not mean replication of the data or replication of the source. The data source metadata for each data source type includes things like how the data source:

- Defines and stores data, including catalogs, schemas, tables, and columns
- Accepts and responds to requests for data
- Handles insert, update, and delete transactions
- Executes stored procedures
- Makes data-related comparisons

A virtual layer of information about data source capabilities allows the TDV Query Engine to create efficient query execution plans that leverage data source strengths and inherent advantages of preprocessing data at the source.

Adding the data source requires choosing an adapter for your data source and configuring the adapter with the connection and authentication settings for that physical source instance.

Introspection collects metadata for all the selected data source resources. Introspected resources are added to the TDV Studio resource tree for use in the Studio modeling layer, where you can then create views, procedures, and other resources based on those data resources.

Using Data Source Resources

- If a connection has already been made to the data source, but not all resources have been introspected for modeling, use the Add/Remove Resources feature on the existing resource to introspect the missing resources. This maintains one connection to each data source from within TDV.
- If a connection has not already been made to the data source, create a new connection and introspect only those objects required for modeling. The introspected objects can be refined later using the Add/Remove Resources feature.

Tips and Tricks

- Introspect only those objects needed for modeling. This prevents unintended access to data, and keeps the resource tree free of unneeded clutter.
- Schedule re-introspection during off-peak hours to automatically capture metadata changes to physical data source objects
- Create a custom data source adapter to modify the default capabilities of a data source for a project-specific data source with non-standard configuration (i.e. case sensitivity and trailing spaces differences)
- Configure connection pool settings for relational data sources to avoid overloading the data source
- Configure a connection validation query for relational data sources so TDV can validate that connections pulled out of the connection pool are still valid. Sometimes an intervening router or firewall will shut down idle connections without TDV knowing about it.
- When configuring data sources, it is best to have only one data source resource in TDV for each physical data source instance being consumed. This allows the TDV optimizer to group data source operations and effectively push down relations
- Use application/service accounts to access data sources. Using these accounts allows for better security at the data source, and keeps the effects of password changes, locked accounts, etc. from impacting development and production

For more information on data sources, please see the TIBCO Data Virtualization User Guide section titled: Working With Data Sources.

2.2 Definition Sets
A definition set is a set of definitions you create that can be shared by other TDV resources. A definition set includes SQL data types, exceptions and constants, XML types and elements, or WSDL definitions. You can refer to the resources in a definition set from any number of procedures. The changes you make in a definition set are propagated to all the procedures using the definition set. You cannot publish a definition set. The definition sets you create are reusable and can be shared with any user with appropriate access privileges in the system.

For more information on definition sets, please see the TIBCO Data Virtualization User Guide section titled: Definition Sets.

TDV supports the following three types of definition sets.

2.2.1 SQL Definition Set
The SQL definition set editor lets you create your own SQL type definitions, exceptions, and constants.

Using SQL Definition Set Resources
- Use a SQL Definition Set to store relational constants, type definitions and exception messages reused across multiple scripts

Tips and Tricks
- Define constants, types, and exceptions that are used in more than one TDV procedure in one SQL Definition Set. This allows a global reference for types, constant values such as field names, and exceptions that are frequently handled and processed.
- Use the PATH keyword to add "paths" to your procedure's environment so that you don't need to specify the full path to a definition set every time it is used. For example, adding "PATH /lib/resource;" before your declaration statements allows you to declare a variable of type "ResourceDefs.ParameterValues" instead of "/lib/resource/ResourceDefs.ParameterValues". This will make your code cleaner and easier to read. Note that
/lib/debug and /lib/util are added to a procedure's environment by default, which is why you can use "CALL PRINT(...)" instead of having to use "CALL /lib/debug/PRINT( ...)".

### 2.2.2 XML Definition Set

The XML definition set editor lets you create XML schema definitions. You can import definitions from existing XSD files, validate definitions, format definitions, and create definitions from an XML instance.

