Energy Components

Chevron Data Quality Tool:

Design and Overview

© 2010 Chevron Corporation. All rights reserved.

This document contains confidential and proprietary information of Chevron Corporation, its subsidiaries and affiliates (Company). Its use, reproduction, or disclosure outside Company requires the prior written authorization of Company.

Table of Contents

Revision History 5

1.0 Executive Summary 6

1.1 Document Audience 6

1.2 Change Log 6

2.0 Design Overview 7

2.1 Basic Data Flow 7

2.2 Custom Energy Components Windows 8

2.3 Target Users & Skillsets 8

3.0 Defining Rules & Groups 9

3.1 Rule Groups: 9

3.1.1 Table Definitions 9

3.1.1.1 CT\_DQ\_RULE\_GROUP 9

3.1.1.2 CT\_DQ\_RULE\_GRP\_COMBINATION (Read Only) 10

3.1.2 Class Definitions 10

3.1.2.1 TV\_CT\_DQ\_RULE\_GROUP 11

3.1.2.2 TV\_CT\_DQ\_RULE\_BY\_GROUP (Read Only) 11

3.1.3 Start and End Date Sources 11

3.1.3.1 How to Populate Date Sources 11

3.1.3.2 Usage as SQL Parameters 11

3.1.3.3 Impact on Deletes 12

3.1.4 Report Only Groups 13

3.1.5 EC Security 13

3.2 Rules 13

3.2.1 Table Definitions 13

3.2.1.1 CT\_DQ\_RULE 13

3.2.1.2 CT\_DQ\_RULE\_GRP\_COMBINATION 17

3.2.2 Class Definitions 17

3.2.2.1 TV\_CT\_DQ\_RULE 17

3.2.2.2 TV\_CT\_DQ\_RULE\_BY\_GROUP 17

3.2.3 Rule Definitions – What is a Rule? 17

3.2.4 Rule Description 18

3.2.5 Rule Organization 18

3.2.6 Hierarchy Object Type 19

3.2.7 SQL Source Key Fields (OBJECT\_ID\_SOURCE, DAYTIME\_SOURCE, ALT\_UNIQUE\_KEY\_SOURCE) 20

3.2.7.1 What if I need to override where to get Object ID and Daytime? 20

3.2.7.2 What if Object ID and Daytime are not enough? 21

3.2.8 Additional Information Field 21

3.2.8.1 Why populate the Additional Information Field? 22

3.2.8.2 How to populate the Additional Information Field 22

3.2.9 From SQL Field (FROM\_SQL) 24

3.2.9.1 Start Date and End Date Variables 24

3.2.9.2 From SQL Complexities 25

3.2.9.3 From SQL Mistakes 26

3.2.10 SQL Flex Fields (SQL\_FLD\_1\_SOURCE, etc.) 26

3.2.10.1 SQL Flex Fields Example (Update User ID) 27

3.2.11 Validating SQL 28

3.2.12 Generic Flex Fields (TEXT1, DATE1, etc.) 29

3.2.12.1 Generic Flex Fields Example (Spotfire Grouping) 30

3.2.13 Logging Level 30

3.2.14 Result Retention Days 31

3.2.15 Rule Role ID 32

3.2.16 EC Screen Component 32

3.2.17 EC Security 33

4.0 Executing Rules 34

4.1 Run Rules 34

4.1.1 Table Definitions 34

4.1.1.1 CT\_DQ\_RUN\_LOG 34

4.1.1.2 CT\_DQ\_RULE\_RUN\_LOG 35

4.1.2 View Definitions 36

4.1.2.1 CV\_DQ\_RULE\_GROUP\_LOG 36

4.1.3 Class Definitions 37

4.1.3.1 TV\_CT\_DQ\_RULE\_GROUP\_LOG 37

4.1.3.2 TV\_CT\_DQ\_RULE\_RESULT\_LOG 37

4.1.4 Running by Rule Group or Rule Group and Rule 37

4.1.5 Running by Rule Group and Rule 40

4.1.6 EC Security 41

4.2 Run Rules Scheduling 42

4.3 Execution Restrictions 42

5.0 Reviewing Rule Results 43

5.1 Rule Results 43

5.1.1 Table Definitions 44

5.1.1.1 CT\_DQ\_RULE\_RESULTS 44

5.1.2 Rule Result Hierarchies 46

5.1.3 Rule Result Retention 46

5.1.4 Rule Result History 47

5.1.5 Group Model Type & Navigator Default Values 47

5.1.6 Object Partitioning 49

5.1.7 Reprocessing Rule Results 50

5.1.8 Go to EC Business Function 50

5.1.8.1 Passing Data to the Business Function 51

5.1.8.2 Launching when there is no Business Function 51

5.1.9 Restriction to Returned Detail Records 51

5.1.10 Miscellaneous Scenarios 51

5.1.10.1 What if I delete/inactivate a Rule after it has been run? 51

5.1.10.2 What if I change the SQL after a Rule has been run? 52

5.1.11 EC Security 52

5.2 Rule Run Logs 53

5.2.1 Table Definitions 53

5.2.1.1 CT\_DQ\_RULE\_RUN\_LOG 53

5.2.1.2 CT\_DQ\_RULE\_RUN\_DETAIL\_LOG 54

5.2.2 Class Definitions 54

5.2.2.1 TV\_CT\_DQ\_RULE\_RUN\_LOG 54

5.2.2.2 TV\_CT\_DQ\_RULE\_LOG\_PARMS 54

5.2.2.3 TV\_CT\_DQ\_RULE\_RUN\_DET\_LOG 54

5.2.3 Rule Run Types 54

5.2.4 Runtime Parameters 55

5.2.5 Rule Detailed Logging 55

5.2.6 EC Security 56

6.0 Unique Situation Examples 56

6.1 Objects without Object ID’s (e.g. User ID’s) 56

7.0 Installation and Maintenance Guide 58

7.1 Performing a Fresh Installation 58

7.2 Uninstallation 60

8.0 Appendix 61

8.1 Component Overview 61

9.0 Glossary of Terms 64

Revision History

In the following table, replace the Revision 1 ***author*** and ***date*** with the correct information. Each time you publish an edit to this document, briefly describe your changes, starting at Revision 2. Always list the latest revision first.

| Rev # | Author | Date of Revision | Reason for Revision |
| --- | --- | --- | --- |
| 1.0 | SUKF | Dec 16, 2014 | Initial Version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Executive Summary

Proactively managing data quality is a challenge for any system, including Energy Components. The current approach taken by most EC implementations is to write custom reports using some 3rd party reporting tool in order to identify specific data situations which need to be addressed. These data situations might include missing data, data outside of set or statistical ranges, functional rules, etc. Using reports for this purpose is an adequate approach; however formal reports have the following challenges:

* they lack the flexibility to quickly add or modify the desired quality check logic
* the inventory of the data quality rules is not readily visible since they are imbedded into the reports
* reports do not provide a means to track progress over time
* being outside of Energy Components, the result delivery/management is not always consistent with the EC screens

Energy Components does include functionality to write simple 'check rules' to perform basic data quality checks. The limitation with this delivered product functionality includes the following:

* it is very restrictive on the data which can be assessed (it must be a reporting class)
* it does not provide a means to track progress over time,
* it lacks delivered means to organize/search for rules
* it only supports simple messaging to describe the data in question
* it can be restrictive in the presentation of the results due to the notification approach and the limitation to linking results to existing EC screens

Due to these challenges, a new tool was created to improve the management of data quality checks. By providing the ability to quickly create and manage data quality rules within the EC application, the expected value is to promote the usage of proactive data quality evaluations - which should improve the overall data quality of the production operations data and eliminate many of the subsequent dependent damage caused by the usage of invalid data. This may include both operational improvements since decisions will be made using higher quality information as well as financial improvement by avoiding the cost and effort to correct historical data issues after the data problems have been identified in owner checks, regulatory filings, etc.

## Document Audience

This document is written for support personnel and architects.

## Change Log

This section notes the important changes between versions of the Data Quality Tool and the relevant sections of this document.

# Design Overview

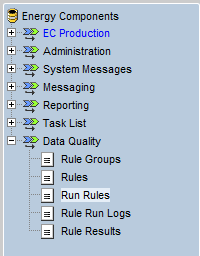
## Basic Data Flow

The following shows the basic data flow for the Data Quality Tool. As shown, Rules are collected into Rule Groups for execution.



## Custom Energy Components Windows

The Data Quality Tool utilizes four new custom EC business functions to define the data quality rules & groups, to execute the rules, and to review the results of the data quality rules. These custom screens utilize the standard EC security functionality to allow the access to these screens to be controlled.



## Target Users & Skillsets

The definition of Rules depends on knowledge of SQL, so it is expected that the person responsible for maintaining the Rules has a background in writing SQL statements. As with SQL from any source (reports, TOAD, ACCESS, Spotfire, etc.) it is possible to write incorrect/extremely inefficient SQL which could utilize database resources to the point of negatively impacting the performance of Energy Components for all users & processes.

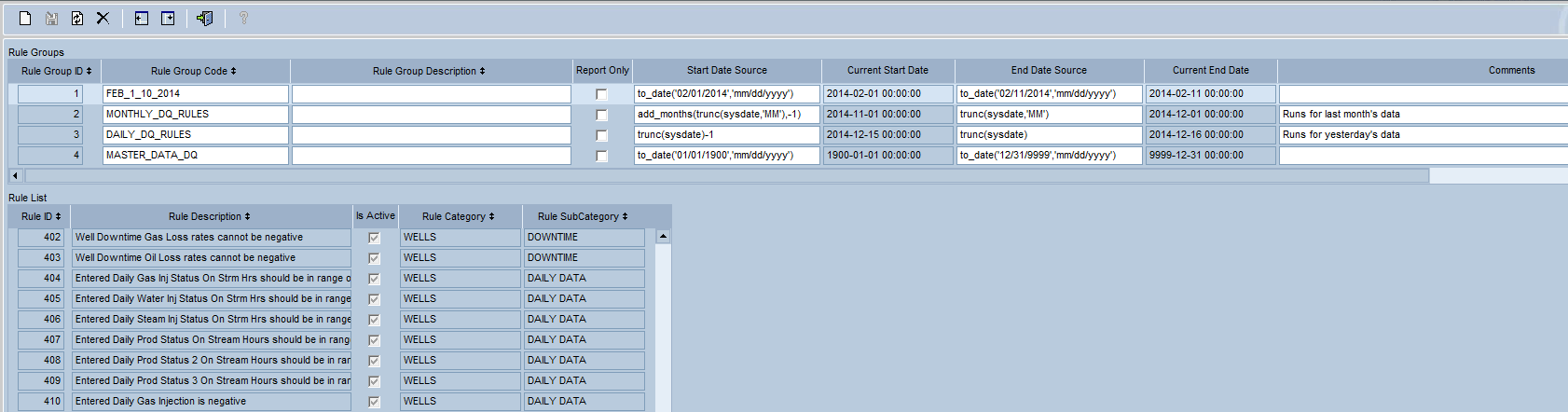
It is a best practice to evaluate the SQL for accuracy and efficiency (e.g. reviewing explain plans) prior to implementing the rules. It is also safest to test Rules/SQL in a non-production environment prior to implementing them in production. Each Rule will log the records identified and run time duration, so this information can be useful in a test environment to identify questionably Rules prior to being implemented in production.

If the maintenance of the Rules is determined to be inappropriate for a BU then it is also possible to discuss the possibility of having the central team develop the Rules and release them as part of the standard EC update releases (similar to how many class updates are made centrally and delivered to BU’s).

# Defining Rules & Groups

The core of the data quality solution is a collection of Rules to evaluate the EC data. These rules are collected into one or more Rule Groups.

## Rule Groups:



Rule Groups define a collection of rules. The two primary purposes of using Rule Groups are:

1. To execute a collection of Rules
2. To provide a means to report a collection of rule results (report only Rule Groups)

### Table Definitions

#### CT\_DQ\_RULE\_GROUP

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Usage |
| RULE\_GROUP\_ID | Number | A unique generated identifier for the rule group. |
| RULE\_GROUP\_CODE | String | A unique user supplied unique identifier for the rule group. |
| RULE\_GROUP\_DESCRIPTION | String | A free form description of the rule group. |
| IS\_REPORT\_ONLY | Boolean | Identifies if the rule group is only used for reporting. If it is set to Y then you will not be able to execute the rule group. |
| IS\_DEFAULT | Boolean | Identifies if the rule group is defined as the default group. If it is set to Y then this is the group used to determine the Rule inventory and the Start Date and End Date when running rules by hierarchy. |
| COMMENTS | String | Free form additional information about the rule group. |
| START\_DATE\_SOURCE | String | A function to be used to determine the Start Date parameter to pass to the linked Rule SQL. For example, entering ‘trunc(sysdate)’ will use the start of yesterday as the Start Date. |
| END\_DATE\_SOURCE | String | A function to be used to determine the End Date parameter to pass to the linked Rule SQL. For example, entering ‘sysdate’ will use the current date as the End Date. The End Date is the first day outside of the included date range. |
| TEXT\_1 – TEXT\_10 | String | Extendable text fields which can be used in the Class configuration to capture additional information on Rule Group. Note that this additional information is for reporting only – there is no usage in the Data Quality Tool processing or screen navigation. |
| VALUE\_1 – VALUE\_5 | Number | Extendable number fields which can be used in the Class configuration to capture additional information on Rule Group. Note that this additional information is for reporting only – there is no usage in the Data Quality Tool processing or screen navigation. |
| DATE\_1 – DATE\_5 | Date | Extendable date fields which can be used in the Class configuration to capture additional information on Rule Group. Note that this additional information is for reporting only – there is no usage in the Data Quality Tool processing or screen navigation. |

#### CT\_DQ\_RULE\_GRP\_COMBINATION (Read Only)

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Usage |
| RULE\_GROUP\_ID | Number | A unique identifier for the rule group. |
| RULE\_ID | Number | A unique identifier for the rule. |

### Class Definitions

#### TV\_CT\_DQ\_RULE\_GROUP

Class to define the Rule Groups; built on table CT\_DQ\_RULE\_GROUP.

#### TV\_CT\_DQ\_RULE\_BY\_GROUP (Read Only)

Read only class to show the Rules associated with the Rule Group; built on table CT\_DQ\_RULE\_GRP\_COMBINATION.

### Start and End Date Sources

One of the primary uses for Rule Groups is to define the date range to apply to the data quality rule.

Setting the dates at the Rule Group level provides the following advantages (over setting the dates in each Rule):

* Increases visibility as to the utilized dates (i.e. you do not need to search each Rule)
* Allows easier maintenance of the dates (i.e. you do not need to change each Rule)
* Allows Rules to be reused for different dates (e.g. you can set up one Rule Group to check yesterday, and another Rule Group to check MTD)

These date parameters have two main uses:

* To pass into the Rules to limit the selected Rule data
* To specify the date range being processed by the Rule Run in order to be able to identify the records which have been resolved

**NOTE:** Rule Groups generally work on the assumption that all Rules within a Group will evaluate data for the same date range. This generally means that master data rules should typically be collected in Rule Groups separate from the Rule Groups used for transactional data (since the master data date range evaluation would typically be far larger).

#### How to Populate Date Sources

The Start and End Date sources are meant to be entered as functions to determine the dates, not necessarily hard-coded dates. For example, a common data quality check would be to run a check every day on yesterday’s data to identify missing or invalid data. In this case you would not want to hard-code to yesterday’s date, since this would only be the correct value for one day (and then you would need to manually change the date every day to check the next day’s records). Instead, the group date functions should be defined using the sysdate function – so instead to check yesterday you could enter the Start Date Source equal to trunc(sysdate) - 1.

If left blank, then the End Date Source will default to the high-date of 12/31/9999.

In the event that you do need to hard code a date (e.g. in a mass long-range check you need to set a hard-coded start date), you could enter the date using the TO\_DATE function, such as to\_date(‘01/01/2000’,’MM/DD/YYYY’).

#### Usage as SQL Parameters

Instead of hard-coding the date restrictions into each rule, the date limits are set at the Rule Group level. When a Rule Group is executed it will automatically restrict the Rule Results by the appropriate date range as specified on the Rule Group. Therefore, it is NOT normally required to restrict the Rule SQL by a date range (e.g. you do not need to add a WHERE clause saying ‘daytime >= ‘ a date.

**NOTE:** The Rule Group will restrict by the Daytime, as specified in the Date Source field on the Rule configuration.

Even though the Rule Group will automatically restrict by the date range, in rare cases it may be required or beneficial (e.g. for performance) to add date restriction logic directly to the Rule SQL. In these rare cases, you can reference two predefined variables for the Start Date and End Date. These predefined variables are :START\_DATE and :END\_DATE. At Rule Group execution, these variables will be replaced with the values of the Rule Group Start Date Source and End Date Source.

For example, you could write a Rule SQL to say ‘where daytime >= :START\_DATE and daytime < :END\_DATE’.

