

DQ Scoring Framework – Technical Design

Version 1.0

November 18, 2019

**Table of Contents**

[**1.0** **Overall Solution** 8](#_Toc23404771)

[**2.0** **Technical Design** 9](#_Toc23404772)

[2.1 Module Diagram 9](#_Toc23404773)

[2.2 Data/DB/Case Store structures 9](#_Toc23404774)

[**2.2.1** **IGX\_DS\_DG\_DATA\_ELEMENT\_RULES\_ORG** 9](#_Toc23404779)

[**2.2.2** **IGX\_DS\_DG\_DATA\_ELEMENT\_RULES** 12](#_Toc23404780)

[**2.2.3** **IGX\_DS\_DG\_RULE\_ASSET\_TYPE** 15](#_Toc23404787)

[**2.2.4** **IGX\_DB\_RULE\_REFRESHER\_LOG** 16](#_Toc23404789)

[**2.2.5** **IGX\_DS\_DQ\_RESULTS** 16](#_Toc23404790)

[**2.2.5** **IGX\_DS\_DQ\_DETAIL\_SCORE** 18](#_Toc23404799)

[**2.2.6** **IGX\_DS\_DQ\_ELEMENT\_SCORE** 19](#_Toc23404809)

[**2.2.7** **IGX\_DS\_DQ\_RULE\_SCORE** 20](#_Toc23404813)

[**2.2.8** **IGX\_DS\_DQ\_ROLLUP\_SCORE** 21](#_Toc23404814)

[**2.2.9** **IGX\_DS\_DG\_HIERARCHY** 22](#_Toc23404815)

[**2.2.10** **IGX\_DB\_SCORING\_ENGINE\_LOG** 23](#_Toc23404816)

[**2.2.11** **IGX\_DB\_DG\_SCORING\_CONFIG** 23](#_Toc23404831)

[**2.2.12** **IGX\_DB\_SCORE\_REFRESHER\_LOG** 24](#_Toc23404832)

[**2.2.13** **IGX\_DS\_DQ\_HIERARCHICAL\_SCORE** 25](#_Toc23404833)

[**2.2.14** **IGX\_DS\_LATEST\_DS\_DQ\_DETAIL\_SCORE** 26](#_Toc23404834)

[**2.2.15** **IGX\_DS\_LATEST\_DS\_DQ\_RESULTS** 26](#_Toc23404835)

[**2.2.6** **IGX\_DB\_DG\_DATA\_ENTITY\_CONFIG** 28](#_Toc23404836)

[**2.2.7** **IGX\_DB\_RULE\_EVALUATOR\_LOG** 29](#_Toc23404837)

[2.3 Pipeline Structure 29](#_Toc23404842)

[2.4 Process Model Design 29](#_Toc23404843)

[**2.4.1** **IGX\_PM\_Rule\_Refresher** 29](#_Toc23404844)

[**2.4.2** **IGX\_PM\_Rule\_Evaluator** 31](#_Toc23404845)

[**2.4.3** **IGX\_PM\_Scoring\_Engine** 31](#_Toc23404846)

[**2.4.4** **IGX\_PM\_Score\_Refresher** 31](#_Toc23404847)

[**2.4.5** **IGX\_PM\_Monitor\_SR\_Status** 32](#_Toc23404848)

[**2.4.6** **IGX\_PM\_Refresh\_Dashboard\_Data** 32](#_Toc23404849)

[**2.4.7** **IGX\_PM\_Evaluate\_Data\_Quality** 33](#_Toc23404850)

[2.5 Analysis Design 33](#_Toc23404857)

[**2.5.1** **IGX\_PRC\_Rule\_Refresher** 33](#_Toc23404858)

[**2.5.2** **IGX\_PRC\_Rule\_Refresher\_Log** 35](#_Toc23404859)

[**2.5.3** **IGX\_PRC\_Evaluate\_DQ\_<Data entity>** 35](#_Toc23404860)

[**2.5.4** **IGX\_PRC\_DQ\_Rules\_Executor** 36](#_Toc23404861)

[**2.5.5** **IGX\_PRC\_Build\_Hierarchy** 38](#_Toc23404862)

[**2.5.6** **IGX\_PRC\_Scoring\_Engine** 40](#_Toc23404863)

[**2.5.7** **IGX\_PRC\_Scoring\_Engine\_Log** 43](#_Toc23404864)

[**2.5.8** **IGX\_PRC\_FIND\_NEW\_FOR\_SCORE\_REFRESHER** 43](#_Toc23404865)

[**2.5.9** **IGX\_PRC\_POST\_SCORES\_TO\_GOVERN** 44](#_Toc23404866)

[**2.5.10** **IGX\_PRC\_MONITOR\_RESULT\_POSTING\_STATUS** 45](#_Toc23404867)

[**2.5.11** **IGX\_PRC\_SCORE\_REFRESHER\_END\_LOG** 46](#_Toc23404868)

[**2.5.12** **IGX\_PRC\_DS\_DQ\_HIERARCHICAL\_SCORE** 46](#_Toc23404869)

[**2.5.13** **IGX\_PRC\_DS\_Latest\_DS\_DQ\_Detail\_Score** 47](#_Toc23404870)

[**2.5.14** **IGX\_PRC\_DS\_Latest\_DS\_DQ\_Results** 48](#_Toc23404871)

[**2.5.15** **IGX\_PRC\_Find\_Ready\_Data\_Entities** 48](#_Toc23404872)

[**2.5.16** **IGX\_PRC\_Log\_Rule\_Evaluator** 49](#_Toc23404873)

[**2.5.17** **IGX\_PRC\_Scoring\_Engine\_Execution\_Check** 49](#_Toc23404874)

[2.6 Data Views 50](#_Toc23404882)

[2.7 Dashboards 51](#_Toc23404883)

[**2.7.1** **Overall DQ Health** 51](#_Toc23404884)

[**2.7.2** **Score Trend by Asset Type** 52](#_Toc23404885)

[**2.7.3** **Failed and Error Records by Data Entity, Element & Rule** 53](#_Toc23404886)

[**2.7.4** **Pass Rate by Hierarchy (Last 30 Days)** 55](#_Toc23404887)

[2.8 Case Management 56](#_Toc23404888)

[**2.8.1** **Case Summary Workflow** 56](#_Toc23404889)

[**2.8.2** **Case Summary Search Screen** 56](#_Toc23404890)

[**2.8.3** **Case Summary Edit/View Screen** 57](#_Toc23404891)

[**2.8.4** **Case Detail Search Screen** 57](#_Toc23404892)

[**2.8.5** **Case Detail Edit/View Screen** 57](#_Toc23404893)

[**3.0** **Configuration Data / Initial Data** 58](#_Toc23404894)

[3.1 Reference Lists 58](#_Toc23404896)

[**3.1.1** **Rule Status** 58](#_Toc23404897)

[**3.1.2** **Rule Dimension** 58](#_Toc23404898)

[**3.1.3** **IGX Data Element Group** 59](#_Toc23404899)

[**3.1.4** **IGX Scoring Config** 59](#_Toc23404900)

[**3.1.5** **IGX Rule Operator List** 60](#_Toc23404901)

[**3.1.6** **IGX Comparison Rule Operator List** 60](#_Toc23404902)

[**3.1.7** **IGX Pass Skip List** 60](#_Toc23404907)

[**3.1.8** **IGX Data Entity Config** 61](#_Toc23404908)

[3.2 Artifacts 61](#_Toc23404909)

[**3.2.1** **IGX Data entity** 61](#_Toc23404915)

[**3.2.2** **IGX Data Element** 61](#_Toc23404916)

[3.3 Predicates 62](#_Toc23404922)

[3.4 Relationship Type 62](#_Toc23404924)

[3.5 Artifact Fields 62](#_Toc23404925)

[**3.5.1** **IGX Data Element** 63](#_Toc23404929)

[**3.5.2** **IGX Data Entity** 63](#_Toc23404930)

[**3.5.3** **Field to Capture DQ Result** 64](#_Toc23404931)

[3.6 Rule Types 64](#_Toc23404932)

[**3.6.1** **IGX Consistency Check** 64](#_Toc23404942)

[**3.6.2** **IGX Duplicate Check** 65](#_Toc23404943)

[**3.6.3** **IGX Not Blank** 66](#_Toc23404944)

[**3.6.4** **IGX Reference Lookup** 66](#_Toc23404945)

[**3.6.5** **IGX Regex** 67](#_Toc23404946)

[**3.6.6** **IGX Numeric Range** 67](#_Toc23404947)

[**3.6.7** **IGX Attribute Comparison** 68](#_Toc23404948)

[**3.6.8** **IGX Conformity Check** 69](#_Toc23404949)

[**3.6.9** **IGX JS Expression** 70](#_Toc23404950)

[3.7 Sample Data 70](#_Toc23404951)

[**3.7.1** **IGX Data Entity** 70](#_Toc23404962)

[**3.7.2** **Sample Rule Implementation** 70](#_Toc23404963)

[**3.7.3** **IGX Data Element** 71](#_Toc23404964)

[**3.7.4** **Sample Relationship** 71](#_Toc23404965)

[**3.7.5** **PROCESS\_LIST** 71](#_Toc23404966)

[**4.0** **Common Design Elements** 72](#_Toc23404967)

[**5.0** **Security** 73](#_Toc23404968)

[**6.0** **Assumptions** 74](#_Toc23404969)

[7.0 **Appendix** 75](#_Toc23404970)

[7.1 Recommended Configuration file 75](#_Toc23404971)

**Document Version History**

| Author(s) | Version | Amendments/Reasons | Date |
| --- | --- | --- | --- |
| Shalabh Mehrotra | 0.1 | Initial Draft | 14/08/2019 |
| Shalabh Mehrotra | 0.2 | Incorporated Review comments | 22/08/2019 |
| Ajay Gera | 0.3 | Included the design of Rule Evaluator | 23/08/2019 |
| Shalabh Mehrotra | 0.4 | Added/removed few columns in first 2 data stores | 23/08/2019 |
| Ajay Gera | 0.5 | Made some changes in rule evaluator   * Added new attribute “Rule Attribute Data Type” in ‘Attribute Comparison’ rule type definition * Capturing the rule results at record level so that output can be used by ‘Scorer’ * Removed igx\_prc\_restore\_dq\_results analysis as results will be saved by igx\_prc\_re\_evaluate\_dq\_rules * Added ‘JS Expression’ rules | 30/08/2019 |
| Shalabh Mehrota | 0.6 |  | 19/09/2019 |
| Ajay Gera | 0.7 | * Added the design of Rule Refresher   Changes to section 2.5.4 (Rule evaluator)   * Renamed “Pass if Blank Or Null” to “Treat Blank Or Null As” in rule template and accordingly in their handling in rule evaluator * Added the point to remove store name from data element name * Read active and non-deleted rules | 19/09/2019 |
| Ajay Gera | 0.8 | * Added section 2.2.13 - Created igx\_ds\_dq\_hierarchical\_score store to feed dashboards * Added system\_version field in igx\_ds\_dq\_results store so that Scorer can update the status for Group fields * Added error handling in Score Refresher | 04/10/2019 |
| Shalabh Mehrotra | 0.9 | * Added new table IGX\_DS\_DG\_RULE\_ASSET\_TYPE and made related changes in IGX\_PRC\_Rule\_Refresher and IGX\_PRC\_Scoring\_Engine for population of Rule’s Asset Type and fetching of same. | 11/10/2019 |
| Ajay Gera | 0.10 | * Changes to store transformed rule result in igx\_ds\_dq\_results. Changes made in section 2.2.5, 2.5.4 and 2.5.6 | 14/10/2019 |
| Rajiv Bansal | 0.11 | * Section 2.6 contains the definition of Data views for Dashboard * Section 2.7 contains the definition of Dashboard widgets * Section 2.2.5 Added another column, Fail Description, to IGX\_DS\_DQ\_RESULTS. * Section 2.2.13 Added another column, Rule Dimension, to IGX\_DS\_DQ\_HIERARCHICAL\_SCORE. | 14/10/2019 |
| Rajiv Bansal | 0.12 | * Section 2.2.14 and 2.2.15 added for additional datastores. * Section 2.5.12, 2.5.13 and 2.5.14 added for additional analysis * Section 2.6 updated for Dataview definition using above new stores. * Section 2.2.13 Added another column, Rule Dimension, to IGX\_DS\_DQ\_HIERARCHICAL\_SCORE. | 15/10/2019 |
| Ajay Gera | 0.13 | Added Asset\_Class field in igx\_ds\_dg\_hierarchy and igx\_ds\_dq\_rollup\_score stores and made changes to populate them.  Changed following sections  2.2.8, 2.2.9, 2.5.5, 2.5.6 and 3.1.4 | 15/10/2019 |
| Rajiv Bansal | 0.14 | Changes in Section 2.2.9, added next asset class & next asset type and Section 2.5.12 for correction in analysis | 18/10/2019 |
| Ajay Gera | 0.15 | Added the design for End-to-End scheduling  Added sections  2.2.6, 2.2.7, 2.4.2, 2.4.6, 2.4.7, 2.5.15, 2.5.16, 2.5.17, 3.1.8, 3.7.5  Modified sections  2.4.1, 2.4.4, 2.5.1, 2.5.3, 2.5.4 | 30/10/2019 |
| Rajiv Bansal | 0.16 | Updated section 2.6 for retention of dataviews  Updated section 2.4.8 for process model of dashboard  Updated 2.5.12 as per implementation of analysis.  For Record Versioning, following changes:  Added sections 2.2.18, 2.2.19, 2.2.20, 2.5.18, 2.5.19, 2.5.20 | 31/10/2019 |
| Ajay Gera | 0.17 | Updated the diagram in section 2.4.7  Added section 2.5.21 and 2.5.22 | 11/07/2019 |
| Rajiv Bansal | 0.18 | Updated section 2.5.12 for fixes in analysis | 08/11/2019 |
| Shiv Mangal Rahi | 0.19 |  |  |
| Ajay Gera | 0.20 | Included the design to populate rule input values in igx\_ds\_dq\_result store  Updated section 2.2.5, 25.4, 2.5.14 | 14/11/2019 |
| Naveen Seth | 0.21 | Update TDD section 2.5.5 with “Assignment of ownership and how to generate its score” | 15/11/2019 |
| Monideepa Chakravarty | 1.0 | Final technical design document | 18/11/2019 |

# **Overall Solution**

As part of FDA project there is a requirement to develop and baseline "Framework" for DQ Govern scoring.

Scope of the Framework:

• This framework is to support defining rules in Govern, scoring the data against those rules in DQ+ and posting back the result to Govern.

• Based on the hierarchy defined in Govern scoring needs to rolled-up.

# **Technical Design**

## Module Diagram



## Data/DB/Case Store structures



### **IGX\_DS\_DG\_DATA\_ELEMENT\_RULES\_ORG**

This DS stores the original DG rules associated with a Data element for reference purpose. It will be a HDFS store. Older data needs to be purged beyond specified retention period.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| NA | NA (Keeping latest version logic needs to be applied) (Keeping latest version logic needs to be applied) | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Identifier?** | **Data Type** | **Description** |
| DATA\_ENTITY\_NAME |  | String | Data entity Name e.g. FakeNameGenerator |
| DATA\_ENTITY\_UID | Y | String | Data entity UID (Asset UID) |
| DATA\_ELEMENT\_NAME |  | String | Data Element Name e.g. City |
| DATA\_ELEMENT\_UID | Y | String | Data Element UID (Asset UID) |
| RULE\_TYPE |  | String | Rule Type e.g. RE\_RULE\_NOT\_BLANK |
| RULE\_NAME |  | String | Rule Name e.g. IGX000002 |
| RULE\_UID | Y | String | Rule UID |
| RULE\_DESCRIPTION |  | String | RuleDescription e.g. Field should not be blank |
| RULE\_STATUS |  | String | Rule Status e.g. Active, Draft, Inactive |
| DIMENSION |  | String | Dimension e.g. Completeness |
| THRESHOLD |  | Decimal | Threshold e.g. 0.9; Applicable for all Dimensions |
| CONSIDER\_NULL\_AS\_BLANK |  | Boolean | ConsiderNullAsBlank -True/False; Applicable for Completeness Dimension |
| CONSIDER\_ALL\_SPACES\_AS\_BLANK |  | Boolean | ConsiderAllSpacesAsBlank - True/False; Applicable for Consistency Dimension |
| PRE\_CONDITION\_ATTRIBUTE |  | String | PreConditionAttribute ; Applicable for Consistency Dimension |
| PRE\_CONDITION\_OPERATOR |  | String | PreConditionOperator ; Applicable for Consistency Dimension |
| PRE\_CONDITION\_VALUE |  | String | PreConditionValue; Applicable for Consistency Dimension |
| PRE\_CONDITION\_VALUE\_CASE\_SENSITIVE |  | Boolean | PreConditionValueCaseSensitive - True/False; Applicable for Consistency Dimension |
| RULE\_OPERATOR |  | String | RuleOperator; Applicable for Consistency Dimension |
| RULE\_VALUE |  | String | RuleValue; Applicable for Consistency Dimension |
| RULE\_VALUE\_CASE\_SENSITIVE |  | Boolean | RuleValueCaseSensitive -True/False; Applicable for Consistency Dimension |
| TREAT\_BLANK\_OR\_NULL\_AS |  | String | TreatBlankorNullAs - Pass/Skip; Applicable for Duplication, Conformity, Conformity (Regex) and Accuracy Dimension |
| REFERENCE\_LIST |  | String | ReferenceList e.g. IGX Phone Number Blacklist; Applicable for Conformity Dimension |
| MATCH\_REFERENCE\_LIST |  | Boolean | MatchReferenceList -True/False; Applicable for Conformity Dimension |
| REFERENCE\_LIST\_DATA |  | String(4000) | Reference List data; Applicable for Conformity Dimension |
| REGULAR\_EXPRESSION |  | String | RegularExpression; Applicable for Conformity Dimension (Regex) |
| MATCH\_REGULAR\_EXPRESSION |  | Boolean | MatchRegularExpression - True/False; Applicable for Conformity Dimension (Regex) |
| LOWER\_BOUND |  | Integer | LowerBound; Applicable for Accuracy Dimension |
| LOWER\_BOUND\_INCLUSIVE |  | Boolean | LowerBoundInclusive - True/False; Applicable for Accuracy Dimension |
| UPPER\_BOUND |  | Integer | UpperBound; Applicable for Accuracy Dimension |
| UPPER\_BOUND\_INCLUSIVE |  | Boolean | UpperBoundInclusive - True/False; Applicable for Accuracy Dimension |
| WITHIN\_RANGE |  | Boolean | WithinRange - True/False; Applicable for Accuracy Dimension |
| RULE\_ATTRIBUTE\_DATA\_TYPE |  | String | Data type of the attributes for ‘Attribute Comparison’ rule |
| JS\_EXPRESSION |  | String | RegularExpression; Applicable for IGX JS Expression rule type |
| MATCH\_JS\_EXPRESSION |  | Boolean | MatchJSExpression - True/False; Applicable for IGX JS Expression rule type |
| DATA\_ELEMENT\_GROUP |  | String | DataElementGroup e.g. Phone; Applicable for Completeness and Conformity Dimension (Multiple) |
| RE\_HANDLE\_BLANK\_NULL\_AS |  | String | Parameter value from the reference list to be populated here |
| DATA\_ENTITY\_CREATED\_ON |  | DateTime | Data entity CreatedOn |
| DATA\_ENTITY\_UPDATED\_ON |  | DateTime | Data entity UpdatedOn |
| DATA\_ELEMENT\_CREATED\_ON |  | DateTime | Data Element CreatedOn |
| DATA\_ELEMENT\_UPDATED\_ON |  | DateTime | Data Element UpdatedOn |
| RULE\_CREATED\_ON |  | DateTime | Rule CreatedOn |
| RULE\_UPDATED\_ON |  | DateTime | Rule UpdatedOn |
| DELETE\_FLAG |  | Integer | Deleted Flag, 0 – Default, 1 - Deleted |
| D3S\_PROCESS\_ID |  | String | Populate runtime PM Process Id |
| D3S\_WORK\_ID |  | String | For each execution, Work Id will be same for all records |
| D3S\_RECORD\_ID |  | String | UUID; Unique for each record |
| D3S\_SYSTEM\_CREATED\_TS |  | DateTime | Date/Time at which record inserted into data store |
| D3S\_SYSTEM\_UPDATED\_TS |  | DateTime | Date/Time at which record logically updated into data store |
| D3S\_SYSTEM\_RECORD\_VERSION | Y | Big Integer | Version no. of a record. It gets incremented whenever a record logically updated. It starts with one for every PK combination (ignoring this field).  This column will also become part of the PK. |
| D3S\_USER\_DEFINED\_1 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_2 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_3 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_4 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_5 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_6 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_7 |  | String | Spare field |
| D3S\_USER\_DEFINED\_8 |  | String | Spare field |
| D3S\_USER\_DEFINED\_9 |  | String | Spare field |
| D3S\_USER\_DEFINED\_10 |  | String | Spare field |
| D3S\_USER\_DEFINED\_11 |  | String | Spare field |
| D3S\_USER\_DEFINED\_12 |  | String | Spare field |

### **IGX\_DS\_DG\_DATA\_ELEMENT\_RULES**

This DS stores the transformed DG rules associated with a Data element and can have additional columns defined to store transformed output. It will be a HDFS store. This data store will finally be used in subsequent processed. Older data needs to be purged beyond specified retention period.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| NA | NA (Keeping latest version logic needs to be applied) | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Identifier?** | **Data Type** | **Description** |
| DATA\_ENTITY\_NAME |  | String | Data entity Name e.g. FakeNameGenerator |
| DATA\_ENTITY\_UID | Y | String | Data entity UID (Asset UID) |
| DATA\_ELEMENT\_NAME |  | String | Data Element Name e.g. City |
| DATA\_ELEMENT\_UID | Y | String | Data Element UID (Asset UID) |
| RULE\_TYPE |  | String | Rule Type e.g. RE\_RULE\_NOT\_BLANK |
| RULE\_NAME |  | String | Rule Name e.g. IGX000002 |
| RULE\_UID | Y | String | Rule UID |
| RULE\_DESCRIPTION |  | String | RuleDescription e.g. Field should not be blank |
| RULE\_STATUS |  | String | Rule Status e.g. Active, Draft, Inactive |
| DIMENSION |  | String | Dimension e.g. Completeness |
| THRESHOLD |  | Decimal | Threshold e.g. 0.9; Applicable for all Dimensions |
| CONSIDER\_NULL\_AS\_BLANK |  | Boolean | ConsiderNullAsBlank -True/False; Applicable for Completeness Dimension |
| CONSIDER\_ALL\_SPACES\_AS\_BLANK |  | Boolean | ConsiderAllSpacesAsBlank - True/False; Applicable for Consistency Dimension |
| PRE\_CONDITION\_ATTRIBUTE |  | String | PreConditionAttribute ; Applicable for Consistency Dimension |
| PRE\_CONDITION\_OPERATOR |  | String | PreConditionOperator ; Applicable for Consistency Dimension |
| PRE\_CONDITION\_VALUE |  | String | PreConditionValue; Applicable for Consistency Dimension |
| PRE\_CONDITION\_VALUE\_CASE\_SENSITIVE |  | Boolean | PreConditionValueCaseSensitive - True/False; Applicable for Consistency Dimension |
| RULE\_OPERATOR |  | String | RuleOperator; Applicable for Consistency Dimension |
| RULE\_VALUE |  | String | RuleValue; Applicable for Consistency Dimension |
| RULE\_VALUE\_CASE\_SENSITIVE |  | Boolean | RuleValueCaseSensitive -True/False; Applicable for Consistency Dimension |
| TREAT\_BLANK\_OR\_NULL\_AS |  | String | TreatBlankorNullAs - Pass/Skip; Applicable for Duplication, Conformity, Conformity (Regex) and Accuracy Dimension |
| REFERENCE\_LIST |  | String | ReferenceList e.g. IGX Phone Number Blacklist; Applicable for Conformity Dimension |
| MATCH\_REFERENCE\_LIST |  | Boolean | MatchReferenceList -True/False; Applicable for Conformity Dimension |
| REFERENCE\_LIST\_DATA |  | String(4000) | Reference List data; Applicable for Conformity Dimension |
| REGULAR\_EXPRESSION |  | String | RegularExpression; Applicable for Conformity Dimension (Regex) |
| MATCH\_REGULAR\_EXPRESSION |  | Boolean | MatchRegularExpression; Applicable for Conformity Dimension (Regex) |
| LOWER\_BOUND |  | Integer | LowerBound; Applicable for Accuracy Dimension |
| LOWER\_BOUND\_INCLUSIVE |  | Boolean | LowerBoundInclusive - True/False; Applicable for Accuracy Dimension |
| UPPER\_BOUND |  | Integer | UpperBound; Applicable for Accuracy Dimension |
| UPPER\_BOUND\_INCLUSIVE |  | Boolean | UpperBoundInclusive - True/False; Applicable for Accuracy Dimension |
| WITHIN\_RANGE |  | Boolean | WithinRange - True/False; Applicable for Accuracy Dimension |
| RULE\_ATTRIBUTE\_DATA\_TYPE |  | String | Data type of the attributes for ‘Attribute Comparison’ rule |
| JS\_EXPRESSION |  | String | RegularExpression; Applicable for IGX JS Expression rule type |
| MATCH\_JS\_EXPRESSION |  | Boolean | MatchJSExpression - True/False; Applicable for IGX JS Expression rule type |
| DATA\_ELEMENT\_GROUP |  | String | DataElementGroup e.g. Phone; Applicable for Completeness and Conformity Dimension (Multiple) |
| RE\_HANDLE\_BLANK\_NULL\_AS |  | String | Parameter value from the reference list to be populated here |
| DATA\_ELEMENT\_GROUP\_HANDLING |  | String | Multiple Data Element Handling e.g. AtLeastOneMustPass and AllMustPass; Applicable for Completeness and Conformity Dimension (Multiple) |
| DATA\_ENTITY\_CREATED\_ON |  | DateTime | Data entity CreatedOn |
| DATA\_ENTITY\_UPDATED\_ON |  | DateTime | Data entity UpdatedOn |
| DATA\_ELEMENT\_CREATED\_ON |  | DateTime | Data Element CreatedOn |
| DATA\_ELEMENT\_UPDATED\_ON |  | DateTime | Data Element UpdatedOn |
| RULE\_CREATED\_ON |  | DateTime | Rule CreatedOn |
| RULE\_UPDATED\_ON |  | DateTime | Rule UpdatedOn |
| DELETE\_FLAG |  | Integer | Deleted Flag, 0 – Default, 1 - Deleted |
| D3S\_PROCESS\_ID |  | String | Populate runtime PM Process Id |
| D3S\_WORK\_ID |  | String | For each execution, Work Id will be same for all records |
| D3S\_RECORD\_ID |  | String | UUID; Unique for each record |
| D3S\_SYSTEM\_CREATED\_TS |  | DateTime | Date/Time at which record inserted into data store |
| D3S\_SYSTEM\_UPDATED\_TS |  | DateTime | Date/Time at which record logically updated into data store |
| D3S\_SYSTEM\_RECORD\_VERSION | Y | Big Integer | Version no. of a record. It gets incremented whenever a record logically updated. It starts with one for every PK combination (ignoring this field).  This column will also become part of the PK. |
| D3S\_USER\_DEFINED\_1 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_2 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_3 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_4 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_5 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_6 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_7 |  | String | Spare field |
| D3S\_USER\_DEFINED\_8 |  | String | Spare field |
| D3S\_USER\_DEFINED\_9 |  | String | Spare field |
| D3S\_USER\_DEFINED\_10 |  | String | Spare field |
| D3S\_USER\_DEFINED\_11 |  | String | Spare field |
| D3S\_USER\_DEFINED\_12 |  | String | Spare field |