**Using XML Definition Set Resources**

- Use a XML Definition Set to import the XSD document or create a webservice schema for XML instance documents
- These definition sets are used as the basis of a XQuery transform. Create this resource before a XQuery resource

**Tips and Tricks**

- Each distinct XML schema should be created in its own definition set. This allows compartmentalization of schemas and easy updates to schema definitions

### 2.2.3 WSDL Definition Set

The WSDL definition set editor lets you create your own WSDL definition set. You can import WSDL documents that include schema definitions, input and output message types, bindings, etc.

**Using WSDL Definition Set Resources**

- For each contract-first webservice implementation, a separate WSDL definition set will be required. Create the definition set holding the WSDL document to be implemented prior to developing any webservice implementation resources

**Tips and Tricks**

- Each distinct WSDL should be created in its own definition set. This allows compartmentalization of definitions and easy updates should the WSDL document change or require versioning

### 2.3 Views

A TDV view is a virtual data table defined by SQL and TDV metadata. A view defines a SQL query that comprises a SELECT statement and any ANSI-standard SQL clauses and functions. A view can JOIN or UNION to any resource defined in the TDV virtual data layer.

Views can be selectively published to make them available through client applications or the Web. If you want to make a view available to client programs, you must publish that view.

**Using View Resources**

- Use a view to manipulate data source resources, such as tables, to rename (alias), join, aggregate
- Use a view to apply column or row-level security
- Use a view to insulate upstream TDV resources from changes to underlying data source implementations

**Tips and Tricks**

- Periodically test views that you are developing to ensure that execution returns proper data and performs as expected.
- Build views as building blocks, and reuse them to avoid duplication of work
- Layer objects, making small and manageable changes at each layer. View definitions will be flattened at execution time, but development will be more manageable with modularized changes
• Add virtual index and foreign key metadata where appropriate to top-level views (i.e. layer of objects that are subsequently published) to allow BI and other tools to effectively join and filter data.

For more information on views, please see the TIBCO Data Virtualization User’s Guide section titled: Designing Procedures.

2.4 Procedures

Procedures are predefined programs that can be executed to query and manipulate data stored in a data source. They have scalar input and output parameters. Some of them return one or more cursors, which represent tabular output data.

For more information on procedures, please see the TIBCO Data Virtualization User’s Guide section titled: Views.

TDV supports the following types of procedures.

2.4.1 SQL Script

A SQL Script is a procedure written in TDV’s SQL script language, as described in “TDV SQL Script” in the TIBCO Data Virtualization Reference Guide.

SQL Scripts are akin to Oracle PL/SQL Procedures and SQL Server T-SQL Procedures.

Using SQL Script Resources

• When complex data processing is required, such as row-by-row manipulation and filtering
• When complex conditional logic is required, such as performing different actions based on input parameter values
• When compound statements are required, such as selecting data and updating based on the selected data
• When a repeatable function can be performed requiring inputs and producing outputs
• When a WHERE clause must be enforced, or where security / data volume is a concern

Tips and Tricks

• SQL Scripts should be unit tested using either another SQL Script or view resource to execute with inputs and validate or display the output
• Build SQL Scripts to perform one functional work unit. This keeps code easy to trace and modify. It also allows the creation of building blocks which can be reused across projects
• Use the PATH keyword to add "paths" to your procedure's environment. This allows the use of shortened names to called procedures and referenced definition sets. For example, adding "PATH /lib/resource;" before your declaration statements allows you to declare a variable of type "ResourceDefs.ParameterValues" instead of "/lib/resource/ResourceDefs.ParameterValues". This will make your code cleaner and easier to read. Note that /lib/debug and /lib/util are added to a procedure's environment by default, which is why you can use "CALL PRINT(...)" instead of having to use "CALL /lib/debug/PRINT(...)".
• As an alternative to SQL definition sets, you can create a procedure that contains more complex constant and data type definitions (row, cursor, and vector datatypes and constants can't be defined in definition sets, for example.) The procedure need not (and probably should not) execute any logic, but should merely be a container of public declarations.