**NOTE:** The End Date is assumed to be the first day outside of the included date range, so do not write SQL to retrieve records BETWEEN the :START\_DATE and :END\_DATE but rather write SQL where the retrieved date is **greater than or equal to** the :START\_DATE and **less than** the :TO\_DATE

#### Impact on Deletes

When a Rule is executed it will identify a list of records that meet the rule criteria, and then each of these results will be logged onto a rule result table (CT\_DQ\_RULE\_RESULTS). Presumably, this list of results will cause data corrections to occur to eliminate some or all of the identified issues. At this point, when the Rule Group is rerun, it would identify a lower number of records – as the corrected records will no longer be returned as issues.

Since the same Rule can be run under multiple Rule Groups (e.g. for different date ranges), the process does not assume that it can simply delete all starting results and populate the new results during each Rule Group execution. Instead, the Rule execution logic only impacts the date range identified in the Rule Group via the Start Date Source and End Date Source fields.

For example, imagine that you had a Rule to identify Well Tests that had not been moved from the NEW status, and that this Well Test Rule was linked to a Rule Group that checked yesterday’s data. After a week’s execution we are on June 11 and the Rule has identified NEW well tests on the 1st, 5th and 10th day of the month. When we re-process the Rule Group on the 11th, it only evaluates well tests on the 10th. Therefore, since we do not know whether the issue on the 1st and 5th have been resolved (we would not know until those dates were re-run), the execution logic is designed to only evaluate the records within the set date range. In our example, the well test SQL will be run against the 10th – any selected records will be added/updated on the result table, and any other records (i.e. those that are not identified via the SQL for the 10th) are assumed to have been resolved.

**NOTE: Because the rule execution uses the Start Date Source and End Date Source to determine the evaluated date range it is important to NOT manipulate the dates within the individual SQL.** For example, if you set the Start Date Source to yesterday (sysdate – 1) and then wrote the Rule SQL to say ‘where daytime >= :START\_DATE – 30’ , then the SQL would successfully select records from 30 days before the Start Date; however the Delete logic would only evaluate the records on or after the Start Date.

**NOTE:** The Delete of records will evaluate all records greater than or equal to the :START\_DATE and **less than the :END\_DATE**, so any SQL using these parameters should also use this logic to retrieve records (i.e. not BETWEEN :START\_DATE and :END\_DATE).

### Report Only Groups

If a Rule Group is marked as a Report Only Group then it is not used to execute a group of Rules (you are also prevented from specifying Start Date Source and End Date Source). Instead, report only groups are created to support reporting of a collection of rule results.

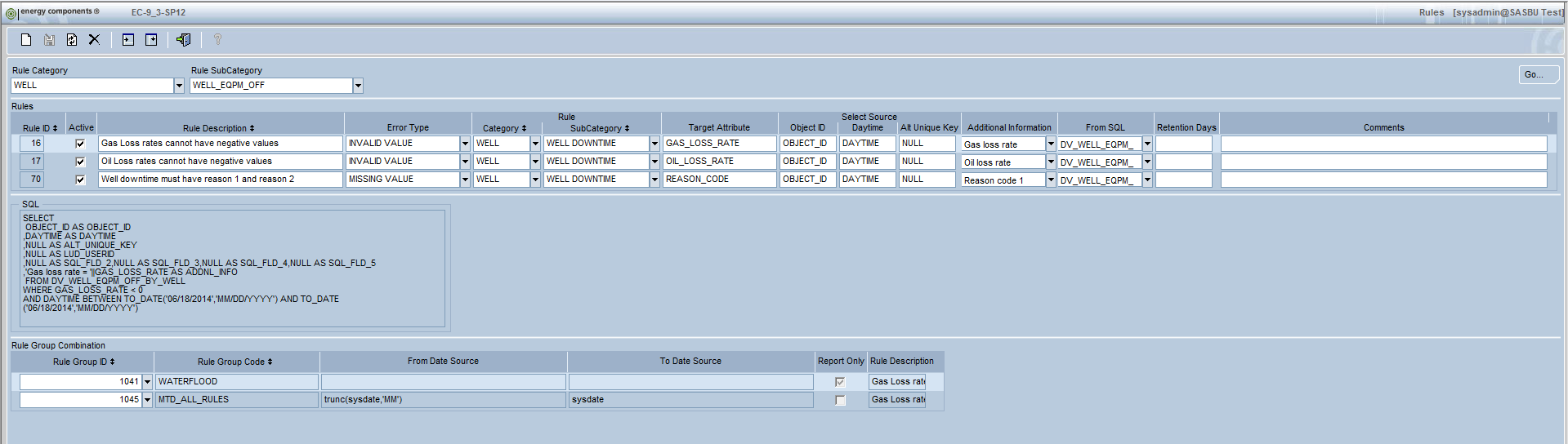
For example, a select number of the data quality Rules may be of interest to a Waterflood implementation. By creating a report only Rule Group for Waterflood containing these rules, it provides the relationship for ad hoc reporting, custom reports, Spotfire, etc. to limit the rule results to only these rules.

**NOTE**: Report Only Rule Groups are not used by the delivered Data Quality EC screens.

### EC Security

The Rule Group screen is defined as object /com.ec.cvx.common.screens/ct\_dq\_create\_rule\_group. Access to this object will be required for a user to maintain Rule Groups.

## Rules



The Data Quality Rules are essentially dynamic SQL statements that will be constructed by combining various user supplied data elements.

### Table Definitions

#### CT\_DQ\_RULE

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Usage |
| RULE\_ID | Number | A unique generated identifier for the rule group. |
| IS\_ACTIVE | Boolean | Identifies if the Rule is active. Inactive Rules will not be validated on the screen or included when a Rule Group is run. |
| RULE\_DESCRIPTION | String | A free form description of the rule group. This description will be displayed on the rule results, so it should be descriptive enough to describe the identified situation. |
| COMMENTS | String | Free form additional information about the rule group. This data is informational only, and has no impact on the Rule processing. |
| ERROR\_TYPE | String | A BU configurable EC Codes table (CT\_DQ\_ERROR\_TYPE) used to define the type of error (e.g. Missing Value, Inconsistent Record, Range Violation, etc.). This data is informational only, and has no impact on the Rule processing. |
| RULE\_CATEGORY | String | A BU configuration EC Codes table (CT\_DQ\_RULE\_CATEGORY) used to organize the Data Quality Rules. This field is used on the Rules BF (required) as well as the Rule Results BF (optional). |
| RULE\_SUBCATEGORY | String | A BU configuration EC Codes table (CT\_DQ\_RULE\_SUBCATEGORY) used to organize the Data Quality Rules. This field is used on the Rules BF (required) as well as the Rule Results BF (optional). |
| OBJECT\_TYPE | String | This defines the object type of the objects being evaluated by the rule. This object type determines how to retrieve the hierarchy information. |
| ATTRIBUTE\_NAME | String | A free form user supplied field to help identify the target of the Rule. This information is only used to help locate a specific rule. This data is informational only, and has no impact on the Rule processing. |
| OBJECT\_ID\_SOURCE | String | The column which should be used to populate the object\_id column on the rule results. This is required on all Rules |
| DAYTIME\_SOURCE | String | The column which should be used to populate the daytime column on the rule results. If there is no daytime/date on the evaluated record then enter the word ‘NULL’. |
| ALT\_UNIQUE\_KEY\_SOURCE | String | The column which should be used to populate the alternate unique key column on the rule results. If the object id and daytime are enough to identify a single record on the evaluated data then enter the word ‘NULL’. |
| RESULT\_RETENTION\_DAYS | Number | Identifies the length of time in days that results should be retained. If you leave this value blank then all records will be retained until they are resolved. |
| APP\_ID | String | The Application ID associated with the entered Role ID |
| ROLE\_ID | String | The Role ID allows Rules to be associated with a Role in order to restrict the Rule Results available for certain users. If the Role ID is left blank then the corresponding Rule Results will be available to all users (assuming they have access to the appropriate hierarchy) |
| SCREEN\_COMPONENT\_ID | String | Optionally, each Rule can be associated with an EC screen/business function. This allows the user of the Rule Results screen to launch directly to the applicable EC screen to make date corrections. |
| LOGGING\_LEVEL | String | By default, when a Rule Group is executed, the counts of total Rule Results for the Run will be logged at a Rule level (e.g. 10 errors for Rule #1, 12 errors for Rule #2). In some cases, a more detailed level of logging might be desired (e.g. logging counts by Rule and Area) in order to be able to do trending analysis on historical data quality Runs. |
| SQL\_FLD\_1\_SOURCE – SQL\_FLD\_5\_SOURCE | String | If a site wants to add additional data on the populated rule results then these five fields identify the source for this additional data. For example, you might want to store the update user on every rule result, so you would use one of these fields to identify the source of this user id (e.g. last\_updated\_by). Class configuration can be used to rename and/or hide these fields as needed. |
| FROM\_SQL | String | This field defines the SQL statement starting with the ‘FROM’ and including the appropriate WHERE clause. For example, typically you would start with the table/class being read – followed by the appropriate WHERE clause. The only variables supported in this field are :START\_DATE and :END\_DATE. |
| ADDNL\_INFO | String | The object\_id, daytime and additional unique key should identify a specific record in error. If you want to also supply any additional information about the record then this field is used to construct a message made up of hard coded text plus any appropriate variables (identified by a preceding ‘:’) from the selected source (table, class, etc.) |
| TEXT\_1 – TEXT\_10 | String | Extendable text fields which can be used in the Class configuration to capture additional information on Rule Group. Note that this additional information is for reporting only – there is no usage in the Data Quality Tool processing or screen navigation. |
| VALUE\_1 – VALUE\_5 | Number | Extendable number fields which can be used in the Class configuration to capture additional information on Rule Group. Note that this additional information is for reporting only – there is no usage in the Data Quality Tool processing or screen navigation. |
| DATE\_1 – DATE\_5 | Date | Extendable date fields which can be used in the Class configuration to capture additional information on Rule Group. Note that this additional information is for reporting only – there is no usage in the Data Quality Tool processing or screen navigation. |

#### CT\_DQ\_RULE\_GRP\_COMBINATION

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Usage |
| RULE\_GROUP\_ID | Number | A unique identifier for the rule group. |
| RULE\_ID | Number | A unique identifier for the rule. |

### Class Definitions

#### TV\_CT\_DQ\_RULE

Class to define the Rules; built on table CT\_DQ\_RULE.

#### TV\_CT\_DQ\_RULE\_BY\_GROUP

Class to define the relationship of Rules to Rule Groups; built on table CT\_DQ\_RULE\_GRP\_COMBINATION.

### Rule Definitions – What is a Rule?

The customizable SQL for a Rule allows a great flexibility in writing a Data Quality Rule; however it is best to keep each Rule specific to an individual problem versus trying to combine multiple rules into one giant IF/CASE statement or collection of UNIONs.

By being specific, it allows for some benefits, including:

* The rule description can define a specific problem
* The rule additional information can provide details specific to one issue versus listing many items
* The rule grouping, via the category, subcategory and attribute can be specific, which improves the management and searching for rules
* Specific rule make data analysis easier to target specific issues, instead of simply making generic conclusions
* Specific rules allow the designation of the EC Screen to enable direct launching from the Rule Results screen

For example, if you wanted to analyze a well test for valid Gas Lift rates on gas lift wells, non-NULL Gas Rates, non-NULL wellhead pressures, and tests left in NEW status, instead of combining the rules into one giant rule such as:

Rule #1: Description = Invalid data on well test, Rule Category = Well, Rule SubCategory = Well Test, Attribute = ??

SELECT OBJECT\_ID, DAYTIME, 'Production Method = ' || PROD\_METHOD ||', Gas Lift Volume = ' || GL\_RATE\_MSCFPERDAY || ', Wellhead Pressure = ' || WH\_PRESS\_PSIG || ', STATUS = ' || STATUS as ADDNL\_INFO

FROM RV\_PWEL\_RESULT

WHERE (PROD\_METHOD = 'GL' AND GL\_RATE\_MSCFPERDAY IS NULL)

OR GAS\_RATE\_ADJ\_MSCFPERDAY IS NULL

OR WH\_PRESS\_PSIG IS NULL

OR STATUS = 'NEW';

While the above example would work as a Data Quality Rule, a better approach might be to split the rules into logic components, such as the following. While this approach will allow the same records to be evaluated by multiple rules, it creates more targeted rules.

Rule #1: Description = Gas lift wells must have a supplied gas lift rate on well tests’, Rule Category = Well, Rule SubCategory = Well Test, Attribute = Gas Lift Rate

SELECT OBJECT\_ID, DAYTIME, NULL as ADDNL\_INFO

FROM RV\_PWEL\_RESULT

WHERE (PROD\_METHOD = 'GL' AND GL\_RATE\_MSCFPERDAY IS NULL);

Rule #2: Description = Well Tests should be Rejected or Accepted’, Rule Category = Well, Rule SubCategory = Well Test, Attribute = Gas Lift Rate

SELECT OBJECT\_ID, DAYTIME, NULL as ADDNL\_INFO

FROM RV\_PWEL\_RESULT

WHERE STATUS = 'NEW';

### Rule Description

Each Rule has a Rule Description. Since this decryption will appear with the Rule Results, it should be descriptive enough to fully describe the situation being identified.

Vague Rule Description: ‘Pressure cannot be NULL’. Notice that this message is likely not descriptive enough since it does not define which pressure (bottom hole, tubing, casing, etc.) and it does not specify what type of record captured this pressure (daily status, well test, well event, etc.).

Specific Rule Description: ‘Wellhead Pressure cannot be NULL on Production Well Daily Status records’. Notice that the message defines the specific data element in error, as well as the record type where the data was captured.

### Rule Organization

One of the challenges with the Rules will be managing the inventory of Rules as the number gets larger. Since the Rule business logic will be imbedded into the SQL statements, it can be a challenge to locate and prevent duplicates from being created.

To alleviate this challenge, each Rule is designated in a Rule Category and SubCategory. These elements are linked to EC Code tables (CT\_DQ\_RULE\_CATEGORY, CT\_DQ\_RULE\_SUBCATEGORY) which also use the code dependency configuration to tie subcategories to categories.

Categories and Subcategories can be customized by each BU to organize the Rules in the most logical groupings. These groupings can be object based (WELL, STREAM, etc.), product based (OIL, GAS, etc.), frequency based (DAILY, MONTHLY, etc.), etc. to provide the most efficient organizational methodology for the rules.

The Category and Subcategory will be used on the Rules screen when creating/maintaining Rules, as well as on the Rule Results screen as an optional method to limit the returned results. So for example, if all of the Well Test Rules are grouped together (e.g. WELL – WELL TEST) then the Rule screen will allow the list of Rules to be filtered to only the Well Test Rules, and the Rule Results screen will allow the Rule Results to only show the Well Test Rule Results (plus a limitation using the hierarchy).

In addition to the Rule Category and Subcategory, each Rule has an Attribute data element. This Attribute element is free-form (i.e. not necessarily linked to the actual class\_attribute table) and allows each rule to indicate which element is being targeted by the rule. The Attribute element has no processing implications – it is only there to help classify the Rule to make it easier to quickly understand the focus of the rule (versus having to look into the actual SQL statement).

### Hierarchy Object Type

Each Rule has a mandatory association with an object type. This object type defines the type of object being evaluated by the Rule. For example, a well test rule would be a Well object type. Besides helping to describe the purpose of the rule, the object type is also critical in determining how to populate the hierarchy information for the Rule Results.

The population of the object type field on the Rules screen does not use the standard EC object type full list. Instead, this drop down is sourced from a new custom code table CT\_DQ\_HIER\_OBJ\_TYPE, and then each of these configured object types is linked to an entry on a configuration table used to defined how to retrieve the hierarchy information (CT\_DQ\_HIER\_DETERMINATION).

The hierarchy determination configuration on CT\_DQ\_HIER\_DETERMINATION is similar to the Rules configuration. Each entry – defined by object type – defines the source fields and FROM\_SQL field for the SQL which will be used to determine how to populate the hierarchies. This table comes pre-populated with some standard hierarchy determination methods for the basic EC object types, but can also be customized as needed.

The SQL source fields are comprised of source fields for object id, daytime, end\_date, and then source fields for the operational, collection point, and geographical hierarchies. These hierarchy source fields also determine if an installation wants to populate certain hierarchy fields on the Rule Results. By leaving fields NULL on the hierarchy determination table, you are indicating that these data elements are not to be populated on the Rule Results (e.g. if you do not care about the geographical hierarchy then you can leave all of the GEO Source fields blank/NULL).

**NOTE**: The Object ID Source needs to point to the same object id from the Rule. For example, if the Rule evaluates Well Bores then the Object ID Source must be a Well Bore Object ID (not the Well Object ID, even though Well was used to get the hierarchy).

