**Data Validation Module Documentation**

* Terms used:
  + Parent record - The parent record for all child records in a given data stream. Each parent record has a foreign key reference (PTA\_ERROR\_ID) to the associated parent error record.
    - For example the SPT\_VESSEL\_TRIPS table contains the parent records for the RPL data stream and SPT\_UL\_TRANSACTIONS table contains the parent records for the UL data stream.
  + Parent error record (DVM\_PTA\_ERRORS) - The parent record that all error type and error records are associated with when the data validation module is processed.
  + Error record (DVM\_ERRORS) - These records represent individual instances of data validation issues identified by the data validation module and are associated with parent error records.
* System Requirements:
  + PHP 5.4.x
  + SVN repository "Shared\_Library" directory contents: svn://badfish.pifsc.gov/CRED/PHP\_code/web/crei/Shared\_Library
  + Git repository application\_code directory contents from Git repository: [git@pichub.pifsc.gov:sptt/sptt.git](mailto:git@pichub.pifsc.gov:sptt/sptt.git)
  + Windows Operating System configured to execute PHP from the command line (add the PHP module directory containing php.exe to the Windows system PATH variable)
  + php.ini file used when executing PHP via the command line (located in c:\Windows by default) must be configured to load the PHP OCI extension (e.g. extension=php\_oci8.dll) to enable PHP to connect to the Oracle database
  + OCI driver must be installed and configured (instructions can be found here: <http://pichub.pifsc.gov/centralized-data-tools/access-to-oracle-migration/blob/master/docs/Oracle_OCI_driver_install.docx>)
* Overview: The module was developed to provide a framework to validate data entered in the database based on flexible data validation criteria that can be developed and implemented in Oracle Views by a data manager/developer without requiring application development skills. A series of QC Views can be developed to identify problematic values in the database and implemented in the framework to allow each criteria to be evaluated on the given data stream. A given parent record (e.g. SPT\_VESSEL\_TRIPS) of a given data stream (e.g. RPL) and all associated child records (e.g. SPT\_VESSEL\_TRIP\_EVTS, SPT\_SET\_CATCH, etc.) will be validated as a group based on the QC Views defined. Each validation issue that is identified will be saved as a separate error record (DVM\_ERRORS) that includes detailed information about the type of error and a descriptive error message with contextual information that will allow a given issue to be quickly identified, these error records are associated with the parent error record (DVM\_PTA\_ERRORS). The error records can be queried and exported easily in a single unified report for resolution by data management staff.
* Features:
  + Point in time architecture (PTA): The module will save all error criteria that were active at the time that the data was first validated by the module. This will allow the data to be re-validated with the same criteria that were active when it was entered which was implemented so new criteria would not be evaluated on older data which can potentially cause some problems.
  + Data validation issues can be accepted based on manual annotations and an Error Resolution Type. This allows data issues that cannot be resolved or false positives to be marked as valid with the annotation as an explanation.
* Implementation
  + Current Implementation
    - Standalone PL/SQL Package (DVM\_PKG) that can be executed to validate any parent record that has been configured and enabled in the data validation module. The behavior of the module depends on the state of the given parent record. The first time a parent record is evaluated it will save the active validation criteria for future validation module processing and all error records will be associated with the parent error record. All subsequent times a parent record is evaluated it will re-use the saved validation criteria and remove all obsolete errors and add all newly identified errors
    - QC Queries are implemented for groups of tables that comprise a data stream and the resultant Error records are associated with the given parent error record for a given data stream.
  + How to define data validation criteria
    - See external document [here](How%20to%20Define%20Criteria%20in%20Data%20Validation%20Module.docx)
  + Error Records
    - Each individual data issue identified by the Data Validation Module is represented by a separate Error record (DVM\_ERRORS) that includes a description of the error that contains all relevant database values associated with the given data record(s) at the time of evaluation. The Error Type is specified as well as the severity of the error (e.g. warning vs. fatal error).