2.4.2 Parameterized Queries

Parameterized queries help to rapidly enable the application of input parameters to view / table resources. The editor allows the graphical creation of simple SQL Script procedures.
Using Parameterized Query Resources

- To quickly and easily apply filters based on input parameters, create a parameterized query
- To quickly webservice-enable a view
- To insulate and provide a layer of abstraction between data source stored procedures or webservice operations, and upstream developed TDV resources

Tips and Tricks

- Take care when editing the SQL code of a parameterized query. Editing this code will permanently disable the Model tab
- Parameterized queries can assist with building out request envelopes for SQL Script procedure calls to web service operations. Simply drag and drop the web service operation into the model, plug in the appropriate values, and switch to the SQL Script tab to see the automatically generated request envelope.

2.4.3 Packaged Queries

Packaged queries let you use database-specific SQL syntax with data sources defined within TDV. The SQL code is then sent for execution directly on the targeted data source.

Sometimes, you already have a complex query written for a particular database and it might not be feasible to rewrite that query in TDV's SQL. For example, your query might require a database-specific feature not available in TDV or perhaps the query takes advantage of the database-specific feature for performance reasons. In such cases, your database-specific query can be created as a packaged query and subsequently be used in other queries.

Every packaged query is associated with a specific data source. A packaged query is stored in TDV with the associated data source’s metadata, and it functions like a stored procedure, accepting input parameters and producing rows of data. It must have exactly one output parameter that is a cursor with at least one column.

Because the TDV system cannot automatically determine the required inputs and outputs of a database-specific query, it is necessary for the user to supply this information. No other optimization is applied to packaged queries.

Using Packaged Query Resources

- Packaged queries are the only resource in TDV that can support data-source-specific SQL syntax
- Packaged queries can be used to store definitions of cache tables and cache configuration tables. These can be re-executed in the future or on different environments
- Packaged queries can be used to create indexes on cache tables

Tips and Tricks

- Multiple database-specific SQL statements can be executed sequentially by the same packaged query as long as only the last SQL statement returns a single cursor with at least a single column output. Use "<version 2>multipartseparator=:+;" as the first line of your packaged query, and end each statement with "; ;"
- TDV can automatically push down filters to your packaged query allowing it to be treated as a derived table and the target of semijoin optimization. To enable this, your packaged query must only contain a SELECT statement, and you must check the “Single Select” checkbox in the editor
- Apart from the above pushing down of filters, the TDV Query Engine cannot optimize packaged queries. It is strongly recommended to use these objects as a last resort. Only create packaged queries where all other options for performing these functions inside TDV, including modifying adapter configurations, have been exhausted
2.4.4 XQuery Procedure
The XQuery Procedure provides a way to create XQuery functions that may be used for advanced processing in TDV XQuery transformations.

Using XQuery Procedure Resources
- XQuery Procedures are the only way to include external XQuery functions and classes in TDV. Create these objects for any external functions you will need in XQuery transformations.

Tips and Tricks
- Create a separate function for each XQuery function that you would like to use. This keeps definitions compartmentalized and easily tested / updated.

2.5 Triggers
A trigger is a resource that initiates a specific system activity to be performed when specified conditions occur. Triggers use a JMS event, a system event, a timer event, or a user-defined event to initiate specific system actions like:

- Execute a procedure
- Gather statistics
- Re-introspect data sources
- Send an e-mail

Triggers can be scheduled to be performed periodically or can occur based on specified conditions. The triggering conditions can cause actions without requiring user interaction.

Triggers provide the following benefits:
- One user can set multiple schedules (using separate triggers) on one resource
- Many users with appropriate privileges can share the same schedule on a resource
- Users can define their own events to trigger several actions asynchronously
- Tasks and actions can be triggered in response to a system event, such as a request failure or the status of a data source
- Procedures can be executed without sending notifications
- Views and procedures can be executed and the results can be sent to one or more recipients

Using Trigger Resources
- Triggers should be used when an action must be performed as the result of some external event taking place (i.e. a user event or system event)
- Triggers are needed to process JMS messages. A trigger will be executed for each JMS message received over a TDV JMS Connector
- Triggers can perform set actions at set times

Tips and Tricks
- Use triggers to create cache windows by dynamically enabling / disabling a cache at a set time each day
Use triggers to execute sanity checks after nightly re-introspection windows have taken place

Triggers are not a replacement for standard job scheduling software products. Do not use triggers to start ETL-like processes inside TDV

For more information on triggers, please see the TIBCO Data Virtualization User’s Guide section titled: Triggers

2.6 Custom Java Procedures
Java procedures are programs that are written in Java to access and use TDV resources. These procedures are added as TDV instance data sources, and can then be executed against data source objects, used in views or procedures, and published as TDV database resources or webservices.