In most situations the Hierarchy Determination configuration is simply associating an object type with an object view (e.g. STREAM with OV\_STREAM, WELL with OV\_WELL, etc.); however by using a separate custom object type table, this allows an installation to do the following:

* Inactivate the object types not associated with their data quality rules
* Build custom hierarchy determination methods for a single EC object. For example, if there was a need to see the hierarchy differently for two stream related rules, then two different stream hierarchy object types could be configured (each with its unique hierarchy determination configuration).
* By using hierarchy object type, we avoid requiring that every Rule SQL be written to select all of the hierarchy attributes, the EC Data Quality Tool uses the concept of hierarchy object type to determine how to associate the Rule Results with a hierarchy
* Some of the EC Objects do not have a direct linkage to hierarchies (e.g. Well Bores, Perforation Intervals, etc.). In these cases, instead of hard coding the relationship, the Hierarchy Determination Table defines how to go from these objects to a hierarchy (e.g. for Well Bores, you need to go from the Well Bore table to the Well table to get hierarchy).
* Hierarchy object types allow for the creation of rules for non-object id objects (e.g. user id’s) which can then be assigned to hierarchy levels through these object types

### SQL Source Key Fields (OBJECT\_ID\_SOURCE, DAYTIME\_SOURCE, ALT\_UNIQUE\_KEY\_SOURCE)

Every Rule SQL is essentially a SELECT statement to return records which meet specific criteria. One of the critical needs from every one of the Rules is to uniquely identify the specific record which is identified. For example, if the Rule identifies Well Downtime records which are missing a reason code, then the Rule needs to return the unique key for the Well Downtime record (object\_id and daytime).

In most cases in EC, the unique key will simply be an object\_id and a daytime. Therefore, to support most Rules, the Rule must identify the data columns to use to populate object\_id and daytime. This is where the OBJECT\_ID\_SOURCE and DAYTIME\_SOURCE fields are used.

**NOTE**: Currently every Rule must have been associated with an Object ID. This is due to the fact that the Object ID is used to determine the hierarchy (which is used to navigate to the Rule Results).

#### What if I need to override where to get Object ID and Daytime?

By default, the generated SQL statements will assume that the SQL needs to return the columns OBJECT\_ID and DAYTIME; however in some cases this may not be completely accurate, so the SOURCE fields allow for the specification of the ‘override’ fields to be used.

Here are some examples:

1. The Rule does not have a DAYTIME, such as a SCADA Tag mapping rule to see where the Tag mappings are mapped to inactive data elements. In this case you would enter the word NULL into the DAYTIME\_SOURCE in order to specify that nothing will be returned for this data element.
2. The targeted class/table/etc. has a different name for OBJECT\_ID or DAYTIME. For example, a Rule evaluating the Stream Set Lists (TV\_STREAM\_SET\_LIST) would need to enter STREAM\_ID into the OBJECT\_ID\_SOURCE field.
3. The SQL will use a join of several tables/classes/etc. so the Rule must specify the correct source for the OBJECT\_ID and/or DAYTIME. For example, if the Rule SQL joined the Equipment Event Data (DV\_EQPM\_EVENT) to an equipment object class, such as Compressor (OV\_COMPRESSOR) then the OBJECT\_ID\_SOURCE would need to specify which OBJECT\_ID to use (both are the same; however the SQL needs to be told which one), and the DAYTIME\_SOURCE would need to be set to DV\_EQPM\_EVENT.DAYTIME since the DAYTIME from OV\_COMPRESSOR would not be appropriate to identify a particular Equipment Event record in error.

#### What if Object ID and Daytime are not enough to uniquely identify a record?

As previously stated, most Rules will uniquely identify records using only the Object\_id and Daytime; however there are a few exceptions when these two fields are not enough. For example, standard EC allows the same well and daytime to have multiple well tests – so in this case the OBJECT\_ID and DAYTIME alone may be shared by several Well Tests. In order to provide the additional information needed to uniquely locate a record, the Rules have a field called ALT\_UNIQUE\_KEY\_SOURCE. This data element (which defaults to NULL) can be used to specify the source of the additional data element(s) to be used to uniquely identify a single record. In the case of Well Tests, this can be done by entering RESULT\_NO into the ALT\_UNIQUE\_SOURCE\_ID. This way, every Well Test identified by the Rule will store the OBJECT\_ID, DAYTIME, and RESULT\_NO.

In the event that multiple additional data elements are required to uniquely define a record then you simply need to concatenate multiple fields together in the ALT\_UNIQUE\_KEY\_SOURCE field. For example, if you needed to specify RESULT\_NO and VALID\_FROM\_DATE then you would set the ALT\_UNIQUE\_KEY\_SOURCE equal to RESULT\_NO || VALID\_FROM\_DATE (or RESULT\_NO || ‘,’ || TO\_CHAR(VALID\_FROM\_DATE,’MM/DD/YYYY’) if you prefer a more readable approach – although both work the same).

**NOTE**: The Alternate Unique Key is a text field, so if you need to store a date you might want to configure the ALT\_UNIQUE\_KEY\_SOURCE with a to\_char function in order to specify the desired date format. For example TO\_DATE(VALID\_FROM\_DATE,’MM/DD/YYYY’).

### Additional Information Field

As previously mentioned, the Rule Results will uniquely identify the record which meets the Rule criteria. In many cases, this plus the description of the Rule is enough to immediately understand the issue.

Here are some Rule examples where knowing the Well (OBJECT\_ID) and Daytime are enough to understand the problem:

* Well Test should be Accepted or Rejected, not NEW
* Well Downtime record missing a Reason Code
* On Stream Hours is NULL for the Production Well Daily Status record

On the other hand, in many Rules, there is additional information needed to be able to understand the identified problem. In these cases, there is an optional field on the Rules called the Additional Information Field (ADDNL\_INFO) which can be used to provide more data.

#### Why populate the Additional Information Field?

Here are some examples of how it could be used:

* The Rule identifies where a value is outside of a range, so the actual value of the field would be useful (e.g. A Rule for Well Head Pressure is > 5000 should show the value of the Well Head Pressure)
* The Rule identifies a problem, but some master data information would be useful to see (e.g. The Rule identifies where a Well Test is missing a value, and we would also like to show the Well Class to help understand how critical this omission might be)
* The Rule identifies a problem, but it would also be useful to see some of the change control information (e.g. The Rule identifies a record in error, but we want to also show who updated the record last and when the update occurred)

#### How to populate the Additional Information Field

The ADDNL\_INFO field can be comprised of hard-coded portions as well as variables combined into a single SQL field.

This field is not intended to repeat the description of the Rule. For example, if the Rule Description is ‘Well Test well head pressure should not be negative’ then we would NOT want to set the Additional Information Field to say ‘Well Test well head pressure is negative’.

The variable portion of this field is identified by preceding the variable with a colon. For example, to show the value of the well head pressure you would enter :wh\_press into this field. Variables can also be functions. In all cases, when the SQL is executed, the colon will be removed, and everything after the colon (until the next space) will be assumed to be non-literals.

For variables, the Additional Information Field can refer to any data element available from the selected data class/table/etc. For example, if the SQL selects from DV\_PWEL\_RESULT then any field from this class (VALID\_FROM\_DATE, GOR, CREATED\_BY, etc.) can be included in the ADDNL\_INFO. Obviously the reverse is also true – that fields not available from the class are not available (e.g. WELL\_CLASS, NAME, etc.).

Here are some examples of populating the Additional Information Field:

1. Showing the value of a variable

ADDNL\_INFO = The value of the well head pressure is :wh\_press

1. Showing multiple variables

ADDNL\_INFO = The value of the well head pressure is :wh\_press and the record was created on :created\_date by :created\_by

1. Using functions (shows the word NULL if the value is NULL)

ADDNL\_INFO = The value of the well head pressure is :decode(wh\_press,NULL,’NULL’,wh\_press)

1. Using ‘special’ characters at the end of variable

ADDNL\_INFO = The value of the well head pressure is :wh\_press, the gas lift pressure is :gl\_diff\_press

\*\*\* the SQL interpreter will understand that the comma after :wh\_press is part of the literal part of the additional information instead of part of the variable.

Since the variable is defined as the information after a colon until the next space, there are some opportunities for errors due to formatting:

1. Failing to leave a space (you do not need a space for the last variable)

ADDNL\_INFO = Pressures are :wh\_press:gl\_diff\_press

\*\* since the two variables are jammed together, the SQL interpreter will assume that the variable is a single field named wh\_press:gl\_diff\_press

1. Using a colon as a literal instead of variable

ADDNL\_INFO = The well head pressure is: :wh\_press

\*\* The first : after the word ‘is’ will be eliminated since it will be assumed to be part of a variable

1. Adding extra spaces within a function

ADDNL\_INFO = The well head pressure is :decode(wh\_press, NULL,wh\_press)

\*\* The space within the decode statement will indicate to the SQL interpreter that the variable has ended, so the result will be a variable decode(wh\_press followed by a literal , NULL,wh\_press)

**NOTE**: The Additional Information is a text field, so if you need to store a date you might want to configure with a to\_char function in order to specify the desired date format. For example:

The valid from date is :TO\_CHAR(VALID\_FROM\_DATE,’MM/DD/YYYY’)

Notice that the TO\_CHAR is preceded by a colon to indicate that it is not a literal, and there are no spaces in the function.

### From SQL Field (FROM\_SQL)

The From SQL field is meant to contain the SQL logic used to identify the records meeting the specific criteria. This field starts with the word ‘FROM’ in a select statement and includes the appropriate WHERE clause.

For example, to identify well tests that are still in a NEW status the FROM\_SQL field would be entered as the following:

FROM DV\_PWEL\_RESULT

WHERE STATUS = ‘NEW’

**NOTE:** Do not end the FROM SQL with a semicolon.

#### Start Date and End Date Variables

As mentioned in the discussion of the Rule Groups, the executable Rule Groups are defined with a Start Date and End Date Source in order to define a target date range. The execution of the Rule Group will automatically restrict the returned results by the appropriate data range established on the Rule Group, so typically Rule SQL will **NOT** need to include a date restriction.

For example, a Rule to identify Well Tests still in the NEW status could have a FROM\_SQL written as follows:

FROM DV\_PWEL\_RESULT

WHERE STATUS = ‘NEW’

If this Rule was executed as part of a Rule Group limited to a single day then the retrieved ‘NEW’ Well Tests would be limited to that single day, while the same Rule executed a YTD Rule Group would restrict to ‘NEW’ Well Tests for YTD.

**NOTE** that the date restriction will be performed against the value of the Daytime field – as determined by the DAYTIME\_SOURCE.

In very rare cases, it is possible to reference the Rule Group Start Date and End Date parameters in the FROM SQL. For example, a complex Rule SQL may have performance benefit from duplicating the date restriction within the SQL. In these rare cases, in order to pass these dates to the Rule SQL statements, two predefined date variables are utilized. Similar to the variables in the Additional Information Field, these date variables are defined with preceding colons :START\_DATE and :END\_DATE.

For example, if there was a Rule which relied on a DISTINCT or UNION there might be performance benefit to add the restrictions directly to the Rule SQL. The following shows the required usage of these predefined variables (although the example SQL does not justify the usage of these variables):

FROM DV\_PWEL\_RESULT

WHERE STATUS = ‘NEW’

AND DAYTIME >= :START\_DATE

AND DAYTIME < :END\_DATE

**NOTE** that the SQL is written to pull records less than the :END\_DATE (not less than or equal).

**NOTE** that if you specify the date restriction in the SQL then it needs to have the same impact as the restriction auto-generated at the Rule Group execution (i.e. the resulting date ranges need to be identical). Differing logic will result in unexpected determination (and corresponding deletes) of resolved records.

#### From SQL Complexities

Since the From SQL is relatively free form, it is possible to add complexity beyond a simple select. Here are some examples:

1. Joins (well test for allocated wells where the oil volume is NULL)

From dv\_pwel\_result a, ov\_well w

where w.object\_id = a.object\_id

and w.daytime <= a.daytime

and nvl(w.end\_date,a.daytime + 1) > a.daytime

and w.alloc\_flag = ‘Y’

and a.net\_oil\_vol is NULL

\*\*\*\* Note that this SQL would require the object\_id\_source and daytime\_source fields to be populated to specify whether the values should be used from the data class (a.object\_id, a.daytime) or the object class (w.object\_id, w.daytime). Also, in this case RV\_PWEL\_RESULT could have been used instead of the Join.

1. Subselects (Wells without a well test)

From ov\_well w, system\_days s

where

w.daytime <= s.daytime

and s.daytime < nvl(w.end\_date,s.daytime +1)

and not exists

(select object\_id from dv\_pwel\_result t

where w.object\_id = t.object\_id

and valid\_from\_date = s.daytime)

\*\*\* Note that in this case the DAYTIME\_SOURCE will need to be set to the DAYTIME from SYSTEM\_DAYS

1. Subqueries (Monthly allocated water not equal to sum of daily allocated water)

FROM

(select

daytime,

object\_id,

nvl(ALLOC\_WATER\_VOL,0) as monthly\_vol

FROM DV\_PWEL\_MTH\_ALLOC

) a,

(select

object\_id,

sum(nvl(alloc\_WATER\_vol,0)) as daily\_vol,

trunc(daytime,'MM') as daytime

from Dv\_pwel\_day\_alloc

group by object\_id, trunc(daytime,'MM')

) b

where a.object\_id = b.object\_id

and a.daytime = b.daytime

and b.daily\_vol <> a.monthly\_vol

\*\*\* Note that any time a subquery is used the subquery must select all of the data elements referred in the OBJECT\_ID\_SOURCE, DAYTIME\_SOURCE, ALT\_UNIQUE\_KEY\_SOURCE, ADDNL\_INFO, SQL\_FLD\_1\_SOURCE, SQL\_FLD\_2\_SOURCE, SQL\_FLD\_3\_SOURCE, SQL\_FLD\_4\_SOURCE, SQL\_FLD\_5\_SOURCE. In this case we have the object\_id and daytime, and have access to the monthly\_vol and daily\_vol for the ADDNL\_INFO field

\*\*\*\* Note that this SQL would require the object\_id\_source and daytime\_source fields to be populated to specify whether the values should be used from the monthly class (a.object\_id, a.daytime) or the daily class (b.object\_id, b.daytime).

#### From SQL Mistakes

* Do NOT end the FROM\_SQL with a semi-colon
* Do NOT try to add too many separate checks into a single Rule (e.g. using UNIONS)
* Do NOT try to use any variables other than :START\_DATE and :END\_DATE

### SQL Flex Fields (SQL\_FLD\_1\_SOURCE, etc.)

As previously stated, since the Rule Results will all be stored on the same table and reviewed through the same custom EC screen, the columns retrieved in the Select statement have to be consistent – and are by default the object\_id, daytime, alt\_unique\_key, and addnl\_info.

In some situations, a BU might decide that – for all Rules – they want to explicitly store (i.e. beyond adding it to every Rule’s addition information field) an additional attribute. To support this need, there are 5 generic SQL columns on the Rules table (SQL\_FLD\_1\_SOURCE through SQL\_FLD\_5\_SOURCE). Any of these 5 fields can be used – along with the renaming in the Class configuration, to store additional information.

On the Rule configuration, these 5 fields are used to specify the source of the data element (i.e. which table/class column will be used to populate the data element). Later in this document when the Rule Results are described, we will see that the Rule Results table (CT\_DQ\_RULE\_RESULTS) has 5 corresponding table columns which would be used to store the values for each Rule Result.

When looking at the generated SQL (as discussed in more detail in 3.2.9), the disabled/unmapped SQL fields will default to NULL. Any SQL field mapped on the Rule Results Class will appear in the SQL with the ATTRIBUTE\_NAME used in the Class configuration.

**NOTE**: The SQL Flex fields are text fields, so if you need to store a date you might want to configure the SQL Flex fields with a to\_char function in order to specify the desired date format. For example TO\_DATE(VALID\_FROM\_DATE,’MM/DD/YYYY’).

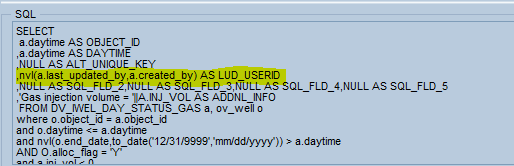
#### SQL Flex Fields Example (Update User ID)

One example of how these SQL Flex fields could be used is a situation where a BU wants to see the last update user id on every Rule Result. This might be desirable to identify over time if there are particular users who are responsible for the majority of the data issues – in order to be able to focus training, communication, etc. to those particular people.

Here are the steps for using a flex field to store the last updated user:

1. Alter the CT\_DQ\_RULE class to map one of the SQL flex source fields to a new attribute for the last updated user source (e.g. LUD\_USERID\_SOURCE)
   1. Add the new attribute (CLASS\_ATTRIBUTE)
   2. Map the column to SQL\_FLD\_1\_SOURCE (CLASS\_ATTR\_DB\_MAPPING)
   3. Create the attribute presentation as needed – making it free form versus linked to EC users, etc. (CLASS\_ATTR\_PRESENTATION)
   4. If applicable, add a class trigger action to default the new data element – for example you could default to nvl(last\_updated\_by, created\_by) (CLASS\_TRIGGER\_ACTION)
   5. Generate the Class
2. Alter the CT\_DQ\_RULE\_RESULTS class to map one of the SQL flex fields to a new attribute for the last updated user (e.g. LUD\_USERID)
   1. Add the new attribute (CLASS\_ATTRIBUTE)
   2. Map the column to SQL\_FLD\_1 (CLASS\_ATTR\_DB\_MAPPING)
   3. Create the attribute presentation as needed – readonly=true (CLASS\_ATTR\_PRESENTATION)
   4. Generate the Class

At this point, the SQL generated from the Rule screen will show that the data entered into LUD\_USERID\_SOURCE data element will be selected into attribute defined on CT\_DQ\_RULE\_RESULTS.