### **IGX\_DS\_DG\_RULE\_ASSET\_TYPE**

This DS stores the Asset Type for each Rule as implemented in Govern.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| 1 | NA | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Identifier?** | **Data Type** | **Description** |
| ASSET\_TYPE\_NAME |  | String | Asset Type at specific level e.g. Rule, Data Element, etc. |
| ASSET\_TYPE\_UID | Y | String | Asset Type UID at specific level |
| RULE\_NAME |  | String | Rule Name e.g. IGX000002 |
| RULE\_UID | Y | String | Rule UID |
| D3S\_PROCESS\_ID |  | String | Populate runtime PM Process Id |
| D3S\_WORK\_ID |  | String | For each execution, Work Id will be same for all records |
| D3S\_RECORD\_ID |  | String | UUID; Unique for each record |
| D3S\_SYSTEM\_CREATED\_TS |  | DateTime | Date/Time at which record inserted into data store |
| D3S\_USER\_DEFINED\_1 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_2 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_3 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_4 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_5 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_6 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_7 |  | String | Spare field |
| D3S\_USER\_DEFINED\_8 |  | String | Spare field |
| D3S\_USER\_DEFINED\_9 |  | String | Spare field |
| D3S\_USER\_DEFINED\_10 |  | String | Spare field |
| D3S\_USER\_DEFINED\_11 |  | String | Spare field |
| D3S\_USER\_DEFINED\_12 |  | String | Spare field |



### **IGX\_DB\_RULE\_REFRESHER\_LOG**

This DB store stores log of the Rule refresher process. This store will be of DB type.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| 1 | 1 year (TBC) | NA | NA |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field** | | **Identifier?** | | **Data Type** | | **Description** | |
| D3S\_PROCESS\_ID | | Y | | String | | Process id of the process | |
| D3S\_WORK\_ID | |  | | String | | Work id of the process | |
| RUN\_DATE | | Y | | Date | | Date on which process executed | |
| START\_TS | |  | | DateTime | | Start time of the process | |
| END\_TS | |  | | DateTime | | End time of the process | |
| D3S\_SYSTEM\_CREATED\_TS | |  | | DateTime | | Date/Time at which record inserted into data store | |
| D3S\_USER\_DEFINED\_1 | |  | | Big Integer | | Spare field | |
| D3S\_USER\_DEFINED\_2 | |  | | Big Integer | | Spare field | |
| D3S\_USER\_DEFINED\_3 | |  | | Date | | Spare field | |
| D3S\_USER\_DEFINED\_4 | |  | | Date | | Spare field | |
| D3S\_USER\_DEFINED\_5 | |  | | DateTime | | Spare field | |
| D3S\_USER\_DEFINED\_6 | |  | | DateTime | | Spare field | |
| D3S\_USER\_DEFINED\_7 | |  | | String | | Spare field | |
| D3S\_USER\_DEFINED\_8 | |  | | String | | Spare field | |
| D3S\_USER\_DEFINED\_9 | |  | | String | | Spare field | |
| D3S\_USER\_DEFINED\_10 | |  | | String | | Spare field | |
| D3S\_USER\_DEFINED\_11 | |  | | String | | Spare field | |
| D3S\_USER\_DEFINED\_12 | |  | | String | | Spare field | |

### **IGX\_DS\_DQ\_RESULTS**

This store maintains the DQ store at the lowest level. Initially Rule evaluator executes the DQ rules and stores the results in this store. Later, Scoring Engine re-evaluates the results for Group fields and updates the rule result if there is any change. This updated result is maintained in Rule\_Result\_Transformed

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| 1 | 30 days | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Identifier?** | **Data Type** | **Description** |
| DATA\_KEY | Y | String | Concatenated value of the identifier fields’ value. |
| RUN\_TIMESTAMP | Y | DateTime | Run timestamp of the process. Value will be same for all output records generated from the DQ of a data entity in a particular run |
| DATA\_ENTITY\_NAME |  | String | Data entity to which result belongs to |
| DATA\_ENTITY\_UID | Y | String | Data entity uid |
| DATA\_ELEMENT\_NAME |  | String | Attribute name |
| DATA\_ELEMENT\_UID | Y | String | Attribute uid |
| DATA\_ELEMENT\_GROUP |  | String | Group of the data element |
| DATA\_ELEMENT\_GROUP\_RULE\_TYPE |  | String | Whether one of the field should pass or all source pass |
| RULE\_NAME |  | String | Rule name |
| RULE\_UID | Y | String | Rule uid |
| DIMENSION |  | String | Dimension of the rule |
| RULE\_RESULT |  | String | Rule result – Pass, Fail, Skip, Error |
| RULE\_RESULT\_TRANSFORMED |  | String | Transformed rule result considering Group field – Pass, Fail, Skip, Error |
| RULE\_INPUT\_VALUES |  | String | List of Column Name and their values, used for evaluation of Rule.  <Column1 Name>: <Column1 Value>; <Column2 Name>: <Column2 Value> |
| DATA\_BATCH\_IDENTIFIER |  | String | Data batch identifier available from source |
| IGX\_SYSTEM\_CREATED\_TS |  | DateTime | Timestamp at which records inserted in store |
| IGX\_SYSTEM\_RECORD\_VERSION | Y | Integer | Version no. of the record for handling logical updates. Scoring engine may update the rule\_result for Group fields |
| IGX\_PROCESS\_ID |  | String | Process id of the process which inserted the record |
| IGX\_WORK\_ID |  | String | Work id of the process which inserted the record |



### **IGX\_DS\_DQ\_DETAIL\_SCORE**

Scoring Engine summarizes IGX\_DS\_DQ\_RESULTS data, generate score at lowest level for each rule after due transformation and store in this DS.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| NA | 30 days | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Identifier?** | **Data Type** | **Description** |
| RUN\_TIMESTAMP | Y | DateTime | Run timestamp of the process. Value will be same for all output records |
| RR\_RECORD\_ID | Y | String | IGX\_DS\_DG\_DATA\_ELEMENT\_RULES. D3S\_RECORD\_ID |
| DATA\_ENTITY\_NAME |  | String | Data entity to which result belongs to |
| DATA\_ENTITY\_UID | Y | String | Data entity uid |
| DATA\_ELEMENT\_NAME |  | String | Attribute name |
| DATA\_ELEMENT\_UID | Y | String | Attribute uid |
| RULE\_NAME |  | String | Rule name e.g. IGX000002 |
| RULE\_UID | Y | String | Rule UID |
| ROW\_COUNT |  | Integer | No. of records in which rule got validated |
| PASS\_COUNT |  | Integer | No. of records in which rule got passed |
| FAIL\_COUNT |  | Integer | No. of records in which rule got failed |
| SKIP\_COUNT |  | Integer | No. of record values skipped because rule could not be evaluated |
| ERROR\_COUNT |  | Integer | No. of records errored out |
| SCORE |  | Float | Lowest level Attribute + Rule level score generated |
| RE\_WORKID |  | String | Rule evaluator work id whose data was used in execution |
| D3S\_PROCESS\_ID |  | String | Populate runtime PM Process Id |
| D3S\_WORK\_ID |  | String | For each execution, Work Id will be same for all records |
| D3S\_RECORD\_ID |  | String | UUID; Unique for each record |
| D3S\_SYSTEM\_CREATED\_TS |  | DateTime | Date/Time at which record inserted into data store |
| D3S\_USER\_DEFINED\_1 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_2 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_3 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_4 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_5 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_6 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_7 |  | String | Spare field |
| D3S\_USER\_DEFINED\_8 |  | String | Spare field |
| D3S\_USER\_DEFINED\_9 |  | String | Spare field |
| D3S\_USER\_DEFINED\_10 |  | String | Spare field |
| D3S\_USER\_DEFINED\_11 |  | String | Spare field |
| D3S\_USER\_DEFINED\_12 |  | String | Spare field |



### **IGX\_DS\_DQ\_ELEMENT\_SCORE**

Scoring Engine summarizes IGX\_DS\_DQ\_DETAIL\_SCORE data, generate score at lowest level without rule and store in this DS.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| NA | 30 days | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Identifier?** | **Data Type** | **Description** |
| RUN\_TIMESTAMP | Y | DateTime | Run timestamp of the process. Value will be same for all output records |
| DATA\_ENTITY\_NAME |  | String | Data entity to which result belongs to |
| DATA\_ENTITY\_UID | Y | String | Data entity uid |
| DATA\_ELEMENT\_NAME |  | String | Attribute name |
| DATA\_ELEMENT\_UID | Y | String | Attribute uid |
| ROW\_COUNT |  | Integer | No. of records in which rule got validated |
| PASS\_COUNT |  | Integer | No. of records in which rule got passed |
| FAIL\_COUNT |  | Integer | No. of records in which rule got failed |
| SKIP\_COUNT |  | Integer | No. of record values skipped because rule could not be evaluated |
| ERROR\_COUNT |  | Integer | No. of records errored out |
| SCORE |  | Float | Lowest level Attribute + Rule level score generated |
| D3S\_PROCESS\_ID |  | String | Populate runtime PM Process Id |
| D3S\_WORK\_ID |  | String | For each execution, Work Id will be same for all records |
| D3S\_RECORD\_ID |  | String | UUID; Unique for each record |
| D3S\_SYSTEM\_CREATED\_TS |  | DateTime | Date/Time at which record inserted into data store |
| D3S\_USER\_DEFINED\_1 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_2 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_3 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_4 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_5 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_6 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_7 |  | String | Spare field |
| D3S\_USER\_DEFINED\_8 |  | String | Spare field |
| D3S\_USER\_DEFINED\_9 |  | String | Spare field |
| D3S\_USER\_DEFINED\_10 |  | String | Spare field |
| D3S\_USER\_DEFINED\_11 |  | String | Spare field |
| D3S\_USER\_DEFINED\_12 |  | String | Spare field |



### **IGX\_DS\_DQ\_RULE\_SCORE**

Scoring Engine summarizes IGX\_DS\_DQ\_DETAIL\_SCORE data, generate score at lowest level without data element and store in this DS.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| NA | 30 days | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Identifier?** | **Data Type** | **Description** |
| RUN\_TIMESTAMP | Y | DateTime | Run timestamp of the process. Value will be same for all output records |
| DATA\_ENTITY\_NAME |  | String | Data entity to which result belongs to |
| DATA\_ENTITY\_UID | Y | String | Data entity uid |
| RULE\_NAME |  | String | Rule name e.g. IGX000002 |
| RULE\_UID | Y | String | Rule UID |
| ROW\_COUNT |  | Integer | No. of records in which rule got validated |
| PASS\_COUNT |  | Integer | No. of records in which rule got passed |
| FAIL\_COUNT |  | Integer | No. of records in which rule got failed |
| SKIP\_COUNT |  | Integer | No. of record values skipped because rule could not be evaluated |
| ERROR\_COUNT |  | Integer | No. of records errored out |
| SCORE |  | Float | Lowest level Attribute + Rule level score generated |
| D3S\_PROCESS\_ID |  | String | Populate runtime PM Process Id |
| D3S\_WORK\_ID |  | String | For each execution, Work Id will be same for all records |
| D3S\_RECORD\_ID |  | String | UUID; Unique for each record |
| D3S\_SYSTEM\_CREATED\_TS |  | DateTime | Date/Time at which record inserted into data store |
| D3S\_USER\_DEFINED\_1 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_2 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_3 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_4 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_5 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_6 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_7 |  | String | Spare field |
| D3S\_USER\_DEFINED\_8 |  | String | Spare field |
| D3S\_USER\_DEFINED\_9 |  | String | Spare field |
| D3S\_USER\_DEFINED\_10 |  | String | Spare field |
| D3S\_USER\_DEFINED\_11 |  | String | Spare field |
| D3S\_USER\_DEFINED\_12 |  | String | Spare field |

### **IGX\_DS\_DQ\_ROLLUP\_SCORE**

This DS stores rolled-up score at each level.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| NA | 30 days | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Identifier?** | **Data Type** | **Description** |
| RUN\_TIMESTAMP | Y | DateTime | Run timestamp of the process. Value will be same for all output records |
| ASSET\_TYPE\_NAME |  | String | Asset Type at specific level e.g. Rule, Data Element, etc. |
| ASSET\_TYPE\_UID |  | String | Asset Type UID at specific level |
| ASSET\_NAME |  | String | Asset name at specific level |
| ASSET\_UID | Y | String | Asset UID at specific level |
| ASSET\_CLASS |  | String | Class of the Asset |
| LEVEL |  | Integer | Asset’s level; Starts at 1 then incremented by 1. |
| WEIGHTAGE |  | Decimal | Weightage; Defaults to 1; used in calculation of next level score |
| ROW\_COUNT |  | Integer | No. of records in which rule got validated |
| PASS\_COUNT |  | Integer | No. of records in which rule got passed |
| FAIL\_COUNT |  | Integer | No. of records in which rule got failed |
| SKIP\_COUNT |  | Integer | No. of record values skipped because rule could not be evaluated |
| ERROR\_COUNT |  | Integer | No. of records errored out |
| SCORE |  | Float | Rolled-up score |
| D3S\_PROCESS\_ID |  | String | Populate runtime PM Process Id |
| D3S\_WORK\_ID |  | String | For each execution, Work Id will be same for all records |
| D3S\_RECORD\_ID |  | String | UUID; Unique for each record |
| D3S\_SYSTEM\_CREATED\_TS |  | DateTime | Date/Time at which record inserted into data store |
| D3S\_USER\_DEFINED\_1 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_2 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_3 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_4 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_5 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_6 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_7 |  | String | Spare field |
| D3S\_USER\_DEFINED\_8 |  | String | Spare field |
| D3S\_USER\_DEFINED\_9 |  | String | Spare field |
| D3S\_USER\_DEFINED\_10 |  | String | Spare field |
| D3S\_USER\_DEFINED\_11 |  | String | Spare field |
| D3S\_USER\_DEFINED\_12 |  | String | Spare field |

### **IGX\_DS\_DG\_HIERARCHY**

This DS stores the hierarchy as implemented in Govern.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| 1 | NA | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Identifier?** | **Data Type** | **Description** |
| ASSET\_TYPE\_NAME |  | String | Asset Type at specific level e.g. Rule, Data Element, etc. |
| ASSET\_TYPE\_UID |  | String | Asset Type UID at specific level |
| ASSET\_NAME |  | String | Asset name at specific level e.g. Birth Date |
| ASSET \_UID | Y | String | Asset UID at specific level |
| ASSET\_CLASS |  | String | Class of the Asset |
| NEXT\_ASSET \_NAME |  | String | Asset name at next higher level e.g. Business Term – Birth Date |
| NEXT\_ASSET \_UID | Y | String | Asset UID at next higher level; Will be Null at the top |
| NEXT\_ASSET\_TYPE\_NAME |  | String | Asset Type at specific level e.g. Rule, Data Element, etc. |
| NEXT\_ASSET\_CLASS |  | String | Class of Asset |
| PREDICATE \_NAME |  | String | Associated predicate name e.g. IGX Has Field |
| PREDICATE \_UID | Y | String | Predicate UID |
| WEIGHTAGE |  | Decimal | Weightage; Defaults to 1 |
| HIERARCHY | Y | String | Identifier for specific Hierarchy e.g. 1, 2, 1a, 1b etc. Defaults to 1. |
| D3S\_PROCESS\_ID |  | String | Populate runtime PM Process Id |
| D3S\_WORK\_ID |  | String | For each execution, Work Id will be same for all records |
| D3S\_RECORD\_ID |  | String | UUID; Unique for each record |
| D3S\_SYSTEM\_CREATED\_TS |  | DateTime | Date/Time at which record inserted into data store |
| D3S\_USER\_DEFINED\_1 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_2 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_3 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_4 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_5 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_6 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_7 |  | String | Spare field |
| D3S\_USER\_DEFINED\_8 |  | String | Spare field |
| D3S\_USER\_DEFINED\_9 |  | String | Spare field |
| D3S\_USER\_DEFINED\_10 |  | String | Spare field |
| D3S\_USER\_DEFINED\_11 |  | String | Spare field |
| D3S\_USER\_DEFINED\_12 |  | String | Spare field |

### **IGX\_DB\_SCORING\_ENGINE\_LOG**

This DB store stores log of the Scoring Engine process. This store will be of DB type.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| 1 | 3 months | NA | NA |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | | **Identifier?** | | **Data Type** | | **Description** |
| D3S\_PROCESS\_ID | | Y | | String | | Process id of the process |
| D3S\_WORK\_ID | |  | | String | | Work id of the process |
| RUN\_DATE | | Y | | Date | | Date on which process executed |
| EVAL\_RUN\_TS | |  | | DateTime | | Max(IGX\_DS\_DQ\_RESULTS.RUN\_TIMESTAMP) |
| START\_TS | |  | | DateTime | | Start time of the process |
| END\_TS | |  | | DateTime | | End time of the process |
| D3S\_SYSTEM\_CREATED\_TS | |  | | DateTime | | Date/Time at which record inserted into data store | |
| D3S\_USER\_DEFINED\_1 | |  | | Big Integer | | Spare field |
| D3S\_USER\_DEFINED\_2 | |  | | Big Integer | | Spare field |
| D3S\_USER\_DEFINED\_3 | |  | | Date | | Spare field |
| D3S\_USER\_DEFINED\_4 | |  | | Date | | Spare field |
| D3S\_USER\_DEFINED\_5 | |  | | DateTime | | Spare field |
| D3S\_USER\_DEFINED\_6 | |  | | DateTime | | Spare field |
| D3S\_USER\_DEFINED\_7 | |  | | String | | Spare field |
| D3S\_USER\_DEFINED\_8 | |  | | String | | Spare field |
| D3S\_USER\_DEFINED\_9 | |  | | String | | Spare field |
| D3S\_USER\_DEFINED\_10 | |  | | String | | Spare field |
| D3S\_USER\_DEFINED\_11 | |  | | String | | Spare field |
| D3S\_USER\_DEFINED\_12 | |  | | String | | Spare field |



### **IGX\_DB\_DG\_SCORING\_CONFIG**

This stores the IGX Scoring Config reference list configured in Govern.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| NA | NA | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Identifier?** | **Data Type** | **Description** |
| KEY | Y | String | IGX Scoring Config.Code |
| VALUE |  | String | IGX Scoring Config.Description |
| D3S\_PROCESS\_ID |  | String | Populate runtime PM Process Id |
| D3S\_WORK\_ID |  | String | For each execution, Work Id will be same for all records |
| D3S\_RECORD\_ID |  | String | UUID; Unique for each record |
| D3S\_SYSTEM\_CREATED\_TS |  | DateTime | Date/Time at which record inserted into data store |
| D3S\_USER\_DEFINED\_1 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_2 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_3 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_4 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_5 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_6 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_7 |  | String | Spare field |
| D3S\_USER\_DEFINED\_8 |  | String | Spare field |
| D3S\_USER\_DEFINED\_9 |  | String | Spare field |
| D3S\_USER\_DEFINED\_10 |  | String | Spare field |
| D3S\_USER\_DEFINED\_11 |  | String | Spare field |
| D3S\_USER\_DEFINED\_12 |  | String | Spare field |

### **IGX\_DB\_SCORE\_REFRESHER\_LOG**

This DB store stores log of the Score Refresher process. This store will be of DB type.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| NA | 90 days | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Identifier?** | **Data Type** | **Description** |
| RUN\_DATE | Y | DateTime | Run timestamp of the process. |
| START\_TS |  | DateTime | Start timestamp of the process |
| END\_TS |  | DateTime | End timestamp of the process |
| ERROR\_MSG |  | String | Description of the error encountered during result posting in Govern |
| SCORING\_RUN\_TS |  | DateTime | Scoring engine run timestamp upto which its results have been considered by Score Refresher |
| IGX\_SYSTEM\_RECORD\_VERSION | Y | Integer | Version of the record |
| IGX\_SYSTEM\_CREATED\_TS |  | DateTime | Timestamp at which records inserted in store |
| IGX\_PROCESS\_ID |  | String | Process id of the process which inserted the record |
| IGX\_WORK\_ID |  | String | Work id of the process which inserted the record |

### **IGX\_DS\_DQ\_HIERARCHICAL\_SCORE**

This store is used store the hierarchical score to feed the Scoring dashboard.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| NA | 30 days | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Identifier?** | **Data Type** | **Description** |
| RUN\_TIMESTAMP | Y | DateTime | Run timestamp of the process. Value will be same for all output records |
| RULE\_DIMENSION |  | String | Dimension of the Rule |
| RULE\_NAME |  | String | Rule name e.g. IGX000002 |
| RULE\_UID | Y | String | Rule UID |
| DATA\_ELEMENT\_NAME |  | String | Attribute name |
| DATA\_ELEMENT\_UID | Y | String | Attribute uid |
| DATA\_ENTITY\_NAME |  | String | Data entity to which result belongs to |
| DATA\_ENTITY\_UID | Y | String | Data entity uid |
| BUSINESS\_TERM |  | String | Business Term to which result belongs to |
| BUSINESS\_TERM\_UID | Y | String | Business Term uid |
| MODEL\_NAME |  | String | Model to which result belongs to |
| MODEL\_UID | Y | String | Model uid |
| OWNER |  | String | Data entity to which result belongs to |
| OWNER\_UID | Y | String | Data entity uid |
| ROW\_COUNT |  | Integer | No. of records in which rule got validated |
| PASS\_COUNT |  | Integer | No. of records in which rule got passed |
| FAIL\_COUNT |  | Integer | No. of records in which rule got failed |
| SKIP\_COUNT |  | Integer | No. of record values skipped because rule could not be evaluated |
| ERROR\_COUNT |  | Integer | No. of records errored out |
| SC\_WORKID |  | String | Scorer work id whose data was used to populate this record |
| IGX\_PROCESS\_ID |  | String | Populate runtime PM Process Id |
| IGX\_WORK\_ID |  | String | For each execution, Work Id will be same for all records |
| IGX\_SYSTEM\_CREATED\_TS |  | DateTime | Date/Time at which record inserted into data store |

### **IGX\_DS\_LATEST\_DS\_DQ\_DETAIL\_SCORE**

Contains scores from IGX\_DS\_DQ\_DETAIL\_SCORE for latest run for each data entity. It will be used for building the dataview for dashboard.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| NA | NA | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Identifier?** | **Data Type** | **Description** |
| RUN\_TIMESTAMP | Y | DateTime |  |
| RR\_RECORD\_ID | Y | String | IGX\_DS\_DG\_DATA\_ELEMENT\_RULES. D3S\_RECORD\_ID |
| DATA\_ENTITY\_NAME |  | String | Data entity to which result belongs to |
| DATA\_ENTITY\_UID | Y | String | Data entity uid |
| DATA\_ELEMENT\_NAME |  | String | Attribute name |
| DATA\_ELEMENT\_UID | Y | String | Attribute uid |
| RULE\_NAME |  | String | Rule name e.g. IGX000002 |
| RULE\_UID | Y | String | Rule UID |
| ROW\_COUNT |  | Integer | No. of records in which rule got validated |
| PASS\_COUNT |  | Integer | No. of records in which rule got passed |
| FAIL\_COUNT |  | Integer | No. of records in which rule got failed |
| SKIP\_COUNT |  | Integer | No. of record values skipped because rule could not be evaluated |
| ERROR\_COUNT |  | Integer | No. of records errored out |
| SCORE |  | Float | Lowest level Attribute + Rule level score generated |
| D3S\_PROCESS\_ID |  | String | Populate runtime PM Process Id |
| D3S\_WORK\_ID |  | String | For each execution, Work Id will be same for all records |
| D3S\_RECORD\_ID |  | String | UUID; Unique for each record |
| D3S\_SYSTEM\_CREATED\_TS |  | DateTime | Date/Time at which record inserted into data store |

### **IGX\_DS\_LATEST\_DS\_DQ\_RESULTS**

Contains scores from IGX\_DS\_DQ\_RESULTS for latest run for each data entity. It will pick the latest record (to get the transformed status) for the rule evaluation. It will be used for building the data view for dashboard.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| 1 | NA | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Identifier?** | **Data Type** | **Description** |
| DATA\_KEY | Y | String | Concatenated value of the identifier fields’ value. |
| RUN\_TIMESTAMP | Y | DateTime |  |
| DATA\_ENTITY\_NAME |  | String | Data entity to which result belongs to |
| DATA\_ENTITY\_UID | Y | String | Data entity uid |
| DATA\_ELEMENT\_NAME |  | String | Attribute name |
| DATA\_ELEMENT\_UID | Y | String | Attribute uid |
| DATA\_ELEMENT\_GROUP |  | String | Group of the data element |
| DATA\_ELEMENT\_GROUP\_RULE\_TYPE |  | String | Whether one of the field should pass or all source pass |
| RULE\_NAME |  | String | Rule name |
| RULE\_UID | Y | String | Rule uid |
| DIMENSION |  | String | Dimension of the rule |
| RULE\_RESULT |  | String | Rule result – Pass, Fail, Skip, Error |
| RULE\_RESULT\_TRANSFORMED |  | String | Transformed rule result considering Group field – Pass, Fail, Skip, Error |
| RULE\_INPUT\_VALUES |  | String | List of Column Name and their values, used for evaluation of Rule.  <Column1 Name>: <Column1 Value>; <Column2 Name>: <Column2 Value> |
| DATA\_BATCH\_IDENTIFIER |  | String | Data batch identifier available from source |
| IGX\_SYSTEM\_CREATED\_TS |  | DateTime | Timestamp at which records inserted in store |
| IGX\_SYSTEM\_RECORD\_VERSION | Y | Integer | Version no. of the record for handling logical updates. Scoring engine may update the rule\_result for Group fields |
| IGX\_PROCESS\_ID |  | String | Process id of the process which inserted the record |
| IGX\_WORK\_ID |  | String | Work id of the process which inserted the record |

### **IGX\_DB\_DG\_DATA\_ENTITY\_CONFIG**

This store will be refreshed by Rule Refresher by sourcing data from “IGX Data Entity Config” reference list. It will be full refresh so existing data will be purged before loading new data. It will be a database store

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| 1 | NA | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Identifier?** | **Data Type** | **Description** |
| DATA\_ENTITY\_NAME | Y | String | Name of the source or data entity |
| INGESTION\_MODE |  | String | D – Direct i.e. Source is always up to date  T – Time based loading; no log of data load is available  L – Load log is available to indicate the load |
| FREQ |  | String | D – Daily  W – Weekly  M – Monthly |
| DAYS |  | String | It is the days on which file is expected e.g.  If Freq is ‘W’  1,3,5 – represents Monday, Wednesday and Friday  If Freq is ‘M  10,20 – 10th and 20th of the month |
| DATA\_LOAD\_TIME |  | String | This field is applicable when ingestion\_mode is ‘L’ or ‘T’   * Time by which data is assumed to get loaded in case Ingestion\_mode is ‘T’ * Maximum time by which to wait for data load when ingestion\_mode = ‘L’. If data is not loaded by this time, data entity will be skipped by Rule Evaluator   Format: HH24:MI |
| IGX\_SYSTEM\_CREATED\_TS |  | DateTime | Timestamp at which records inserted in store |
| IGX\_PROCESS\_ID |  | String | Process id of the process which inserted the record |
| IGX\_WORK\_ID |  | String | Work id of the process which inserted the record |

### **IGX\_DB\_RULE\_EVALUATOR\_LOG**

This store contains the logs of Rule Evaluator process. A new record will be inserted in this store for every execution of each source. It is to be created as a database store

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| NA | NA | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Identifier?** | **Data Type** | **Description** |
| DATA\_ENTITY\_NAME | Y | String | Name or short name of the source |
| RUN\_DATE | Y |  |  |
| START\_TS |  | DateTime | Start timestamp of the process |
| END\_TS |  | DateTime | End timestamp of the process |
| IGX\_SYSTEM\_CREATED\_TS |  | DateTime | Timestamp at which records inserted in store |
| IGX\_PROCESS\_ID |  | String | Process id of the process which inserted the record |
| IGX\_WORK\_ID |  | String | Work id of the process which inserted the record |

### **IGX\_DS\_ASSET\_VERSION**

This store contains the latest version(rolled-up) for each ASSET. This store will be refreshed everytime the process populates it.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| NA | NA | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Identifier?** | **Data Type** | **Description** |
| ASSET\_NAME |  | String |  |
| ASSET\_UID | Y | String |  |
| ASSET\_TYPE |  | String |  |
| ASSET\_TYPE\_UID |  | String |  |
| ASSET\_CLASS |  | String |  |
| Level |  | Integer |  |
| ASSET\_CREATE\_DATE |  | DateTime |  |
| ASSET\_UPDATE\_DATE |  | DateTime |  |
| ASSET\_VERSION |  | Integer |  |
| IGX\_SYSTEM\_CREATED\_TS |  | DateTime | Timestamp at which records inserted in store |
| IGX\_PROCESS\_ID |  | String | Process id of the process which inserted the record |
| IGX\_WORK\_ID |  | String | Work id of the process which inserted the record |

### **IGX\_DS\_ASSET\_VERSION\_TEMP**

This is a temporary store used while identifying the version of assets. It structure will be same as IGX\_DS\_ASSET\_VERSION. This store will be refreshed everytime before the process populates it.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| NA | NA | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Identifier?** | **Data Type** | **Description** |
| ASSET\_NAME |  | String |  |
| ASSET\_UID | Y | String |  |
| ASSET\_TYPE |  | String |  |
| ASSET\_TYPE\_UID |  | String |  |
| ASSET\_CLASS |  | String |  |
| Level |  | Integer |  |
| ASSET\_CREATE\_DATE |  | DateTime |  |
| ASSET\_UPDATE\_DATE |  | DateTime |  |
| ASSET\_VERSION |  | Integer |  |
| IGX\_SYSTEM\_CREATED\_TS |  | DateTime | Timestamp at which records inserted in store |
| IGX\_PROCESS\_ID |  | String | Process id of the process which inserted the record |
| IGX\_WORK\_ID |  | String | Work id of the process which inserted the record |

### **IGX\_DS\_DG\_PREVIOUS\_HIERARCHY**

This DS stores the previous version of hierarchy.

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition number & Type** | **Retention Policy** | **Index** | **Index type** |
| 1 | NA | NA | NA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Identifier?** | **Data Type** | **Description** |
| ASSET\_TYPE\_NAME |  | String | Asset Type at specific level e.g. Rule, Data Element, etc. |
| ASSET\_TYPE\_UID |  | String | Asset Type UID at specific level |
| ASSET\_NAME |  | String | Asset name at specific level e.g. Birth Date |
| ASSET \_UID | Y | String | Asset UID at specific level |
| ASSET\_CLASS |  | String | Class of the Asset |
| NEXT\_ASSET \_NAME |  | String | Asset name at next higher level e.g. Business Term – Birth Date |
| NEXT\_ASSET \_UID | Y | String | Asset UID at next higher level; Will be Null at the top |
| NEXT\_ASSET\_TYPE\_NAME |  | String | Asset Type at specific level e.g. Rule, Data Element, etc. |
| NEXT\_ASSET\_CLASS |  | String | Class of Asset |
| PREDICATE \_NAME |  | String | Associated predicate name e.g. IGX Has Field |
| PREDICATE \_UID | Y | String | Predicate UID |
| WEIGHTAGE |  | Decimal | Weightage; Defaults to 1 |
| HIERARCHY | Y | String | Identifier for specific Hierarchy e.g. 1, 2, 1a, 1b etc. Defaults to 1. |
| D3S\_PROCESS\_ID |  | String | Populate runtime PM Process Id |
| D3S\_WORK\_ID |  | String | For each execution, Work Id will be same for all records |
| D3S\_RECORD\_ID |  | String | UUID; Unique for each record |
| D3S\_SYSTEM\_CREATED\_TS |  | DateTime | Date/Time at which record inserted into data store |
| D3S\_USER\_DEFINED\_1 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_2 |  | Big Integer | Spare field |
| D3S\_USER\_DEFINED\_3 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_4 |  | Date | Spare field |
| D3S\_USER\_DEFINED\_5 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_6 |  | DateTime | Spare field |
| D3S\_USER\_DEFINED\_7 |  | String | Spare field |
| D3S\_USER\_DEFINED\_8 |  | String | Spare field |
| D3S\_USER\_DEFINED\_9 |  | String | Spare field |
| D3S\_USER\_DEFINED\_10 |  | String | Spare field |
| D3S\_USER\_DEFINED\_11 |  | String | Spare field |
| D3S\_USER\_DEFINED\_12 |  | String | Spare field |



## Pipeline Structure

/DQ Govern Scoring/Process Models/

/DQ Govern Scoring/Analysis/

/DQ Govern Scoring/Data Stores/

/DQ Govern Scoring/Data Views/

/DQ Govern Scoring/Dashboards/

**Note**: Above structure will be under agreed path for all FDA components e.g. INFOGIX\_FRAMEWORK.

## Process Model Design

### **IGX\_PM\_Rule\_Refresher**

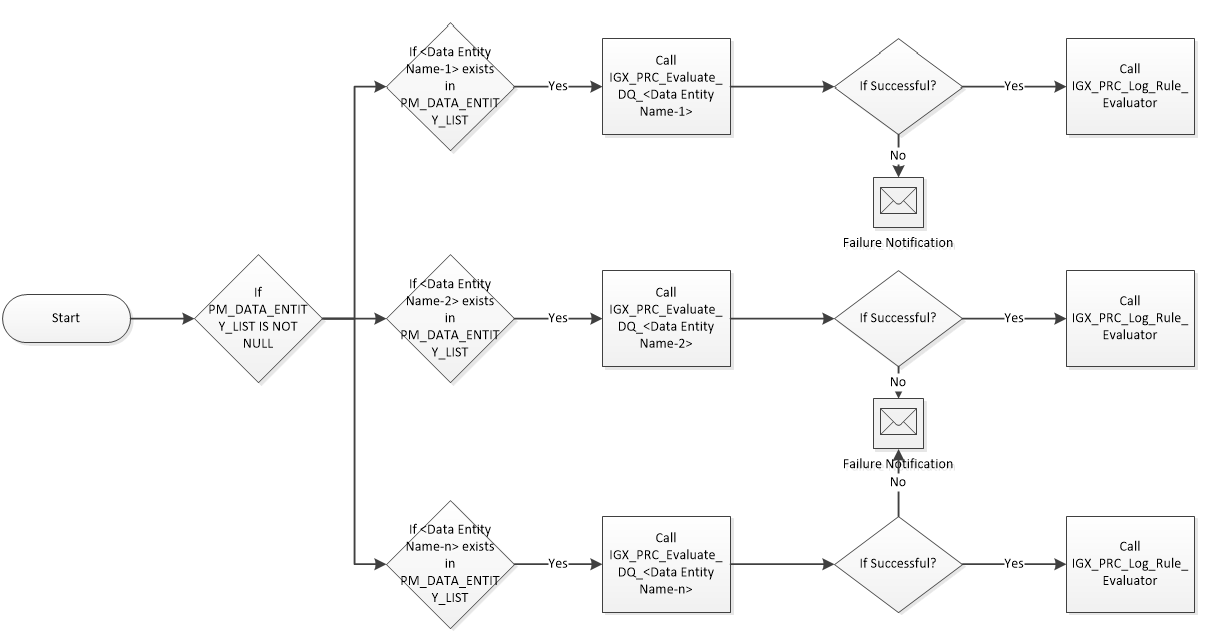
This is the process model which will call the rule refresher and other related analysis as below. It takes PM\_RR\_EXEC\_FLAG as input. If value of this variable is ‘Y’, it continues with the execution else comes out from PM gracefully

1. Define PM\_RR\_EXEC\_FLAG and assign it a default value of ‘Y’
2. If PM\_RR\_EXEC\_FLAG = ‘Y’
   1. Purge igx\_db\_dg\_data\_entity\_config data store. Let existing purging code as is.
   2. IGX\_PRC\_Rule\_Refresher
   3. IGX\_PRC\_Build\_Hierarchy
   4. IGX\_PRC\_BUILD\_HIERARCHY (Move it before calling refresher log analysis)
   5. IGX\_PRC\_Rule\_Refresher\_Log
3. Else EXIT from PM

### **IGX\_PM\_Rule\_Evaluator**

This process model is to execute the rule evaluator process for each data entity. It will take PM\_DATA\_ENTITY\_LIST variable as input which it will get from the previous analysis IGX\_PRC\_FIND\_READY\_DATA\_ENTITIES.

* Separate branch is to be created for each Data Entity for which rule evaluator is to be executed.
* First decision box in each of the branch in below diagram will define process model variable PM\_DATA\_ENTITY\_NAME and assign it the name of the data entity for which that branch has been created. It will check if value defined for PM\_DATA\_ENTITY\_NAME exists in PM\_DATA\_ENTITY\_LIST. If it exists, the execution will proceed in that branch.
* Multiple branches may execute in parallel



### **IGX\_PM\_Scoring\_Engine**

This is the process model which will call the scoring engine and other related analysis. PM will call the below 2 analyses in sequence where each analysis will pass its output to the next analysis.

1. IGX\_PRC\_Scoring\_Engine
2. IGX\_PRC\_Scoring\_Engine\_Log

### **IGX\_PM\_Scoring\_Engine\_Rollup\_Score**

This process model is created to recursively call IGX\_PRC\_Scoring\_Engine\_Rollup\_Score. In each call it will pick assets of next level and roll up their score. It will exit the recursive loop once all the assets are done for score roll up.

Implementation Logic

1. External Start – define variable for max hierarchy level and next hierarchy level.
2. Call analysis IGX\_PRC\_Scoring\_Engine\_Rollup\_Score. Take PM variable for loop flag and next level to be processed.
3. If loop flag is 1 then continue and call again IGX\_PRC\_Scoring\_Engine\_Rollup\_Score for next level, else stop and exit.

### **IGX\_PM\_Score\_Refresher**

This process model is to publish the DQ results to Govern. PM will call the below 3 analysis in sequence.

1. ~~IGX\_PRC\_Find\_New\_For\_Score\_Refresher~~ (Moved it to igx\_pm\_evaluate\_data\_quality)
2. ~~Gateway – If PM\_CURRENT\_TS is not null and <> default date of the variable then proceed to next analysis ELSE exit gracefully~~ (Moved it to igx\_pm\_evaluate\_data\_quality)
3. IGX\_PRC\_POST\_SCORES\_TO\_GOVERN
4. IGX\_PM\_MONITOR\_SR\_STATUS

### **IGX\_PM\_Monitor\_SR\_Status**

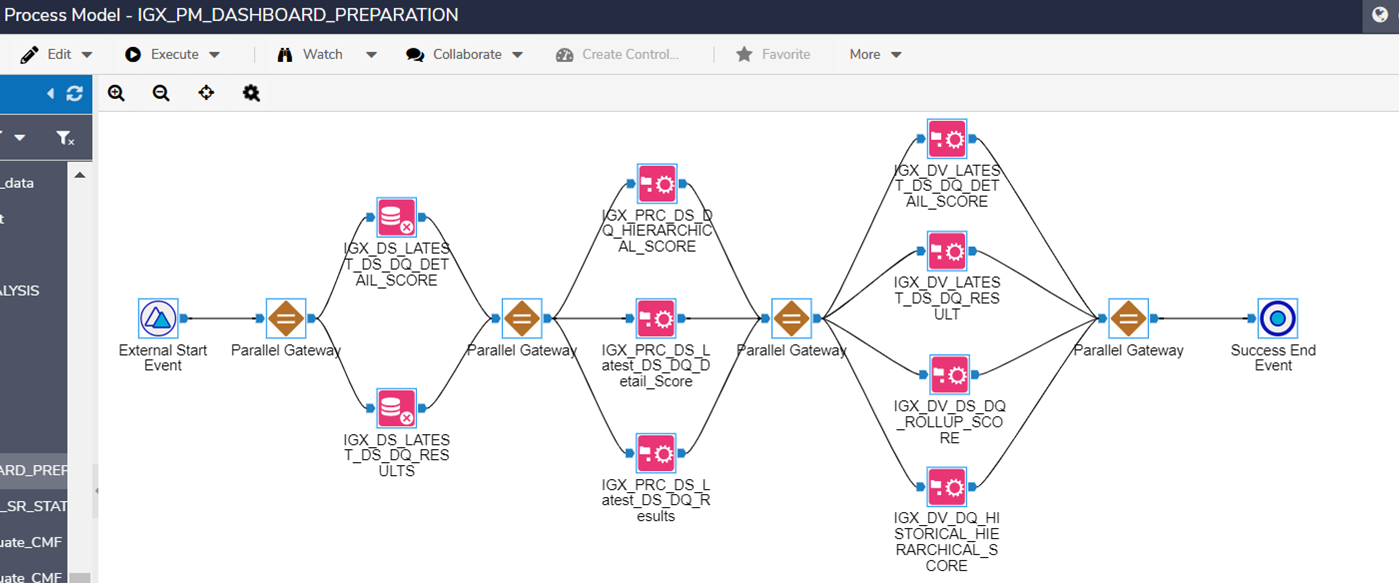
This process model is to monitor the status of DQ result posting API and mark the score refresher process completed

1. IGX\_PRC\_MONITOR\_RESULT\_POSTING\_STATUS
2. If PM\_COMPLETE\_FLAG = ‘Y’ then call IGX\_PRC\_CREATE\_RULE\_REFRESHER\_LOG
3. If PM\_COMPLETE\_FLAG = ‘N’ then wait for 15 seconds and recursively call IGX\_PM\_MONITOR\_SR\_STATUS process model. Pass the received process model variables PM\_FAILED\_ASSET\_TYPE\_ID and PM\_EXECUTION\_ID

### **IGX\_PM\_DASHBOARD\_PREPARATION**

This is the process model which will refresh the data in dataviews for the dashboard. It will also execute any required analysis for data preparation, as needed. Below is the process model diagram. It performs following :

1. Truncate data from IGX\_DS\_LATEST\_DS\_DQ\_DETAIL\_SCORE and IGX\_DS\_LATEST\_DS\_DQ\_RESULTS.
2. Executes analysis IGX\_PRC\_DS\_DQ\_HIERARCHICAL\_SCORE, IGX\_PRC\_DS\_Latest\_DS\_DQ\_Detail\_Score and IGX\_PRC\_DS\_Latest\_DS\_DQ\_Results
3. Refresh dataviews IGX\_DV\_Latest\_DS\_DQ\_Detail\_Score, IGX\_DV\_Latest\_DS\_DQ\_Results, IGX\_DV\_DS\_DQ\_ROLLUP\_SCORE and IGX\_DV\_DQ\_HISTORICAL\_HIERARCHICAL\_SCORE

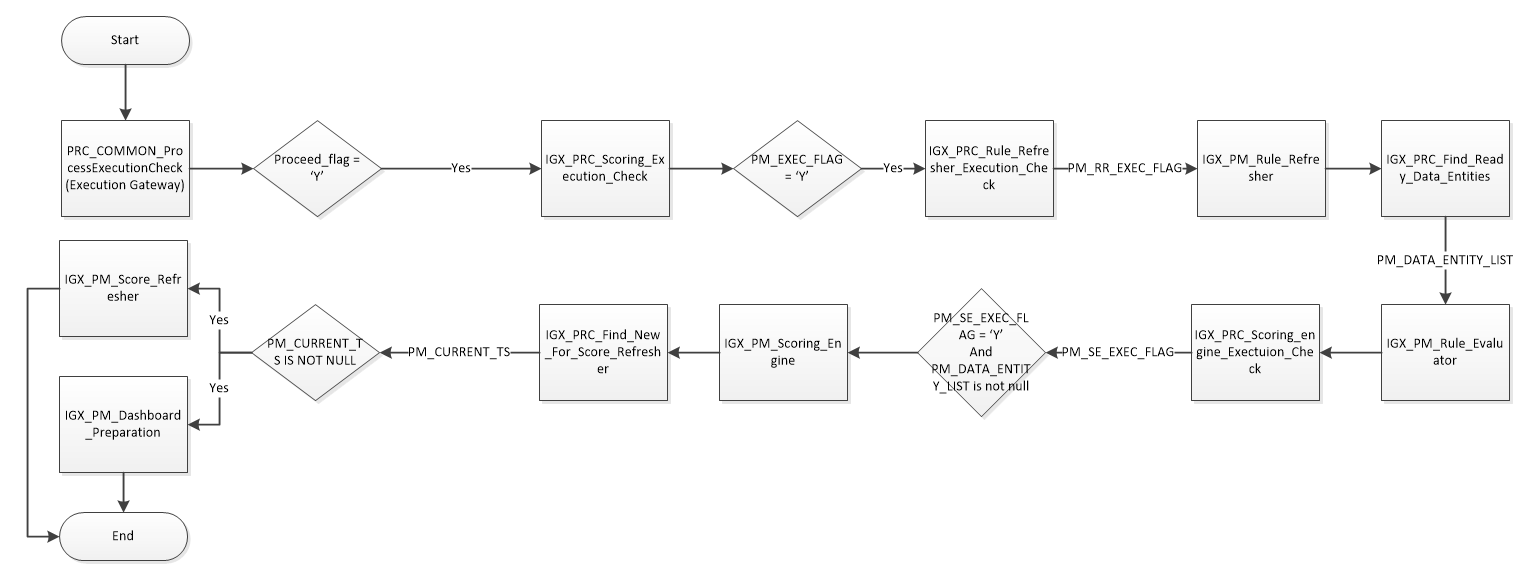


### **IGX\_PM\_Evaluate\_Data\_Quality**

This is the grand process model to trigger various components of the application. Process model is depicted through below diagram

**Note:**

1. Configure the failure notification property of the process model so that notification email goes out if process model fails. Create a super group “Support Team” and send the notification to this group.
2. PRC\_COMMON\_ProcessExecutionCheck analysis part of the ‘Execution Gateway’ framework and may be import in current environment.





## Analysis Design

### **IGX\_PRC\_Rule\_Refresher**

This analysis fetches Data Element and associated rules data from Data Govern and inserts the records in IGX\_DS\_DG\_DATA\_ELEMENT\_RULES\_ORG and IGX\_DS\_DG\_DATA\_ELEMENT\_RULES.

Implementation Logic

1. *Fetch the records:* Fetch all the Data Element and associated rules data using REST API.
   1. Get All Asset Types
   2. Store “IGX Data Entity Config” into data store
      1. Take a branch out from step a
      2. Filter data where name = ‘IGX Data Entity Config’
      3. Call ${governURL}/api/v2/assets/${uid} API
      4. Parse the JSON returned by the API and store relevant 5 fields of the reference list in igx\_db\_dg\_data\_entity\_config
   3. Build a reference key value pair for Data entity, Data element, Predicate and rules by Calling Govern API to fetch the values of reference list ‘IGX Scoring Config’. This API takes the uid of the reference list. API to be called is https://<host>/api/v2/assets/${uid\_of\_reference\_list}. Code of Reference list will be Key and Long description will be Value. Output needs to be stored in IGX\_DB\_DG\_SCORING\_CONFIG which will be a complete refresh in each execution.
   4. Join each value built above with Asset Type Name in order to derive associated Asset type UID.
      1. When the Key is AT\_DATA\_ENTITY using Asset type UID (IGX Data entity UID) and Predicate UID (PE\_DATA\_ENTITY\_TO\_DATA\_ELEMENT\_PREDICATE\_UID) to get all data entities and related data elements. Set returned Relationships.AssetUID as DATA\_ELEMENT\_UID. API to be called is https://<host>/api/v2/assets/ ${uid\_of\_IGXDataEntity}/?\_predicateUid=${predicateUid}
      2. When the Key is AT\_DATA\_ELEMENT using Asset type UID (IGX Data Element UID) and Predicate UID (PE\_DATA\_ELEMENT\_TO\_RULE\_PREDICATE\_UID) get all data elements and related rule’s UID. Set returned Relationships.AssetUID as RULE\_UID. API to be called is https://<host>/api/v2/assets/ ${uid\_of\_IGXDataElement}/?\_predicateUid=${predicateUid}
      3. For others, using Asset type UID (Rule Type UID) derive the rules. API to be called is https://<host>/api/v2/assets/ ${uid\_of\_IGX\_rule\_type}. Add AssetTypeUid and AssetTypeName in the flow to be used while populating IGX\_DS\_DG\_RULE\_ASSET\_TYPE.
      4. When there is associated reference list in the rule (IGX Reference Lookup) then build the reference list data in format as [“Data1”,”Data2”,…] and populate ReferenceListData. API to be called is https://<host>/api/v2/assets/${uid\_of\_reference\_list}.
   5. Join DATA\_ELEMENT\_UID derive in step c(i) with AssetUID derive in c(ii) in order to associate actual data elements with data entities.
   6. Join RULE\_UID derive in step c(ii) with AssetUID derive in c(iii) in order to associate actual rules with data elements.

Use below link for reference –

<https://cafe-modeloffice.infogix.com/desktop/index.html#analysis/67793>

1. *Multiple Data Element handling:* In case DATA\_ELEMENT\_GROUP column is populated then determine ”Type Of Check” from “IGX Data Element Group” reference list for related “Group”. Set DATA\_ELEMENT\_GROUP\_HANDLING based on “Type of Check” value.
2. *Identify the change records:* If the record already exists in IGX\_DS\_DG\_DATA\_ELEMENT\_RULES\_ORG then for each DATA\_ENTITY\_UID, DATA\_ELEMENT\_UID and RULE\_UID combination compare DATA\_ENTITY\_UPDATED\_ON, DATA\_ELEMENT\_UPDATED\_ON and RULE\_UPDATED\_ON with the data retrieved currently. If any of DATA\_ENTITY\_UPDATED\_ON, DATA\_ELEMENT\_UPDATED\_ON and RULE\_UPDATED\_ON mismatches then only insert a new record in IGX\_DS\_DG\_DATA\_ELEMENT\_RULES\_ORG and increment the version in D3S\_SYSTEM\_RECORD\_VERSION. Determine the same set of records for IGX\_DS\_DG\_DATA\_ELEMENT\_RULES.
3. *Insert the records:* Insert the records in IGX\_DS\_DG\_DATA\_ELEMENT\_RULES\_ORG and IGX\_DS\_DG\_DATA\_ELEMENT\_RULES. Set RULE\_TYPE as Key returned in 1(b). Ensure only new records identified by distinct RULE\_UID is added to IGX\_DS\_DG\_RULE\_ASSET\_TYPE incrementally i.e. No Update/Delete on this table.

### **IGX\_PRC\_Rule\_Refresher\_Log**

This analysis makes log entry in IGX\_DB\_RULE\_REFRESHER\_LOG.

Implementation Logic

1. *Insert the Log:*
   1. Insert a new record.
   2. Populate current date in run\_date, current timestamp in END\_TS and process\_id/work id in respective fields. Also Set START\_TS as initialized start time done in process model.

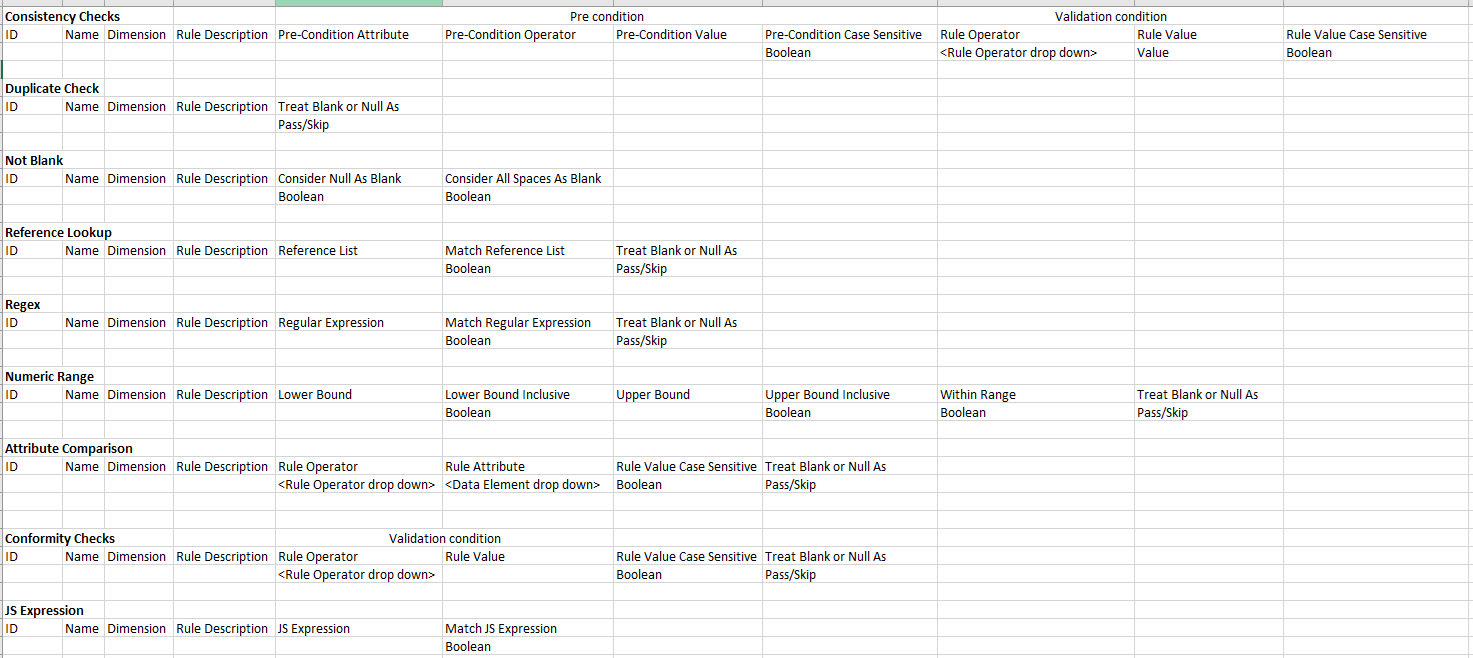
### **IGX\_PRC\_Evaluate\_DQ\_<Data entity>**

This analysis prepares the input data for rule evaluator. It will read the incremental data or newly loaded data from input data store. Criteria to identify the incremental data may vary from data store to data store and will be implemented as appropriate in project. This type of analysis will be written for each data source on which data quality runs are to be run.

1. Read source data
   1. Convert all non-string fields to string which are likely to be used in DQ rules
   2. Transform data into columnar data i.e. key-value pair and name them as field\_name and field\_value. Output of this step should be sequential no., source’s identifier fields concatenated by ‘|’ (alias it as Data\_Key), field\_name, field\_value
   3. Add field Run\_date and assign it current timestamp. Make sure that same value is populated in all records
   4. Add another field data\_batch\_identifier which batch of data can be identified. It may be some real batch identifier or date in yyyymmdd format
   5. Populate DATA\_ENTITY in the process model variable PM\_DATA\_ENTITY and pass to next analysis.
   6. Take out another branch from step d, group the data based on data\_key, run\_date and data\_batch\_identifier so that you get the field\_name and field\_value in associative array. This step is needed to evaluate JS expression based rules in next analysis. If you don’t have the requirement to build expression based rules on a data source, this step can be skipped and rather a dummy flow may be created to fulfil the input requirement of next analysis
2. Call igx\_prc\_dq\_rules\_executor analysis using Nested Analysis node and input the output of both step d (output-1) and f (output-2) to it
3. From step 1c, assign the value of run date to process model variables PM\_RUN\_DATE

### **IGX\_PRC\_DQ\_Rules\_Executor**

Analysis will support following type of rules. Each rule type follows its own style or template of rule definition. Following rule templates are to be supported



1. Take output of IGX\_PRC\_RE\_Read\_Source\_<Data entity> analysis
2. Read latest version of records from IGX\_DS\_DG\_DATA\_ELEMENT\_RULES and filter records where DATA\_ENTITY\_name = PM\_DATA\_ENTITY and rule\_status = ‘Active’ and delete\_flag = 0.
3. Take out two branches by applying filters
   1. Non-expression based rules i.e.
   2. Expression based rules i.e.
4. Inner join step1 (output-1) with step 3a based on data\_element\_name (case insensitive). Data element name may contain data store name also, extract the data element name/field name as the content after the last dot “.” E.g. if value is “customer.name”, extract “name” as the data element name. If no dot is present then whole content to be considered as data element name.
5. Cross join step1 (output-2) with 3b and apply rule execution logic given for Expression rules
6. Apply the rule processing logic as applied at the existing analysis <https://cafe-modeloffice.infogix.com/desktop/index.html?tenantId=Infogix#analysis/67793> on the output of step4. This analysis has separate branch for each rule type however now we won’t create such branches. Rather in “Rule Executor” JS node, have conditional branching of code depending upon the rule\_type
7. Following changes are to be made in rule processing logic for changed or newly added rule types
   1. Consistency (changed)

In existing implementation, rule was considered assign to data element of pre-condition whereas it should be related to rule’s data element. As a result of it, Pre-condition Data Element has been added and “Compare To” has been removed from the configuration however both of them are available from the IGX\_DS\_DG\_ELEMENT\_RULES. Handle this change appropriately.

If pre-condition of the rule fails then consider that rule as skipped. Mark the rule result as ‘Skip’

* 1. Data Element Comparison (New)

This check involves comparing two data elements of same data type however data in columnar format gets converted to string. As per the data type mentioned in the Rule\_attribute\_data\_type, convert the value to respective data type before comparing. To get the values of both attributes, it would require joining with step4 two times to get the value of both the attributes. If condition of the rule gets satisfied, rule is considered Passed otherwise Fail. “Treat Blank Or Null As” being “Skip” means that rule result should be ‘Skip’ if any of the data elements is blank/null otherwise ‘Pass’

* 1. Conformity Checks (New)

This check is same as consistency check but without the Pre-condition

* 1. Duplicate Check (Changed)

If value of the data element is blank or null, rule result will be as defined by rule parameter “Treat Blank Or Null As”.

* 1. Reference Lookup (Changed)

If value of the data element is blank or null, rule result will be as defined by rule parameter “Treat Blank Or Null As”.

* 1. Regex (Changed)

If value of the data element is blank or null, rule result will be as defined by rule parameter “Treat Blank Or Null As”.

* 1. Numeric Range (Changed)

If value of the data element is blank or null, rule result will be as defined by rule parameter “Treat Blank Or Null As”.

* 1. JS Expression (New)

Refer the POC analysis for the execution of Expression rules - http://10.2.20.72:8080/desktop/index.html#analysis/55172

* 1. Follow JS user-defined function

Use function based approach to achieve rule execution in JS and use rule library to keep JS functions. It will increase the code re-usability and keep the code concise.

* 1. Exception handling

Exception handling is to be done in all the rules so that if any exception occurs in execution of JS mainly due to data type mismatch, we catch the exceptions and gracefully mark the output of the rule as ‘Error’. Analysis should not terminate due to such errors.

1. Store the output of the rule evaluation result in IGX\_DS\_DQ\_RESULTS store. While storing the result, populate rule result i.e. Pass, Fail, Skip or Error in both Rule\_Result and Rule\_Result\_Transformed fields
2. Store input values in rule\_Input\_values field of igx\_ds\_dq\_results store
   1. Store one or multiple field values used in rule evaluation in pre-defined format i.e. <Field1 Name>: <Field1 Value>; <Field2 Name>: <Field2 Value>; <Field*n* Name>: <Field*n* Value>
   2. These field values are to be populated for all records irrespective of the rule evaluation result
   3. JS rules may use one or more fields. Consistency check and Attribute Comparison rules uses two fields for rule evaluation. Rest of the rules use only one field for rule evaluation. Accordingly, above mentioned string to be prepared

|  |
| --- |
| **Note:** Following type of rule scenario considered but currently not supported by current design   * Expression based rules supported in rule library e.g. rule to validate sales commission should be 10% of salary, rule library in DQ+ allows to write expression like “commission = (salary\*10)/100” – **Now supported through JS Expression Rule type** * Composite rules e.g. if data has multi line/field home address and office address. If requirement is to consider an address passed only if all lines/fields of an address are passed and address information is considered passed if atleast one of the address is passed. – **Now supported** * Reference Lookup from source external to Govern * Composite field lookup rule e.g. checking if City and Zip code combination valid by looking up in Govern reference list (To be explored) |

### **IGX\_PRC\_Build\_Hierarchy**

This analysis fetches related assets from Data Govern, builds hierarchy and insert the records in IGX\_DS\_DG\_HIERARCHY. Whenever this is called need to refresh IGX\_DS\_DG\_HIERARCHY completely i.e. first Truncate IGX\_DS\_DG\_HIERARCHY from PM and then call this analysis.

Implementation Logic

1. *Fetch the records:* Fetch all the assets hierarchy using REST API.
   1. Get all the response of API “/api/v2/assets/types” assets.
   2. Get All Asset Types and their Class
   3. Filter out the records with Class Name as “User” , “Group” & “Refernces”.
   4. Fetch data from IGX\_DB\_DG\_SCORING\_CONFIG and filter on RE, PR and PE prefix. Maintain three lists based on RE, PR and PE prefix separately.
   5. Get all the predicates using /api/v2/relationships/predicates. Filter the output list based on PR\_HIERARCHY\_PARTICIPANT\_PREDICATES list. Remember it is a comma separated list. Also fetch the predicate uid for PR\_RULE\_TO\_LOWEST\_LEVEL\_PREDICATE.
   6. From IGX\_DB\_DG\_SCORING\_CONFIG, filter records for “'AT\_DATA\_ELEMENT'” . join these records with the Assets returned from point (C) above. This will give the Asset UID for lowest level Assets i.e. for “Data Element”.
   7. Get all the response of API “/api/v2/assets/${uid}” . This will give all the objects/records for “Data Elements” defined in the system with AssetUid. This will become the lowest/starting item for hierarchy. Find the higher level item using lowest level to next level predicate uid e.g. “IGX Has Field” as stored in PE\_DATA\_ENTITY\_TO\_DATA\_ELEMENT\_PREDICATE\_UID configuration available in PE list. Use API https://<host>/api/v2/assets/ ${assetTypeUid}/?predicateUid=${predicateUid}
      1. Relationships section will provide AssetUID of higher level item.
   8. In order to determine higher level Asset Type UID, check for all the glossary items as received in point (a), match the returned AssetUID with point (f.i) (e) AssetUID.
   9. Now pass all the predicates as received in point (d) minus the predicates already used to fetch next level hierarchy item. Use API https://<host>/api/v2/assets/ ${assetTypeUid}/?\_predicateUid=${predicateUid}
      1. Relationships section will provide AssetUID of higher level item.
   10. Repeat step h and i till no response is received.
   11. Now fetch the responsibility associated with top level item using API /api/v2/responsibilities/types/{assetTypeUid?
   12. Care should be taken for multi-hierarchy scenario where an asset at any level can be associated with multiple predicates and accordingly build the hierarchy. By default, set 1 in HIERARCY column. In case of multi-hierarchies increment the value as applicable or branch them as 1a, 1b, etc.
   13. Weightage column is populated using data fetched from relationships API
       1. Call GET /api/v2/relationships/types?predicateUid=${predicateUid} for all predicates UID as applicable using output of point d. This will give uid which will become RelationshipTypeUid to be passed to below API call.
       2. GET /api/v2/relationships?RelationshipTypeUid=${RelationshipTypeUid}. Output needs to be joined as below –
       3. Relationships.AssetUid as obtained at each level above = Subject.Uid
       4. AssetUid as obtained at each level above = Object.Uid
   14. Insert the records in to IGX\_DS\_DG\_HIERARCHY
   15. Get Distinct AssetTypeUid and call the API ${governURL}/api/v2/responsibilities/assignments?\_assetTypeUid=${AssetTypeUIDTo}&\_pageSize=250 to get Owners & Group with Uid and Related Asset Uid.
   16. For the Groups retrieved from Step o , get all the related owners/users by calling API ${governURL}/api/v2/membership/groups/${ToAssetUid}/members.
   17. Retrieve the Application User by Calling API ${governURL}/api/v2/assets/types?Class=User.
   18. Assign the retrieved Application User in step q to all records retrieved in Step p i.e. populate AssetTypeUidTo & AssetTypeNameTo. This will give all the owners in the system with related AssetsUid .
   19. Join the records from Step m with step r to get the owners of all the assets except lower level i.e. Data Element.
   20. Join the records from Step m with Step f to get Lowest Level records i.e. Data Elements objects.
   21. Join records received in step t with step r to get the owners of lowest level records i.e. Data Elements.
   22. Union the records from Step r and Step t and insert in Table IGX\_DS\_DG\_HIERARCHY
   23. Get all the response of API “/api/v2/assets/types” using parameter as “Rule”. This will return all the Rule Types like ‘IGX Not Blank”, “IGX Regex” etc.
   24. For each returned Rule Type pass related predicate uid e.g. as that of “IGX Rules Applies to Field” as stored in PE\_DATA\_ELEMENT\_TO\_RULE\_PREDICATE\_UID configuration available in PE list. Use API https://<host>/api/v2/assets/ ${assetTypeUid}/?\_predicateUid=${predicateUid}
   25. This will give Asset Type of Rule. Step (w) will give you the Class of rule.

### **IGX\_PRC\_Scoring\_Engine**

This analysis fetches data from Evaluator engine output, performs transformation, builds summary score and finally rolled-up score using generated hierarchy.

Implementation Logic

1. *Fetch the records:* Fetch all the records from IGX\_DS\_DQ\_RESULTS where RUN\_TIMESTAMP is greater than MAX(IGX\_DB\_SCORING\_ENGINE\_LOG.EVAL\_RUN\_TS). First time pick all the records. Pass the Max(RUN\_TIMESTAMP) value through process model to next analysis IGX\_PRC\_Scoring\_Engine\_Log.
2. *Data Element Group handling:* In case DATA\_ELEMENT\_GROUP column is populated in a row then find related records having same DATA\_ELEMENT\_GROUP value within same DATA\_KEY set. Based on DATA\_ELEMENT\_GROUP\_TYPE\_VALUE transform the RULE\_RESULT i.e. if value is AtLeastOneMustPass and if single participant result is Pass then all the participants rows will be transformed to Pass. Similarly, if value is AllMustPass and if single participant result is Fail then all the participants’ rows will be transformed to Fail. Keep this value in TRANSFORMED\_STATUS derived field; if no transformation done for other rows copy the RULE\_RESULT value as is.
3. Update transformed status in igx\_ds\_dq\_results
   1. Take out a branch from step 2 above
   2. Filter records where Rule\_Result is not equal to transformed\_status
   3. Insert the records in igx\_ds\_dq\_results. While inserting, increment the version no. and populate in igx\_system\_record\_version field. Populate transform\_status in rule\_result\_transformed field. For rest of the fields, copy the values as they were in previous version of record
4. *Detail Score generation:* 
   1. Group the records based on RUN\_TIMESTAMP, DATA\_ENTITY\_NAME, DATA\_ENTITY\_UID, DATA\_ELEMENT\_NAME, DATA\_ELEMENT\_UID, RULE\_NAME and RULE\_UID and determine PASS\_COUNT, FAIL\_COUNT, SKIP\_COUNT and ERROR\_COUNT from TRANSFORMED\_STATUS.
   2. Calculate Score as (*pass\_count)/(fail\_count + pass\_count)\*100.* Skip count won’t be considered in score calculation. If both fail count and pass count are zero, populate score as -1 i.e. N/A.
   3. Prepare to store in IGX\_DS\_DQ\_DETAIL\_SCORE.
5. *Summary Score generation:* 
   1. Group the records based on RUN\_TIMESTAMP, DATA\_ENTITY\_NAME, DATA\_ENTITY\_UID, DATA\_ELEMENT\_NAME, DATA\_ELEMENT\_UID and determine PASS\_COUNT, FAIL\_COUNT, SKIP\_COUNT and ERROR\_COUNT.
   2. Calculate Score as (*pass\_count)/(fail\_count + pass\_count)\*100.*
   3. Prepare to store in IGX\_DS\_DQ\_ELEMENT\_SCORE.
   4. Group the records based on RUN\_TIMESTAMP, DATA\_ENTITY\_NAME, DATA\_ENTITY\_UID, RULE\_NAME, RULE\_UID and determine PASS\_COUNT, FAIL\_COUNT, SKIP\_COUNT and ERROR\_COUNT.
   5. Calculate Score as (*pass\_count)/(fail\_count + pass\_count)\*100.*
   6. Prepare to store in IGX\_DS\_DQ\_RULE\_SCORE.
6. *Roll-up Score generation:* 
   1. Lowest Level
      1. For each DATA\_ELEMENT\_NAME get the latest record (using RUN\_TIMESTAMP) from IGX\_DS\_DQ\_ELEMENT\_SCORE if it belongs to different DATA\_ENTITY\_UID which are not available in this run.
      2. Group the records based on DATA\_ELEMENT\_NAME, DATA\_ELEMENT\_UID and determine PASS\_COUNT, FAIL\_COUNT, SKIP\_COUNT and ERROR\_COUNT.
      3. Calculate Score as sum(*pass\_count)/sum(fail\_count + pass\_count)\*100*
      4. Join with IGX\_DS\_DG\_HIERARCHY to fetch ASSET\_TYPE\_UID, ASSET\_TYPE\_NAME and ASSET\_CLASS for associated DATA\_ELEMENT\_UID (Join on IGX\_DS\_DG\_HIERARCHY.ASSET\_UID).
      5. Prepare to store in IGX\_DS\_DQ\_ROLLUP\_SCORE
      6. For each RULE\_NAME get the latest record (using RUN\_TIMESTAMP) from IGX\_DS\_DQ\_RULE\_SCORE if it belongs to different DATA\_ENTITY\_UID which are not available in this run.
      7. Group the records based on RULE\_NAME and RULE\_UID and determine PASS\_COUNT, FAIL\_COUNT, SKIP\_COUNT and ERROR\_COUNT. Calculate Score as sum(*pass\_count)/sum(fail\_count + pass\_count)\*100*
      8. Join with IGX\_DS\_DG\_RULE\_ASSET\_TYPE to fetch ASSET\_TYPE\_UID and ASSET\_TYPE\_NAME for associated RULE\_UID. Consider Asset Class as ‘Rule’
      9. Prepare to store in IGX\_DS\_DQ\_ROLLUP\_SCORE
   2. Higher Levels
      1. Group the records of current run based on RUN\_TIMESTAMP, DATA\_ENTITY\_NAME, DATA\_ENTITY\_UID and determine PASS\_COUNT, FAIL\_COUNT, SKIP\_COUNT and ERROR\_COUNT. Records to be grouped is at the level of IGX\_DS\_DQ\_DETAIL\_SCORE.
      2. Calculate Score as sum(*pass\_count)/sum(fail\_count + pass\_count)\*100*
      3. Join with IGX\_DS\_DG\_HIERARCHY to fetch ASSET\_TYPE\_UID, ASSET\_TYPE\_NAME and ASSET\_CLASS for the associated DATA\_ENTITY\_UID (Join on IGX\_DS\_DG\_HIERARCHY.ASSET\_UID).
      4. Prepare to store in IGX\_DS\_DQ\_ROLLUP\_SCORE
      5. Maintain a list of all assets (temporary store) which are candidate to be stored in IGX\_DS\_DQ\_ROLLUP\_SCORE in this run.
      6. Determine the next level Asset by joining IGX\_DS\_DQ\_ELEMENT\_SCORE.DATA\_ELEMENT\_UID of current run and IGX\_DS\_DG\_HIERARCHY.ASSET\_UID and fetch NEXT\_ASSET\_UID. It is possible that multiple NEXT\_ASSET\_UID is fetched from IGX\_DS\_DG\_HIERARCHY due to multi-hierarchies. Handle each hierarchy by maintaining a list of pending hierarchy and striking of those from list which have been handled. Skip those Assets which are included in temporary store as mentioned in above point, however ensure that higher level asset is still picked for skipped asset.
      7. For each fetched NEXT\_ASSET\_UID get the latest record (using RUN\_TIMESTAMP) from IGX\_DS\_DQ\_ROLLUP\_SCORE for associated ASSET\_UID which are not available in this run or other related Assets of this run. Association can again be derived from IGX\_DS\_DG\_HIERARCHY by joining derived NEXT\_ASSET\_UID with IGX\_DS\_DG\_HIERARCHY.NEXT\_ASSET\_UID and get all the related ASSET\_UID minus ASSET\_UID of the current run.
      8. Determine Asset Type UID, Name and Asset Class from IGX\_DS\_DG\_HIERARCHY at each level.
      9. If Weightage is available, then
         1. Calculate Score as sum((*score\*weightage) + (n.score\*weightage ))* for eachNEXT\_ASSET\_UID; In case score is -1 skip it*.*

Else

Calculate Score as sum(*pass\_count)/sum(fail\_count + pass\_count)\*100*

* + 1. Keep on updating the list of all assets (temporary store) as mentioned in point (v) above if not already exists which have participated in this run (due to being start of a hierarchy or being fetched from next level asset being associated to it) so that same asset is not rolled-up multiple times due to different hierarchies.
    2. Repeat above two steps joining the fetched NEXT\_ASSET\_UID with ASSET\_UID till NEXT\_ASSET\_ID return no rows for each determined hierarchy.
    3. Prepare to store in IGX\_DS\_DQ\_ROLLUP\_SCORE

1. *Insert the records:* Insert the records in IGX\_DS\_DQ\_DETAIL\_SCORE, IGX\_DS\_DQ\_ELEMENT\_SCORE, IGX\_DS\_DQ\_RULE\_SCORE and IGX\_DS\_DQ\_ROLLUP\_SCORE.