Using Custom Java Procedures
- Custom Java Procedures are useful for integrating non-JDBC-compliant data sources, and data sources requiring the use of an advanced Java API or special processing
- Custom Java Procedures are useful for generating functions not supported in ANSI SQL, or that require access to Java API's and frameworks

Tips and Tricks
- Use a java procedure to integrate data from unsupported data sources that have no JDBC driver
- Use a java procedure to implement advanced processing functions that can only be specified using a full, procedural, programming language

For more information about the TDV APIs for custom Java procedures and examples of Java procedures, see “Java APIs for Custom Procedures” and “Java for Custom Procedures Examples” in the TIBCO Data Virtualization Reference Guide.
3 Annotation Standards

Each resource in TDV has an Info panel. This info panel is where annotations – or general comments – about each resource developed are to be stored. As much detail as possible should be included in this panel so other developers may effectively reuse resources, and so future development on any given resource has full details about a resource requiring changes or updates. The following attributes should be captured in the annotations:

- Description: A detailed long description for the resource
- Input: A list of input parameters that the resource accepts including a description of each. If the resource is a procedure that does not accept any inputs, this should include a single N/A entry with no description.
  
  This can be omitted for views
- Output: A list of outputs that the resource returns including a description of each. If the resource is a procedure that does not return any outputs, this should include a single N/A entry with no description.
  
  This can be omitted for views
- Exceptions: Relevant exceptions. Can be omitted for views
- VersionHistory: Resource change history. Note the author of the change.

At a minimum, the following information should be included:

3.1 Description

Provide a detailed functional overview of the intended purpose of the resource. Include a technical process flow.

Example:

/* Description:
This procedure is used to copy all of the TDV resources from a source folder to a target folder. If the target folder does not exist, this procedure will create it and then copy the source folder resources
   - Check if the target folder exists
     o If not, create it
   - For each resource in the source folder
     o If the resource is a folder, recursively execute this procedure with the source as this object, and target as a subfolder of the target folder
     o Otherwise, execute admin API to copy the resource definition to the target folder
 */
3.2 Input Parameters
Provide an entry for each input parameter required to execute a procedure resource. For views, provide a typical WHERE clause that should be applied when executing the view.

Example:
/* Input:
sourceFolderPath - TDV source folder path. Folder must exist.
   Values: e.g. /shared/Utilities
targetFolderPath - TDV target folder path. Gets created if it does not exist.
   Values: e.g. /shared/tmp
*/

3.3 Output Parameters
Provide an entry for each output parameter or output cursor column.

Example:
/* Output:
success    - true(1) or false(0)
faultResponse - null if successful otherwise contains a fault response.
   Values: Fault Response XML value
*/

3.4 Exceptions
Add all exceptions that are expected from this resource. Any exceptions that may be raised by this resource, and any exceptions that are commonly encountered while executing this resource (but are not explicitly caught,) should be detailed here.

Example:
/* Exceptions:
System.NotFoundException – Raised if the source resource does not exist
TargetCreationFailure    – Raised if the target folder or resource cannot be created
*/

3.5 Version History
For auditing purposes, include a version history section that should be updated any time this resource is changed. Include the date the resource was modified, the author, TDV version used to develop the resource and a brief description of changes made.

Example:
/*
Version History:
*/
3.6 Example
Here is a complete example of an annotation:

/*
Description: Procedure extracts specifically formatted tags from resource annotations in TDV into separate columns for reporting.

Input:
minChangeDate – If this value is populated, only extract annotations with a change date that is greater or equal to the input value. Otherwise return all annotations if this is null.