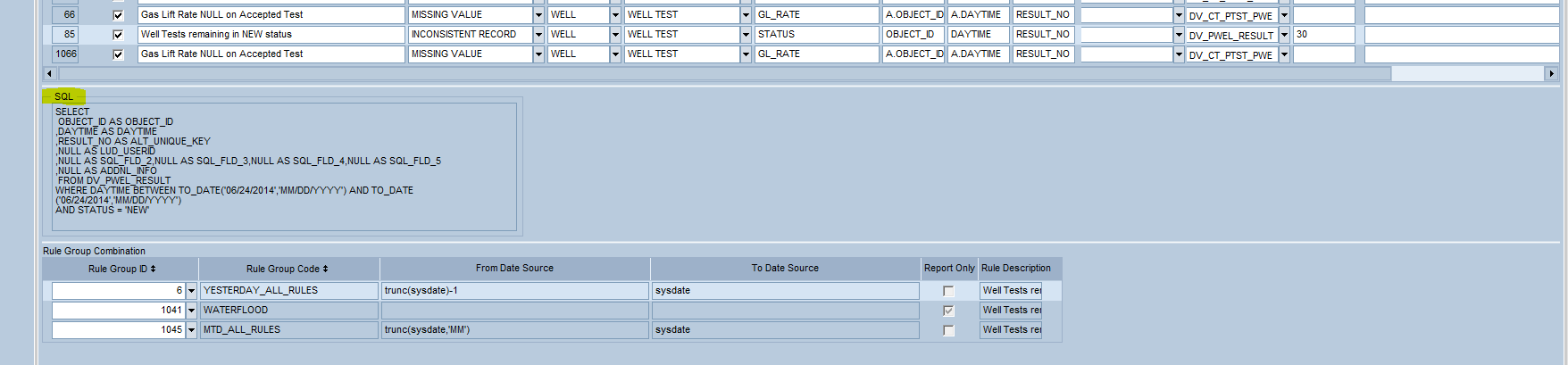


This also means that any records identified by this rule will store this information on the Rule Results.

### Validating SQL

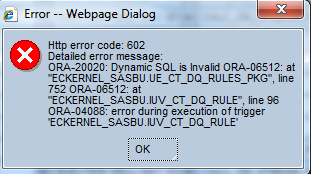
The generated SQL statement is made up of multiple components, so there are several opportunities for making entry mistakes which would result in invalid SQL.

To help resolve any issues, and to assist in the confirmation that the SQL is functionally written correctly, the Rule screen shows the generated SQL.



In addition, to prevent these issues from being passed to the Rule Run execution (and the execution failing), the Rule screen is built with SQL validation logic.

When any active rule is saved on the Rule screen, the SQL is generated and validated. If there are any issues an error message is generated indicating that the Dynamic SQL is invalid.



**HINT**: Since the generated SQL will not display until the record is saved, and the validation function will prevent the record from being saved, there is a process which makes the validation of SQL easier.

1. Start with the rule marked as inactive (do not check the active box). Since only active Rules are validated, this allows the data entry to be completed, the Rule to be saved, and the SQL to be generated
2. Once you are completely done entering the Rule configuration, then mark the Rule as active and save the record. This will validate the SQL
3. If there is a validation error, then manually review the generated SQL – or copy & paste the SQL into TOAD, etc. to help identify the issue
4. Alter the configuration and save the Rule
5. Try to activate and save the Rule again (repeat correction loop as needed)
6. Once the Rule can be saved as active then you have confirmed that the SQL is valid

### Generic Flex Fields (TEXT1, DATE1, etc.)

The Rule table (CT\_DQ\_RULE) also contains several generic fields (TEXT1 – TEXT10, DATE1 – DATE5, VALUE1 – VALUE 5). These flex fields are similar to the flex field throughout the rest of EC. They can be utilized in custom BU class configuration to allow for the capture of additional information on Rules.

**NOTE**: The delivered Data Quality processing does not utilize these fields, so they would typically only be used to capture additional information for custom BU reporting.

Similar to the standard EC tables, there are functions available to retrieve the field values for Rules (EC\_CT\_DQ\_RULE) and Rule Groups (EC\_CT\_DQ\_RULE\_GROUP). These functions are particularly useful if you have chosen to use a custom flex field on either class, since the functions will allow you to add a custom attribute to the Rule Results class (CT\_DQ\_RULE\_RESULTS) to display this information.

For example, you could add a custom attribute on the Rule Results class to map to the function call EC\_CT\_DQ\_RULE.TEXT\_1(RULE\_ID) in order to show the value of the TEXT\_1 field.

#### Generic Flex Fields Example (Spotfire Grouping)

One example of a usage for these flex fields on the Rules would be to support the presentation of the Rule Results in Spotfire.

**NOTE**: The usage of Spotfire would be a BU specific approach to Rule Result delivery.

Spotfire data presentations are typically grouped by tabs – e.g. a tab for pre-allocation checks, one for post allocation checks, one for ECIS checks, etc. While it is possible to use the standard Rule Category and Rule Subcategory fields to equate to these tabs, it would also be possible to utilize these flex fields specifically for Spotfire grouping.

Here are the steps for using a flex field to store the Spotfire grouping:

1. Alter the CT\_DQ\_RULE class to map one of the flex fields (e.g. TEXT\_1) to a new attribute for the Spotfire grouping code (e.g. SPOTFIRE\_GROUP\_CODE)
   1. Add the new attribute (CLASS\_ATTRIBUTE)
   2. Map the column to TEXT\_1 (CLASS\_ATTR\_DB\_MAPPING)
   3. Create the attribute presentation as needed – making it free form versus linked to EC Codes table, etc. (CLASS\_ATTR\_PRESENTATION)
   4. Generate the Class
2. If desired, also add the new data element to the Rule Results class (CT\_DQ\_RULE\_RESULTS)
   1. Add the new attribute (CLASS\_ATTRIBUTE)
   2. Map the column to a function retrieving the TEXT\_1 value from the Rule (CLASS\_ATTR\_DB\_MAPPING)
   3. Create the attribute presentation as needed – making it read only, etc. (CLASS\_ATTR\_PRESENTATION)
   4. Generate the Class

At this point, the Rule screen would allow the entry of a Spotfire Grouping Code and this Spotfire Grouping Code would be available on the Rule Results to be used to filter the results to the appropriate Spotfire tab.

### Logging Level

By default, each execution of a Rule Group will log the total number of errors (broken down by the number of new and old errors) for each Rule. In some cases, this total error by Rule may not be detailed enough to support the analysis of historical error count trends.

The logging level on the Rule configuration allows for the designation of a logging level by Rule. The logging level options are defined by an EC Codes table (CT\_DQ\_LOGGING\_LEVEL), which can be customized by inactivating the delivered options, or adding new entries.

The Logging options on the EC Codes table are limited to the table fields from the Rule Results table (CT\_DQ\_RULE\_RESULTS). Typically, a level in one of the hierarchies will be selected as a more detailed logging level. For example, by choosing the Operational Area as the logging level for a particular Rule, it will mean that instead of simply tracking that an execution of a Rule Group identified 100 errors for a Rule – the logging level configuration will instead make the logging track at the Operational Area and Rule. So in the example of 100 total errors, you might see that for a Rule the log now shows 80 errors for Area #1 and 20 errors for Area #2.

The format of the EC Code entries for CT\_DQ\_LOGGING\_LEVEL is to specify the attribute to GROUP BY in the ALT CODE column. For example, by adding OP\_AREA\_ID into the ALT\_CODE attribute, the Rule will be logged at the Operational Area. Notice that the Logging Level of Rule ID has nothing populated in the ALT\_CODE – indicating that no additional subtotaling is required.

Logging Level will default to the EC Code on CT\_DQ\_LOGGING\_LEVEL which is marked with IS\_DEFAULT = ‘Y’. If no codes are marked as default (or multiple are marked as default) then the Logging Level will be stored as NULL.

When no logging level is set (attribute is NULL) then the default will be to log at the Rule level.

**NOTE**: If you go too detailed (FC1, Object ID, etc.) then you will create a lot of log entries.

### Result Retention Days

The Result Retention Days field allows you to specify how long a Rule Result record should be retained. By default, or if this field is left NULL, the expectation is that the Rule Results will be retained until the issue is resolved.

For example, if you run a Rule looking for Stream Volumes where the Net Volume is NULL then any identified records will remain on the Rule Result table (CT\_DQ\_RULE\_RESULT) until the Net Volume is changed to a non-NULL value and the Rule is re-executed. This approach is likely appropriate for most of the Rules (i.e. the Result Retention Days should be left NULL for most Rules) since most of the Rules will have identified invalid data which needs to be corrected.

In some less frequent cases, retaining the results forever might not be appropriate. For example, if you were to create a Rule that identified Well Tests which were entered late (e.g. Well Tests are expected to be entered by 9:00 am on the following work day), then the Rule Results can actually never be corrected (i.e. there is no way to change the Well Tests to be on time once they are late). In this case, if you were to retain the Rule Results forever then the Results table would just continue to append more and more results with nothing ever removing them. One way to address this situation would be to use the Result Retention Days field to enter how long the Results should be retained.

A value of 0 means that only the current results would be retained and all others would be removed from the table. A value of 5 would mean that 5 days of results would be retained (assuming that you ran a daily rule).

Keep in mind that the age of the Rule Results is based on the last time the Rule Result was identified (i.e. the last run date), not the daytime of the record. For example, if today you ran a set of Rules to evaluate all of 2013 data and you identified a record from January 2013 in issue, then the age of the record would be measured from today (since it was identified today), not January 2013. If you then reran the same Rule for all of 2013 again without correcting the issue, then the last run date of the Rule Result would be updated again.

### Rule Role ID

There is a single EC transaction to review the Rule Results. As discussed in the Rule Result section of this document, the Rule Results are retrieved by hierarchy (operational or collection point),

The usage of the navigator hierarchy can allow a user to focus on returning only the Rule Results for their area of responsibility; however the hierarchy alone may not provide the ability to completely segregate the Rule Results to those responsible for their correction.

For example, within a level of the hierarchy it is probable that there are unique groups of people for the different record types within EC – such as one group responsible for the equipment data and another for the well data, or one group responsible for the injection wells and another for the producing wells. Using hierarchy alone would mean that the results would be commingled whenever either group reviewed the Rule Results. While procedurally each group could simply ignore the results for the Rules which do not apply to each group, another option is to use the Role ID on the Rule set-up.

By assigning a Role ID to a Rule, the impact will be that the Rule Result screen will be limited to only those Rules (and corresponding Rule Results) associated with Roles linked to the user.

So using one of the example above, if the Rules associated with the injection wells were linked to a Role uniquely tied to the Injection Well users then only the injection well users would be able to see the Rule Results for the injection well Rules.

**NOTE**: If the Role ID is omitted from a Rule then all users with access to the Rule Result screen and the appropriate hierarchy will be able to see the Rule Results.

### EC Screen Component

Optionally, each DQ Rule can be associated with an EC Screen. This enables the ability for the user of the Rule Results screen to launch directly to the appropriate EC screen to correct the data in issue.

**NOTE**: The intention to use the EC Screen functionality on EC Rules dictates that the EC Rules should be defined to only evaluate one EC Screen’s record. For example, a combined rule looking at all Production Well Daily Status records would create a challenge in selecting a single EC screen since multiple EC screens would be included in the Rule (Data\_1, Data\_2, etc.). A more effective strategy would be to create unique EC Rules for each EC Screen.

### EC Security

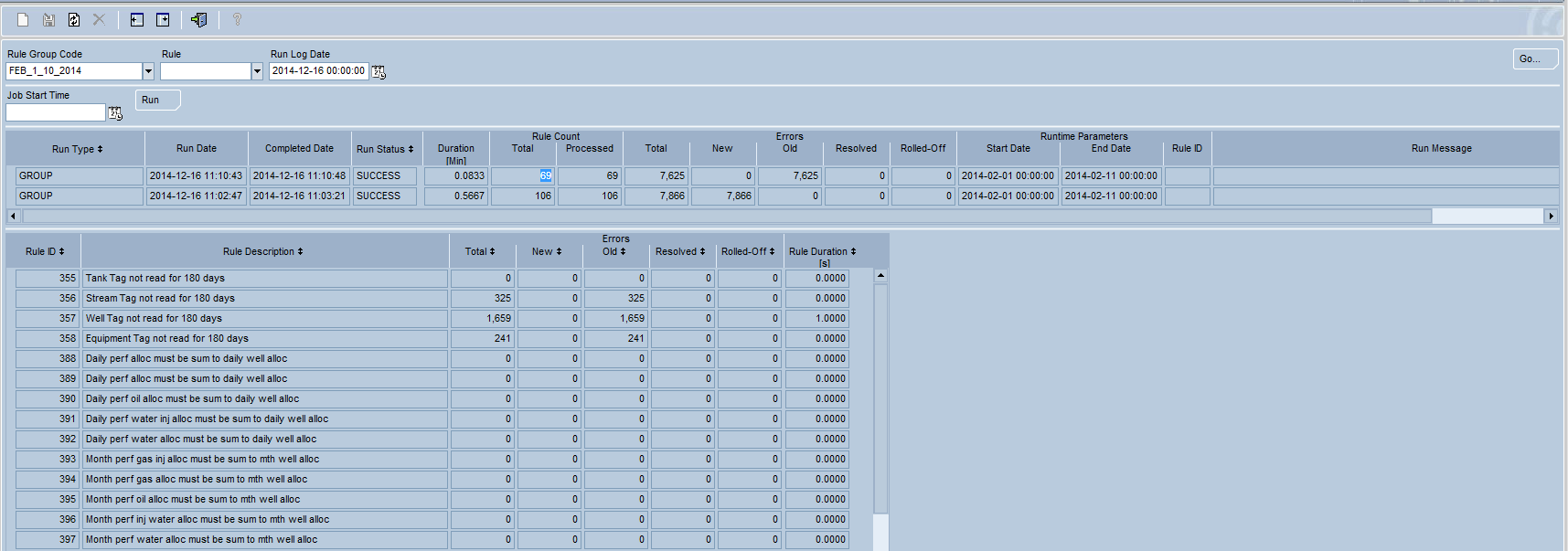
The Rule screen is defined as object /com.ec.cvx.common.screens/ct\_dq\_rules. Access to this object will be required for a user to maintain Rules and Rule to Rule Group Combinations.

In addition, as defined above, if Rules will be defined to ROLE\_ID’s then the appropriate users will need to be assigned to these Rule Roles in order to be able to see the Rule Results.

# Executing Rules

Once the Rules have been associated with at least one executable Rule Group they can be executed. There are two methods to run a Rule Group: by using the Run Rules business function, or by scheduling a job using the EC Scheduler. In both cases, the summary results (counts, times) can be reviewed through the Run Rules business function.

## Run Rules



The Run Rules business function allows you to both run a Rule Group and review the overall results.

### Table Definitions

#### CT\_DQ\_RUN\_LOG

This table has one record for each job execution.

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Usage |
| RUN\_ID | Number | The system generated unique number associated with this rule run. |
| RUN\_TYPE | String | Identifies if an entire group was run (GROUP), and individual rule within the group (RULE), or a specific rule result was reprocessed (RESULT). |
| RULE\_GROUP\_ID | Number | The unique identifier for the rule group. |
| RULE\_ID | Number | The unique identifier for the rule, if the Run was executed by Rule. |
| HIER\_GROUP\_TYPE | Number | The group type (operational, collection point) used if the Run was executed for a given hierarchy. |
| HIER\_OBJECT\_ID | Number | The hierarchy object (area\_id, fc1\_id, etc.) used if the Run was executed for a given hierarchy. |
| REC\_OBJECT\_ID | Number | The object id (well, stream, etc.) if the Run was executed for a particular record. |
| REC\_DAYTIME | Number | The daytime if the Run was executed for a particular record. |
| REC\_ALT\_UNIQUE\_KEY | Number | The alternate unique ID if the Run was executed for a particular record. |
| PARM\_START\_DATE | Number | The value of the Start Date when the Run was executed. |
| PARM\_END\_DATE | Number | The value of the End Date when the Run was executed. |
| RUN\_DATE | Date | The datetime for the start of the rule execution. |
| COMPLETED\_DATE | Date | The datetime when the run completed. |
| RUN\_STATUS | String | Tracks the run from RUNNING to COMPLETE (or FAILURE). |
| RUN\_MESSAGE | String | If the Run fails then this message provides the error information. |

#### CT\_DQ\_RULE\_RUN\_LOG

This table has one record for each rule within a job execution.