    - The ability to re-run validation has been implemented preliminarily. The framework purges all existing Error records and re-uses the existing DVM\_ERRORS and DVM\_ERROR\_TYPES records when evaluating the associated Error Types that existed when the parent record was first evaluated (defined by DVM\_PTA\_ERR\_TYP\_ASSOC records).
  + Error Resolution
    - When an error record represents a legitimate value or legitimate set of values a data manager has the ability to enter a manual annotation on the corresponding DVM\_ERRORS record itself and set the ERROR\_NOTES value to an explanation of the reason the error should be considered a false positive (e.g. fishing in IATTC area) or is otherwise exempted (e.g. there is no way to determine the field value). Also, the ERR\_RES\_TYPE\_ID value can be set to the PK of an existing Error Resolution Types (DVM\_ERR\_RES\_TYPES) record that corresponds to the resolution type (currently: No Data Available, Manually Reviewed and Accepted, and No Resolution Can be Reached Yet) to allow the resolution types to be grouped.
    - \*\*Note: there is currently no interface that has been developed to enter accept and annotate data validation issues. This can be developed potentially as part of a larger data management system.
  + Error Report Queries (all Views have comments on all columns and the object itself):
    - DVM\_PTA\_ERRORS\_V – (PTA Errors (View)) This View returns all unresolved Errors associated with a given PTA Error record that were identified during the last evaluation of the associated PTA Error Types. A PTA Error record can be referenced by any data table that represents the parent record for a given data stream (e.g. SPT\_VESSEL\_TRIPS for RPL data). The query returns detailed information about the specifics of each error identified as well as general information about the given Error's Error Type. Each associated date/time is provided as a standard formatted date in MM/DD/YYYY HH24:MI format.
    - DVM\_PTA\_ERROR\_TYPES\_V – (PTA Error Types (View)) This View returns all Error Types associated with a given PTA Error Type record. The PTA Error Type record is defined the first time the given data stream is first entered into the database, all active Error Types at this point in time are saved and associated with a new PTA Error Type record and this is referenced by a given parent record of a given data stream (e.g. SPT\_VESSEL\_TRIPS for RPL data). Each associated date/time is provided as a standard formatted date in MM/DD/YYYY HH24:MI format. This relationship is used to determine the Error Types that were associated with a data stream when the given parent record is first entered into the database.
    - SPT\_XML\_PTA\_ERROR\_TYPE\_V – (XML File PTA Error Types (View)) This View returns all PTA Error Types associated with each XML file that has been processed to date based on the date the given data stream was first entered into the database (defined by CREATE\_DATE). The purpose of this table is to save all active QC Error Type criteria and associate it with a given data stream when it is first entered so that the same validation criteria can be re-evaluated instead of just the QC criteria that is currently active.
      * \*\*Note: this is used specifically for the eTunaLog XML Import module
    - SPT\_XML\_PTA\_ERRORS\_V – (XML File PTA Errors (View)) This View returns all outstanding PTA Errors associated with each XML file that has been processed to date based on the last time the QC validation was evaluated (defined by LAST\_EVAL\_DATE). All dates are returned as native data types as well as standard formatted date strings and all reference table values are included in the result set.
      * \*\*Note: this is used specifically for the eTunaLog XML Import module
  + Error Rule Query
    - DVM\_QC\_CRITERIA\_V – (QC Criteria (View)) This View returns all QC Criteria (Error Types) defined in the database and their associated QC Object, Error Severity, and Error Category. This query is used to define all PTA Error Types when a data stream is first validated in the database
* Algorithm Used:
  + Actual QC validation process is performed by the DVM\_PKG. VALIDATE\_PARENT\_RECORD package procedure defined in the PL/SQL definition file: docs\Data Validation Module\code \ DVM\_PKG.sql.
    - \*\*Note: This process is executed by running a PL/SQL block of code (see [Ex. 1](#Eval_QC_Criteria))
  + The data stream code supplied to the VALIDATE\_PARENT\_RECORD() procedure is queried to determine the given data stream's parent table so that the necessary database queries can be generated.
  + The given parent table is queried to check if the given parent record identified by the p\_PK\_ID primary key value exists, if it doesn't the module does nothing otherwise the process continues.
  + The given parent table's parent error record is queried to check if it exists. If so, the validation criteria that was active when the parent record was first evaluated is loaded for evaluation. If not, a new parent error record is added to the database and associated with the parent record and the currently active validation criteria is loaded and saved in the database for future validation processing.