Output:
Result – The set of extracted annotation values

Exceptions:
None

VersionHistory:
Modified Date:       Modified By:       TDV Version:       Reason:
08/02/2016           Some Developer     7.0.1            Created new
08/10/2016           Some Developer     7.0.2            Added auth column
*/
4 Code Formatting

It is important that resources developed in TDV follow code formatting standards to make resources more readable and easier to understand. This section, while more applicable to SQL Script and other procedure resources that are not generated using the Grid and Model panels, is applicable to all code not machine generated.

Some guidelines follow:

4.1 Indentation

Code should be indented to clearly delineate functional blocks. For example, one level of indentation for SQL Script parameters sections, one level of indentation for code between a BEGIN and END statement, an extra level of indentation in an IF block, etc.

- Increase the indentation level when changing scope through a loop or conditional block such as WHILE, FOR AS, IF, CASE, etc.
- Increase the indentation level when changing SQL code block sections, such as SELECT, FROM, WHERE, GROUP BY, etc.
- Increase the indentation level when text or code statements wrap a line
- Increase the indentation level for each open element in a XML document hierarchy or nested SQL block (see above)

4.2 Capitalization

Capitalization helps separate code block statements from test logic, procedure calls, resource element names, etc. making it easier to read and understand variable and procedure names, as well as finding particular sections of code in complex resources.

- Set in UPPERCASE any internal function, code block definition, data type, or other system reserved word, such as IF, SELECT, CALL, DECLARE, VARCHAR, etc.
- Set in camelCase any parameter or variable name. Capitalize the first letter of a new word making up the name of a parameter or variable, such as thisParameterHoldsSomeValue, thisVariableIsForSomethingInteresting
- Set in InitCaps any defined exceptions, such as MyException, ResourceNotFoundException, etc.
- Set in UPPER_CASE any declared constants, using underscore to separate words, such as CACHE_TABLE, RESOURCE_TYPE_PROCEDURE, etc.
- Leave the case of any underlying data source object as-is, such as a table called foo with a column Bar, when referenced, should be SELECT Bar from foo
- Set in camelCase any new resource developed in TDV. For example, a view using underlying physical layer tables shouldBeNamedLikeThis
4.3 Resource, Column, Variable and Parameter Naming
Columns, variables and parameters should be named for easy identification of the data contained in the object, even at the expense of long names. Further, developers must ensure that the names their objects use reference appropriate ontology naming.

- Sacrifice long names over poor descriptions such as useThisObjectName instead of useObjNm
- Defer external-facing names, such as published resource, column and parameter names, to data leads and architects to ensure proper use of common model, reference data, and ontologies
- Be as descriptive as possible. Remember, it may be a different developer opening this resource for troubleshooting or extension
- Do not use short forms or abbreviations unless directed by data leads or architects

4.4 Inline Comments and Code Blocking
Code blocking is important to keep logical groups of actions or objects together for easy navigation and modification. Inline comments are important to describe the process flow or operations performed by a given code block or set of statements.

- Group logical statements together in such a way as to easily identify groups of related variables, parameters and statements. Add spaces between these logical groups
- Add at least one inline comment to each logical statement group to identify its purpose
- In general, add an inline comment for each line of code or statement that may not be immediately obvious to other developers. Remember, it may be a different developer opening this resource for troubleshooting or extension

4.5 Formatted Example
PROCEDURE changePassword(
    IN oldPassword VARCHAR(255),
    IN newPassword VARCHAR(255),
    IN confirmNewPassword VARCHAR(255),
    OUT result VARCHAR(255)
) BEGIN
    DECLARE userid VARCHAR(255);
    DECLARE domainid VARCHAR(255);
    DECLARE resultXML XML;
    DECLARE faultXML XML;
    DECLARE UpdateUserFaultException EXCEPTION;
    DECLARE NL CONSTANT CHAR(1) DEFAULT ' ';

    CALL GetProperty('CURRENT_USER_NAME', userid);
    CALL GetProperty('CURRENT_USER_DOMAIN', domainid);