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Usage |
| RUN\_ID | Number | The system generated unique number associated with this rule run. |
| RULE\_ID | Number | The unique identifier for the rule. |
| TOTAL\_ERRORS | Number | The total number of rule results identified for this rule during this run. |
| TOTAL\_NEW\_ERRORS | Number | The number of rule results for this rule which were newly identified in this run. |
| TOTAL\_OLD\_ERRORS | Number | The number of rule results for this rule which were previously identified prior to this run. |
| TOTAL\_DELETED\_ERRORS | Number | The number of rule results which were deleted/resolved for this rule in this run. |
| TOTAL\_ROLLED\_ERRORS | Number | The number of rule results which were rolled off for this rule due to the rule retention days parameter in this run. |
| STARTED\_DATE | Date | The datetime that the rule execution began. The run time is the difference between this time and the completed\_date |
| COMPLETED\_DATE | Date | The datetime that the rule execution completed. The run time is the difference between this time and the started\_date |

### View Definitions

#### CV\_DQ\_RULE\_GROUP\_LOG

This view is derived as a combination of the overall Run information (CT\_DQ\_RUN\_LOG) and the individual Rule log information (CT\_DQ\_RULE\_RUN\_LOG). It is grouped by the Rule Group ID in order to show the overall Rule Group Run information.

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Usage |
| RUN\_ID | Number | The system generated unique number associated with this rule run. |
| RUN\_TYPE | String | The Run Type identifies if the Rule was run as part of a Rule Group (GROUP), as an individual Rule (RULE). This view does not show records for the reprocessing of individual Rule Results (RESULT). |
| RULE\_GROUP\_ID | Number | The unique identifier for the rule group. |
| RULE\_ID | Number | The unique identifier for the rule, if the Run was executed by Rule. |
| HIER\_GROUP\_TYPE | Number | The group type (operational, collection point) used if the Run was executed for a given hierarchy. |
| HIER\_OBJECT\_ID | Number | The hierarchy object (area\_id, fc1\_id, etc.) used if the Run was executed for a given hierarchy. |
| REC\_OBJECT\_ID | Number | The object id (well, stream, etc.) if the Run was executed for a particular record. |
| REC\_DAYTIME | Number | The daytime if the Run was executed for a particular record. |
| REC\_ALT\_UNIQUE\_KEY | Number | The alternate unique ID if the Run was executed for a particular record. |
| PARM\_START\_DATE | Date | The value of the Start Date used during the execution of the rule group. |
| PARM\_END\_DATE | Date | The value of the End Date used during the execution of the rule group. |
| RUN\_DATE | Date | The datetime for the start of the overall rule group execution. |
| COMPLETED\_DATE | Date | The datetime for the end of the overall rule group execution. |
| RUN\_STATUS | String | Tracks the run from RUNNING to COMPLETE (or FAILURE). |
| RUN\_MESSAGE | String | If the Run fails then this message provides the error information. |
| DURATION | Number | The duration of the overall Group run, as computed by the difference between the completed date and the run date. |
| RULE\_COUNT | Number | The number of Rules associated with the Rule Group – as determined by the number of executed Rules. |
| TOTAL\_ERRORS | Number | The summed total number of rule results identified for this run. |
| TOTAL\_NEW\_ERRORS | Number | The summed number of rule results which were newly identified in this run. |
| TOTAL\_OLD\_ERRORS | Number | The summed number of rule results which were previously identified prior to this run. |
| TOTAL\_DELETED\_ERRORS | Number | The summed number of rule results which were deleted/resolved in this run. |
| TOTAL\_ROLLED\_ERRORS | Number | The summed number of rule results which were rolled off due to the rule retention days parameter in this run. |

### Class Definitions

#### TV\_CT\_DQ\_RULE\_GROUP\_LOG

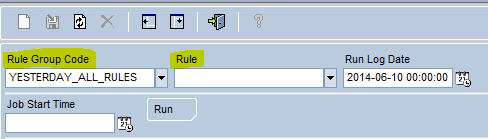
Class to show the overall Job execution; built on view CV\_DQ\_RULE\_GROUP\_LOG.

#### TV\_CT\_DQ\_RULE\_RESULT\_LOG

Class to show the individual Rule Results within a Job; built on table CT\_DQ\_RULE\_RUN\_LOG.

### Running by Rule Group or Rule Group and Rule

All Rules are run within a Rule Group. If you run the entire Rule Group (i.e. leave the Rule blank) then all Rules within the Rule Group are executed. The Run Log Date in the navigator will restrict the returned records (when the GO button is selected) to only those run on or after the entered Run Log Date. This date defaults to Yesterday.



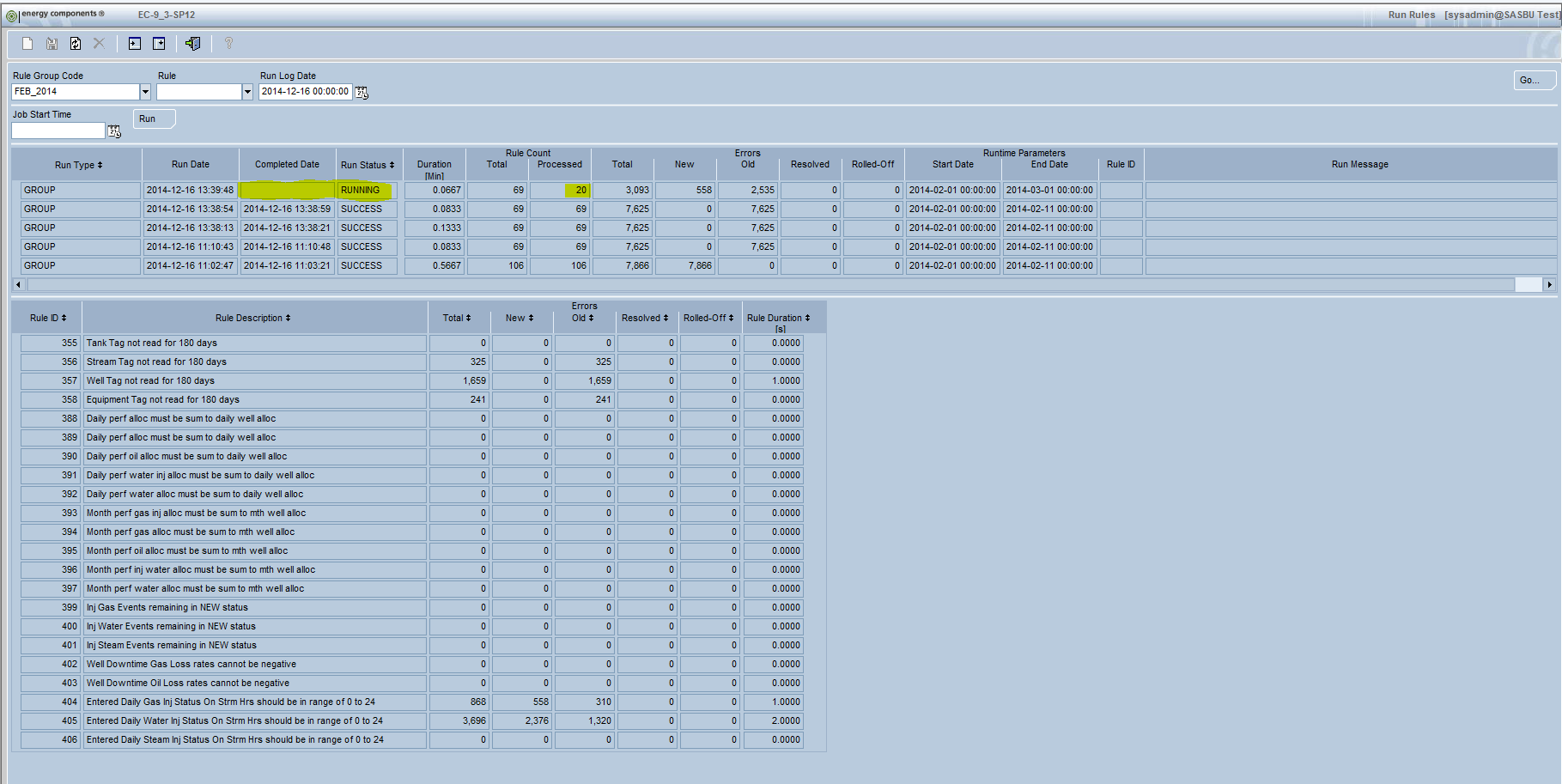
When an entire Rule Group is executed, you will see the generated log entry shown with a Run Type of ‘GROUP’ and the Rule Count Total will show the number of Rules defined in the Rule Group (in the example below there are 33 Rules in the Rule Group).

**NOTE**: The Total Rule Count is dynamically computed by looking at the Rule Group Combination table, so if you Run a Rule Group and then add/remove Rules you will see that the Processed Rules (which is fixed) no longer equal the total rules (which is dynamic).

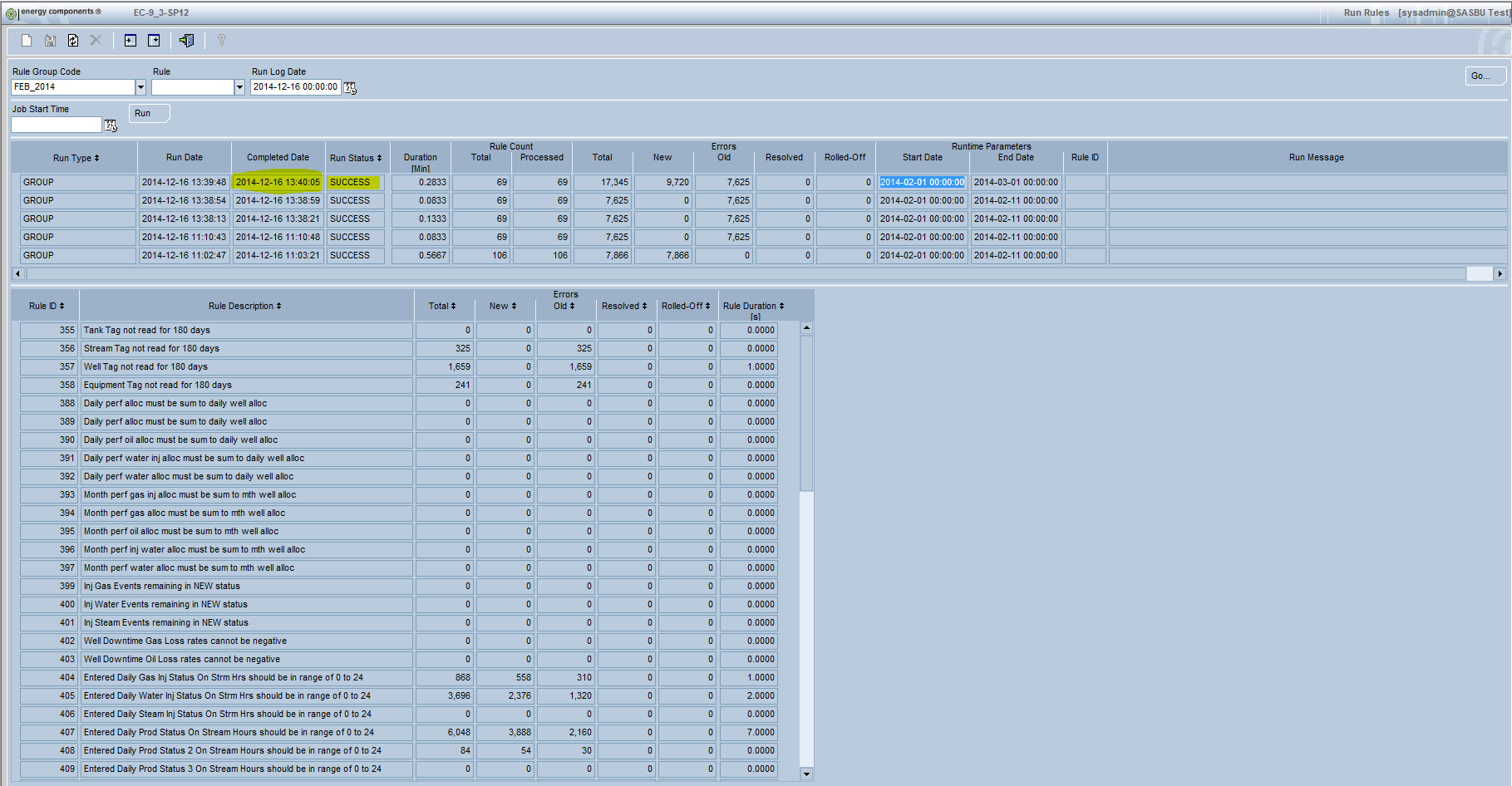
The Run log also shows the values for the Start Date and End Date parameters which were used in the execution of the Rules.

Identified errors are classified as ‘New’ if they have not been previously logged to the Result table, or ‘Old’ if they already exist on the Rule Result table. As previously discussed, this is why it is important on the Rule configuration to make sure that the Object ID, Daytime and Alternate Unique Key fields combine to identify a single record. If the key is incorrect (e.g. leaving off RESULT\_NO from a well test Rule), then the possible symptom would be to see unexpected ‘Old’ errors on Runs where only new data was expected (e.g. if you run a daily check on yesterday’s data and in the first run it identifies an ‘Old’ record then it is possible that two identified errors have ended up creating a single Rule Result). This situation will also be identified by the number of generated Rule Results being less than the number of identified Total Errors.

As the Rules are executed, the Run Status will show RUNNING. In addition, the Processed Rule Count will continue to be incremented – along with the Duration and Error Counts – until all Rules are processed. In the example below you can see that after 0.0667 minutes we have processed 20 out of the 69 total rules and have identified around 3,000 total errors. The lower section of the screen will show the details for the completed Rules.

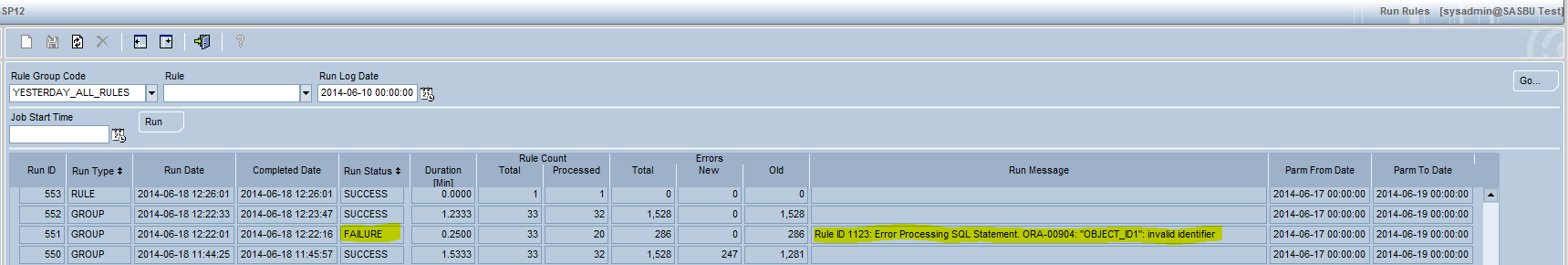


Once the entire Rule set has been processed, the Run Status will be changed to SUCCESS, and the Duration and Error counts now represent the results for all of the Rules. Also notice that the Rule Total matches the Processed Rule Total.

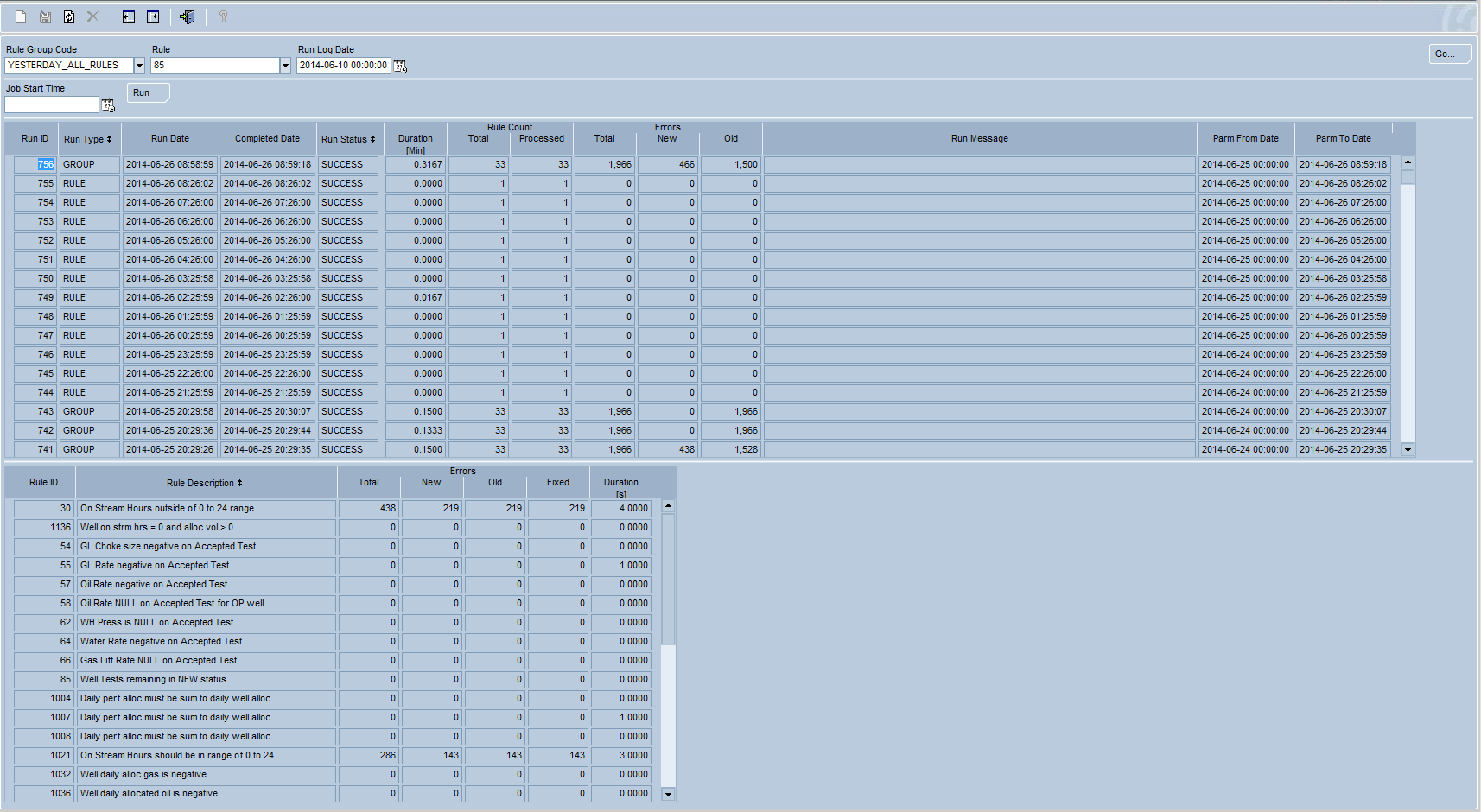


In the event that a technical failure occurs during the execution of one of the Rules, the Run Status will show FAILURE, and the Run Message will detail the reason for the failure.