  + All criteria is retrieved and stored in a PL/SQL package variable for the data validation module evaluation
  + Each QC View object that is defined for the validation criteria is evaluated for the given parent record and each corresponding QC validation check is checked to see if the given error condition was found. Each error condition that is found (indicated by an indicator field value of 'Y' defined by DVM\_ERROR\_TYPES.IND\_FIELD\_NAME on the given QC View) will have a new pending error added to a PL/SQL package variable with an error description generated based on the values returned by the given row of the QC View object query and the placeholders defined in the given error type's ERR\_TYPE\_COMMENT\_TEMPLATE value.
  + Pending/Existing error processing
    - If the parent record has been evaluated before all existing error records that match pending errors (based on error description and error type) are left unmodified. All existing error records that do not match pending errors will be deleted as those error conditions no longer exist on the parent record and associated child records. All pending errors that do not match an existing error record are inserted into the database as they represent new error conditions.
    - If the parent record has not been evaluated before then it will be evaluated using all of the active data validation criteria at the time it is evaluated, these specific validation criteria are saved so they can be re-evaluated at any time in the future on the given parent record. All error conditions will be inserted into the database and associated with the given parent error record.
* Core tables (all tables have comments on all columns and the object itself):
  + DVM\_QC\_OBJECTS – (Data QC Objects) This is a reference table that defines all of the QC validation views that are executed on the data model after a given data stream is loaded into the database (e.g. SPT\_VESSEL\_TRIPS, SPT\_UL\_TRANSACTIONS, SPT\_APP\_XML\_FILES).
    - DVM\_ERROR\_TYPES – (Data Error Types) This is a reference table that defines the different QC Data Error Types that indicate how to identify QC errors and report them to end-users for resolution. Each Data Error will have a corresponding Data Error Type.
      * DVM\_DATA\_STREAMS - (Data Streams) This is a reference table that defines all data streams that are implemented in the SPTT data set. This reference table is referenced by the DVM\_ERROR\_TYPES to define the data stream that the given error type is associated with. This reference table is also referenced by the SPT\_APP\_XML\_POST\_PROC to define the data stream that a given post processing procedure is executed for. Examples of data streams are RPL, eTunaLog, UL, FOT, LFSC. This is used to filter these records based on the given context of the processing/validation
      * DVM\_ERR\_SEVERITY – (Error Severity) This is a reference table that defines all error severities for error type criteria. This indicates the status of the given error type criteria (e.g. warnings, data errors, violations of law, etc.)
      * DVM\_ERRORS – (Data Errors) This is an error table that represents any specific data error instances that a given data table/entity contains (e.g. SPT\_VESSEL\_TRIPS, SPT\_UL\_TRANSACTIONS, SPT\_APP\_XML\_FILES, etc.). These records will be used to communicate errors to the data entry and data management staff
        + DVM\_ERR\_RES\_TYPES - (Error Resolution Types) This is a reference table that defines all error resolutions types that will be used to define the status of a given QC data validation issue. When an error is marked as resolved it will have an annotation to explain the resolution of the error. The types of error resolutions include no data available, manually reviewed and accepted, no resolution can be reached yet. Certain error resolutions types will cause an error to be filtered out from the standard error reports.)
  + Parent Table – This is the given parent table that is enabled in the data validation module (e.g. SPT\_VESSEL\_TRIPS, SPT\_CANN\_TRANSACTIONS, etc.).
    - DVM\_PTA\_ERRORS – (Errors (PTA)) This table represents a generalized intersection table that allows multiple Error records to reference this consolidated table that allows multiple Errors to be associated with the given table record (e.g. SPT\_VESSEL\_TRIPS, SPT\_UL\_TRANSACTIONS, etc.).
      * DVM\_PTA\_ERR\_TYP\_ASSOC – (Error Type Associations (PTA)) This intersection table allows multiple Error Types to be associated with a given table (e.g. SPT\_VESSEL\_TRIPS, SPT\_UL\_TRANSACTIONS, etc.). These associations represent the Error Types that are defined at the time that a given table record is created so that the specific rules can be applied for subsequent validation assessments over time.

Legend:

Completed

* Future plans for additional/modified functionality:
  + Develop import process to allow the QC template spreadsheet to be imported directly into the DVM\_QC\_OBJECTS and DVM\_ERROR\_TYPES tables to remove the requirement to manually enter this data which can be time-consuming and potentially error-prone (possibly just import into temp table and use merge queries to insert/update records into the corresponding tables as necessary).
  + Update algorithm to associate the error records directly with the parent record (e.g. SPT\_VESSEL\_TRIPS vs. SPT\_APP\_XML\_FILES since only the RPL data entered via XML import module will have an associated SPT\_APP\_XML\_FILES record). When this change is made there would be no distinction between parent error records and parent records.
  + Enhance the validation module to maintain existing error records that are identical to the new error records and retain any annotations made to these records so as not to lose the manual work completed by data management staff. This is done by comparing the ERROR\_DESCRIPTION and ERROR\_TYPE\_ID values between existing and pending error records to determine if existing records should be deleted and/or new records should be added
  + Implement the Data Validation Module as a stand-alone PL/SQL package that can be used directly in the database and executed via APEX, PHP, etc. The logic in the evaluate\_QC\_criteria() and re\_run\_validation() methods would need to be implemented in stored procedures in a package.
    - Query update - query for the primary key field name based on the DATA\_STREAM\_PAR\_TABLE field in the DVM\_DATA\_STREAMS table so that it can be used as an argument in the data validation package.
    - Include a stored procedure that can batch re-evaluate records
  + Implement URL generating capabilities into the framework to allow the QC Views to generate their own URLs for application pages that will allow the given error to be resolved easily (e.g. APEX edit trip event link, PHP DM page link). We would need to add to the fields in the DVM\_ERRORS table and define a designated IND\_FIELD to pull this information from (e.g. DATA\_URL). Could implement in foundational views as well and just pull directly into QC query as the link could be valuable in other contexts
  + There is a slow query that needs to be fixed when time permits (SPT\_QC\_EVT\_DIST\_ISSUE\_V). The execution slows down considerably as the database is populated.
  + Develop simple generalized interface to allow errors to be annotated in web interface
    - Could be part of a more generalized application that could be used throughout the Center
  + Look into combining and generalizing the SPT\_PTA\_ERRORS and SPT\_PTA\_ERROR\_TYPES tables (e.g. SPT\_PTA\_VALIDATIONS) since they seem to be redundant. Could use a single key that two different tables reference. This would simplify the logic and reduce the requirements for implementing framework on other data streams.
  + Generalize objects and use less-specific names for the different database entities. (e.g. should not refer to things as errors since some are warnings, maybe use the term "data issue" instead). All object names should be changed accordingly as well.

**Ex 1:**

--sample usage for data validation module:

DECLARE

P\_DATA\_STREAM\_CODE SPTT\_DATA\_VALIDATOR.DVM\_PKG.VARCHAR\_ARRAY\_NUM;

P\_PK\_ID NUMBER;

BEGIN

-- Modify the code to initialize the variable

P\_DATA\_STREAM\_CODE(1) := 'RPL';

P\_DATA\_STREAM\_CODE(2) := 'XML';

P\_PK\_ID := :vtid;

DVM\_PKG.VALIDATE\_PARENT\_RECORD(

P\_DATA\_STREAM\_CODES => P\_DATA\_STREAM\_CODE,

P\_PK\_ID => P\_PK\_ID

);

--rollback;

END;

\*\*Note: The P\_DATA\_STREAM\_CODE variable can handle multiple data stream codes if necessary. P\_PK\_ID is the numeric primary key value of the given parent record that will be validated.