    -- make sure the new password and the confirmation password match
    --
    IF (newPassword <> confirmNewPassword) THEN
        RAISE System.IllegalArgumentException VALUE 'The values for newPassword and confirmNewPassword must match.';
-- the Admin API can only change passwords for users in the composite domain
--
IF (domainid <> 'composite') THEN
  IF (domainid = 'dynamic') THEN
    RAISE System.IllegalArgumentException VALUE 'Only users in the "composite" domain can change their passwords. Users of the pass-through domain "dynamic" must use the password changing mechanism of the data source(s) being queried to change their passwords.';
  ELSE
    RAISE System.IllegalArgumentException VALUE 'Only users in the "composite" domain can change their passwords. LDAP users in the domain "| domainid | " must use the password changing mechanism of the LDAP system to change their passwords.,'
  END IF;
END IF;
END IF;

-- execute the password update
--
SELECT
  updateUserResponse,
  fault
INTO
  resultXML,
  faultXML
FROM
  /services/webservices/system/admin/"user"/operations/updateUser(
      <ns1:domainName>composite</ns1:domainName>
      <ns1:userName>' || CAST(XMLTEXT(userid) AS VARCHAR(2147483647)) || '</ns1:userName>
      <ns1:oldPassword>' || CAST(XMLTEXT(oldPassword) AS VARCHAR(2147483647)) || '</ns1:oldPassword>
      <ns1:password>' || CAST(XMLTEXT(newPassword) AS VARCHAR(2147483647)) || '</ns1:password>
    </ns1:updateUser>
  ) updateUser;

-- check for errors
--
IF (faultXML IS NOT NULL) THEN
  RAISE UpdateUserFaultException VALUE 'Unable to update password: ' || NL || CAST(faultXML AS System.Text);
END IF;

SET result = 'Password successfully updated.';
END
5 Help!

Developers may need assistance with building resources and working with TDV.

5.1 Function Support
TDV supports most functions in the ANSI SQL 92/99 standards. For details on each of the functions supported in TDV, including input and output parameters and data types, please see:

- Inside Studio, the Help → Help Topics… option will provide a searchable library of functions
- The TIBCO Data Virtualization Reference Guide installed with each copy of TDV and TDV Studio has full details and code examples as well

5.2 Privilege Issues
Users and groups must have proper privileges assigned in order to access published resources. This includes privileges on all dependent resources and data source objects. For help analyzing whether a resource has the proper privileges assigned, in TDV Manager, see the Security → Resource Management section to analyze the privileges assigned to a user or group.

In general, follow these guidelines:

- Privileges should be assigned at the group level
- Privileges must be assigned at the published resource, the virtual database, schema and catalog, and all dependents including cached objects, tables, procedures, views and columns
- When assigning privileges at the published resource, you may select to apply the privileges to all dependent resources as well
- Remember to check the Copy Privileges from Parent Folder checkbox when creating new resources to match the privileges of the new resource to the parent container at creation time

Details on setting and managing privileges can be found in the TIBCO Data Virtualization Users Guide installed with each TDV Instance and TDV Studio installation

5.3 Missing Data Source Resources
If a particular data source resource (such as a table or procedure) is not available for development, contact the TDV Subject Matter Experts or TDV Architects to arrange to have the resource added to TDV by a Database Administrator and Data Owner.

5.4 Cache Configuration and Refresh Issues
The reason for cache refresh failures will be indicated in the TDV instance logs and cached resource panel. Any and all errors can be tracked and resolved using these error codes. Often, the database administration team will need to assist in debugging database level exceptions and errors.
5.5 Layering
Full details on TIBCO Professional Services recommended layering strategy can be found in the document “Data Virtualization Best Practices White Paper.pdf”

Note the following differences:

For questions or concerns related to proper use of the layering practices and principles, including any exceptions to the practice, please contact the TDV Subject Matter Experts or the TDV Architecture team.

5.6 Who to Call Next
If the above resources do not properly resolve the issue, discuss your issue with other TDV developers.
If you are still unable to resolve your issue, email your TDV Subject Matter Experts or Architects for assistance.

5.7 TIBCO Data Virtualization Community
The TIBCO Data Virtualization Community (https://community.tibco.com/products/tibco-data-virtualization) allows you to collaborate with key data virtualization team members and like-minded customers to directly take part in the planning and development of the TIBCO Data Virtualization product. Customers can also download select TIBCO Professional Services-maintained whitepapers.