Note that the earlier Rules will have completed successfully and posted their results. In the example below 20 out of the 33 Rules were processed successfully. This means that the 286 errors for these rules will exist on the Rule Result table. If the issue was corrected and the Rule Group was reprocessed then these 286 errors would simply appear as ‘Old’ errors on the next run (assuming none were corrected or added in the time between runs).

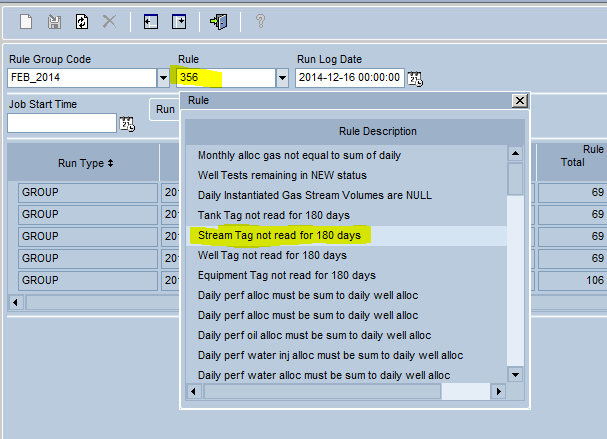


Once a Run is completed (or during), by selecting the Run in the top window section, it will display to details counts for the Run in the lower window section. In the example below, the top window section shows that there are 1,966 total errors for all Rules, while the lower section shows the errors for each individual Rule (adding up to 1,966). The lower window section also shows the duration broken down by each Rule.

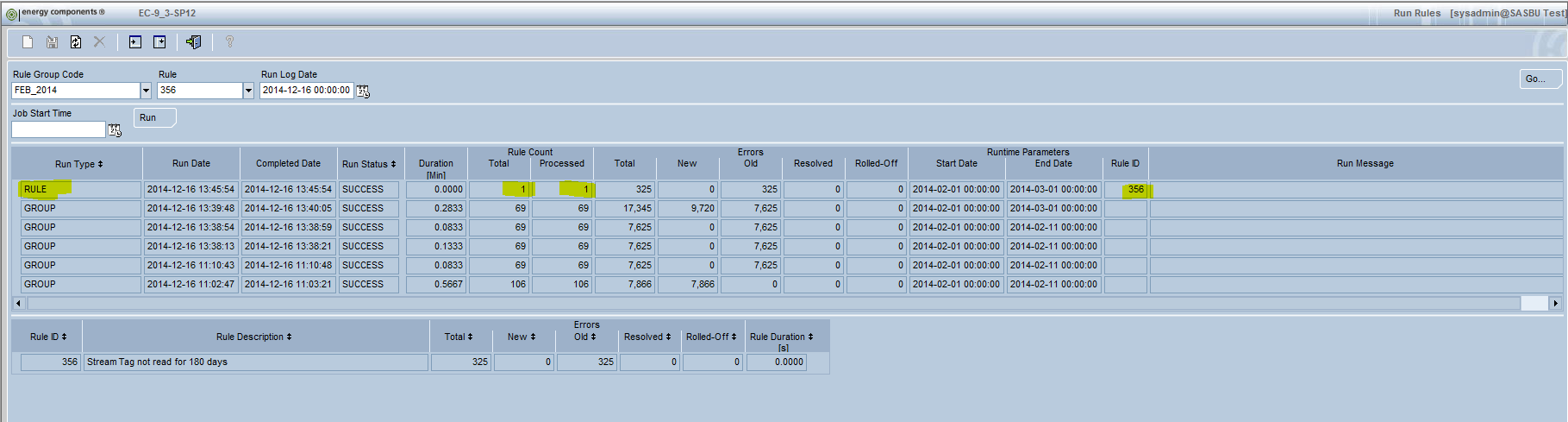


### Running by Rule Group and Rule

It is also possible to simply rerun one Rule within a Rule Group. This is accomplished by populating the Rule field in the Navigator.



When an individual Rule is run within a Rule Group, you will see that the log shows the Run Type of RULE (not GROUP) and the Total Rule Count is shown as 1 (not the total Rules within the Group).



**NOTE**: If the Rule is entered in the Navigator then the screen will only return those Rule Runs which include the selected Rule.

### EC Security

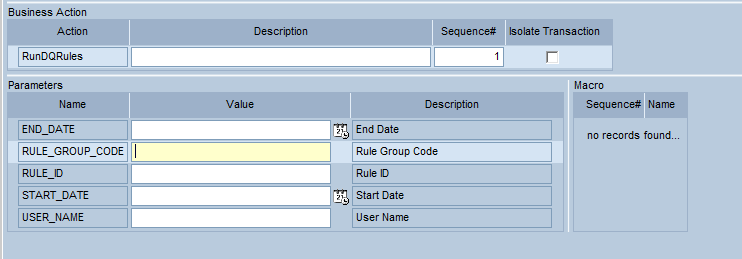
The Run Rule screen is defined as object: /com.ec.cvx.common.screens/ct\_dq\_rules\_group. Access to this object will be required for a user to be able to browse the Run Logs. Note that this is a read-only screen, so only Read access is required.

In order to actually Run a Rule Group, a user will need access to the Run button. This button is defined as object: /com.ec.cvx.common.screens/ct\_dq\_rules\_group/RunRules. An access level of 60 (Edit on APPROVED data) is required to be able to Run Rule Groups.

## Run Rules Scheduling

Another available method for executing Rules is to use the EC Scheduler. For example, it would be typical to have a Rule Group defined to run a group of Rules to evaluate yesterday’s data. This Rule Group could be manually executed through the Run Rule Group window, or it could be set up in the EC Scheduler to automatically run once every day.

The set-up of the Rule job uses the standard method for creating EC jobs (so we will not repeat that information). The uniqueness is in the selected Business Action. There is a new custom business action called RunDQRules which is used to initiate the Rule execution.



As shown above, the Parameters associated with this new business action allow you to specify the inputs to the running of the Rule Group:

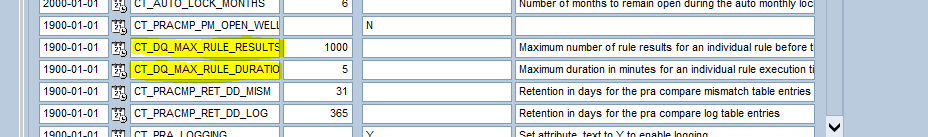
* Rule Group Code (required)
* Rule ID (optional), will run the entire Rule Group if left blank
* User Name (optional), will be associated with the created by field on the Rule Results
* Start Date (optional), will override the Start Date of the Rule Group
* End Date (optional), will override the End Date of the Rule Group

**NOTE**: The logs for any Rule Group or Rule Group & Rule ID run through the EC Scheduler will also appear in the Run Rules business function.

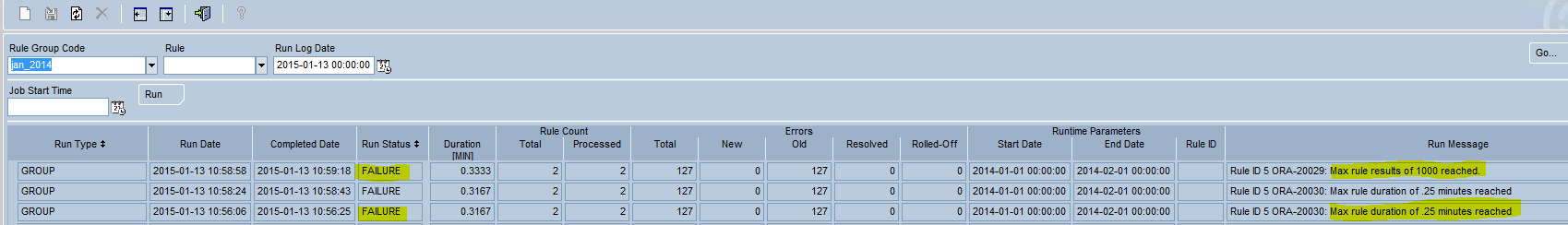
## Execution Restrictions

Since the Data Quality Rules are simply SQL statements, it is possible that, in spite of the QA processes in place, an incorrect SQL statement might be added to the production environment. To prevent extreme negative impact to the production database, there are several system attributes which exist to define upper threshholds on Rule execution.

These system attributes define the maximum number of identified rule results for an individual rule, and the maximum run time in minutes for an individual rule.



In the event that either of these two threshholds are exceeded, the Rule Execution job will be terminated with a FAILURE status.



**NOTE:** These threshholds are meant to identify the extreme situations where there is an error in the SQL definition. Depending on the specific Rule definition at each location, it might be required to change these system attribute values from their defaults of 1000 rule results and 5 minutes.

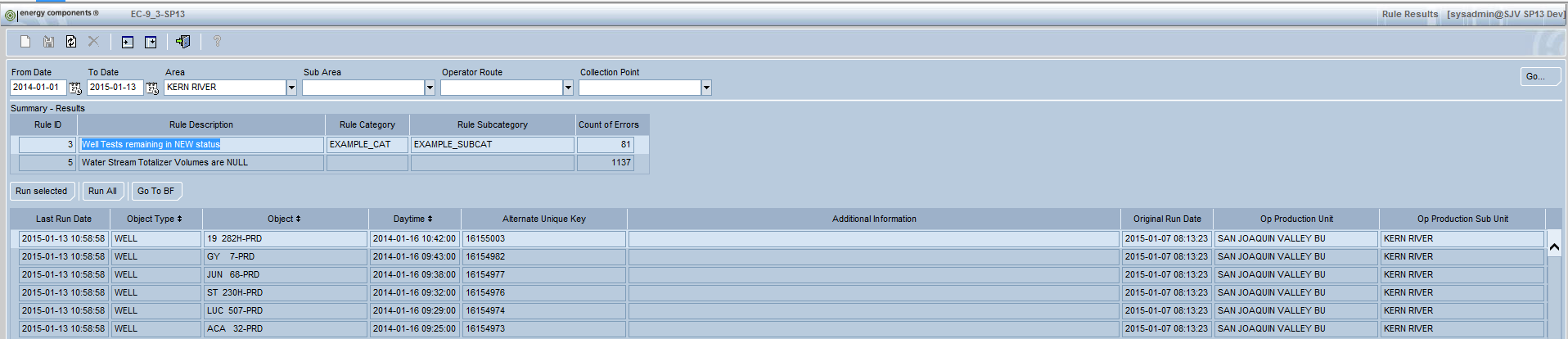
# Reviewing Rule Results

Once the Rules have been executed, the results are stored on a new custom table/class (CT\_DQ\_RULE\_RESULTS). These results are available through a custom Rule Results screen as well as any BU specific reporting solutions.

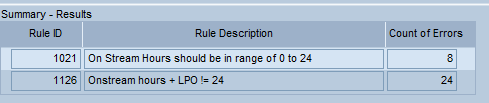
## Rule Results

The Rule Results window is meant to be used by the owner of the data to review any errors identified by the Rules.

This window utilizes the standard EC hierarchy to restrict the returned records associated with that hierarchy level. For example, in the screen print below the results have been limited to the Sub Area level.



In the Summary section of the screen, the total errors by Rule are listed.



Once one of the Rules is selected in the Summary – Results section, the Detail section of the window will show each of the specific records identified by that Rule. For example, once Rule ID 1126 (On Stream hours + LPO != 24) is selected in the Summary section, then the Detail lists all of the Wells, restricted to the date and hierarchy in the navigator, which have records that failed the Rule.



For each of the Rule Results, the object, daytime and Alternate Unique Key are identified in order to specify a unique EC record in error. If the Additional information has been configured, then this information also is stored on each Rule Result. In the example above, the Additional Information was used to show the value of the On stream hours and the LPO hours (the Rule description simply says that the total is not 24), as well as the well type.

Scrolling to the right, the Rule Results also show the Original Run Date (the date the error was first identified) and the Last Run Date (the date the error was most recently identified). Finally, each Rule Result is stored with the Operational, Collection Point, and Geographical hierarchies populated in order to support navigating to the records and/or reporting. If any of these hierarchies are not utilized then the fields can be hidden from the Rule Result screen.

**NOTE**: If Role ID’s are assigned to a Rule then only those users linked to that Role will see the Rule Results.

### Table Definitions

#### CT\_DQ\_RULE\_RESULTS

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Usage |
| RESULT\_ID | Number | The system generated unique number associated with this rule result. |
| ORIG\_RUN\_DATE | Date | The datetime when this rule result was first identified. |
| LAST\_RUN\_DATE | Date | The last datetime when this rule result was identified. |
| LAST\_RUN\_ID | Number | The run id for the last run which identified this rule result. |
| RULE\_ID | Number | The unique identifier for the rule. |
| OBJECT\_TYPE | String | The unique identifier for the object (well, stream, etc.) associated with this rule result. |
| OBJECT\_ID | String | The unique identifier for the object (well, stream, etc.) associated with this rule result. |
| DAYTIME | Date | The daytime (optional) for the record associated with this rule result. |
| ALT\_UNIQUE\_KEY | String | If configured on the rule, then this will be the additional information needed to uniquely identify a single record in error. |
| ADDNL\_INFO | String | The value of the Additional Information field configured on the Rule. |
| SQL\_FLD\_1 – SQL\_FLD\_5 | String | If the rule has utilized the flex SQL field sources then these fields will be used to store that information. Class configuration can be used to rename these fields as needed. |
| OP\_PU\_ID – OP\_WELL\_HOOKUP\_ID | String | The field associated in the operational hierarchy for the associated object. These are system populated in the rule run. |
| CP\_PU\_ID – CP\_COL\_POINT\_ID | String | The field associated in the collection point hierarchy for the associated object. These are system populated in the rule run. |
| GEO\_AREA\_ID – GEO\_SUB\_FIELD\_ID | String | The fields associated in the geographical hierarchy for the associated object. These are system populated in the rule run. |
| GROUP\_REF\_ID\_1 – GROUP\_REF\_ID\_10 | String | The generic group reference fields for the associated object, if utilized in the operational, collection point or geographical hierarchies. These are system populated in the rule run. Class configuration can be used to rename these fields as needed. |

### Rule Result Hierarchies

Currently the Rule Results are stored with explicitly stored hierarchies for operational, collection\_point, and geographic. The decision was made to store the hierarchy versus dynamically determining the hierarchy for the following reasons:

1. The Rule Result screen shows the results for all different object types, so storing the hierarchies eased the ability to browse Rule Results
2. The Rule Result screen allows the results to be viewed at various hierarchy levels (e.g. area, sub area, etc.), so storing the hierarchy results eased the ability to browse Rule Results
3. There are several EC objects which are not directly linked to hierarchies (Perforations, Well Bores, etc.). By storing the hierarchies this allows the Rule execution logic to use various methods to determine the hierarchy to be stored – while the Rule Result screen can avoid having the logic to infer hierarchies
4. To support 3rd party reports, Spotfire, etc. reviewing of the Rule Results, it was deemed easier to store the hierarchy versus requiring these methods to determine the correct hierarchies to use

The implication of this decision are as follows:

1. The Rule Results show the hierarchies as of the time of the Rule execution. If you change an object’s hierarchy after the Rule Results have been generated then the new hierarchy will NOT be reflected on the Rule Results
2. While the Rule Results class can be configured (CT\_DQ\_RULE\_RESULTS), you CANNOT disable the operational or collection\_point hierarchy fields since they are utilized in the Rule Result query. If you do not use these fields and want to hide them from the Rule Result screen you must set the class presentation to viewhidden=true

### Rule Result Retention

When a Rule is re-run, any previous Rule Result which no longer is identified by the Rule (within the Start Date and End Date periods) will assume to be resolved and therefore it will be deleted from the Rule Result table. This way, the Rule Result table always shows the records which still require some type of action to resolve.