### **IGX\_PRC\_Scoring\_Engine\_Part\_B**

This analysis is responsible to calculate levels of various assets beyond data entities. Actually to rollup scores, we need level of objects so that child scores can be calculated/rolled up before its parent.

Based on parent child relations we get in hierarchy output, this analysis measures levels and put entry in temporary table, which will be used by analysis which rolls up the score.

Implementation Logic

1. Read all data from IGX\_DS\_DG\_HIERARCHY and filter/take data where “NEXT\_ASSET\_CLASS <> 'Rule'”
2. Read all data from IGX\_DS\_DG\_HIERARCHY, and take distinct assets where asset type is element and next asset class is not “Rule”. Stamp level as 1 for all these records. Please note that, to filter element type data use configuration reference data i.e. IGX\_DB\_DG\_SCORING\_CONFIG.
3. Use co-group node and group output of #i and #ii based on a dummy field (having same value).
4. Use java script node, to calculate level of assets. JS node will have below logic:
   1. Create a queue and push all the elements with level. List of elements with their default level 1 is in output of #ii.
   2. Loop thru this queue unless it gets empty and do below
      1. Scan thru all the records in hierarchy (output of #i), and check where assetuid matches with queue’s assetuid.
      2. If matches, then increase level by 1. Emit a record as

ASSET\_UID – NEXT\_ASSET\_UID of hierarchy

ASSET\_NAME - NEXT\_ASSET\_NAME of hierarchy

ASSET\_TYPE\_NAME - NEXT\_ASSET\_TYPE\_NAME of hierarchy

CHILD\_ASSET\_NAME - ASSET\_NAME of hierarchy

LEVEL – level increased by 1

* + 1. Push a new entry in queue, with assetuid as NEXT\_ASSET\_UID of matching hierarchy record.
  1. Delete record at peak from queue.

1. On output of #iv, consider records which are not data entity. Apply grouping on assetuid and asset\_name and take maximum level of each asset. Please note to filter out data entities use IGX\_DB\_DG\_SCORING\_CONFIG to read description based on key.
2. Insert the output of #iv in a temporary table i.e. ASSET\_LEVEL\_TEMP
3. Make a custom output, for minimum and maximum level. This will be passed as PM variable and help in recursively calling rollup PM.

### **IGX\_PRC\_Scoring\_Engine\_Rollup\_Score**

This analysis is responsible for rolling up scores and making entry in rollup score data store. This analysis is called recursively using a process model (IGX\_PM\_Scoring\_Engine\_Rollup\_Score) , and in each run it considers records of next level.

Implementation Logic

1. Read records of current level (passed as PM variable) , from data store ASSET\_LEVEL\_TEMP.
2. Join output of #i , with hierarchy data store (GX\_DS\_DG\_HIERARCHY) to fetch child assetuids. Join will be based on temp.asset\_uid=hierarchy.next\_asset\_uid.
3. Read all records of IGX\_DS\_DQ\_ROLLUP\_SCORE, and fetch recent run records of assets.
4. Join #ii and #iii to fetch score of child assets.
5. Rollup score of assets, using below formula
   1. ROW\_COUNT – Sum of children’s row count
   2. PASS\_COUNT – Sum of children’s pass count
   3. FAIL\_COUNT – Sum of children’s fail count
   4. SKIP\_COUNT – Sum of children’s skip count
   5. ERROR\_COUNT – Sum of children’s error count
   6. SCORE - ((SUM(input.PASS\_COUNT\*input.WEIGHTAGE\_DERIVED))/SUM((input.PASS\_COUNT+input.FAIL\_COUNT)\*input.WEIGHTAGE\_DERIVED))\*100.

Note - WEIGHTAGE\_DERIVED is NVL(1,WEIGHTAGE)

1. Insert the output in IGX\_DS\_DQ\_ROLLUP\_SCORE
2. Make a custom output, with next level and loop flag. It will help in stop loop or next call with next level.

### **IGX\_PRC\_Scoring\_Engine\_Log**

This analysis makes log entry in IGX\_DB\_SCORING\_ENGINE\_LOG.

Implementation Logic

1. *Insert the Log:*
   1. Insert a new record.
   2. Populate current date in run\_date, current timestamp in END\_TS and process\_id/work id in respective fields. Also set passed Max(IGX\_DS\_DQ\_RESULTS.RUN\_TIMESTAMP) as EVAL\_RUN\_TS and START\_TS as initialized start time done in process model.

### **IGX\_PRC\_FIND\_NEW\_FOR\_SCORE\_REFRESHER**

This analysis is responsible to find if anything new has been generated by scoring engine which requires posting to Govern.

Implementation Logic

* Read maximum of scoring\_run\_ts from igx\_ds\_score\_refersher\_log where end\_ts is not null
* Select maximum of run\_date from igx\_db\_scoring\_engine\_log where run\_date > the date derived in previous step where end\_ts is not null. Along with this record, output the current timestamp also
* If any record is found, insert a record in igx\_db\_score\_refresher\_log with version 1.
* Pass the retrieved current timestamp to next analysis via PM variable PM\_CURRENT\_TS
* Read value of parameter “SR\_SCORE\_POST\_DAYS” from IGX\_DB\_DG\_SCORING\_CONFIG and send to next analysis via PM variable PM\_SCORE\_POST\_DAYS

### **IGX\_PRC\_POST\_SCORES\_TO\_GOVERN**

This analysis is responsible to post the scores generated by scoring engine to Govern. It will source the data from igx\_ds\_dq\_rollup\_score.

* Read record from igx\_ds\_dq\_rollup\_score and assign them sequence no. based on asset\_uid order by run\_timestamp in descending.
* Filter records where sequence no. is <= PM\_SCORE\_POST\_DAYS
* For each AssetType UID, JSON to be generated containing the DQ results retrieved from above step. This JSON should contain the html formatting tags so that for each Asset UID DQ results when posted to Govern, gets visible in following tabular format. Each AssetType UID with the generated JSON needs to be passed to the following API to post the results to Govern

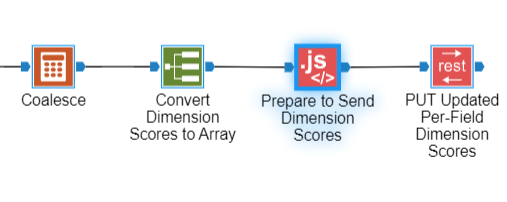
**https://<host>/api/v2/assets/batch/**${AssetTypeUid}

**Note:** Check if complete URL can be made dynamic and used the property of the node. If it is doable, configure the bold part of the URL in IGX\_DB\_DG\_SCORING\_CONFIG store

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Run Date | Version | DQ Score (%) | Passed Count | Failed Count | Skipped Count | Error Count | Total Count |
| 7-Sep-2019 | 2 | 93.5 | 720 | 50 | 210 | 20 | 1000 |
| 6-Sep-2019 | 1 | 92.5 | 620 | 50 | 210 | 20 | 900 |
| 5-Sep-2019 | 1 | 91 | 540 | 50 | 110 | 0 | 700 |

Refer the below analysis where JSON is getting generated however you will need to make necessary changes to have the above mentioned fields in the tabular format.

<https://cafe-modeloffice.infogix.com/desktop/index.html#analysis/67793>



Exception Handling

There is a possibility that the call to REST API might fail due to some reason in submitting the job.

Read the ExecutionID value from the response received from API for each Asset type id. If API calls fails, ExecutionID tag is not available in response. So while parsing JSON, put a try/catch block so that any JS exception could be caught.

Prepare the comma separated list of Asset type ids for which ExecutionID could not be read and populate it in a process model variable PM\_FAILED\_ASSET\_TYPE\_ID to pass on to next analysis.

Prepare the comma separated list of ExecutionIDs for which ExecutionID could be retrieved and populate it in a process model variable PM\_EXECUTION\_ID to pass on to next analysis.

### **IGX\_PRC\_MONITOR\_RESULT\_POSTING\_STATUS**

This analysis will call the below API to find the status of each execution id received in PM\_EXECUTION\_ID variable.

Split the comma separated list of execution IDs into records

Call below API for the each record i.e. execution id

Parse the response received from the API. Status response for each execution id would contain the status of multiple Asset ids.

Convert the response into multiple records having Execution id, Asset type uid, Asset uid and Status

Take out two branches

* One branch with in-progress status - Populate ‘Y’ to PM\_COMPLETED\_FLAG
* Second branch with failed status or Asset uid as a string of all zeros. If asset uid is all zeros, it represents failure in posting. It means that Govern doesn’t really tell the Asset uids which could not be posted in Govern
  + Convert the records into array using Grouping node
  + Parse the array and build a string of error message
  + Split the comma separated values into array and use it to build string of error message
  + Error message to be built like given below

DQ results of Asset Type UIDs <uid1>, <uid2>, <uid-n> could not be submitted. DQ results of one or more assets of Asset types UIDs <uid1>, <uid2>, <uid-n> could not be posted in Govern

* + Assign above Error string to PM\_ERROR\_MSG variable to pass on to next analysis

### **IGX\_PRC\_SCORE\_REFRESHER\_END\_LOG**

This analysis is responsible to make entry into the log store of the Score refresher to mark the process as end

Implementation Logic

* Read latest version of the records
* Find the record with maximum run\_date
* If end\_ts is null, create a new record with same values and populate current timestamp to end\_ts.
* Also populate the value of PM\_ERROR\_MSG to Error\_msg field

### **IGX\_PRC\_DS\_DQ\_HIERARCHICAL\_SCORE**

This analysis will read the data of incremental executions for Scoring Engine and populated the inflated hierarchical dataset in IGX\_DS\_DQ\_HIERARCHICAL\_SCORE datastore for dashboard purpose. Below is the logic for the analysis.

a) Read IGX\_DB\_DG\_SCORING\_CONFIG and filter where Code = ‘AT\_BUSINESS\_TERMS’

b) Read IGX\_DS\_DG\_HIERARCHY and join with IGX\_DB\_DG\_SCORING\_CONFIG(above) where ASSET\_TYPE\_NAME=IGX\_DB\_DG\_SCORING\_CONFIG.Description. (It will return the list of Business Terms) *- Call this join Output as 'Business Terms'*

c) Read incremental records from IGX\_DS\_DQ\_DETAIL\_SCORE (It gives the data by Rule, Entity and Element)

d) Join it with IGX\_DS\_DG\_HIERARCHY where DATA\_ELEMENT\_UID=ASSET\_UID and pick NEXT\_ASSET\_NAME. Left Join it again with 'Business Terms' where NEXT\_ASSET\_UID='Business Terms'.NEXT\_ASSET\_UID. Alias NEXT\_ASSET\_NAME as 'BUSINESS\_TERM' and NEXT\_ASSET\_UID as 'BUSINESS\_TERM\_UID'. Above steps will add the Business Terms in the output dataset. If the BUSINESS\_TERM is null change its value to ‘UnDefined’.

e) Read IGX\_DS\_DG\_HIERARCHY and filter NEXT\_ASSET\_CLASS='Model'. (It will return the list of Models) - *Call this filter Output as 'Models'*

f) Left Join above output of IGX\_DS\_DQ\_DETAIL\_SCORE (after adding business terms)

with ‘Models’ where BUSINESS\_TERM\_UID=ASSET\_UID and pick NEXT\_ASSET\_NAME. Alias NEXT\_ASSET\_NAME as 'MODEL\_NAME' and NEXT\_ASSET\_UID as 'MODEL\_UID' . Above steps will add the Model name in the output dataset. If the MODEL\_NAME is null change its value to ‘UnDefined’.

g) Read IGX\_DS\_DG\_HIERARCHY and filter NEXT\_ASSET\_CLASS='User'. (It will return the list of Owners) - *Call this filter Output as 'Owners'*

h) Left Join above output of IGX\_DS\_DQ\_DETAIL\_SCORE (after adding MODEL\_NAME)

with ‘Owners’ where MODEL\_UID=ASSET\_UID and pick NEXT\_ASSET\_NAME. Alias NEXT\_ASSET\_NAME as 'OWNER\_NAME' and NEXT\_ASSET\_UID as 'OWNER\_UID'. Above steps will add the Owner name in the output dataset. If the OWNER\_NAME is null change its value to ‘UnDefined’.

i) Join above output with IGX\_DS\_DG\_DATA\_ELEMENT\_RULES(latest record) , consider only active rules with not deleted flag, based upon RULE\_UID=RULE\_UID and bring DIMENSION(alias as RULE\_DIMENSION) to the output.

j) Insert the above final output to IGX\_DS\_DQ\_HIERARCHICAL\_SCORE after mapping.

### **IGX\_PRC\_DS\_Latest\_DS\_DQ\_Detail\_Score**

The Analysis will read IGX\_DS\_DQ\_DETAIL\_SCORE datastore for the latest run (based upon RUN\_TIMESTAMP) for each Key (DATA\_ENTITY\_NAME) and pick following additional columns

DATA\_ELEMENT\_NAME

RULE\_NAME

ROW\_COUNT

PASS\_COUNT

FAIL\_COUNT

SKIP\_COUNT

ERROR\_COUNT

SCORE

Output needs to be stored in IGX\_DS\_Latest\_DS\_DQ\_Detail\_Score. Prior to that the output store needs to be truncated.

### **IGX\_PRC\_DS\_Latest\_DS\_DQ\_Results**

The Analysis will be read IGX\_DS\_DQ\_RESULTS datastore for the latest run (based upon RUN\_TIMESTAMP) for each Key (DATA\_ENTITY\_NAME) and then again read the latest record version for the Key (DATA\_KEY, RUN\_TIMESTAMP, DATA\_ENTITY\_NAME, DATA\_ELEMENT\_NAME, DATA\_ELEMENT\_GROUP, DATA\_ELEMENT\_GROUP\_RULE\_TYPE, RULE\_NAME, DIMENSION, RULE\_RESULT, RULE\_INPUT\_VALUES, DATA\_BATCH\_IDENTIFIER) and read following additional columns

RULE\_RESULT\_TRANSFORMED

IGX\_SYSTEM\_RECORD\_VERSION

Output needs to be stored in IGX\_DS\_Latest\_DS\_DQ\_Results. Prior to that the output store needs to be truncated.

### **IGX\_PRC\_Find\_Ready\_Data\_Entities**

This analysis identifies the data sources or entities ready for execution of rule evaluator on current date based on the configuration available in igx\_db\_dg\_source\_config store.

Implementation Logic

1. Read igx\_db\_dg\_data\_entity\_config store
2. Filter records where either of the condition is true
   * Freq = ‘D’
   * Freq = ‘W’ and day of week of the current date is present in the comma separated list present in DAYS field.
   * Freq = ‘M’ and day part of the date (i.e. day of the month) of current date is present in the comma separated list present in DAYS field
3. Read latest run record of each data\_entity\_name from igx\_db\_rule\_evaluator\_log and filter the records where run\_date (excluding time) = current date (excluding time). Here latest run records are being read as there could be multiple runs for data entity in a day whose ingestion\_mode is ‘L’. Alias latest run\_date as latest\_run\_date
4. Left outer join step A with C based on data\_entity\_name
5. Filter records where latest\_run\_date is null OR ingestion\_mode = ‘L’ i.e. file types which are yet to be processed by rule evaluator or data entities for which data load log is available
6. Take out 3 branches, one for each value of ingestion\_mode i.e. ‘D’, ‘T’ and ‘L’
7. For ‘T’ branch, filter records where current time > the time mentioned in data\_load\_time and latest\_run\_date < the time mentioned in data\_load\_time considering current date. While comparing time in first condition, you may extract time in hh24:mi:ss format and do string comparison.
8. For ‘L’ branch – Find the latest load date/time from the log store. Join ‘L’ branch with load log record and filter those records where date/time in load log is > run\_date (leave it unimplemented)
9. Union records from ‘T’, ‘L’ and ‘D’ branch
10. Flatten the records and concatenate all values of CODE in comma separated string
11. Assign the comma separated string to process model variable PM\_DATA\_ENTITY\_LIST

### **IGX\_PRC\_Log\_Rule\_Evaluator**

This analysis logs the execution of rule evaluator for each source after completion of the process.

Implementation Logic

* Take PM\_RUN\_DATE, PM\_DATA\_ENTITY\_NAME as input from the process model
* Insert record in igx\_db\_rule\_evaluator\_log store with respective values. PM\_RUN\_DATE will be populated in both run\_date and start\_ts fields.

### **IGX\_PRC\_Scoring\_Engine\_Execution\_Check**

This analysis checks if rule evaluator has processed all the data entities for the day before proceeding to Scoring Engine.

Implementation Logic

* Check if there is any data entity pending for execution by Rule Evaluator
  + Read igx\_db\_dg\_data\_entity\_config store
  + Filter records where either of the condition is true
    - Freq = ‘D’
    - Freq = ‘W’ and day of week of the current date is present in the comma separated list present in DAYS field.
    - Freq = ‘M’ and day part of the date (i.e. day of the month) of current date is present in the comma separated list present in DAYS field
  + Read igx\_db\_rule\_evaluator\_log and filter the records where run\_date (excluding time) = current date (excluding time)
  + Left outer join step B with C
  + Count records where run\_date is null i.e. rule evaluator is pending
  + Count records where run\_date is null and ingestion\_mode = ‘L’ and current time > the time mentioned in data\_load\_time
  + If count of records in step E is 0 or count in step E and F is same, assign ‘Y’ to PM\_SE\_EXEC\_FLAG else ‘N’

### **IGX\_PRC\_Rule\_and\_Data\_Element\_Version**

This analysis will read Rule and Data Element Assets from Hierarchy and create their version in Version datastore considering any definition or relationship change.

Implementation Logic

* Read the IGX\_DS\_DG\_HIERARCHY store and apply filter ASSET\_TYPE\_NAME ='IGX Data Element' && NEXT\_ASSET\_CLASS ='Rule'
* For Rule versioning, following conditions needs to be checked for each NEXT\_ASSET\_UID. Get the CreateDate and UpdateDate value for **each** Rule(NEXT\_ASSET\_UID)
  + **If** CreateDate or Update Date is greater than previous value in Versions or Rule was not existing previously,
  + **Then** Insert an entry in IGX\_DS\_ASSET\_VERSION\_TEMP store with Version number incremented for the Rule by 1.
  + **ElseIf** the relationship to Dataelements for the rule is changed(added/deleted) i.e. list of related DataElements are different from the list of related DataElements in PreviousHierarchy store
  + **Then** Insert an entry in IGX\_DS\_ASSET\_VERSION\_TEMP store with Version number incremented for the Rule by 1.
  + **Else** Insert an entry in IGX\_DS\_ASSET\_VERSION\_TEMP store with Version number same as previous for the Rule.

\*Note : Set the value of Level as 0(zero).

* For Data Elements, following conditions needs to be checked for each DataElement(ASSET\_UID).
  + **If** the list of related rules are different from the list of related rules in PreviousHierarchy(IGX\_DS\_DG\_PREVIOUS\_HIERARCHY) store or Related rules are Updated since last execution or DataElement was not existing previously
  + **Then** Insert an entry in IGX\_DS\_ASSET\_VERSION\_TEMP store with Version number incremented for the DataElement by 1.
  + **Else** Insert an entry in IGX\_DS\_ASSET\_VERSION\_TEMP store with Version number same as previous for the DataElement

\*Note : Set the value of Level as 1.

### **IGX\_PRC\_RolledUp\_Asset\_Version**

This analysis will rollup the version to higher level assets. It will be iteratively called in process model(one level at a time).

Implementation Logic

* Identify the Assets, in ASSET\_LEVEL\_TEMP store for the passed in PM\_NEXT\_HIERARCHY\_LEVEL(process model variable).
* Search above Assets in IGX\_DS\_DG\_HIERARCHY store as NextAsset and get all their Assets(to which they were related to). For Each NextAsset do following
* Compare the version between IGX\_DS\_ASSET\_VERSION datastore and IGX\_DS\_ASSET\_VERSION\_TEMP datastore for each of its related Asset.
  + **If** NextAsset was not existing in IGX\_DS\_ASSET\_VERSION, or Version for any Asset is higher in IGX\_DS\_ASSET\_VERSION\_TEMP datastore, or list of related Assets in IGX\_DS\_DG\_HIERARCHY are different from the list of related Assets in IGX\_DS\_DG\_PREVIOUS\_HIERARCHY store for this NextAsset,
  + **Then** Insert an entry in IGX\_DS\_ASSET\_VERSION\_TEMP store with Version number incremented for the item by 1.
  + **Else** Insert an entry in IGX\_DS\_ASSET\_VERSION\_TEMP store with Version number same as previous for the item

### **IGX\_PRC\_Copy\_Hierarchy\_and\_Version**

This analysis will copy Hierarchy to PreviousHierarchy and Asset Version from Temp to Asset Version stores.

Implementation Logic

* Copy IGX\_DS\_DG\_HIERARCHY to IGX\_DS\_DG\_PREVIOUS\_HIERARCHY
* Copy IGX\_DS\_ASSET\_VERSION\_TEMP to IGX\_DS\_ASSET\_VERSION

### **IGX\_PRC\_Rule\_Refresher\_Execution\_Check**

This execution check analysis is for Rule Refresher which will help in restrict running Rule Refresher only once for a day

Impelementation Logic

* Read igx\_DB\_Rule\_Refresher\_Log store
* Apply a filter on run\_date equal to current date. While comparing, ignore the time part stored in run\_date.
* If any record found, assign ‘N’ to PM\_RR\_EXEC\_FLAG otherwise ‘Y’

### **IGX\_PRC\_Scoring\_Execution\_Check**

This execution check analysis is for the grand process model and will be called at its beginning. It will check if Scoring Engine has already ran for the day. If it has run, it won’t proceed and come out.

Implementation Logic

* Select run\_date from igx\_db\_scoring\_engine\_log where run\_date = PM\_RUN\_DATE (without considering time).
* If record found, assign ‘Y’ to PM\_EXEC\_FLAG else ‘N’



## Data Views

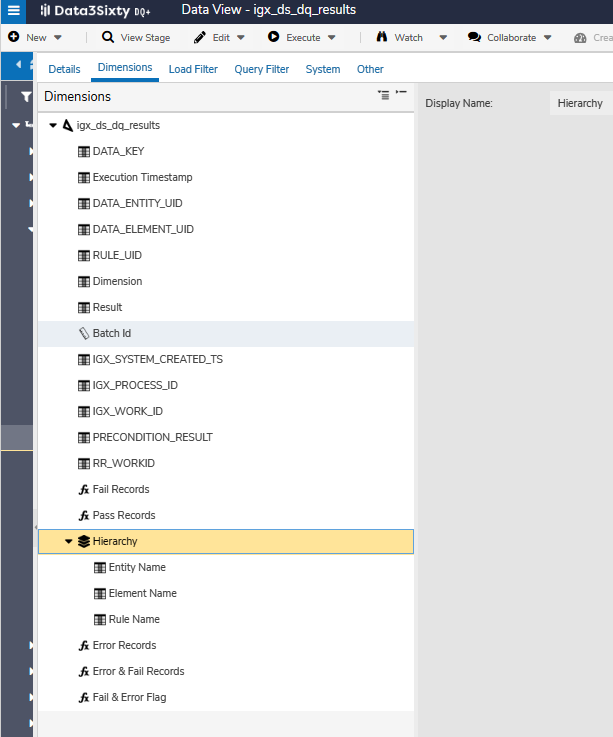
**IGX\_DV\_Latest\_DS\_DQ\_Detail\_Score**

It will be full refresh Dataview. The dataview will be built by reading IGX\_DS\_ Latest\_DS\_DQ\_Detail\_Score.

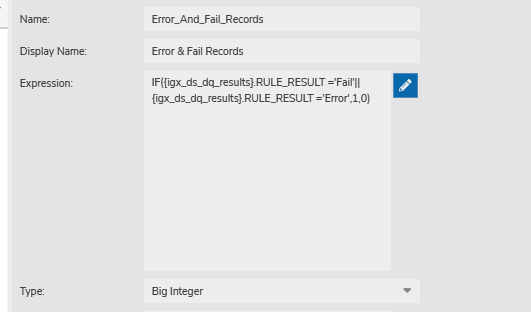
**IGX\_DV\_Latest\_DS\_DQ\_Results**

It will be full refresh Dataview. The dataview will be built by reading IGX\_DS\_Latest\_DS\_DQ\_Results

* 1. Add Drill down field to drill by Entity 🡪 Element 🡪 Rule



* 1. Add computed fields named
     1. Fail and Error Records to store counter as 1 for records that have RULE\_RESULT as fail or Error.



* + 1. Fail & Error Flag store composite flag for error and fail records in single field



**IGX\_DV\_DS\_DQ\_ROLLUP\_SCORE**

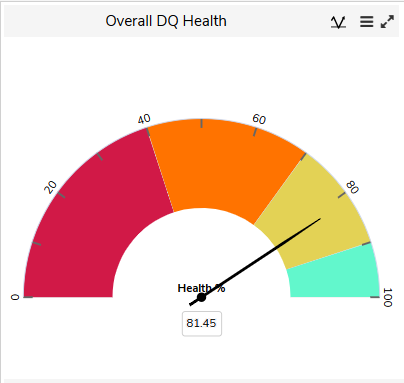
This DataView will be incrementally refreshed. The retention of the dataview should be 1 Full Month. The Dataview will be built by reading IGX\_DS\_DQ\_ROLLUP\_SCORE for the incremental records.

**IGX\_DV\_DQ\_HISTORICAL\_HIERARCHICAL\_SCORE**

This DataView will be incrementally refreshed. The retention of the dataview should be 1 Full Month. The Dataview will be built by reading IGX\_DS\_DQ\_HIERARCHICAL\_SCORE for the incremental records.

## Dashboards

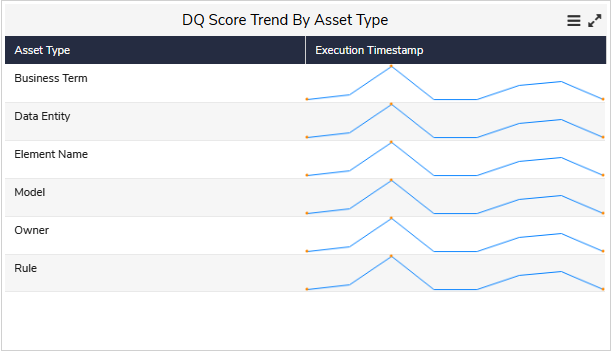
### **Overall DQ Health**



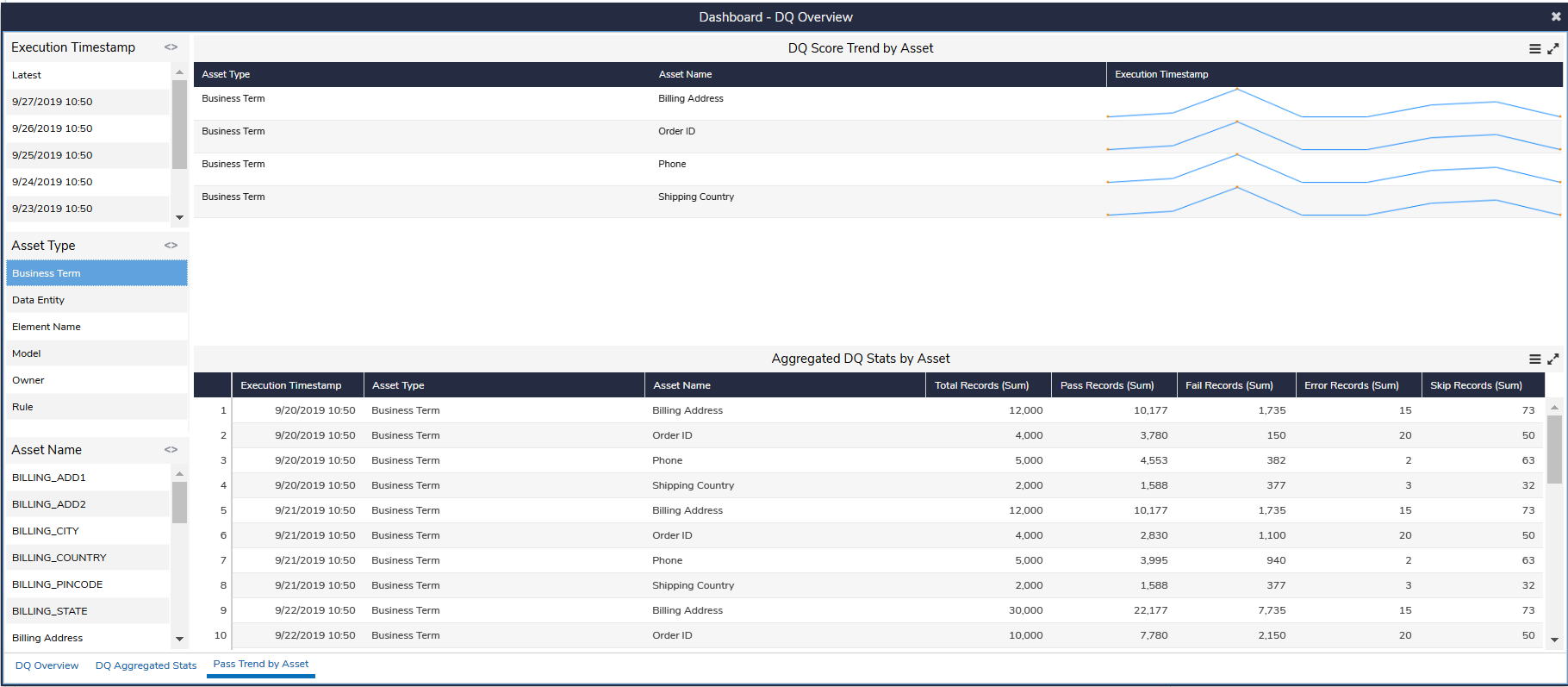
The widget will show the overall DQ score for the entire system. The score will be computed as simple average of scores (for their latest DQ runs) for all Data Entities. The widget will show a gauge on scale of 0 to 100. 0 to 40 will be shown as Red, 40 to 70 will be Orange, 70 to 90 will be light green and more than 90 will be Dark Green. The widget will make use of DataView **IGX\_DV\_Latest\_DS\_DQ\_Detail\_Score.** Formula to be used for computing the score should be

Score = (Sum of Pass \* 100) / (Sum of Pass and Fail)

### **Score Trend by Asset Type**

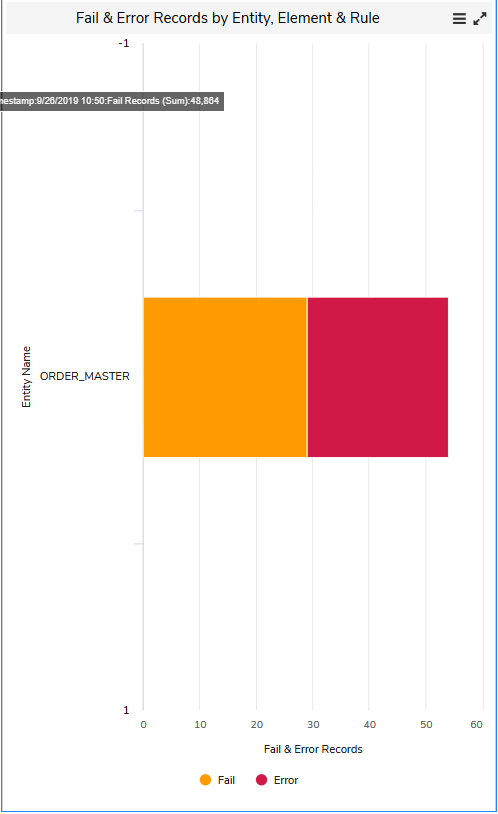


The widget will show the Score trend(last 30 days) of different Asset Types. It will be a linked widget, if clicked will open another tab with more detailed options as shown in below screenshot.

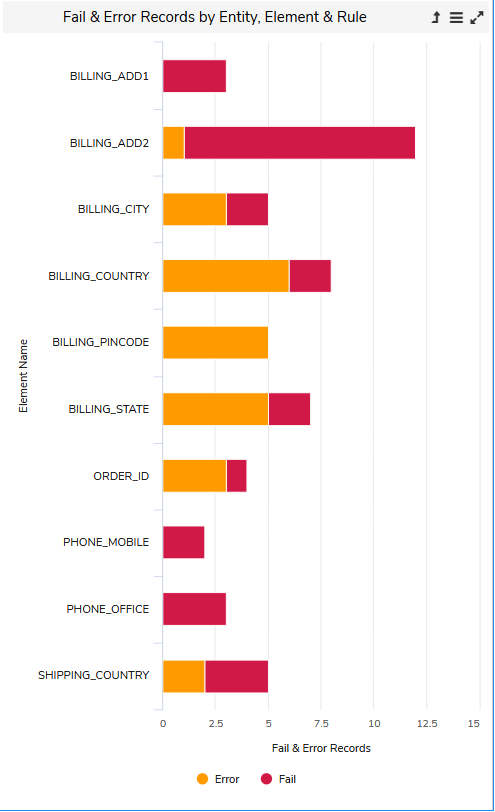


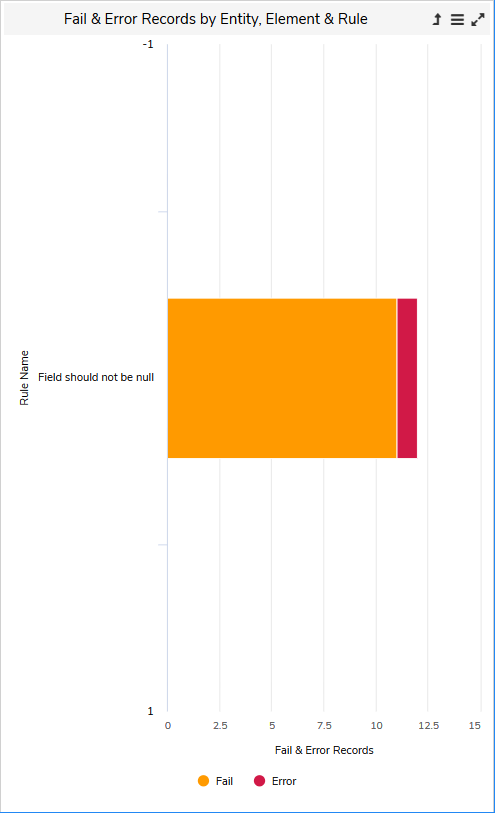
The detailed view will allow user to filter by Execution Timestamp, Asset Type and Asset Name. The upper view will show the graphical trend for the Assets qualified as per filter criteria and the bottom part will show the stats for Total Records, Pass, Fail, Error and Skip record counts. The widgets will make use of DataView **IGX\_DV\_DS\_DQ\_ROLLUP\_SCORE**.

### **Failed and Error Records by Data Entity, Element & Rule**



The widget (stacked column) will show count of records, grouped by Dimension, Failed or Errored in data quality rule validation in the last run for each Data Entity. At starting level the counts will be shown by Data Entity. User can drill down the counts by Data Element and then by Rule, as shown in below screenshots, to investigate the root cause of failure. The widget will make use of DataView **IGX\_DV\_Latest\_DS\_DQ\_Detail\_Score**.





On click on Rule Name in above chart, a tabular listing will show the detailed records of the source, for which the rule evaluation resulted as Failed or Error. This table will be built using DataView **IGX\_DV\_Latest\_DS\_DQ\_Results** . The table will list following columns.

Record Key (DATA\_KEY)

Run Timestamp

Data Entity Name

Data Element Name

Group

Group Rule

Rule Name

Rule Dimension

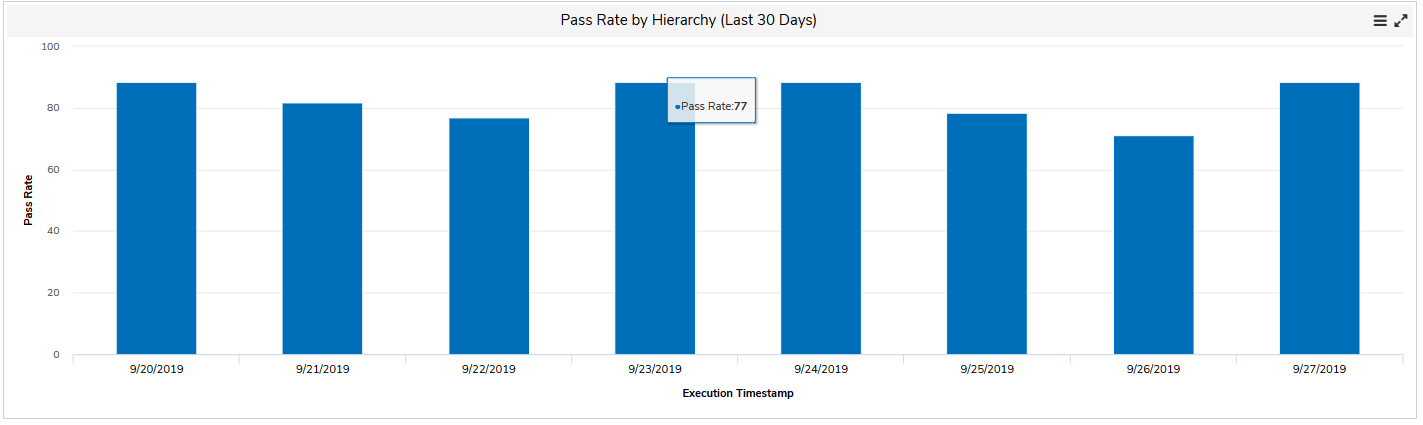
Rule Result

Transformed Rule Result

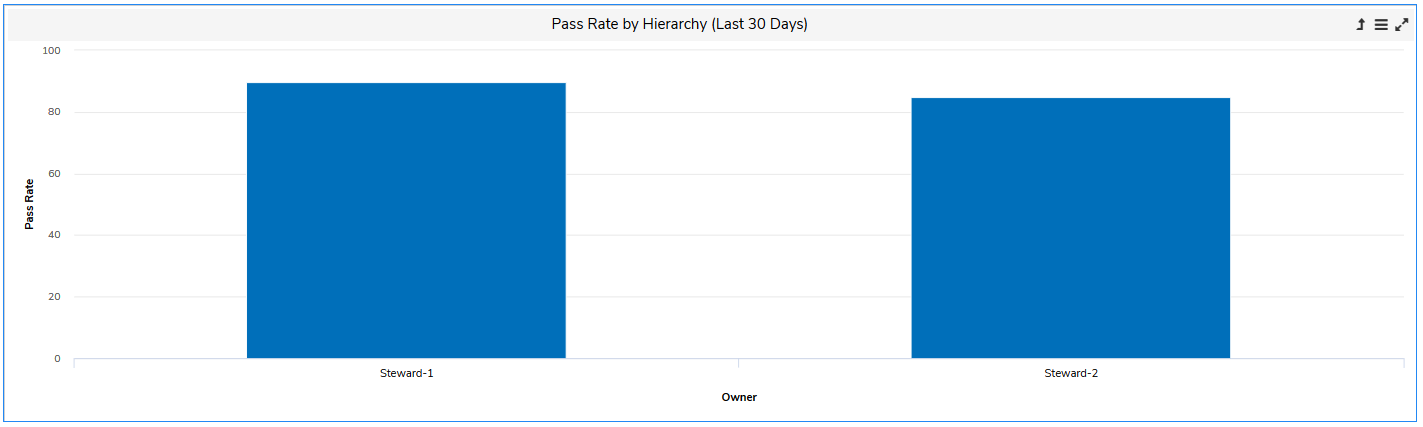
Fail Description

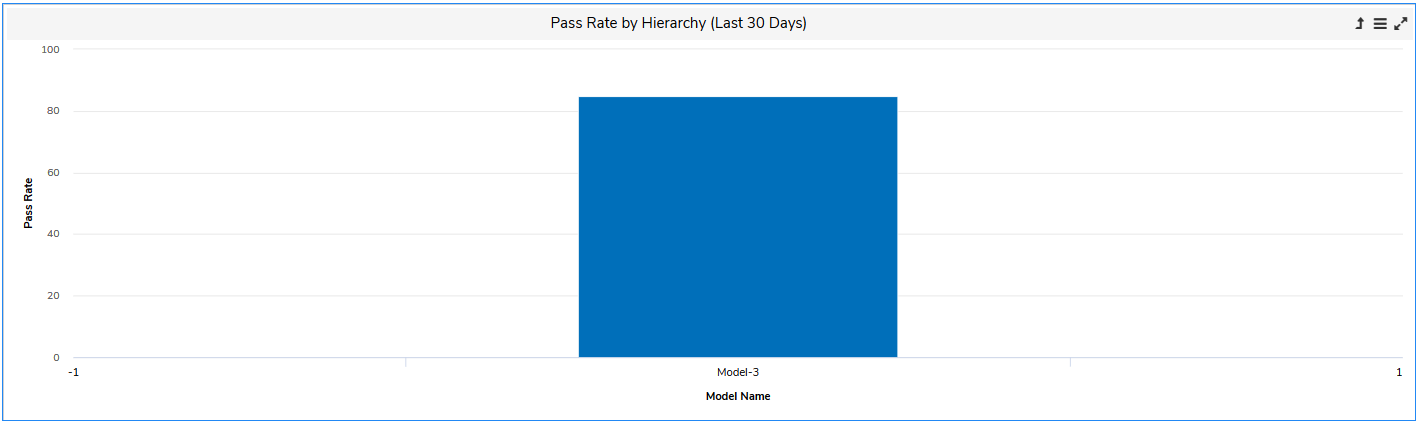
Data Batch Identifier

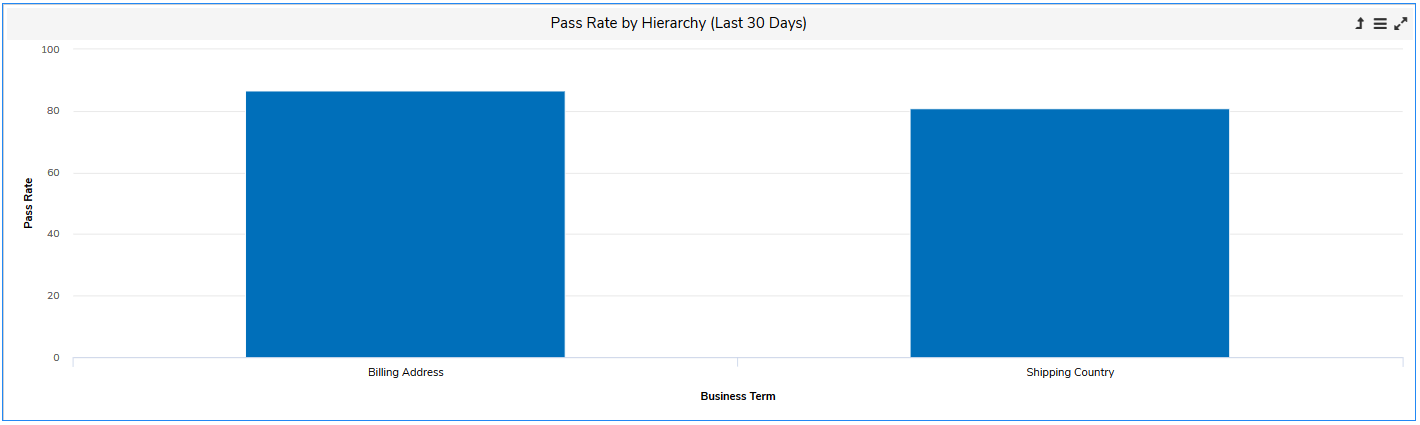
### **Pass Rate by Hierarchy (Last 30 Days)**

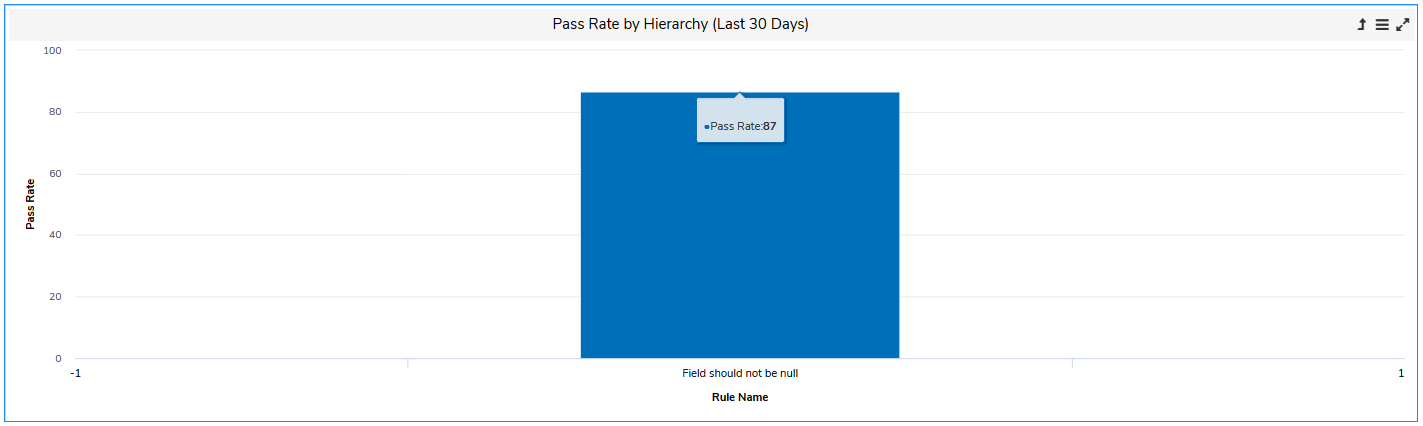


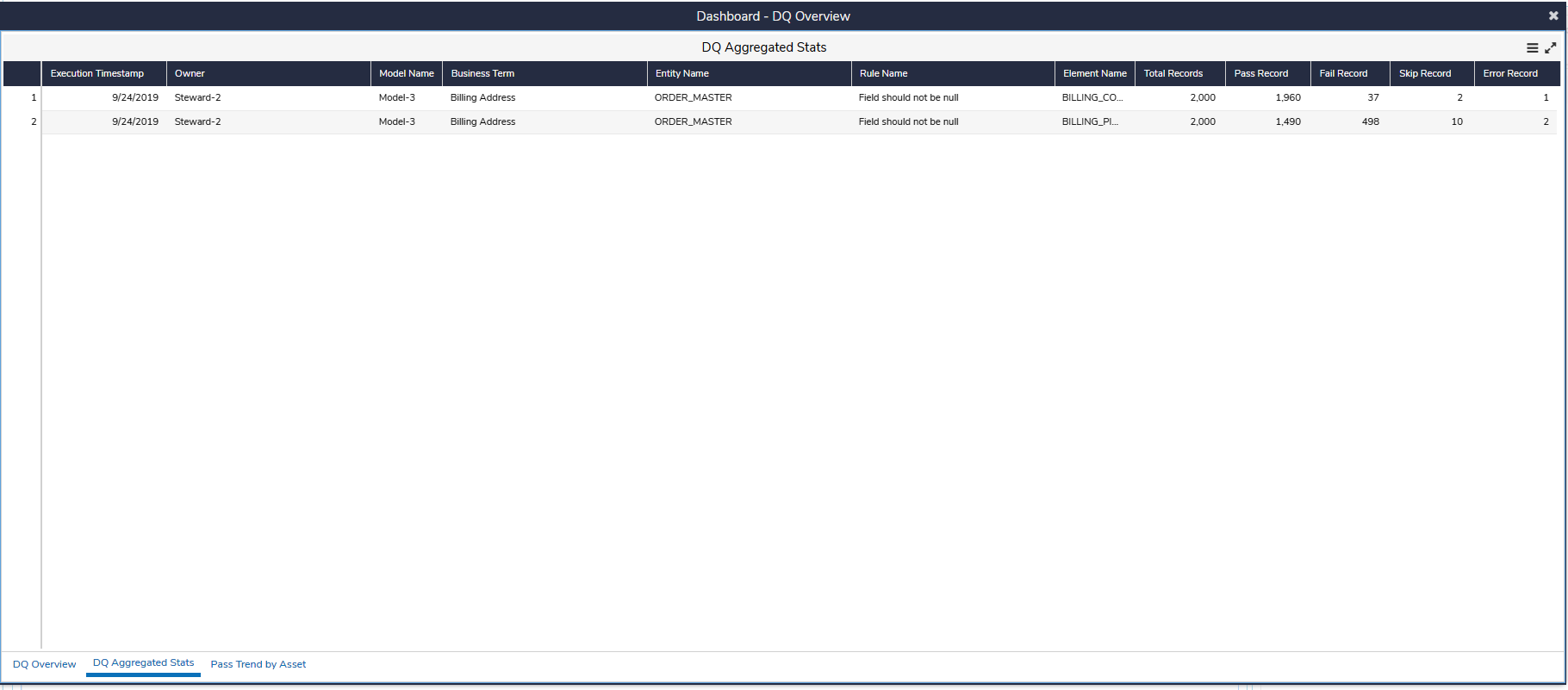
The widget will show the passing rate of different DQ evaluations executed in last 30 days. At starting level the values will be shown by Execution Date. User can drill down it further by Owner to Model to Business Terms to Rule and then a detailed tabular list will appear showing the pass, fail, error and skip counts. The widgets will make use of DataView IGX\_DV\_DQ\_HISTORICAL\_HIERARCHICAL\_SCORE.











## Case Management

### **Case Summary Workflow**

< Define workflow of a case through a diagram showing the associated detail error DB store>

<Define the Case Statuses vis-à-vis D3S workflow Statuses in below table. e.g.>

|  |  |
| --- | --- |
| **Case Status** | **Workflow State** |
| Open | wf-state |

### **Case Summary Search Screen**

<Define the Case Summary Search screen based on functional requirement>

### **Case Summary Edit/View Screen**

<Define the Case Summary Edit/View screen based on functional requirement>

### **Case Detail Search Screen**

<Define the Case Detail Search screen based on functional requirement>

### **Case Detail Edit/View Screen**

<Define the Case Detail Edit/View screen based on functional requirement>

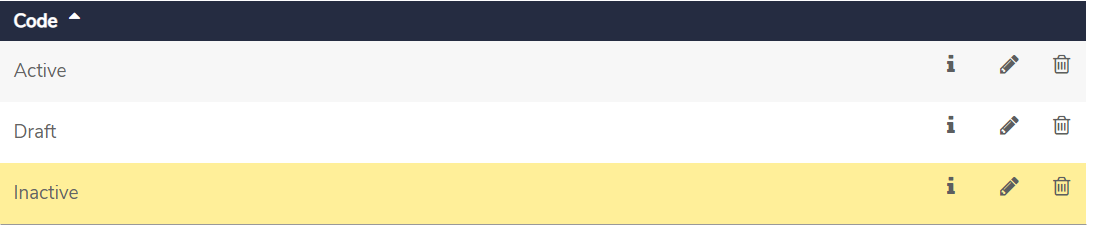
# **Configuration Data / Initial Data**



## Reference Lists

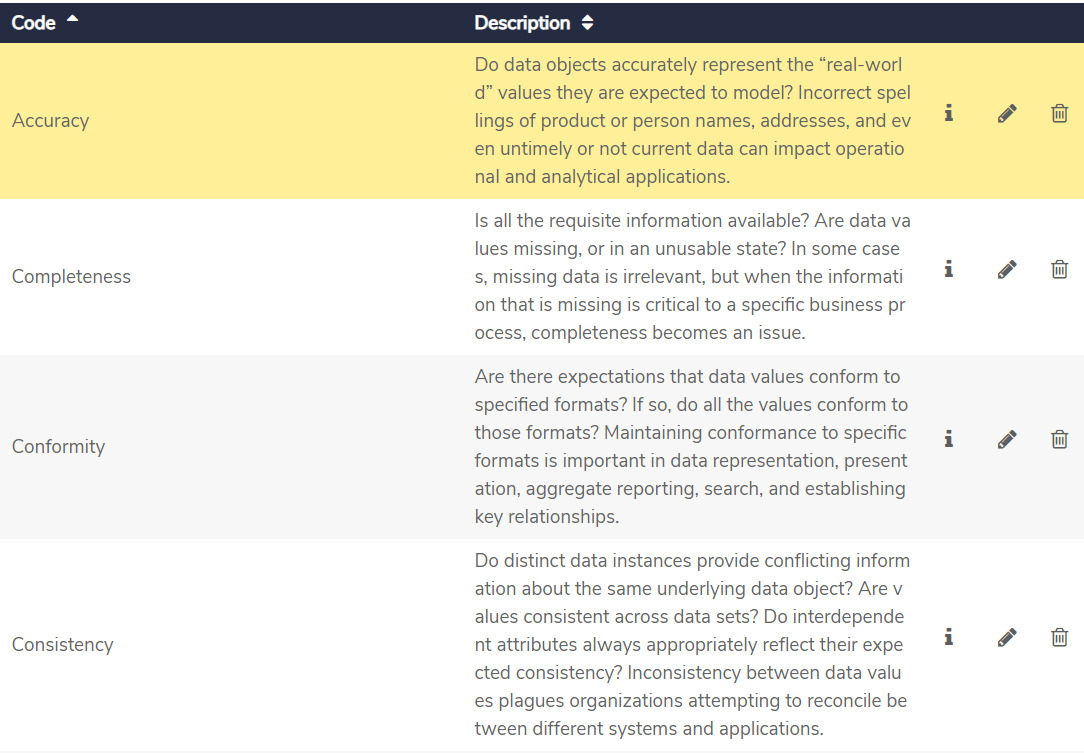
### **Rule Status**

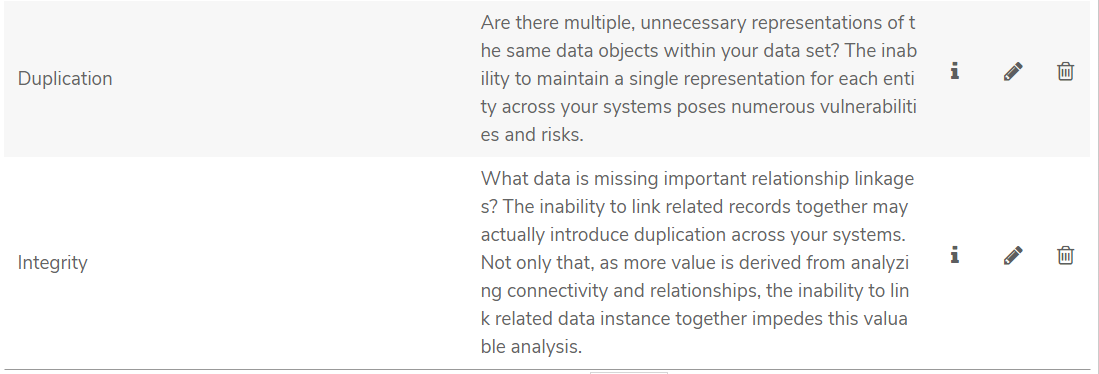
Define rule status if not already defined as below:



### **Rule Dimension**

Define rule dimension if not already defined as below:





### **IGX Data Element Group**

Define Data Element Group used for Multiple data elements of same type. By default, for Phone and Email, AtLeastOneMustPass logic is configured. In case different logic needs to be applied e.g. AllMustPass then separate code needs to be created e.g. Phone2 and in the Type of Check “AllMustPass” should be mentioned. For nested kind of example, Address configuration is provided by default.

|  |  |  |
| --- | --- | --- |
| **Code** | **Type of Check** | **Parent** |
| Phone | AtLeastOneMustPass |  |
| Email | AtLeastOneMustPass |  |
| Address | AtLeastOneMustPass |  |
| Home Address | AllMustPass | Address |
| Office Address | AllMustPass | Address |
| Billing Address | AllMustPass | Address |

### **IGX Scoring Config**

This is a new reference list to be created in Govern. It will act as a configuration store for DQ scoring component. It will mainly contain the entries to identify specific Asset types defined in the project along with few other configurations. PR entry will identify what all predicates participate while building the hierarchy. Entries needs to be configured based on implementation. Below 2 entries in PR\_ PARTICIPANT\_PREDICATES are sample ones only which are comma separated.

|  |  |
| --- | --- |
| **Code** | **Description** |
| RE\_RULE\_NOT\_BLANK | IGX Not Blank |
| RE\_RULE\_REGEX | IGX Regex |
| RE\_RULE\_NUMERIC\_RANGE | IGX Numeric Range |
| RE\_RULE\_DUPLICATE\_CHECK | IGX Duplicate Check |
| RE\_RULE\_REFERENCE\_LOOKUP | IGX Reference Lookup |
| RE\_RULE\_CONSISTENCY | IGX Consistency Check |
| RE\_ATTRIBUTE\_COMPARISON | IGX Attribute Comparison |
| RE\_CONFORMITY\_CHECKS | IGX Conformity Check |
| AT\_DATA\_ENTITY | IGX Data Entity |
| AT\_DATA\_ELEMENT | IGX Data Element |
| PE\_DATA\_ENTITY\_TO\_DATA\_ELEMENT\_PREDICATE\_UID | Actual UID of “IGX Has Field” |
| PE\_DATA\_ELEMENT\_TO\_RULE\_PREDICATE\_UID | Actual UID of “IGX Rules Applies to Field” |
| RE\_RULE\_JSEXP | IGX JS Expression |
| PR\_RULE\_TO\_LOWEST\_LEVEL\_PREDICATE | IGX Rules Applies to Field |
| PR\_HIERARCHY\_PARTICIPANT\_PREDICATES | IGX Has Field |
| SR\_SCORE\_POST\_DAYS | No. of days/executions of scores to be posted to Govern |
| AT\_BUSINESS\_TERM | Name of the Asset Type defined for Business Term e.g. IGX Business Term |

### **IGX Rule Operator List**

|  |  |
| --- | --- |
| **Code** | **Description** |
| Contains | Contains |
| Does Not Contain | Does Not Contain |
| Does Not Match Regular Expression | Does Not Match Regular Expression |
| Ends With | Ends With |
| Is Equal To | Is Equal To |
| Is Greater Than | Is Greater Than |
| Is Greater Than or Equals To | Is Greater Than or Equals To |
| Is In Delimited List | Is In Delimited List |
| Is Less Than | Is Less Than |
| Is Less Than or Equal To | Is Less Than or Equal To |
| Is Not Equal To | Is Not Equal To |
| Is Not In Delimited List | Is Not In Delimited List |
| Is Not Null/Blank | Is Not Null/Blank |
| Is Null/Blank | Is Null/Blank |
| Matches Regular Expression | Matches Regular Expression |
| Starts With | Starts With |

### **IGX Comparison Rule Operator List**

|  |  |
| --- | --- |
| **Code** | **Description** |
| Is Equal To | Is Equal To |
| Is Greater Than | Is Greater Than |
| Is Greater Than or Equals To | Is Greater Than or Equals To |
| Is Less Than | Is Less Than |
| Is Less Than or Equal To | Is Less Than or Equal To |
| Is Not Equal To | Is Not Equal To |



### **IGX Pass Skip List**

|  |  |
| --- | --- |
| **Code** | **Description** |
| Pass | Pass |
| Skip | Skip |

### **IGX Data Entity Config**

This reference list will contain the configuration of source or Data Entity on which DQ rules will be applied. Reference list will have following structure

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| CODE | Name or short name of the source or data entity |
| INGESTION\_MODE | D – Direct i.e. Source is always up to date  C – Loaded through process and log is available to indicate the load  T – Time based loading; no log of data load is available |
| FREQ | D – Daily  W – Weekly  M – Monthly |
| DAYS | It is the days on which file is expected e.g.  If Freq is ‘W’  1,3,5 – represents Monday, Wednesday and Friday  If Freq is ‘M  10,20 – 10th and 20th of the month |
| DATA\_LOAD\_TIME | Time by which data is assumed to have got loaded  Format: HH24:MI |

## Artifacts

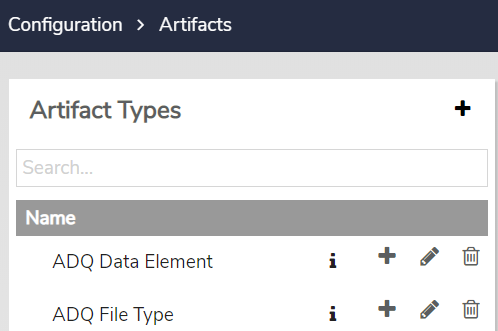


### **IGX Data entity**

Define Artifact “IGX Data entity” in Configuration section under “Artifacts” and define the fields later in order to build relationships.

### **IGX Data Element**

Define Artifact “IGX Data Element” in Configuration section under “Artifacts” and define the fields later in order to build relationships.



Example above - remember to change ADQ to IGX.



## Predicates

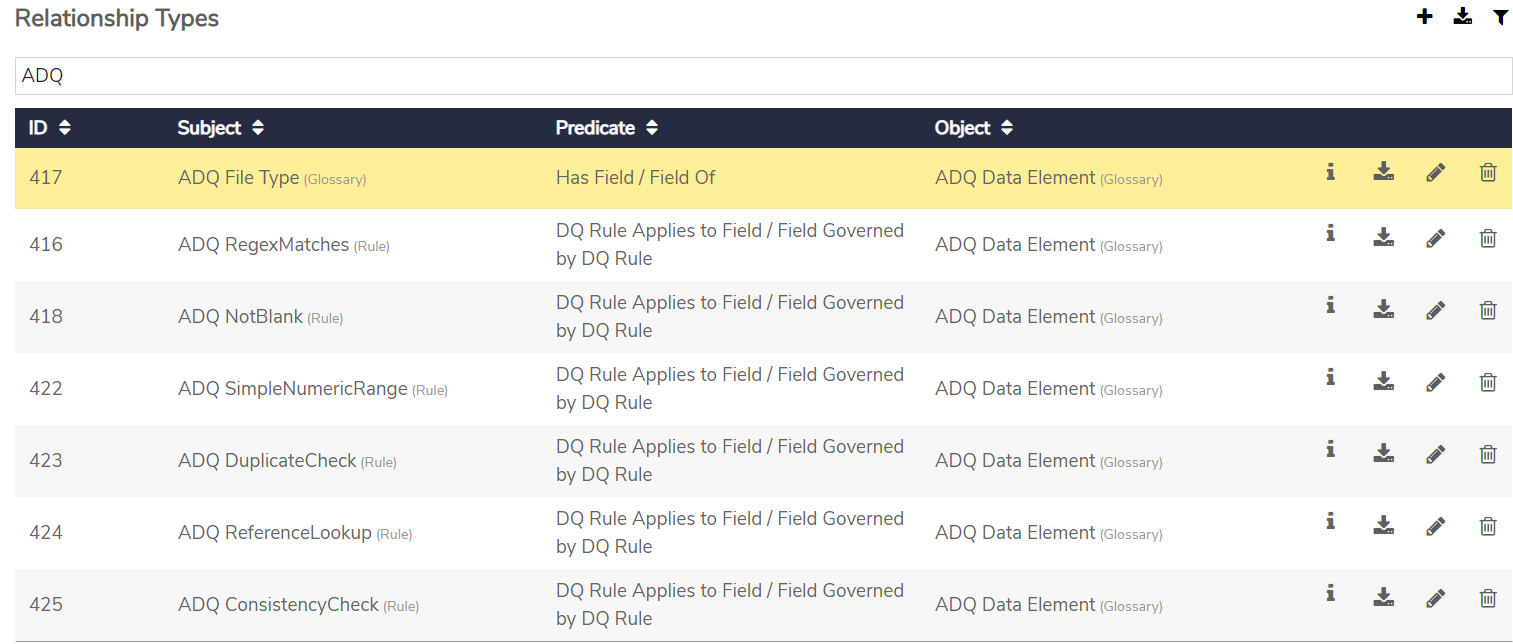
Define a predicate in Configuration section of Govern.

|  |  |  |
| --- | --- | --- |
| **Name** | **Inverse** | **Functional Type** |
| IGX Rules Applies to Field | Field Governed by IGX Rule | Simple |
| IGX Has Field | Field Of | Simple |



## Relationship Type

Relationship types can be defined in Configuration section of Govern between above Rule Types/Object and Predicate as below (remember to change ADQ to IGX and changed name wherever applicable) -

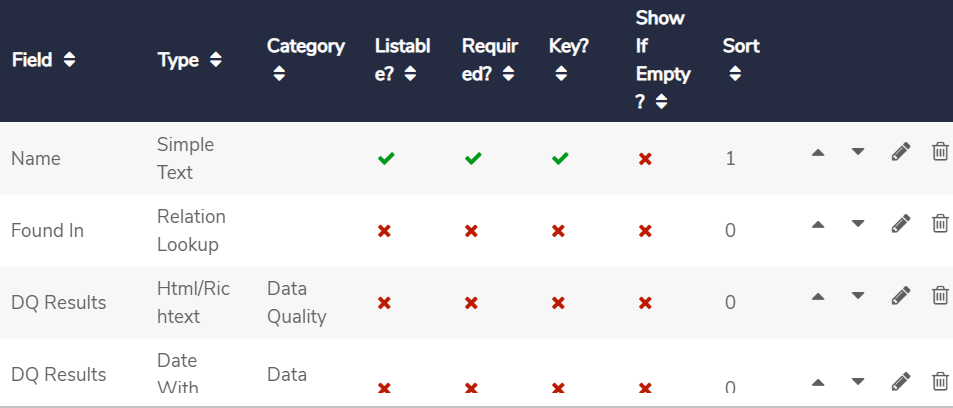


## Artifact Fields

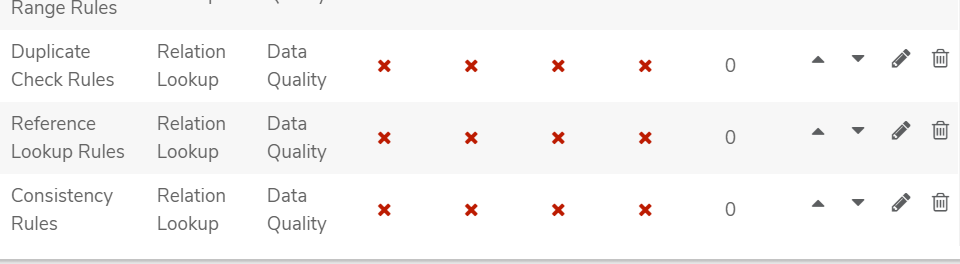


### **IGX Data Element**

Add fields to Artifact “IGX Data Element” in Configuration section as below:







Additional fields – DataElementGroup of List type and should be Listable as well. Associate “IGX Data Element Group” reference list with DataElementGroup column.

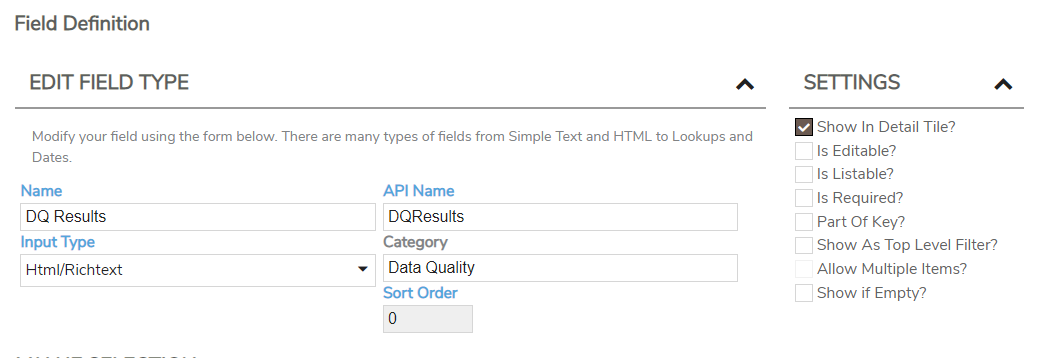
### **IGX Data Entity**

Add fields to Artifact “IGX Data Entity” in Configuration section as below:



### **Field to Capture DQ Result**

Add following field in the definition of various Artefacts, Models, Policies and Rules where roll-up of score is required to be posted. Rule Refresher posts the DQ results in below named HTML field

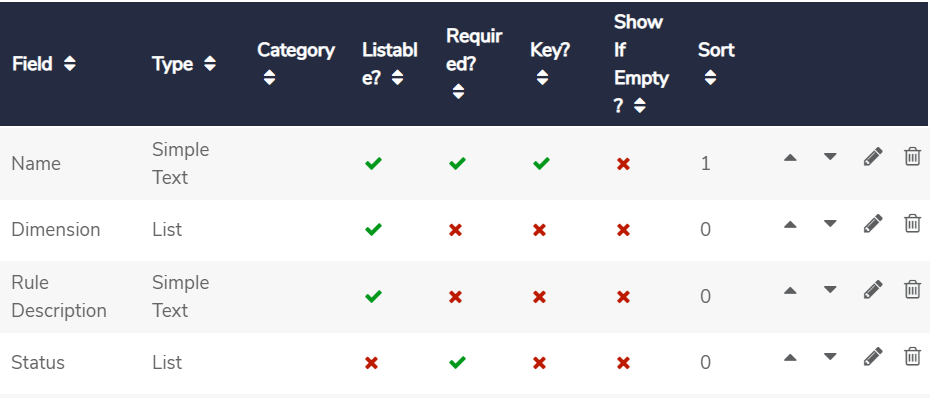


## Rule Types

Rule types can be defined in Configuration section of Govern as below-



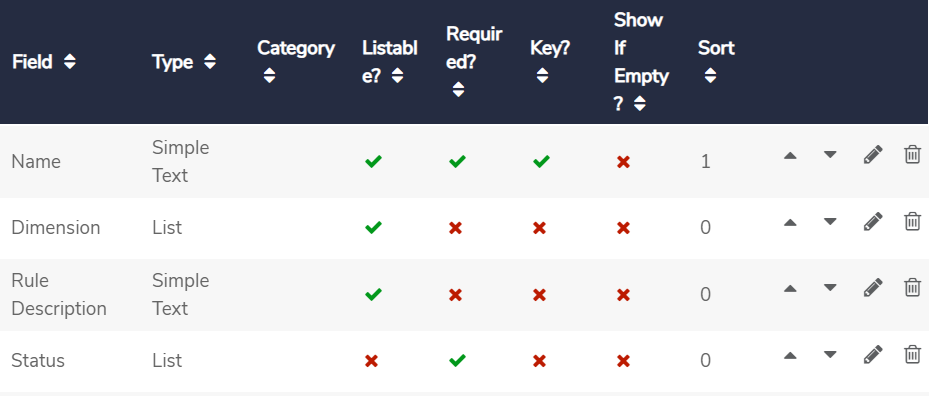
### **IGX Consistency Check**



New fields –

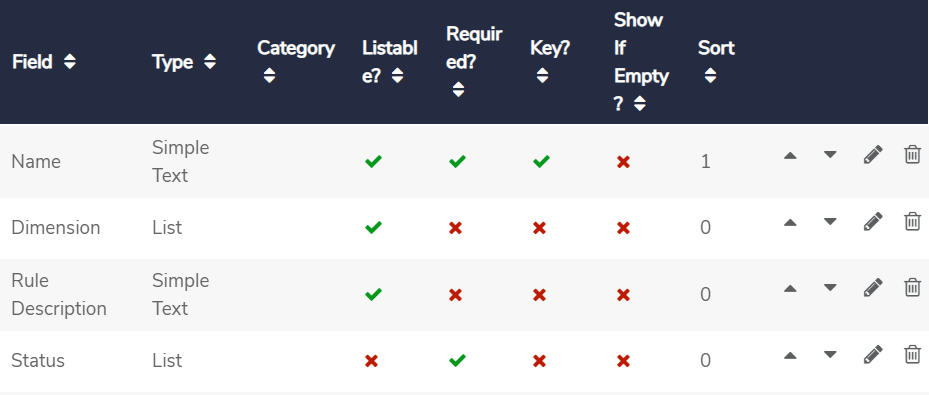
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Type of List** | **Listable** | **Required** |
| Pre-Condition Attribute | List | Artifact: IGX Data Element | Yes | Yes |
| Pre-Condition Operator | List | Reference List Item: IGX Rule Operator List | Yes | Yes |
| Pre-Condition Value | Simple Text |  | Yes | Yes |
| Pre-Condition Value Case Sensitive | True/False |  | Yes | Yes |
| Rule Operator | List | Reference List Item: IGX Rule Operator List | Yes | Yes |
| Rule Value | Simple Text |  | Yes | Yes |
| Rule Value Case Sensitive | True/False |  | Yes | Yes |

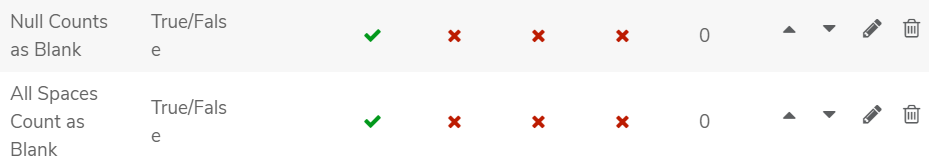
### **IGX Duplicate Check**

Rename field – “Pass if Blank or Null” as “Treat Blank or Null As” of List type having Reference List Item as “IGX Pass Skip List”.

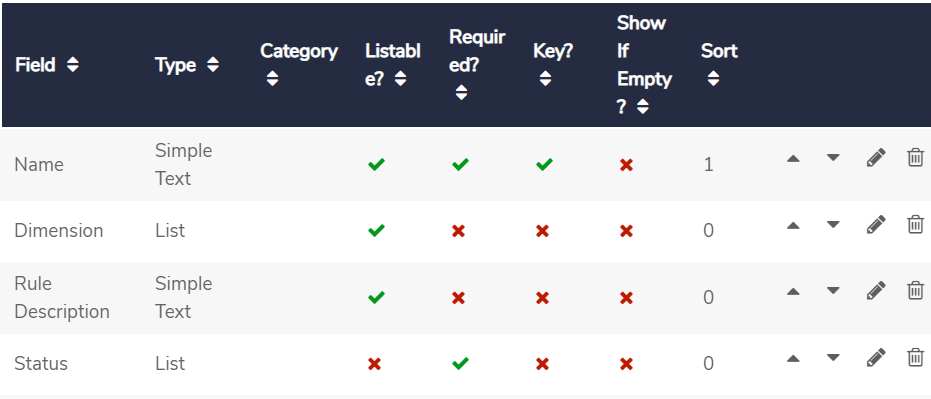
### **IGX Not Blank**

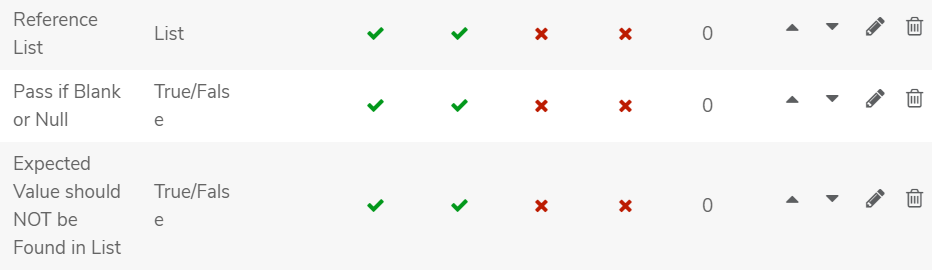




Rename fields – “Null Counts as Blank” as “Consider Null As Blank” and “All Spaces Count as Blank” as “Consider All Spaces As Blank”

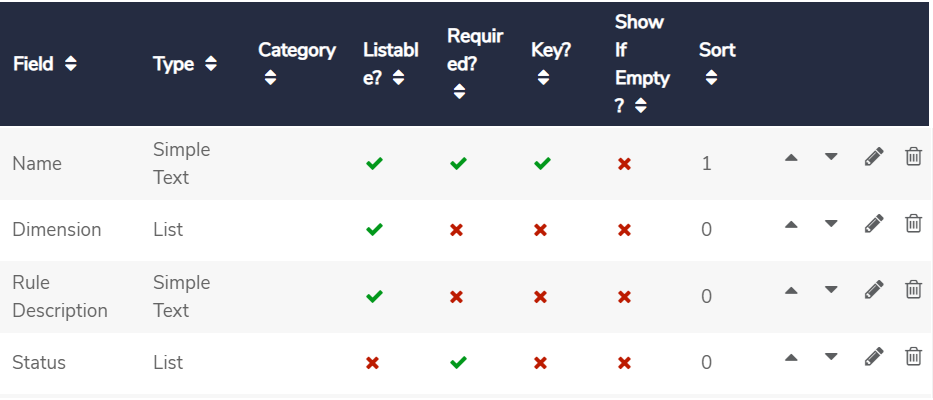
### **IGX Reference Lookup**

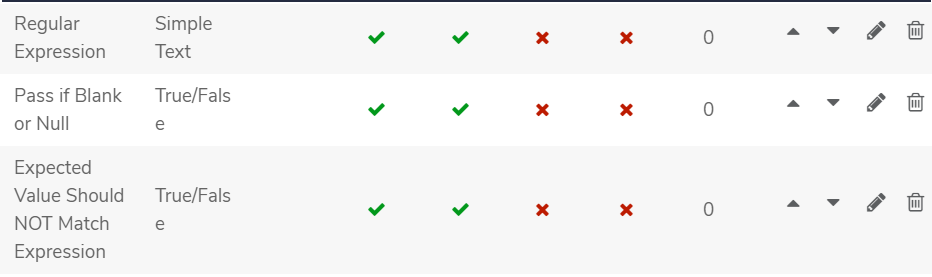




Rename fields – “Expected Value should NOT be Found in List” to “Match Reference List” and “Pass if Blank or Null” as “Treat Blank or Null As” of List type having Reference List Item as “IGX Pass Skip List”.

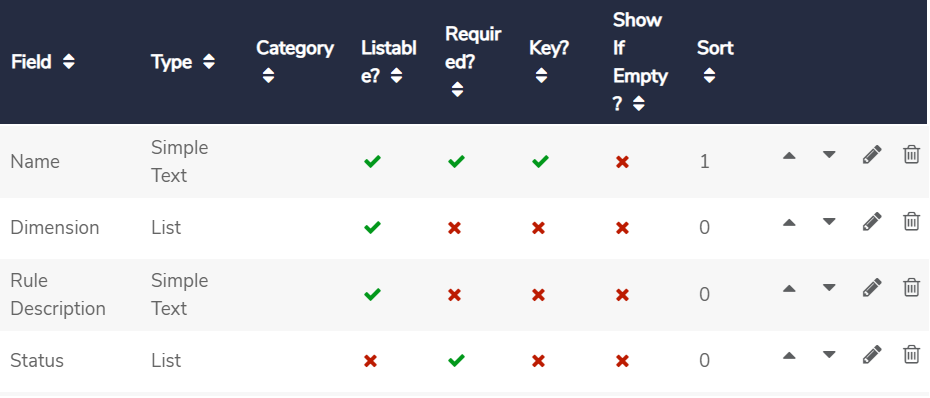
### **IGX Regex**

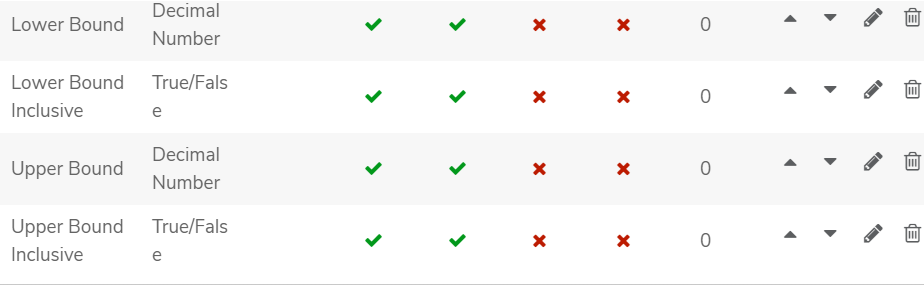


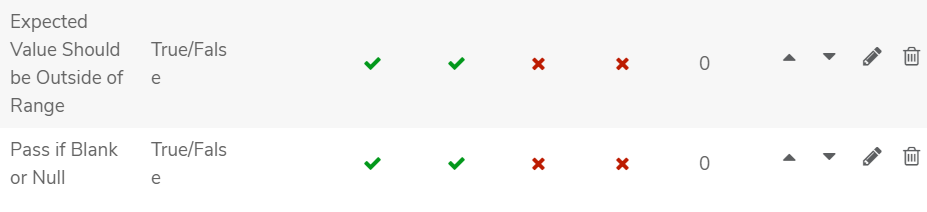


Rename fields – “Expected Value Should NOT Match Expression” to “Match Regular Expression” and “Pass if Blank or Null” as “Treat Blank or Null As” of List type having Reference List Item as “IGX Pass Skip List”.

### **IGX Numeric Range**

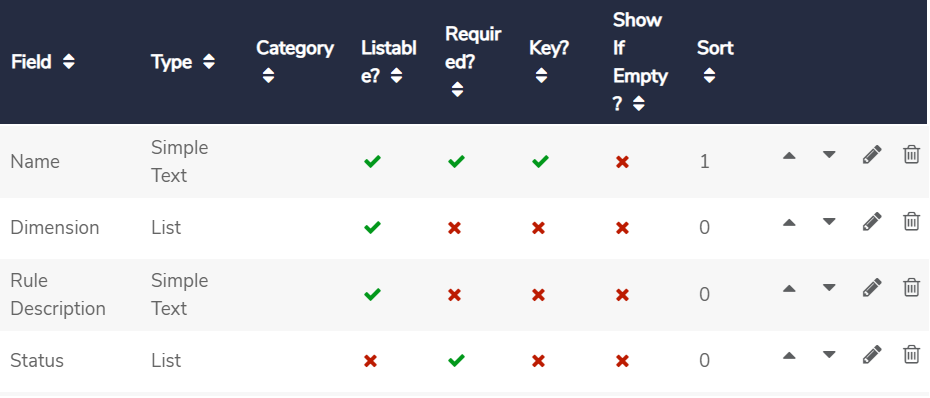






Rename fields – “Expected Value Should be Outside of Range” to “Within Range” and “Pass if Blank or Null” as “Treat Blank or Null As” of List type having Reference List Item as “IGX Pass Skip List”.

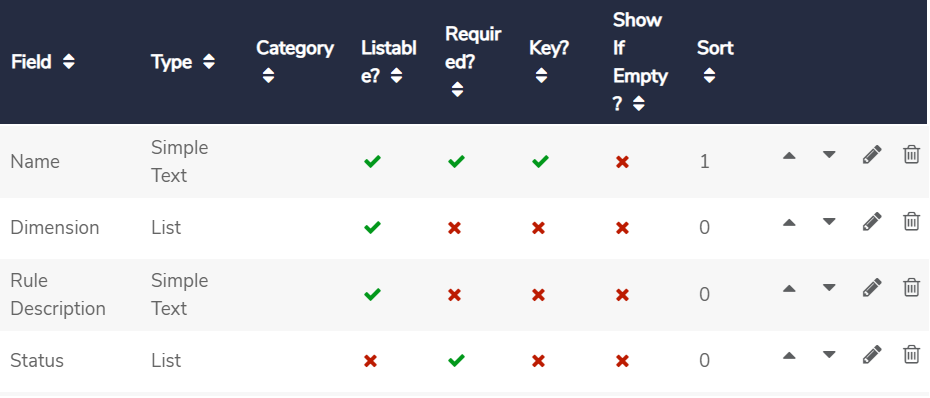
### **IGX Attribute Comparison**



Additional Fields -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Type of List** | **Listable** | **Required** |
| Rule Operator | List | Reference List Item: IGX Comparison Rule Operator List | Yes | Yes |
| Rule Value | List | Artifact: IGX Data Element | Yes | Yes |
| Rule Value Case Sensitive | True/False |  | Yes | Yes |
| Treat Blank or Null As | List | Reference List Item: “IGX Pass Skip List” | Yes | Yes |

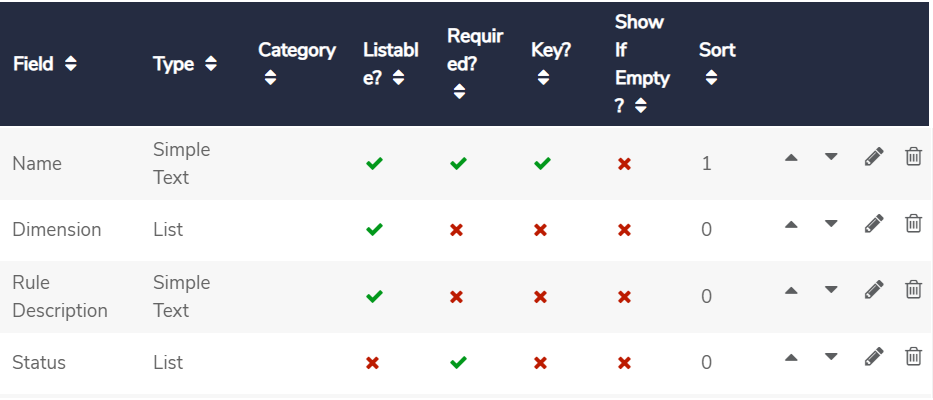
### **IGX Conformity Check**



Additional Fields -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Type of List** | **Listable** | **Required** |
| Rule Operator | List | Reference List Item: IGX Comparison Rule Operator List | Yes | Yes |
| Rule Value | Simple Text |  | Yes | Yes |
| Rule Value Case Sensitive | True/False |  | Yes | Yes |
| Treat Blank or Null As | List | Reference List Item: “IGX Pass Skip List” | Yes | Yes |

### **IGX JS Expression**



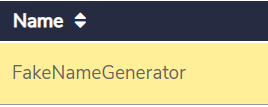
Additional Fields -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Type of List** | **Listable** | **Required** |
| JS Expression | Simple Text |  | Yes | Yes |
| Match JS Expression | True/False |  | Yes | Yes |

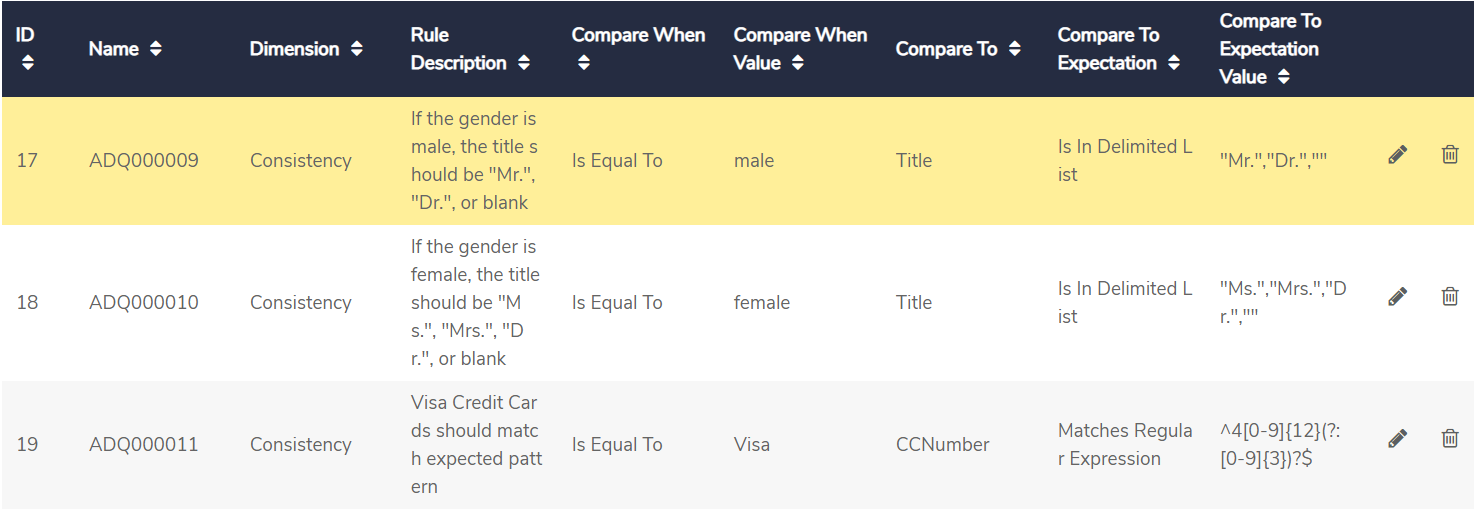
## Sample Data



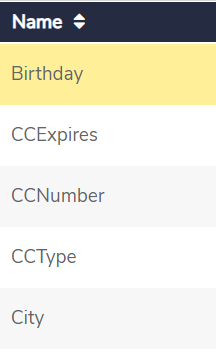
### **IGX Data Entity**



### **Sample Rule Implementation**

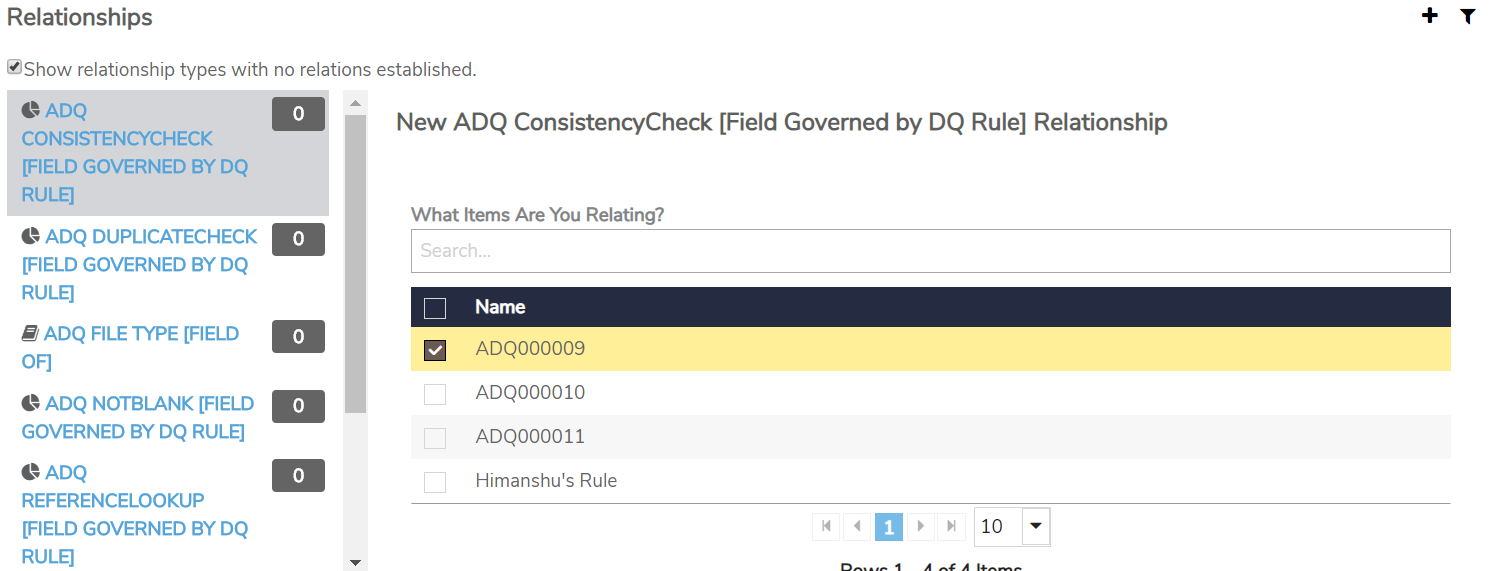


### **IGX Data Element**



### **Sample Relationship**

Create the relationship with rule while defining the data element

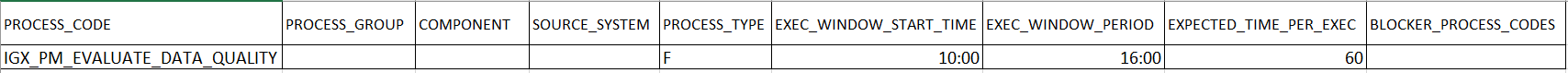


Refer to below link for configuration and sample data

<https://dqsandbox-igx.dev.data3sixty.com/home>

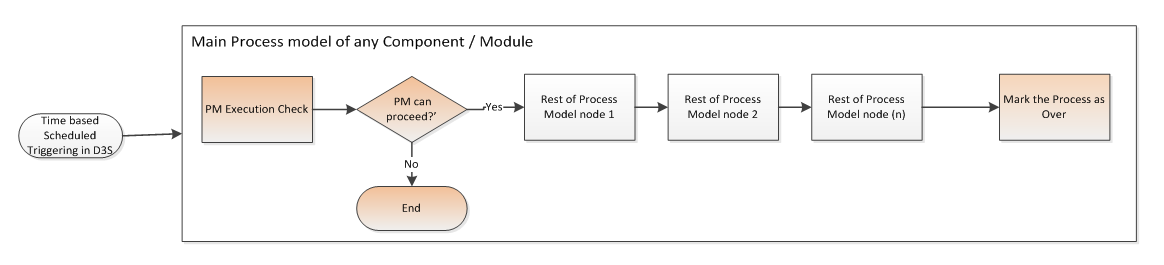
### **PROCESS\_LIST**

Below the sample configuration to be made in process list data store. This configuration is required for Execution Gateway component called as first analysis from the grant process model i.e. igx\_pm\_evaluate\_data\_quality

****

# **Common Design Elements**

PM Execution check will be plugged in module process models as depicted below in orange. “Mark Process as Over” is to mark the process as over at the end of process model



>

<e.g. Detail about triggering of the next process model in the chain i.e. listing the target process model that will be triggered from source process model.

|  |  |  |  |
| --- | --- | --- | --- |
| SNo. | Source Process Model | Target Process Model | Remarks |
|  |  |  |  |
|  |  |  |  |

# **Security**

<Document the permissions for the users and groups on various stages. The implementation of security should explain which users, group were mapped to what. Also include what all permissions need to be provided at every level. >

|  |  |  |
| --- | --- | --- |
| **Super Group** | **Path/Stage** | **Permissions** |
|  |  |  |
|  |  |

|  |  |
| --- | --- |
| **Super Group** | **Group** |
|  |  |
|  |

# **Assumptions**

# **Appendix**

## Recommended Configuration file

<Embed PG\_PURGER\_CONFIG csv file here>