Customers wishing to access the TIBCO Data Virtualization Community must be registered, there is a Register link at the above URL.

5.8 When to Call TIBCO Support
TIBCO Support (also known as the TIBCO Technical Assistance Center) should only be contacted by designated TDV Architects or Subject Matter Experts. All avenues of troubleshooting and documentation should be explored before contacting support.

Before contacting TIBCO Support, gather the following information:

- TIBCO Support Contract Number and TIBCO.com user ID
- Confirm that the issue is repeatable and record how to reproduce the issue
- Document the nature of the issue and provide as complete a description of the issue as possible. Describe symptoms, provide screenshots and the describe expected and observed behavior
- Provide a full export of TDV server logs using the Send/Save Logs for Support tool
- Provide a full export of the TDV resources affected and all dependents for Support
- Document what changes, if any, were made such as: the TDV instance was moved to a new virtual machine, differences in functionality after upgrading from patch version 7.0.x.y.z to 7.0.a.b.c

TIBCO support handles issues including, but not limited to, the following:

- Suspected base product issues
- Suspected performance issues in the product
• TDV or Business Directory product feature requests
• General information requests

TIBCO Technical Assistance Center can be contacted:

• Online: https://support.tibco.com/s/mycases (Requires Registration)
• Phone: +1.650.846.5724 or +1.877.724.8227 (1.877.724.TACS) (US)

For worldwide support numbers, refer to TIBCO Contact Us page:

https://support.tibco.com/s/contact-us

**NOTE:** You can log all levels of priority cases online, however if you have logged an Urgent or High priority case, please follow up with TIBCO by calling the Technical Assistance Center (TAC) at the telephone numbers listed on the Contact Us page above. Leave a message if you reach voicemail and state your name, Case Number, telephone number, and indicate the urgency of the issue.

### 5.9 When to Call TIBCO Professional Services
TIBCO Professional Services (PS) should only be contacted by designated TDV Architects or Subject Matter Experts. PS assistance requires the use of billable consultant time, and handles issues including, but not limited to, the following:

• Assistance with architecting, designing, building or debugging resources in TDV
• Assistance with performance troubleshooting and tuning
• Delivery of TDV training
• Problems with or extensions with code or assets developed by users or PS
• General environment or practice setup, documentation and maintenance

Some examples of issues to call PS for:

• “How do I take advantage of incremental caching to improve performance for my use case?”
• “My views are taking too long to return results and I need help to improve response times”
• “We are getting exceptions from a custom web service that we need help fixing”
6 Code Review Checklist

The following items must be satisfied during the resource review process:

- Was the correct resource type chosen?
  - Views for tables
  - SQL Scripts or Parameterized queries for procedures and specialized processing
  - Packaged Queries for database-specific functions only
- Is there another resource with the same name or performing the same function?
  - If so, why do we need this resource
- Were performance analysis and tuning steps taken
  - Is caching needed
- Has the annotation panel been filled out?
  - Has each tag been filled out?
  - Are the details in each tag sufficient to describe the use and logic in the resource?
- Is the code properly formatted?
  - Can a developer follow the use and logic in the resource?
- Is the code properly indented?
  - Clear code blocks created
  - Clear indentation when scope changes
- Are objects clearly named?
  - Resource, variable, parameter and object names are descriptive
  - Proper use of camel, upper and lower case
- Does the resource fit into the proper layer?
  - Physical resources one to one mappings, no renaming
  - Ontology resources with standard naming, formatting, derived columns
  - Business resources filter, join and union data
  - Client Access resources prepare output formats and structure, apply specialized filters and requirements
  - Resources located in the correct folder
- Were Resource Dependencies Analyzed
  - Resource Privileges were correctly set for the resource and any resources that it depends on, including cache tables
  - Resource was testing using an account with end user privileges
  - Resource lineage does not include references to /users folder or published resources
7 Conclusion

In this document, development guidelines for code formatting, commenting/annotation, resource usage, and tips and tricks have been outlined. For more information, please consult the information sources referenced inline.