For example, a Rule initially identified that a Well Test had a negative choke size and therefore logged the record on the Rule Result table. If that Well Test is corrected to a non-negative choke size, or if the Well Test is deleted from EC then when the Rule is re-run the SQL to identify Well Tests with negative choke size would no longer identify this Well Test – and therefore the Rule Result would be deleted.

As previously discussed in the section on Rule configuration, each Rule also has an optional Result Retention Days element which specifies when a Rule Result should be deleted – relative to the last run date of the Rule Result.

For example, if the Rule Result Retention Days is set to 5 then it means that when the Rule is run then any Rule Result for that Rule will be deleted if the last run date (i.e. the last time that the Rule identified the record as an issue) is earlier that 5 days ago.

**NOTE**: The Rule Result Retention Days ignores the Start Date and End Date parameters when evaluating the records to delete. So for example, if a Rule with retention days of 5 is run to evaluate yesterday’s data then it will also delete any Rule Result older than 5 days regardless of the daytime of the record (i.e. not just data from yesterday).

### Rule Result History

As previously mentioned, the Rule Result table only contains the records which are assumed to still be in error; any record which has been resolved (and the Rule has been rerun) would have been deleted from the table.

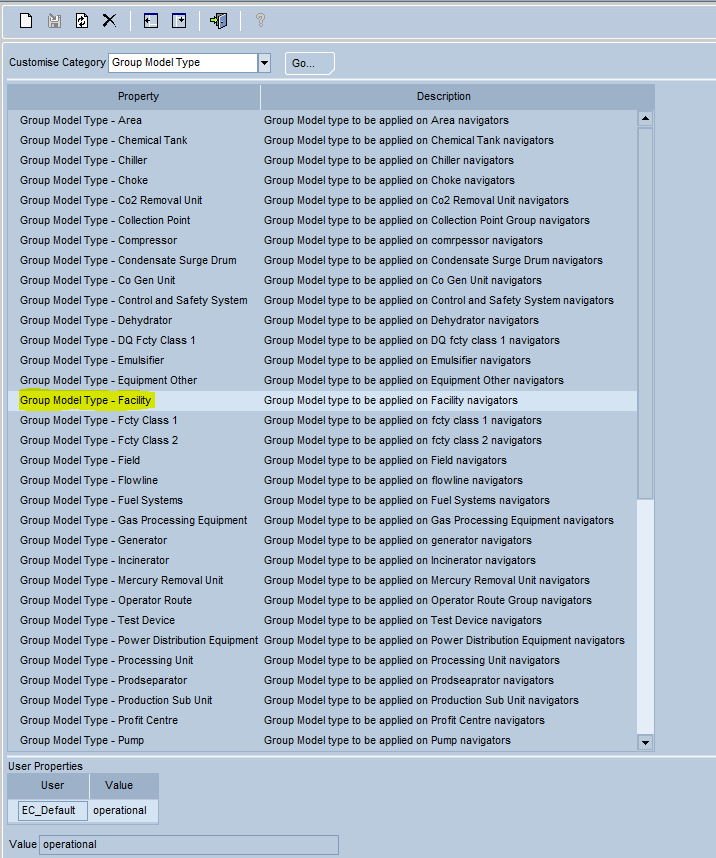
While this approach has the benefit of having the Rule Results show a current list of records in error, there are sometimes where knowing the history of errors could be valuable.

For example, if we were running a Rule to identify late records then it could be useful to see over time which area accounted for the most errors in order to be able to do an accurate root-cause analysis on ways to reduce the number of occurrences in the future.

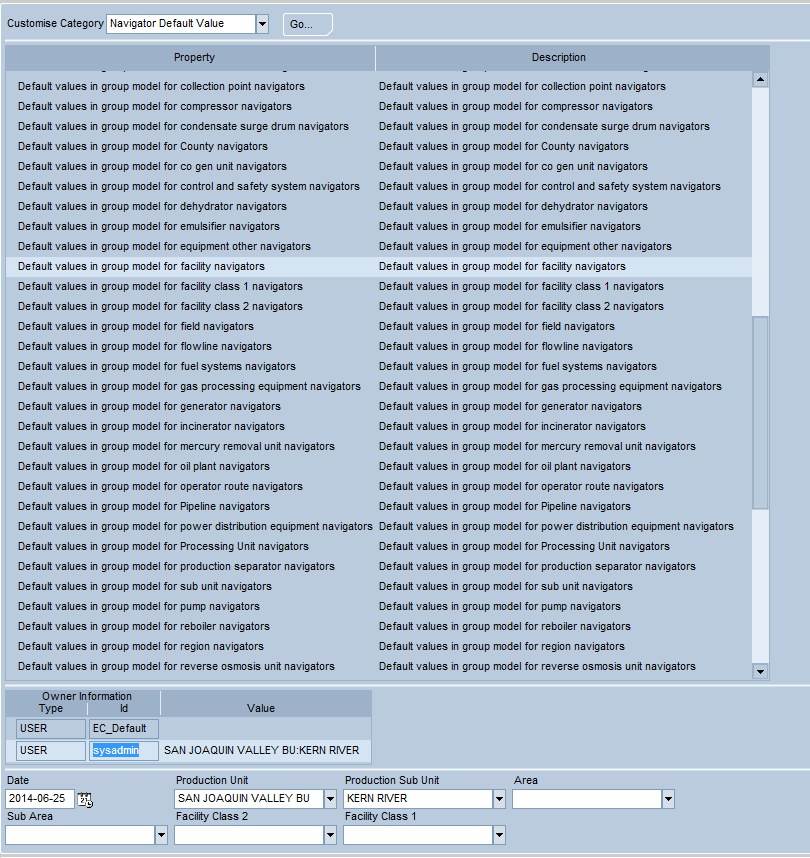
To support this analysis, any Rule Result which is deleted (via a Rule re-run) is logged on the Rule Result Journal table. These records can then be analyzed to see the errors over time in addition to the current errors.

### Group Model Type & Navigator Default Values

The Rule Result screen is associated with the Facility Property. This means that, under Maintain User Setting or Maintain Personal Settings, the system can be configured to set the appropriate group model type (operational, collection point).



In addition, the navigator default values can also be set using the facility navigator’s property.



### Object Partitioning

Any Object Partitioning configuration will apply to the Rule Results screen, thereby restricting the possible navigator values to those assigned to the user/role; however since the Rule Result navigator allows records to be viewed at the higher hierarchy levels (e.g. area, sub-area) this means that a user can currently choose one of these higher levels and therefore they will be able to see some of the results for levels outside of their object partition.

For example, if the object partitioning was set up at the Sub-Area level then the Rule Result screen would correctly limit the Sub-Area drop down to only those appropriate for the user; however if the Rule Result Navigator was entered only up to the Area then the screen would show results for all Sub Areas below this Area.

### Reprocessing Rule Results

From the Rule Results screen, it is possible to have the Rules reevaluate specific Rule Results without having to reprocess the entire Rule Group or Rule.

Typically, an operator would evaluate the Rule Results applicable to his/her responsibility area. Then once the issues have been resolved the Rule Result window allows for the operator to re-assess the Rule Results to confirm that the issues have completely been resolved.

There are two methods of re-running individual results:

* The first method is to highlight one result and then to click the ‘Run Selected’ button. This will cause the highlighted record to be re-assessed
* The second method is to click the ‘Run All’ button. This will rerun all of the Results displayed on the screen (i.e. the results matching the navigator dates and hierarchy, as well as being limited to the selected Rule ID).

In either case, the response to the rerunning will show one of two results:

1. The Result will no longer be relevant (i.e. it has been corrected) and will drop off the results list
2. The Result will be determine to still exist (or another new Result may appear) and the Result will remain on the results list. In this case, if you review the Last Run Date column you can confirm that the Rule Result was recently updated

**NOTE**: Since the reprocessing of individual Rule Results only impact specific records, the re-running of these results will not appear on the Run Rules log and will not be reflected in the Error Counts for the Rule Group/Rule. The only way to update the totals for a Rule Group/Rule is to run the Rule Group/Rule from the Run Rules screen.

### Go to EC Business Function

If the Rule has been configured with an EC Screen/Business Function then highlighting a Rule Result and clicking the Go to BF button will open up a new window with the appropriate EC screen. This allows the user to go to the screen associated with the Rule Result in order to make the necessary corrections.

#### Passing Data to the Business Function

The 9.3 version of EC does not support passing data to the launched business function. Standard default values will be utilized; however the user will likely need to modify the Navigator values to direct the launched screen to the particular data in question.

In 10.4 the EC framework has been modified to allow for the passing of specific data in the launched screens. It is the current plan to take advantage of this ability when the EC Data Quality Tool is migrated to 10.4.

#### Launching when there is no Business Function

In the case when there is no configured Business Function on a Rule, clicking the Go To Business Function will result in an empty EC screen being launched.

### Restriction to Returned Detail Records

There is a practical limit to the number of records which EC can display in and EC screen. Since the Rule Results screen allows various levels of hierarchy in the navigator and various date ranges, it is possible to return large number of Rule Results which can cause EC memory issues – specifically when retrieving the lower screen/detail section of the Rule Results.

To mitigate this issue, the Rule Results screen has the ability to limit the number of retrieved detail records. There is a new custom System Attribute (CT\_DQ\_MAX\_ROWS) which is used to define the number of detail records to return on the screen. By default, this parameter is set to 1,500 records. To see all of the Rule Results beyond the retrieved 1500 records, the best approach is to limit the date range and/or the navigator hierarchy.

**NOTE**: Even if the lower detailed section is restricted to 1500 records, the totals in the header section (total results by Rule) will still accurately display the total record count.

**NOTE**: Even if the lower detailed section is restricted to 1500 records, clicking the Run All button will rerun all of the Rule Results – and will *not* be limited to 1500.

### Miscellaneous Scenarios

#### What if I delete/inactivate a Rule after it has been run?

Since the Rules are run as part of one or more Rule Groups, the Rule execution process will not recognize that a Rule has been eliminated from a Rule Group – and therefore any results will remain on the Rule Result table.

If you want to move a Rule from one group to another then there is no action required, since the Rule Results are stored independently from the Rule Groups. The historical logs will still show the Rule under the old Rule Result, while new runs of the original Rule Group will omit the Rule (and new runs on the newly added Rule Group will start to show the Rule).

If there are no Rule Results, then removing a Rule from a Rule Group creates no issues, since the historical logs would continue to show the now-removed Rule on historical logs, and the future runs of the Rule Groups would simply begin to omit the Rule.

If there are existing Rule Results when you want to eliminate the Rule from all Rule Groups then you have a few choices:

* You can leave the results on the Rule Results table. Without the ability to re-run the Rule the results will remain on the Rule Result table forever
* You can modify the Rule Result table directly (via SQL, not through EC) to remove the Rule Results
* You can first modify the Rule SQL to force no records to be returned (e.g. set the WHERE clause to ‘AND 1=2’) and set the Result Retention Days to 0. Then, when you rerun the Rule it will not add any new Rule Results, and delete all of the existing Rule Results. Now, you can remove the Rule from all Rule Groups.

#### What if I change the SQL after a Rule has been run?

Since the Rule Runs automatically delete any Rule Result that appears to be resolved (i.e. that is no longer identified as part of the Rule SQL), then any changes to the SQL (along with a re-run of the Rule Group) will simply result in the data being re-evaluated and any Rule Results no longer meeting the SQL criteria will be deleted.

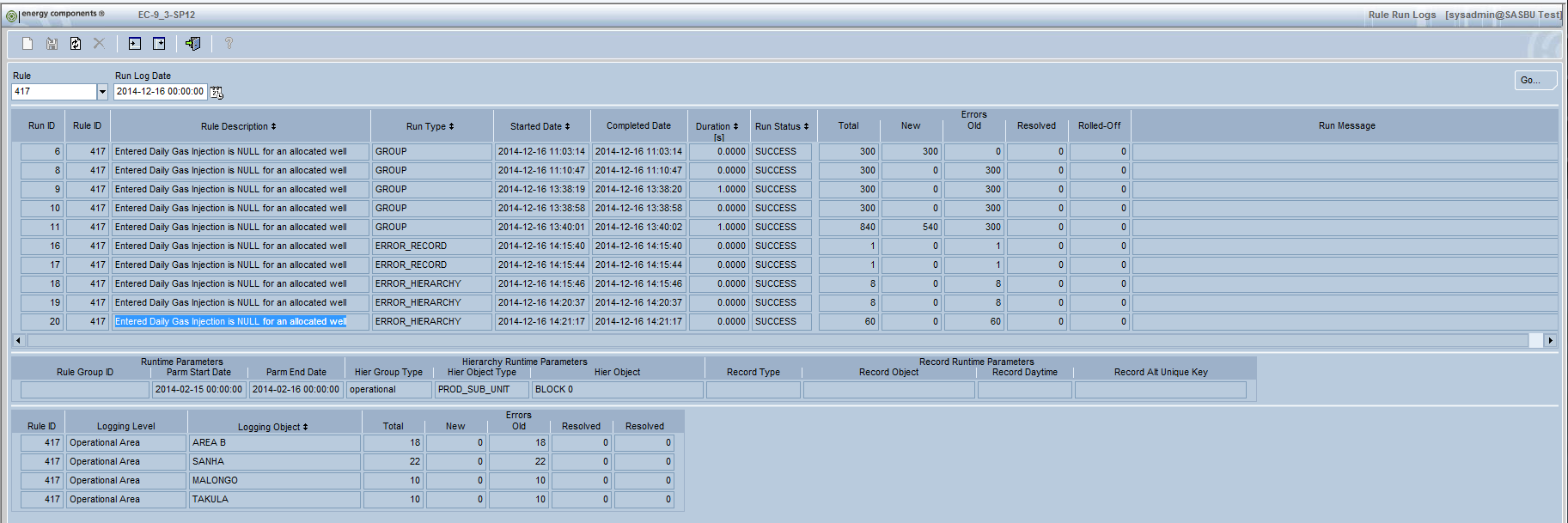
For example, if we initially set up a Rule to compare the monthly well allocation results versus the sum of the daily well allocation results, and that rule identified that one of the Wells differed by .01 mcf. This Rule Result might prompt the Rule SQL to be modified to accept a larger margin of error – for example to rewrite the Rule to identify only those situations where the difference was >= 1. Once the Rule was re-run with this new SQL, the execution process will simply assume that the Rule Result of .01 was now resolved and therefore the Rule Result would be deleted.

### EC Security

The Rule Result screen is defined as object: /com.ec.cvx.common.screens/ct\_dq\_rule\_result. Access to this object will be required for a user to be able to browse the Rule Results.

Access level 60 (Edit on APPROVED data) is required on this screen in order to have access to click the Run Selected and Run All buttons.

## Rule Run Logs



The Rule Run Logs business function allows you to view all runs for a selected Rule ID – regardless of how that Rule was executed.

### Table Definitions

#### CT\_DQ\_RULE\_RUN\_LOG

This table has one record for each rule within a job execution. This is the same table used to populate the lower section of the Run Rules screen.

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Usage |
| RUN\_ID | Number | The system generated unique number associated with this rule run. |
| RULE\_ID | Number | The unique identifier for the rule. |
| TOTAL\_ERRORS | Number | The total number of rule results identified for this rule during this run. |
| TOTAL\_NEW\_ERRORS | Number | The number of rule results for this rule which were newly identified in this run. |
| TOTAL\_OLD\_ERRORS | Number | The number of rule results for this rule which were previously identified prior to this run. |
| TOTAL\_DELETED\_ERRORS | Number | The number of rule results which were deleted/resolved for this rule in this run. |
| TOTAL\_ROLLED\_ERRORS | Number | The number of rule results which were rolled off for this rule due to the rule retention days parameter in this run. |
| STARTED\_DATE | Date | The datetime that the rule execution began. The run time is the difference between this time and the completed\_date |
| COMPLETED\_DATE | Date | The datetime that the rule execution completed. The run time is the difference between this time and the started\_date |

#### CT\_DQ\_RULE\_RUN\_DETAIL\_LOG

This table is populated if the logging level for the Rule is anything other than Rule. For Rule logging, there will be no records added to this table.

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Usage |
| RUN\_ID | Number | The system generated unique number associated with this rule run. |
| RULE\_ID | Number | The unique identifier for the rule. |
| LOGGING\_LEVEL | String | The logging level of the Rule. |
| TOTAL\_ERRORS | Number | The total number of rule results identified for this rule during this run. |
| TOTAL\_NEW\_ERRORS | Number | The number of rule results for this rule which were newly identified in this run. |
| TOTAL\_OLD\_ERRORS | Number | The number of rule results for this rule which were previously identified prior to this run. |
| TOTAL\_DELETED\_ERRORS | Number | The number of rule results which were deleted/resolved for this rule in this run. |
| TOTAL\_ROLLED\_ERRORS | Number | The number of rule results which were rolled off for this rule due to the rule retention days parameter in this run. |

### Class Definitions

#### TV\_CT\_DQ\_RULE\_RUN\_LOG

Class to show the run details for each execution of a Rule; built on table CT\_DQ\_RULE\_RUN\_LOG.

#### TV\_CT\_DQ\_RULE\_LOG\_PARMS

Class to show the input parameters of a Rule Run; built on table CT\_DQ\_RUN\_LOG.

#### TV\_CT\_DQ\_RULE\_RUN\_DET\_LOG

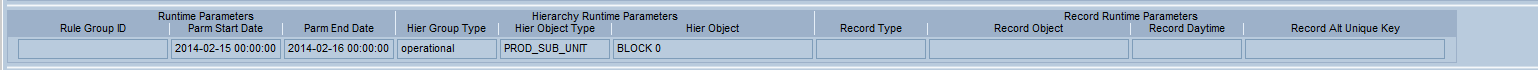
Class to show the detail logging for a Rule; built on table CT\_DQ\_RULE\_RUN\_DETAIL\_LOG.

### Rule Run Types

The Rule Run Logs screen will show the Rule Results for a selected Rule regardless of how that Rule was executed. The following shows the various Run Types:

* GROUP: A Rule is executed as part of a Rule Group (via the Run Rules screen or via the EC Scheduler)
* RULE: A Rule is executed as part of a Rule Group with a supplied Rule ID (via the Run Rules screen or via the EC Scheduler)
* ERROR\_RECORD: An individual Rule Result is executed from the Rule Results screen (via the Run Selected button)
* ERROR\_HIERARCHY: A selection of Rule Results is executed from the Rule Results screen (via the Run All button)

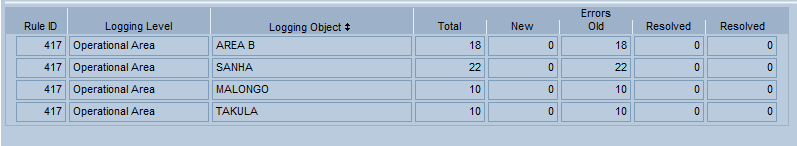
### Runtime Parameters



The Runtime Parameters section of the window shows the input parameters which were used to run the appropriate Rule. These parameters vary for each Run Type .

* GROUP:
  + Start Date
  + End Date
  + Rule Group ID
* RULE:
  + Start Date
  + End Date
  + Rule Group ID
  + Rule ID
* ERROR\_RECORD:
  + Start Date
  + End Date
  + Record Daytime
  + Record Object
  + Record Alternate Unique Key
* ERROR\_HIERARCHY:
  + Start Date
  + End Date
  + Hierarchy Object

### Rule Detailed Logging

Based on the Rule configured logging level, the detail section of the Run Rules screen will optionally show the breakdown of the Rule counts (e.g. New, Old, etc.) at a more detailed level. This functionality allows the historical counts to be tracked at a level which may have more meaning than an overall total.

### EC Security

The Rule Run Logs screen is defined as object: /com.ec.cvx.common.screens/ ct\_dq\_run\_rules\_log. Access to this object will be required for a user to be able to browse the Rule Run Logs. Note that this is a read-only screen, so only Read access is required.

# Unique Situation Examples

The following describes how to use the Data Quality Tool to handle some unique situations.

## Objects without Object ID’s (e.g. User ID’s)

The DQ Tool is based on the assumption that Rules are written against objects with object\_id’s:

* Rules require an object id source
* Object types use object id’s to determine hierarchy values
* Rule Results are displayed showing the object type and object name

While this is true, there is a way to write rules against objects without object ids. For example, you can use the DQ Tool to identify any user who hasn’t logged in for a set number of days. To define this type of rule you would need to do the following:

1. Define a new hierarchy object type by adding a record to CT\_DQ\_HIER\_DETERMINATION. This entry would define how to determine the hierarchy for this new user id rule. Since user id’s do not have hierarchies, you could either choose to hard-code the hierarchy (e.g. to the highest level of the hierarchy, or to a predetermined hierarchy), or leave the hierarchy NULL. Note that if you leave the hierarchy NULL then you would not be able to use the Rule Result screen to browse these Rule Results (you would only be able to see these Rule Results through direct table access, custom reports, etc.).

For this example, we will assume that we want to hard code the production area and leave all other hierarchy values NULL. Therefore when populating the table entry we need to supply the following information:

* Object\_type: This can be set to any value to describe this hierarchy determination method. Since this will be user id related, for this example we will use ‘USER’ as the object\_type.
* From\_SQL: Our goal is to have one record per user\_id to define the hierarchy, so the from\_sql would be set to ‘FROM TV\_T\_BASIS\_USER’
* Object\_id\_source: This is the key to the record which will contain the hierarchy, so in our example we would need to populate this field with the ‘user\_id’
* Daytime\_source: Since we only need one record per user id (i.e. no need for dated versions), we can hard-code this date to ‘to\_date(‘01/01/1900’,’mm/dd/yyyy’)
* End\_date\_source: Since we only need one record per user id (i.e. no need for dated versions), we can hard-code this date to ‘to\_date(‘12/31/9999’,’mm/dd/yyyy’)
* OP\_PU\_ID\_SOURCE: set to either a hard-coded object\_id for the production unit, or set to ‘(select object\_id from ov\_productionunit where code = ‘XXX’)’
* All other hierarchy source fields left NULL

1. Add new EC Code on CT\_DQ\_HIER\_OBJ\_TYPE for the new ‘USER’ hierarchy type, and make this entry active
2. Create a new Rule to check for last log-in. Here are the key field values to enter:

* Object\_id\_source: Like with the hierarchy, we need to set this to USER\_ID (since it will be used to join with the object\_type table to get the hierarchy)
* Daytime\_source: Set to NULL since there is no real daytime associated with this Rule Result
* Object\_type: Set equal to the new ‘USER’ object\_type
* FROM\_SQL (assumes you are checking for no activity for 90 days):

FROM

(

SELECT

DAYTIME AS LAST\_LOGIN\_DATE,

DENSE\_RANK () OVER (PARTITION BY REMOTE\_USER ORDER BY DAYTIME DESC) AS SERIAL\_NO,

REMOTE\_USER as user\_id,

AUTHENTICATED\_IND,

ROUND(SYSDATE - DAYTIME) AS DAYS\_SINCE\_LAST\_LOGIN

FROM TV\_AUDIT\_LOGIN a, t\_basis\_user u

where A.REMOTE\_USER = u.user\_id

and u.active = 'Y'

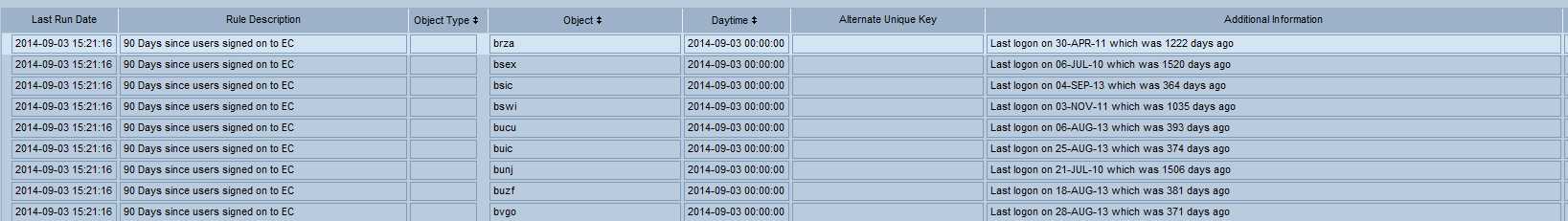
)

WHERE SERIAL\_NO < 2

AND AUTHENTICATED\_IND = 'Y'

AND DAYS\_SINCE\_LAST\_LOGIN > 90

1. When you run the Rule and get Rule Results, you will see the object id as blank (since it cannot decode to a name) and the alternate unique key populated with the user id.



# Installation and Maintenance Guide

Be sure to follow these instructions when installing the Data Quality Tool solution into EC.

## Performing a Fresh Installation

1. **Install the Data Quality Rule Manager**
   1. Obtain the latest version of the Data Quality Tool
   2. Navigate to the downloaded folder.
   3. Double-click “install.cmd.” Enter the site parameters when prompted.
   4. Verify that no issues occurred by reviewing the generated log file.
2. **If utilizing the Colleciton Point Hierarchy then Install the Collection Point update**
   1. Obtain the latest version of the Data Quality Tool
   2. Navigate to the downloaded folder.
   3. Double-click “collection\_point\_install.cmd.” Enter the site parameters when prompted.
   4. Verify that no issues occurred by reviewing the generated log file.
3. **Update CVX ear file**
   1. The custom EC screens for the Data Quality Tool are part of the application server file EC-PROJECT-CVX-???.ear. This ear file needs to replace any existing CVX ear file.
   2. This file needs to be implemented on the application server using the standard process to update an ear file (stop jBoss, rename existing CVX ear file to .bak, install new ear file, restart jBoss)
4. **Determine How to Organize Rules**
   1. The base install is delivered with a proposed Rule Category (CT\_DQ\_RULE\_CATEGORY) and Rule SubCategory (CT\_DQ\_RULE\_SUBCATEGORY) on the EC Codes table.
   2. Add the appropriate entries to the codes and code dependency tables to build the desired hierarchy.
   3. Inactivate any Categories/Subcategories that you do not want to utilize. Remember that there are a few example Rules delivered under the Example Rule Category and Rule Subcategory, so you may want to leave these active/un-deleted until after you have reviewed (and deleted) the example Rules
5. **Define/Refine Object Types**
   1. The base install delivers many of the standard object types (CT\_DQ\_HIER\_OBJ\_TYPE) and hierarchy determination configuration (CT\_DQ\_HIER\_DETERMINATION). If it is known that some object types will never be used in data quality rules then these entries can be set to inactive in order to eliminate them as options on the Rules screen dropdown.
6. **Define Error Types**
   1. The base install delivers some basic Error types (CT\_DQ\_ERROR\_TYPE). This attribute is for information only, so the values can be deleted/disabled as needed.
7. **Define available Rule Logging Levels**
   1. If not populated, the default logging level will be at a Rule ID level. If a more detailed logging is desired then the logging level EC Codes table (CT\_DQ\_LOGGING\_LEVEL) should be used to define the available options.
   2. Inactivate any logging levels which you do not want to be used.
   3. Confirm the default selection (default is Rule). By marking one of the logging levels as a default value, this will result in the selected entry being defaulted when Rules are maintained.
8. **Utilize the flex fields on Rule and/or Rule Group**
   1. If you already know that you want to use one or more of the flex fields on Rule and/or Rule Group (e.g. TEXT\_1, DATE\_3, etc.) then the classes can be modified to be enabled for the BU. Remember that these fields are simply ‘report-only’ data elements which are not stored on the Rule Results.
9. **Utilize the SQL flex fields on Rule**
   1. If you already know that you want to use one or more of the SQL flex fields on the Rule (e.g. SQL\_FLD\_1, SQL\_FLD\_2, etc.) then the Rule and Rule Result classes can be modified to enable these fields for the BU. Remember that these fields are those where additional data is desired to be stored on the Rule Results.
10. **Define the Security Approach for the Data Quality screen**
    1. Typically there will be two roles involved in the Data Quality Tool usage. These roles need to be defined (i.e. they are **not** built in the installation scripts) and then assigned to the correct users – or existing roles will need to be modified.
       1. A Role for the Rule Maintenance screen and Run Rules screen (typically a super-user support person who is familiar with writing SQL)
          1. This Role typically has access to all of the Data Quality screens. Here is a proposed set-up to grant all access to all DQ functions. Notice that the DQ Rule Run Logs is only Read since this is a read-only screen.



* + 1. A Role for the Rule Result screen (typically the end-user operator, etc. who will also be responsible for correcting the data issues)
       1. This Role should be granted access to only the Rule Results screen. By setting access to Edit on APPROVED data, this will allow the Rule Results to be rerun (Rerun Selected and Rerun All buttons).



* 1. Decide on whether Roles will be used to determine who can see certain Rule Results. These can be existing Roles (e.g. the Roles used to grant access to the EC screens) or new Data Quality specific Roles. In the case of Data Quality specific Roles, then these Roles will need to be defined.

## Uninstallation

The Data Quality Tool installation package contains a backout script to uninstall its components.

1. **Remove any manual steps**
   1. Any manual steps taken in the installation (e.g. Roles, EC schedules, etc.) would need to be manually removed for a complete uninstallation.
2. **Run the backout.cmd file**
   1. The script will remove all of the Data Quality tables, views, classes, etc. If the manual steps from Step #1 are not completed then some of the backout steps may not work correctly (e.g. if the EC schedules are not removed then the business action will not be removed in the backout script).
3. **Restore the original CVX ear file**
   1. Restore the CVX ear file back to the original file which was renamed during the installation
   2. This file needs to be implemented on the application server using the standard process to restore an ear file (stop jBoss, delete CVX ear file, rename .bak CVX ear file to .ear, install new ear file, restart jBoss)

# Appendix

## Component Overview

When installed, the Chevron Data Quality Tool solution creates the following database objects:

* Tables
  + CT\_DQ\_RULE
  + CT\_DQ\_RULE\_GROUP
  + CT\_DQ\_RULE\_GRP\_COMBINATION
  + CT\_DQ\_RULE\_RESULT\_LOG
  + CT\_DQ\_RULE\_RESULTS
  + CT\_DQ\_RUN\_LOG
  + CT\_DQ\_HIER\_DETERMINATION
* Journal Tables
  + CT\_DQ\_RULE\_JN
  + CT\_DQ\_RULE\_GROUP\_JN
  + CT\_DQ\_RULE\_GRP\_COMBINATION\_JN
  + CT\_DQ\_RULE\_RESULTS\_JN
* Triggers
  + IU\_CT\_DQ\_RULE
  + IU\_CT\_DQ\_RULE\_GROUP
  + IU\_CT\_RULE\_GRP\_COMBINATION
  + IU\_CT\_DQ\_RUN\_LOG
  + IU\_CT\_DQ\_RULE\_RESULTS
* Journal Triggers
  + JN\_CT\_DQ\_RULE
  + JN\_CT\_DQ\_RULE\_GROUP
  + JN\_CT\_DQ\_RULE\_GRP\_COMBINATION
  + JN\_CT\_DQ\_RULE\_RESULTS
* Views
  + CV\_DQ\_RULE\_GROUP\_LOG
* Generated Maintenance Classes
  + TV\_CT\_DQ\_RULE
  + TV\_CT\_DQ\_RULE\_GROUP
  + TV\_CT\_DQ\_RULE\_GRP\_COMBO
  + TV\_CT\_DQ\_RULE\_GROUP\_LOG
  + TV\_CT\_DQ\_RULE\_RESULT\_LOG
  + TV\_CT\_DQ\_RULE\_RESULTS
* Generated Report Classes
  + RV\_CT\_DQ\_RULE
  + RV\_CT\_DQ\_RULE\_GROUP
  + RV\_CT\_DQ\_RULE\_GRP\_COMBO
  + RV\_CT\_DQ\_RULE\_GROUP\_LOG
  + RV\_CT\_DQ\_RULE\_RESULT\_LOG
  + RV\_CT\_DQ\_RULE\_RESULTS
* Generated Journal Classes
  + TV\_CT\_DQ\_RULE\_JN
  + TV\_CT\_DQ\_RULE\_GROUP\_JN
  + TV\_CT\_DQ\_RULE\_GRP\_COMBO\_JN
  + TV\_CT\_DQ\_RULE\_RESULTS\_JN
* Packages
  + UE\_CT\_DQ\_RULES\_PKG
* Generated Packages
  + EC\_CT\_DQ\_RULE
  + EC\_CT\_DQ\_RULE\_GROUP
* EC Code Tables
  + CT\_DQ\_ERROR\_TYPE
  + CT\_DQ\_HIER\_OBJ\_TYPE
  + CT\_DQ\_LOGGING\_LEVEL
  + CT\_DQ\_RULE\_CATEGORY
  + CT\_DQ\_RULE\_SUBCATEGORY
* Custom EC Screens (part of EC-PROJECT-CVX-???.ear)
  + /com.ec.cvx.common.screens/ct\_dq\_create\_rule\_group
  + /com.ec.cvx.common.screens/ct\_dq\_rules
  + /com.ec.cvx.common.screens/ct\_dq\_rules\_group (Run Rules)
  + /com.ec.cvx.common.screens/ct\_dq\_rule\_result
  + /com.ec.cvx.common.screens/ct\_dq\_run\_rules\_log
* System Attributes
  + CT\_DQ\_MAX\_RULE\_RESULTS (default 1000 results)
  + CT\_DQ\_MAX\_RULE\_DURATION (default 5 minutes)
  + CT\_DQ\_MAX\_ROWS (default 1500 records)

# Glossary of Terms

|  |  |
| --- | --- |
| Term | Definition |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |