Dartec Systems

Data Quality Engine (DQE)

User Guide

Table of Contents

1 Overview

DQE is a set of routines designed to run on the SQL Server platform. A full technical description of the DQE can be found in the DQE_HLD document.

This document describes how a technical user would interact with the DQE tool and covers

- how new rules are created
- how the DQE is executed
- how to review DQE results and activity
- how to use the BI reports to query the DQE results

2 Creating DQ Domain, Entities and Rules

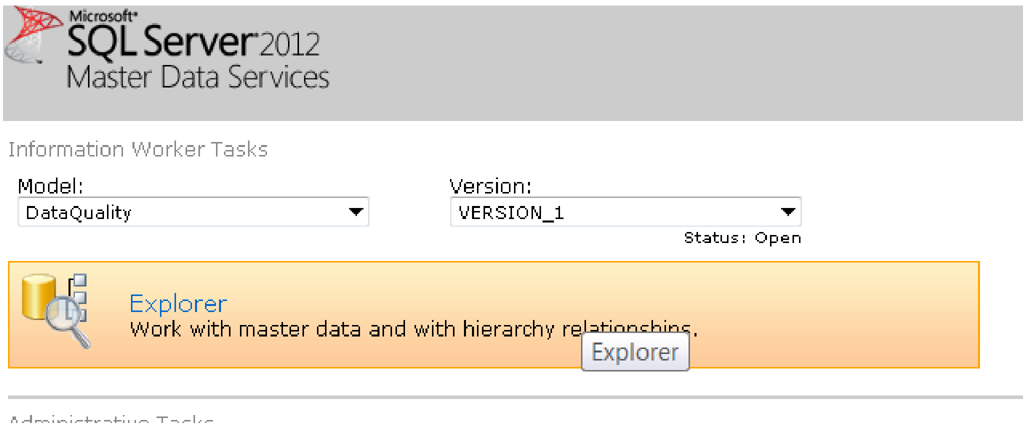
2.1 Rule behaviour and examples

The Technical Overview document contains a table that describes the behaviour of each type of rule currently catered for by the DQE. In addition, the MDS model comes pre-loaded with a range of example rules within the "AdventureWorksStage1" domain. These rules have been created to illustrate the range of functionality available.

2.2 Setting up a new group of Data Quality Rules

The first step to configuring a Data Quality rule, or set of rules, is to ensure you have a container from which the rule can be executed, these containers are the DQ Domains.

To do so log into the "DataQuality" Master Data services package and navigate to explorer.



Information Worker Tasks

Model: DataQuality

Version: VERSION_1

Status: Open

Explorer

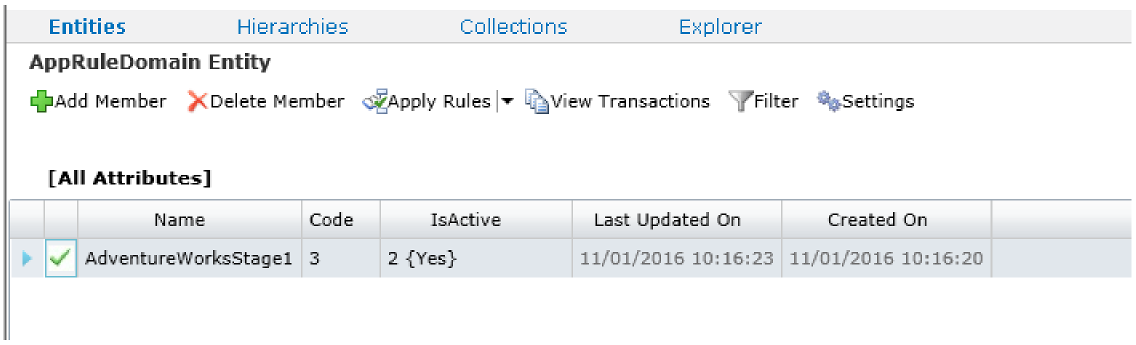
Work with master data and with hierarchy relationships.

2.2.1 Add a new DQ Domain

MDS Dependencies: AppBinarySwitch.

Navigate the "AppRuleDomain" entity.

Enter a new domain; in this example, the domain is "AdventureWorksStage1".



Name	Code	IsActive	Last Updated On	Created On
AdventureWorksStage1	3	2 {Yes}	11/01/2016 10:16:23	11/01/2016 10:16:20

This DQ Domain will act as a container for multiple DQ entities and rules. Typically a DQ Domain would apply to a set of DQ Rules that you want to apply at the same time. It would be possible to create multiple DQ Domains that could be applied at different points within the ETL process.

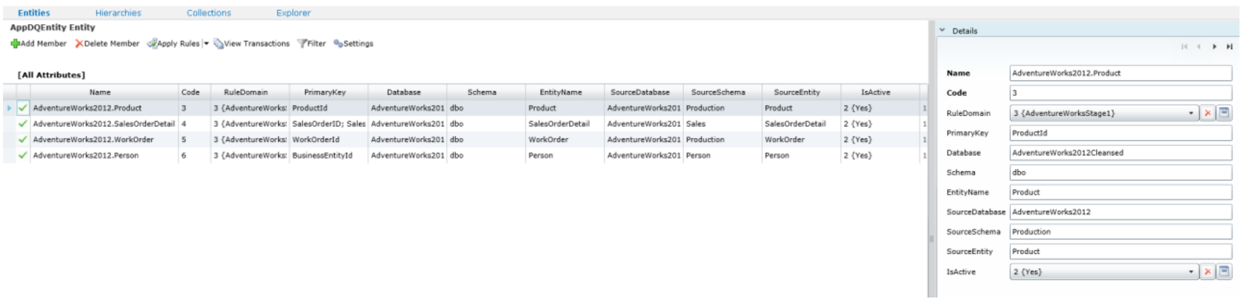
Note, the DQ Domain is the “DomainName” value that is handed into the DQE execution procedure.

2.2.2 <u>Add a new DQ Entity</u>

Dependencies: AppBinarySwitch; AppRuleDomain

Once a DQDomain exists navigate to the “AppDQEntity” screen.

Enter the tables that you want to apply DQRules to within the DQ Domain; in the below example a number of tables have been configured including AdventureWorks2012.SalesOrderDetail & AdventureWorks2012.Product



Name	Code	RuleDomain	PrimaryKey	Database	Schema	EntityName	SourceDatabase	SourceSchema	SourceEntity	IsActive
AdventureWorks2012.Product	3	3 {AdventureWorks:	ProductId	AdventureWorks201	dbo	Product	AdventureWorks201	Production	Product	2 {Yes}
AdventureWorks2012.SalesOrderDetail	4	3 {AdventureWorks:	SalesOrderID; Sales	AdventureWorks201	dbo	SalesOrderDetail	AdventureWorks201	Sales	SalesOrderDetail	2 {Yes}
AdventureWorks2012.WorkOrder	5	3 {AdventureWorks:	WorkOrderId	AdventureWorks201	dbo	WorkOrder	AdventureWorks201	Production	WorkOrder	2 {Yes}
AdventureWorks2012.Person	6	3 {AdventureWorks:	BusinessEntityId	AdventureWorks201	dbo	Person	AdventureWorks201	Person	Person	2 {Yes}

Name	AdventureWorks2012.Product
Code	3
RuleDomain	3 {AdventureWorksStage1}
PrimaryKey	ProductId
Database	AdventureWorks2012Cleansed
Schema	dbo
EntityName	Product
SourceDatabase	AdventureWorks2012
SourceSchema	Production
SourceEntity	Product
IsActive	2 {Yes}

Using this screen it is possible to setup the tables that you will apply DQ Rules. A description of each field is provided below.

Field	Description
Name	The name of the table. Recommendation: This is the value that will be visible Entity_Rule linking table so it is **strongly** recommended that a meaningful name is used that includes the DQDomain and DQEntity name.

Code	The MDS generated unique code for that row.
RuleDomain	Linked domain to the AppRuleDomain. Use this to link the entity to a specific rule domain.
PrimaryKey	Semi-colon separated list of the columns that constitute the primary key on the entity. This value is used by various functions such as determining if duplicates exist.
Database	The database where the DQEntity resides
Schema	The schema where the DQEntity resides
EntityName	The name of the table that the DQ rules will be applied to. **Note: If the rule creates a status or output column this table will be modified.**
SourceDatabase	If you want to create a copy of a table and use this in the DQ activities, specify the database where the original table or view can be found. If a valid sourcedatabase.sourceschema.sourceentity is specified a copy will be created using the values specified in the database.schema.entityname columns. The database where the source DQEntity resides
SourceSchema	If you want to create a copy of a table and use this in the DQ activities, specify the database where the original table or view can be found. If a valid sourcedatabase.sourceschema.sourceentity is specified a copy will be created using the values specified in the database.schema.entityname columns. The schema where the source DQEntity resides
SourceEntity	If you want to create a copy of a table and use this in the DQ activities, specify the database where the original table or view can be found. If a valid sourcedatabase.sourceschema.sourceentity is specified a copy will be created using the values specified in the database.schema.entityname columns. The name of the source table that the DQ rules will be applied to.
IsActive	Indicates whether the entity should be considered as active when the DQDomain rules are applied.

2.2.3 Associate an existing DQ Rule to an DQ Entity

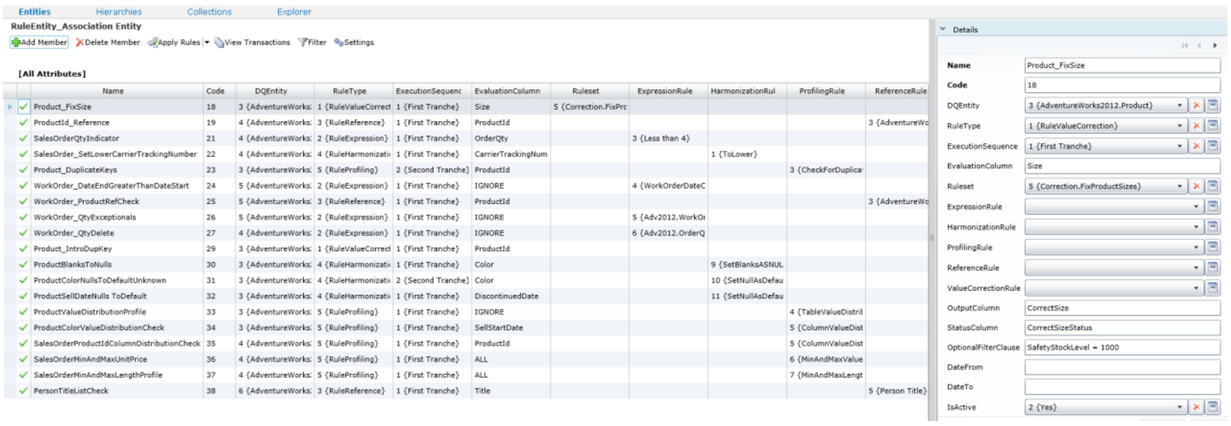
Dependencies: AppBinarySwitch; AppRuleDomain; AppDQEntity

Possible dependencies: AppRuleType; AppExecutionSequence; Ruleset; RuleExpression; RuleHarmonization; RuleProfiling; RuleReference; RuleValueCorrection

Once a DQEntity exists navigate to the “RuleEntity_Association” screen.

The approach adopted by DQE is DQ Rules should be are defined once and applied many times (to DQ Entities). This is achieved by linking DQRules to DQEntities within the “RuleEntity_Association” screen.

The below screen illustrates a number of Rule to Entity associations and the table provides descriptions for each column.



RuleEntity_Association Entity

[All Attributes]

Name	Code	DQEntity	RuleType	ExecutionSequenc	EvaluationColumn	Ruleset	ExpressionRule	HarmonizationRul	ProfilingRule	ReferenceRule
Product_FixSize	18	3 {AdventureWorks:	1 {RuleValueCorrect	1 {First Tranche}	Size	5 {Correction.FixPrc				
ProductId_Reference	19	4 {AdventureWorks:	3 {RuleReference}	1 {First Tranche}	ProductId					3 {AdventureWo
SalesOrderQtyIndicator	21	4 {AdventureWorks:	2 {RuleExpression}	1 {First Tranche}	OrderQty		3 {Less than 4}			
SalesOrder_SetLowerCarrierTrackingNumber	22	4 {AdventureWorks:	4 {RuleHarmonizati	1 {First Tranche}	CarrierTrackingNum			1 {ToLower}		
Product_DuplicateKeys	23	3 {AdventureWorks:	5 {RuleProfiling}	2 {Second Tranche}	ProductId				3 {CheckForDuplica	
WorkOrder_DateEndGreaterThanDateStart	24	5 {AdventureWorks:	2 {RuleExpression}	1 {First Tranche}	IGNORE		4 {WorkOrderDateC			
WorkOrder_ProductRefCheck	25	5 {AdventureWorks:	3 {RuleReference}	1 {First Tranche}	ProductId					3 {AdventureWo
WorkOrder_QtyExceptionals	26	5 {AdventureWorks:	2 {RuleExpression}	1 {First Tranche}	IGNORE		5 {Adv2012.WorkOr			
WorkOrder_QtyDelete	27	4 {AdventureWorks:	2 {RuleExpression}	1 {First Tranche}	IGNORE		6 {Adv2012.OrderQ			
Product_IntroDupKey	29	3 {AdventureWorks:	1 {RuleValueCorrect	1 {First Tranche}	ProductId					
ProductBlanksToNulls	30	3 {AdventureWorks:	4 {RuleHarmonizati	1 {First Tranche}	Color			9 {SetBlanksASNUL		
ProductColorNullsToDefaultUnknown	31	3 {AdventureWorks:	4 {RuleHarmonizati	2 {Second Tranche}	Color			10 {SetNullAsDefau		
ProductSellDateNulls ToDefault	32	3 {AdventureWorks:	4 {RuleHarmonizati	1 {First Tranche}	DiscontinuedDate			11 {SetNullAsDefau		
ProductValueDistributionProfile	33	3 {AdventureWorks:	5 {RuleProfiling}	1 {First Tranche}	IGNORE				4 {TableValueDistrit	
ProductColorValueDistributionCheck	34	3 {AdventureWorks:	5 {RuleProfiling}	1 {First Tranche}	SellStartDate				5 {ColumnValueDist	
SalesOrderProductIdColumnDistributionCheck	35	4 {AdventureWorks:	5 {RuleProfiling}	1 {First Tranche}	ProductId				5 {ColumnValueDist	
SalesOrderMinAndMaxUnitPrice	36	4 {AdventureWorks:	5 {RuleProfiling}	1 {First Tranche}	ALL				6 {MinAndMaxValue	
SalesOrderMinAndMaxLengthProfile	37	4 {AdventureWorks:	5 {RuleProfiling}	1 {First Tranche}	ALL				7 {MinAndMaxLengt	
PersonTitleListCheck	38	6 {AdventureWorks:	3 {RuleReference}	1 {First Tranche}	Title					5 {Person Title}

Details

Name	Product_FixSize
Code	18
DQEntity	3 {AdventureWorks2012.Product}
RuleType	1 {RuleValueCorrection}
ExecutionSequence	1 {First Tranche}
EvaluationColumn	Size
Ruleset	5 {Correction.FixProductSizes}
ExpressionRule	
HarmonizationRule	
ProfilingRule	
ReferenceRule	
ValueCorrectionRule	
OutputColumn	CorrectSize
StatusColumn	CorrectSizeStatus
OptionalFilterClause	SafetyStockLevel = 1000
DateFrom	
DateTo	
IsActive	2 {Yes}

Field	Description
Name	The name of the DQRule & DQEntity association.
Code	The MDS generated unique code for that row.
DQEntity	The DQEntity that the rule will be applied to.
RuleType	Metadata needed by the orchestration. The type of rule this row is.
ExecutionSequence	The sequence in which you want the rules to be applied. If specified it is possible to apply rules in sequential tranches. This allows rules to be sequenced to carry out dependent DQ rules (for example, cleanse in the first tranche & perform character replace in the second tranche OR create an indicator in a first tranche which is used within a second tranche operation).
EvaluationColumn	The name of the column on the DQEntity that the DQRule will be applied to.
Ruleset	Optional: If the DQRules have been grouped into a single Ruleset, the ruleset should be specified in this column.
ExpressionRule/ HarmonizationRule/ ProfilingRule/ ReferenceRule/ ValueCorrectionRule	Optional: Link the DQEntity to a specific DQRule. A rule should not be linked to multiple rules or a rule and a ruleset. If it is the rule will not be executed.
OutputColumn	Optional:

	If the DQRule alters the values in the EvaluationColumn in some way, the outputs can be written to 'cleanse' output column. If no 'OutputColumn' is specified and the rule will cleanse the data in the 'EvaluationColumn'.
StatusColumn	Optional: If the DQRule creates some form of status output (such as a flag), the value is written to the "StatusColumn" if specified.
OptionalFilterClause	Optional: The DQRule can be constrained to apply to only a subset of rows within the DQEntity. To do so a FilterClause should be included. The value will be applied as t-sql WHERE clause and show be written using T-SQL syntax.
DateFrom/ DateTo	Not used in the current release.
IsActive	Indicates whether the row should be considered as active when the DQDomain rules are applied.

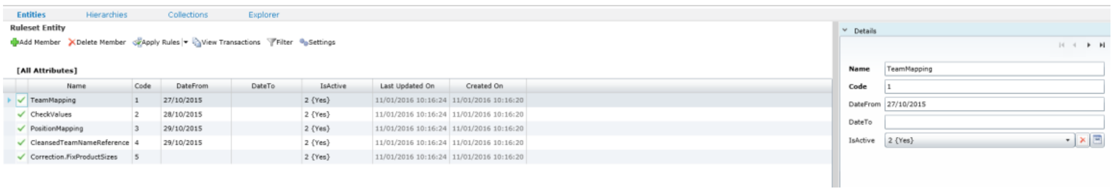
2.2.4 <u>Rules Vs Rulesets</u>

As seen above the RuleEntityAssociation can be linked either to a Rule or a Ruleset. Where a ruleset is used all rules within the ruleset will be applied and the outputs and statuses will be written to the same "OutputColumn" & "StatusColumn".

There are circumstances where the use of Ruleset is the most appropriate such as where ValueCorrection – where you may wish to specify many values to be corrected within a single "EvaluationColumn". Other scenarios may be required where multiple expressions are applied and indicators captured within a single "StatusColumn".

2.2.4.1 Creating a new Ruleset

To create a new ruleset; navigate to the Ruleset screen.



Ruleset Entity

[All Attributes]

	Name	Code	DateFrom	DateTo	IsActive	Last Updated On	Created On
✓	TeamMapping	1	27/10/2015		2 {Yes}	11/01/2016 10:16:24	11/01/2016 10:16:20
✓	CheckValues	2	28/10/2015		2 {Yes}	11/01/2016 10:16:24	11/01/2016 10:16:20
✓	PositionMapping	3	29/10/2015		2 {Yes}	11/01/2016 10:16:24	11/01/2016 10:16:20
✓	CleansedTeamNameReference	4	29/10/2015		2 {Yes}	11/01/2016 10:16:24	11/01/2016 10:16:20
✓	Correction.FixProductSizes	5			2 {Yes}	11/01/2016 10:16:24	11/01/2016 10:16:20

Name: TeamMapping
Code: 1
DateFrom: 27/10/2015
DateTo:
IsActive: 2 {Yes}

Field	**Description**
Name	The name of the ruleset
Code	The MDS generated unique code for that row.
DateFrom/ DateTo	Not enabled
IsActive	Indicates whether the row should be considered as active when the DQDomain rules are applied.

Once created the ruleset will be available to use within the “RuleEntity_Association” screen and with the DQ Rule screens.

2.2.5 <u>Creating a new rule</u>

The DQE contains a number of pre-configured rules. In limited cases these are specific to the DQDomain (for example the Correction Rules) while in most cases these are relatively generic and should be applicable in many circumstances.

Details regarding what each rule does together within a summary of expected outputs can be found in the “Data Quality Rules and Behaviours” section found earlier in Appendix 1.

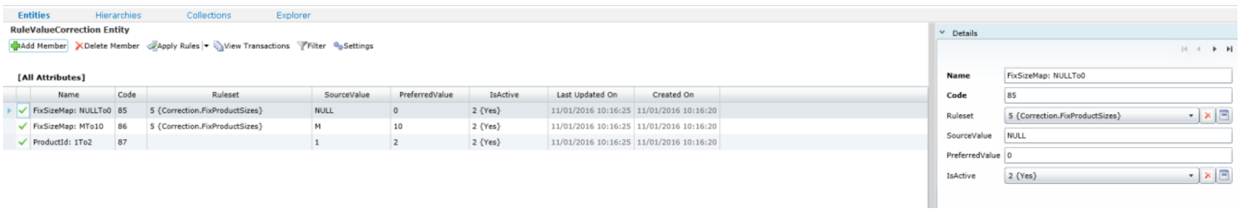
2.2.5.1 Correction Rules

Dependencies: AppBinarySwitch; Ruleset

These rules specify values that should be corrected within the EvaluationColumn.

An example would be finding and replacing the value “St.” with a cleansed value of “Street”.

Correction rules are often best grouped using the Ruleset functionality though this is not mandatory (meaning a single rule can be applied to a DQEntity).



RuleValueCorrection Entity

[All Attributes]

	Name	Code	Ruleset	SourceValue	PreferredValue	IsActive	Last Updated On	Created On
✓	FixSizeMap: NULLTo0	85	5 {Correction.FixProductSizes}	NULL	0	2 {Yes}	11/01/2016 10:16:25	11/01/2016 10:16:20
✓	FixSizeMap: MTo10	86	5 {Correction.FixProductSizes}	M	10	2 {Yes}	11/01/2016 10:16:25	11/01/2016 10:16:20
✓	ProductId: 1To2	87		1	2	2 {Yes}	11/01/2016 10:16:25	11/01/2016 10:16:20

Name	FixSizeMap: NULLTo0
Code	85
Ruleset	5 {Correction.FixProductSizes}
SourceValue	NULL
PreferredValue	0
IsActive	2 {Yes}

Field	Description
Name	The name of the Correction Rule
Code	The MDS generated unique code for that row.
Ruleset	Links to Ruleset entity and allows Correction rules to be grouped.
SourceValue	The value within the “EvaluationColumn” that should be replaced
PreferredValue	The value that should replace the SourceValue within the “EvaluationColumn”
IsActive	Indicates whether the row should be considered as active when the DQDomain rules are applied.

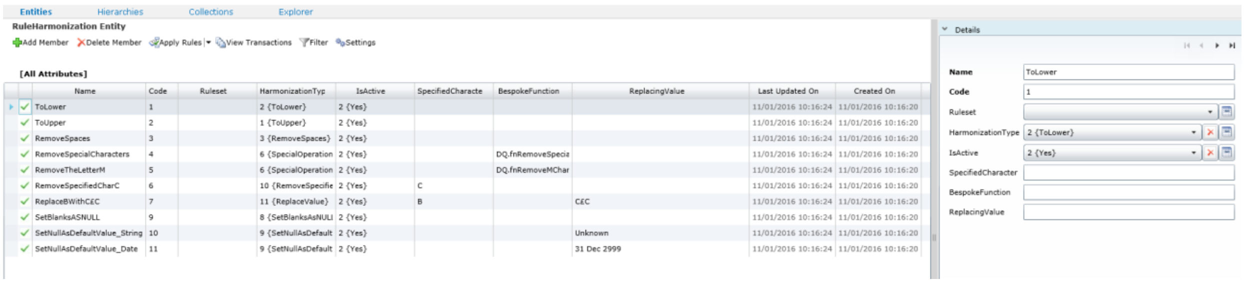
2.2.5.2 Harmonization Rules

Dependencies: AppBinarySwitch; Ruleset; AppHarmonizationType

These rules are tied closely to the underlying DQ stored procedure (sApplyDQRuleHarmonization). Change the names of the HarmonizationType or adding new ones will require changes to the underlying stored procedure.

These rules specify values that should be harmonized within the EvaluationColumn. Examples of harmonization include setting all values to UPPER, lower or removing specified characters.

The SpecialOperation type allows bespoke functions, defined within DataQualityDB, to be defined by the developer.



RuleHarmonization Entity

[All Attributes]

	Name	Code	Ruleset	HarmonizationTyp	IsActive	SpecifiedCharacte	BespokeFunction	ReplacingValue	Last Updated On	Created On
✓	ToLower	1		2 {ToLower}	2 {Yes}				11/01/2016 10:16:24	11/01/2016 10:16:20
✓	ToUpper	2		1 {ToUpper}	2 {Yes}				11/01/2016 10:16:24	11/01/2016 10:16:20
✓	RemoveSpaces	3		3 {RemoveSpaces}	2 {Yes}				11/01/2016 10:16:24	11/01/2016 10:16:20
✓	RemoveSpecialCharacters	4		6 {SpecialOperation	2 {Yes}		DQ.fnRemoveSpecia		11/01/2016 10:16:24	11/01/2016 10:16:20
✓	RemoveTheLetterM	5		6 {SpecialOperation	2 {Yes}		DQ.fnRemoveMChar		11/01/2016 10:16:24	11/01/2016 10:16:20
✓	RemoveSpecifiedCharC	6		10 {RemoveSpecifie	2 {Yes}	C			11/01/2016 10:16:24	11/01/2016 10:16:20
✓	ReplaceBWithC£C	7		11 {ReplaceValue}	2 {Yes}	B		C£C	11/01/2016 10:16:24	11/01/2016 10:16:20
✓	SetBlanksASNULL	9		8 {SetBlanksAsNULl	2 {Yes}				11/01/2016 10:16:24	11/01/2016 10:16:20
✓	SetNullAsDefaultValue_String	10		9 {SetNullAsDefault	2 {Yes}			Unknown	11/01/2016 10:16:24	11/01/2016 10:16:20
✓	SetNullAsDefaultValue_Date	11		9 {SetNullAsDefault	2 {Yes}			31 Dec 2999	11/01/2016 10:16:24	11/01/2016 10:16:20

Name	ToLower
Code	1
Ruleset	
HarmonizationType	2 {ToLower}
IsActive	2 {Yes}
SpecifiedCharacter	
BespokeFunction	
ReplacingValue	

Field	**Description**
Name	The name of the Harmonization Rule
Code	The MDS generated unique code for that row.
Ruleset	Links to Ruleset entity and allows Harmonization rules to be grouped.
HarmonizationType	The type of harmonization rule being applied
SpecifiedCharacter	Optional: Used with the "RemoveSpecifiedCharacter" and the "ReplaceValue" ruletypes. Use this column to specify the character that will be removed or replaced
ReplacingValue	Optional: Used with the "ReplaceValue" & the "SetNullAsDefault" ruletypes. Use this column to specify the character that will replace the character specified in the "SpecifiedCharacter" column.
BespokeFunction	Optional: Use with the "SpecialOperation" ruletype. Specify a function that should be applied to the EvaluationColumn.

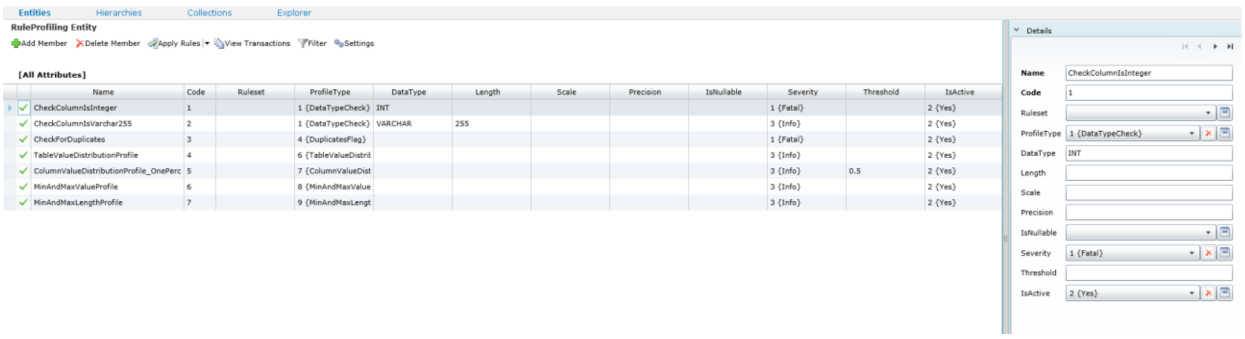
IsActive	Indicates whether the row should be considered as active when the DQDomain rules are applied.

2.2.5.3 Profiling Rules

Dependencies: AppBinarySwitch; Ruleset; AppProfileType; AppSeverity

These rules are tied closely to the underlying stored procedure (sApplyDQRuleProfiling). Changing the names of the ProfileTypes or adding new ones will require changes to the underlyng stored procedure.

These rules can be used to profile a given entity or column. The focus of the profiling rules is to gather information about the DQ Entity rather than changing the data. Examples may include establishing the value distribution on the EvaluationColumn.



RuleProfiling Entity

[All Attributes]

	Name	Code	Ruleset	ProfileType	DataType	Length	Scale	Precision	IsNullable	Severity	Threshold	IsActive
✓	CheckColumnIsInteger	1		1 {DataTypeCheck}	INT					1 {Fatal}		2 {Yes}
✓	CheckColumnIsVarchar255	2		1 {DataTypeCheck}	VARCHAR	255				3 {Info}		2 {Yes}
✓	CheckForDuplicates	3		4 {DuplicatesFlag}						1 {Fatal}		2 {Yes}
✓	TableValueDistributionProfile	4		6 {TableValueDistrit						3 {Info}		2 {Yes}
✓	ColumnValueDistributionProfile_OnePerc	5		7 {ColumnValueDist						3 {Info}	0.5	2 {Yes}
✓	MinAndMaxValueProfile	6		8 {MinAndMaxValue						3 {Info}		2 {Yes}
✓	MinAndMaxLengthProfile	7		9 {MinAndMaxLengt						3 {Info}		2 {Yes}

Name	CheckColumnIsInteger
Code	1
Ruleset	
ProfileType	1 {DataTypeCheck}
DataType	INT
Length	
Scale	
Precision	
IsNullable	
Severity	1 {Fatal}
Threshold	
IsActive	2 {Yes}

Field	Description
Name	The name of the Profile Rule
Code	The MDS generated unique code for that row.
Ruleset	Links to Ruleset entity and allows profiling rules to be grouped.
ProfileType	The type of profiling rule being applied
DataType	Optional: Used with the “DataTypeCheck” rule type. Use this to specify the data type you would like to try and CAST the EvaluationColumn into
Length	Optional: Used with the “DataTypeCheck” rule type. Use this to specify the length you would like to try and CAST the EvaluationColumn into
Scale	Optional: Used with the “DataTypeCheck” rule type.

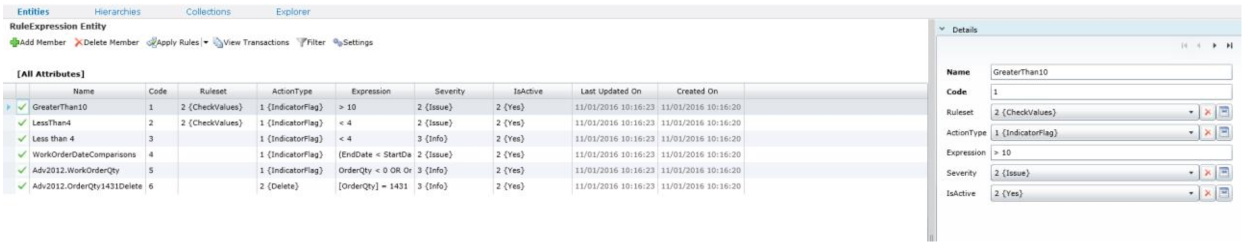
	Use this to specify the scale you would like to try and CAST the EvaluationColumn into
Precision	Optional: Used with the “DataTypeCheck” rule type. Use this to specify the precision you would like to try and CAST the EvaluationColumn into
IsNullable	Not implemented
Severity	Defines the severity of issues once they are detected. Initially 3 arbitrary values for severity have been created, with their suggested meaning as follows:- Fatal – The highlighted data may cause a fatal error e.g. duplicates Issue – The highlighted data may cause an issue but OK to progress Info – The highlighted data is for information only e.g. a min and max value profile. The ‘Severity’ of the data row is written to the Data Quality History table when a data quality rule highlights the data within the row. The severity is expressed as a ‘Severity ID’ and the accompanying ‘Severity Name’
Threshold	Optional: Use with the “ColumnValueDistribution” rule type. This is used to group all values with a distribution less than this value.
IsActive	Indicates whether the row should be considered as active when the DQDomain rules are applied.

2.2.5.4 Expression Rules

Dependencies: AppBinarySwitch; Ruleset; AppActionType; AppSeverity

These rules are tied closely to the underlying stored procedure. Change the names of the ActionType or adding new ones will require changes to the underlyng stored procedure.

These rules allow certain records to be flagged or removed based on an expression (which is in the form of a T-SQL WHERE condition).



RuleExpression Entity

[All Attributes]

Name	Code	Ruleset	ActionType	Expression	Severity	IsActive	Last Updated On	Created On
GreaterThan10	1	2 {CheckValues}	1 {IndicatorFlag}	> 10	2 {Issue}	2 {Yes}	11/01/2016 10:16:23	11/01/2016 10:16:20
LessThan4	2	2 {CheckValues}	1 {IndicatorFlag}	< 4	2 {Issue}	2 {Yes}	11/01/2016 10:16:23	11/01/2016 10:16:20
Less than 4	3		1 {IndicatorFlag}	< 4	3 {Info}	2 {Yes}	11/01/2016 10:16:23	11/01/2016 10:16:20
WorkOrderDateComparisons	4		1 {IndicatorFlag}	(EndDate < StartDa	2 {Issue}	2 {Yes}	11/01/2016 10:16:23	11/01/2016 10:16:20
Adv2012.WorkOrderQty	5		1 {IndicatorFlag}	OrderQty < 0 OR Or	3 {Info}	2 {Yes}	11/01/2016 10:16:23	11/01/2016 10:16:20
Adv2012.OrderQty1431Delete	6		2 {Delete}	[OrderQty] = 1431	3 {Info}	2 {Yes}	11/01/2016 10:16:23	11/01/2016 10:16:20

Name	GreaterThan10
Code	1
Ruleset	2 {CheckValues}
ActionType	1 {IndicatorFlag}
Expression	> 10
Severity	2 {Issue}
IsActive	2 {Yes}

Field	**Description**
Name	The name of the Expression Rule
Code	The MDS generated unique code for that row.

Ruleset	Links to Ruleset entity and allows profiling rules to be grouped.
ActionType	The type of expression rule being applied
Expression	Specify the expression that will be used to identify records within the DQEntity. Note, the expression value is used to construct a T-SQL WHERE clause. This means the expression value should be written as a T-SQL WHERE clause.
Severity	Records the severity of rows identified.
IsActive	Indicates whether the row should be considered as active when the DQDomain rules are applied.

2.2.5.5 Reference Rules

Dependencies: AppBinarySwitch; Ruleset; AppReferenceType; AppSeverity

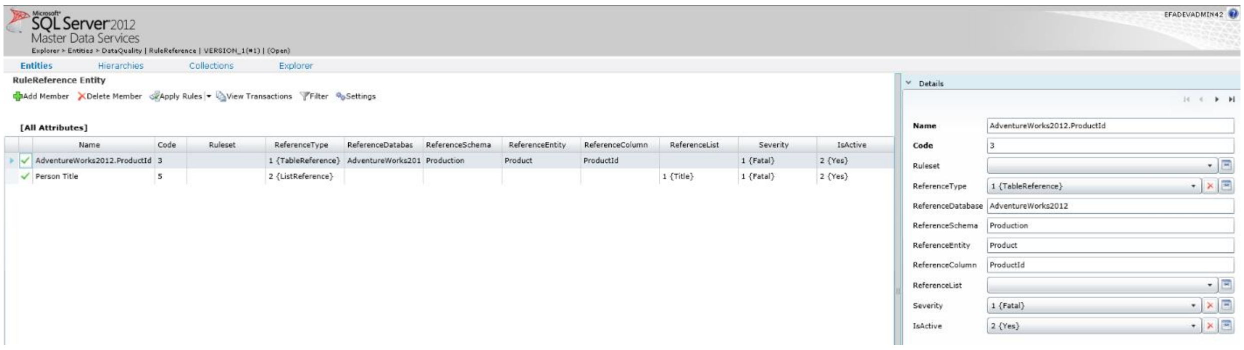
It is likely that you will need to add more rules as you incorporate additional reference checks.

These rule types are tied closely to the underlying stored procedure. Change the names of the ReferenceType or adding new ones will require changes to the underlyng stored procedure.

These rules define either the external table or lookup that the EvaluationColumn will be evaluated against.

All external reference tables must be located on the database server that the reference rule is executed on. The patterns currently used to do this is add an extra import package to the SSIS Project.

Reference checks against smaller controlled lists make use of a DQE screen “AppReferenceLists”, which can be used to store multiple controlled lists.



RuleReference Entity

[All Attributes]

	Name	Code	Ruleset	ReferenceType	ReferenceDatabas	ReferenceSchema	ReferenceEntity	ReferenceColumn	ReferenceList	Severity	IsActive
✓	AdventureWorks2012.ProductId	3		1 {TableReference}	AdventureWorks201	Production	Product	ProductId		1 {Fatal}	2 {Yes}
✓	Person Title	5		2 {ListReference}					1 {Title}	1 {Fatal}	2 {Yes}

Name	AdventureWorks2012.ProductId
Code	3
Ruleset	
ReferenceType	1 {TableReference}
ReferenceDatabase	AdventureWorks2012
ReferenceSchema	Production
ReferenceEntity	Product
ReferenceColumn	ProductId
ReferenceList	
Severity	1 {Fatal}
IsActive	2 {Yes}

Field	**Description**

Name	The name of the Reference Rule
Code	The MDS generated unique code for that row.
Ruleset	Links to Ruleset entity and allows reference rules to be grouped.
ReferenceType	The type of reference rule being applied
ReferenceDatabase ReferenceSchema ReferenceEntity	Optional: Used for “TableReference” checks. These values specify the location of the table that the DQEntity.EvaluationColumn will be evaluated against.
ReferenceColumn	Optional: Used for “TableReference” checks. Specifies the column in the reference table that the evaluation column will be compared against. The comparison is a simple NOT IN statement. The rule will check for values in the EvaluationColumn not found within the ReferenceColumn.
ReferenceList	Optional: Used for “ListReference” checks. Specifies the controlled list within the “AppReferenceLists” that the Evaluation column will be compared against.. The comparison is a simple NOT IN statement. The rule will check for values in the EvaluationColumn not found within the AppReferenceList.
Severity	Records the severity of rows identified.
IsActive	Indicates whether the row should be considered as active when the DQDomain rules are applied.

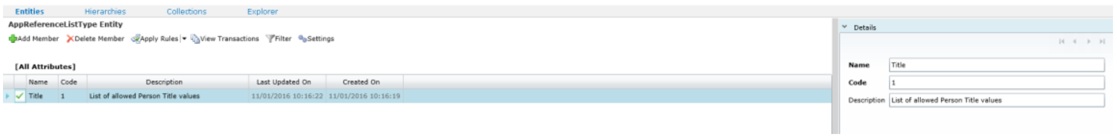
2.2.6 <u>Creating a DQ Reference List</u>

If you have the need to check an incoming evaluation column against a small to medium size list of values, the easiest way of how exposing that list to DQ is by creating the list directly within the Data Quality model.

There are two steps involved in creating a new list of values.

2.2.6.1 Create Reference List Type

Navigate to the “AppReferenceListType” screen



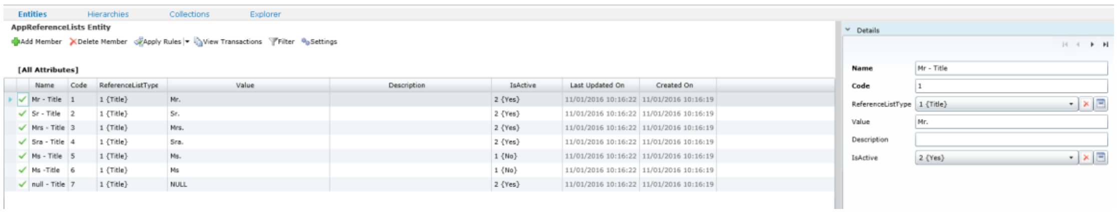
Name	Code	Description	Last Updated On	Created On
Title	1	List of allowed Person Title values	11/01/2016 10:16:22	11/01/2016 10:16:19

Name: Title
Code: 1
Description: List of allowed Person Title values

In the above screen a Reference type of table has been added. This will be used to identify which controlled values relate to the “Title” list.

2.2.6.2 Add the Reference List Values

Navigate to the “AppReferenceList” screen



Name	Code	ReferenceListType	Value	Description	IsActive	Last Updated On	Created On
Mr - Title	1	1 {Title}	Mr.		2 {Yes}	11/01/2016 10:16:22	11/01/2016 10:16:19
Sr - Title	2	1 {Title}	Sr.		2 {Yes}	11/01/2016 10:16:22	11/01/2016 10:16:19
Mrs - Title	3	1 {Title}	Mrs.		2 {Yes}	11/01/2016 10:16:22	11/01/2016 10:16:19
Sra - Title	4	1 {Title}	Sra.		2 {Yes}	11/01/2016 10:16:22	11/01/2016 10:16:19
Ms - Title	5	1 {Title}	Ms.		1 {No}	11/01/2016 10:16:22	11/01/2016 10:16:19
Ms -Title	6	1 {Title}	Ms		1 {No}	11/01/2016 10:16:22	11/01/2016 10:16:19
null - Title	7	1 {Title}	NULL		2 {Yes}	11/01/2016 10:16:22	11/01/2016 10:16:19

Name	Mr - Title
Code	1
ReferenceListType	1 {Title}
Value	Mr.
Description	
IsActive	2 {Yes}

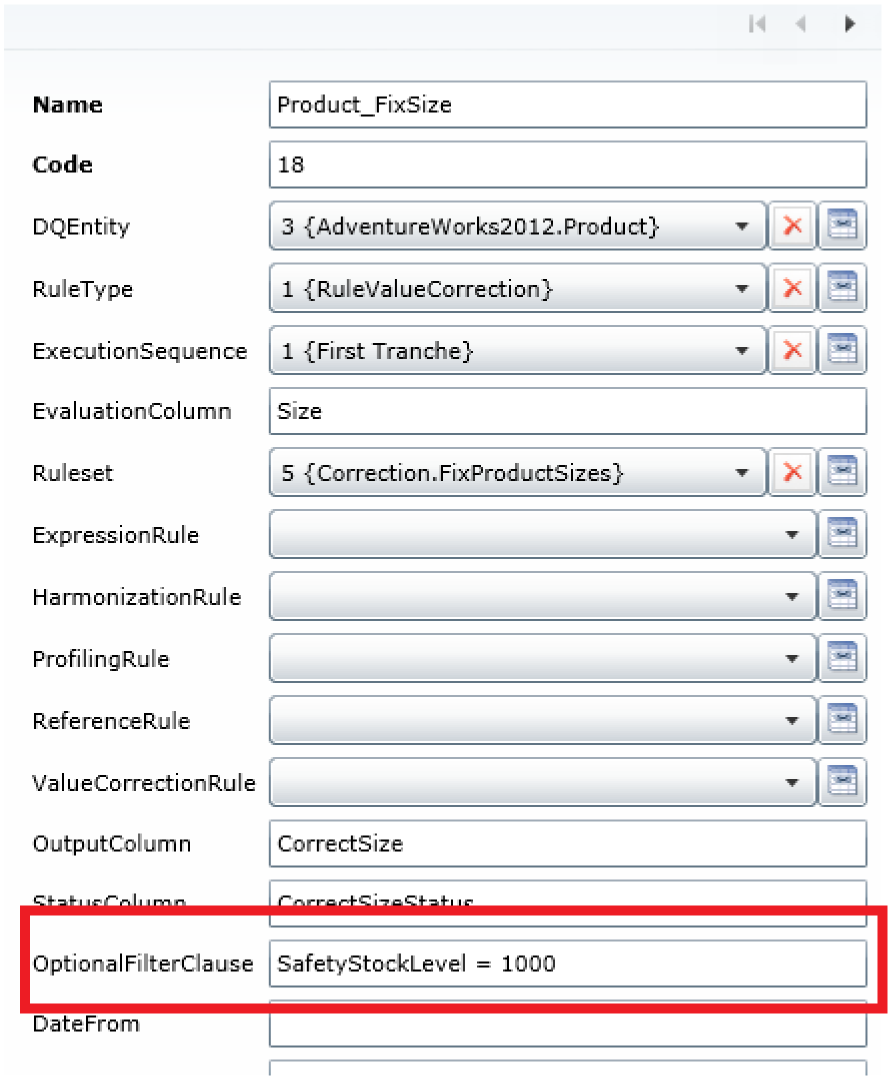
Field	Description
Name	The name of the ruleset
Code	The MDS generated unique code for that row.
ReferenceListType	Use this to define which Reference List Type the row should be linked to. This uses the values available in the AppReferenceListType table.
Value	Use this to add each reference list member
Description	Optional: Use this to add any descriptive information relating to the reference list member
IsActive	Indicates whether the row should be considered as active when the DQDomain rules are applied.

2.2.7 <u>Optional Filter</u>

DQE contains the capability to restrict the rows evaluated as part of the DQ rule. The ‘RuleEntity_Association’ contains an attribute called ‘OptionalFilterClause’ which can be used to restrict the rows within the DQEntity that the rule is applied to.

To utilise this column the restricting logic should be added in the format of a T-SQL WHERE condition. There is no need to include the ‘WHERE’ keyword but other T-SQL keywords are required if the condition is complex.

The below screenshot illustrates how the ‘ValueCorrection’ rule is created but only applied to those records that have a SatefyStockLevel of 1000.



Name	Product_FixSize
Code	18
DQEntity	3 {AdventureWorks2012.Product}
RuleType	1 {RuleValueCorrection}
ExecutionSequence	1 {First Tranche}
EvaluationColumn	Size
Ruleset	5 {Correction.FixProductSizes}
ExpressionRule	
HarmonizationRule	
ProfilingRule	
ReferenceRule	
ValueCorrectionRule	
OutputColumn	CorrectSize
StatusColumn	CorrectSizeStatus
OptionalFilterClause	SafetyStockLevel = 1000
DateFrom	

3 Executing DQE Rules and Domains

DQE can execute DQ Rules either as 'Stand-Alone' rules or in the context of a DQ Domain. The DQE provides the Stored Procedures and a SQL Agent job to support the DQ Rule execution in several contexts.

3.1 Executing Stand-Alone DQ Rules

The DQE contains two mechanisms within DataQualityDB for executing individual DQ Rules. These can be former is useful in the context of testing and troubleshooting.

3.1.1 SQL Agent Job calls

The *preferred* approach to executing individual rules is to use the routine that re-imports all metadata from the MDS model as part of the execution process

The below code snippet illustrates the use of the Execute Job procedure.

```
EXEC DataQualityDB.[DQ].[sExecuteJobDataQualityEngine]
@JobName = 'DataQualityEngineJob'
, @DQDomainName = 'AdventureWorksStage1'
, @ProjectName = 'DataQualityEngine'
, @FolderName = 'DataQualityEngine'
, @EnvironmentName = 'DQEParameters'
, @RuleEntityAssociationCode = 21 -- Optional, the stand-alone code of the rule
you want to run
```

Alternatively it is possible to simply execute a single DQ Rule without the wider activities of re-importing the MDS metadata.

3.1.2 Direct Database calls

The lowest impact approach is to call the sExecuteStandAloneRule directly. Using this approach will make use of the metadata previously been extracted from the MDS model. **This procedure does not re-import the metadata from the MDS model and is best used when troubleshooting the code.**

```
EXEC DataQualityDB.[DQ].[sExecuteStandAloneRule]
        @RuleEntityAssociationCode = 1 -- The Code of the RuleEntity_Association rule to
execute
        , @ParentLoadId = 1 -- This sets the LoadId of the calling procedure, by default this
is set to 1
        , @Debug = 0 -- If set to 1, the constructed SQL is output AND the rule is executed
```

3.2 Executing rules within a DQ Domain

The preferred approach for executing a domain of rules is using the provided DataQualityDB routine.

```
EXEC DataQualityDB.[DQ].[sExecuteJobDataQualityEngine]
@JobName = 'DataQualityEngineJob'
, @DQDomainName = 'AdventureWorksStage1'
, @ProjectName = 'DataQualityEngine'
, @FolderName = 'DataQualityEngine'
, @EnvironmentName = 'DQEParameters'
```

This routine will check that the Job, Project, Folder and SSIS environment exist before attempting to execute the SQL Server Agent job. This routine also updates a number of SSIS environment variables at run time before the scheduled job is executed.

This routine will wait until the job has completed its execution before handing control back to the application.

The stored procedure will execute whichever Scheduled Job is specified in the JobName variable.

A generic scheduled job has been configured to run the DQE, the scheduled job is called “DataQualityEngineJob”.

This job is responsible for executing SSIS project that has been installed in the SSIS Catalog which relies on a set of variables configured in the “DQEParameters” environment file.

4 Reviewing the DQE execution activities

The following types of activity are logged by DQE

- Data Processing logs: Each step within the DQE code is recorded together with a distinct LoadId, Name, Stored Procedure or SSIS package name, Start time, end time and the success status
- DQE Test Result logs: Each DQ Rule generates a test output together with descriptions, test results and the number of records identified
- DQE SQL logs: Most activities within the DQE are based on dynamic SQL. Each ‘chunk’ of actual SQL produced is logged.

All of the auditing information can be linked through the LoadId.

4.1 Processing Execution logs

All executions are logged to a central table which underpins all other auditing tables and views.

```
SELECT *
FROM [Audit].[RoutineLoad]
ORDER BY LoadId DESC
```

4.2 Identifying Errors

All errors are logged to a central table which is used by other auditing tables and views.

```
SELECT *
FROM [Audit].[RoutineError]
ORDER BY LoadId DESC
```

4.3 Data Processing logs

A view (Audit.RoutineLoadHistory) has been created to produce a summary of the steps that have run as part of the execution.

Each step within the ETL has a distinct LoadId which is independently logged together with a ParentLoadId, Start and End time. The view collapses the parent-child relationships into a flat structure which shows the MasterLoadId together with the step LoadId for each activity. The MasterLoadId can be used to link all activities within a single DQE execution.

The below SQL will query this view

```
SELECT *
FROM [Audit].[RoutineLoadHistory]
ORDER BY LoadId DESC
```

All processing errors, both ETL and DQ Rule generated can also be found in the Audit.RoutineLoadHistory view. This information can be used to identify mis-configured DQ Rules or general problems within the processing activities.

The LoadStatusName columns will typically write the ‘FAILURE’ value and the ErrorDescription and ErroredRoutine values can be used to assist with troubleshooting.

4.4 DQE Test Result logs

A [Reports].[DQSummaryResults] view has been created to provide a generic view of the results of the DQ Rules. Note, a fundamental difference between this view and Audit.RoutineLoadHistory is this view only contains information (and errors) relating directly to rule execution – this means errors encountered in the wider ETL activities will not be shown in this view.

Each row shows the result of a specific test together with information about the entity the test has been applied to.

The view contains a number of useful columns including

Column	*Description*
Loadid	The LoadId of the step that applies the rule. This can be used to link this information to a number of auditing tables.
MasterLoadId	The Master Load Id for the DQE execution, all rules run within the same execution will share the same Master Load Id
RuleDomain	The Code and Name of the DQ Domain that the rule is linked to
RuleType	The Code and Name of the type of rule
RuleName	The Code and Name of the Rule (i.e. the Name of the rule that has been associated to the Entity)

RuleEntityAssociation	The Code and Name of the association record found in RuleEntity_Association
Entity	The Code of the entity that has been linked to the Rule
DQEntity	The Name, Schema and Database of the entity that has been evaluated
SourceEntity	The Name, Schema and Database of the original entity (this may or may not be populated based on whether a cleansed copy of the entity has been created)
EvaluationColumn	The name of the column that has been evaluated
DQMessage	A description of the results
RowsAffected	Integer test results
PercentageValue	Numeric test results
Duration in Seconds	The number of seconds it took to execute the rule
Severity	The Code and Name of the severity of the result output
StartTime/ EndTime	The start and end time of the test
ErrorDescription	Where an error is encountered when attempting to execute the rule it is shown here
ErrorRoutine	The location that the error was encountered
MasterStartTime/ MasterEndTime	The start and end time of the overall execution.

To query this view use the below SQL

```
SELECT *
FROM [Reports].[DQSummaryResults]
order by LoadId DESC
```

To review rules that have raised serious issues with the data include the condition on the SeverityName column. If the DQ Rules have been configured correctly the bad results should be flagged with a SeverityName = “Fatal”.

```
SELECT *
```

```
FROM [Reports].[DQSummaryResults]
WHERE [SeverityName] = 'Fatal'
order by LoadId DESC
```

4.5 DQE SQL logs

All DQ rules executed by the DQE are in the format of T-SQL. All T-SQL generated to execute the DQ Rules is stored in a logging table to support auditing and traceability within the system.

All T-SQL used to process and execute the DQ Rules is stored within a single table `[DQ].[RuleExecutionHistory]`. This table holds the T-SQL generated by DQE together with metadata about the LoadId that the T-SQL was generated within.

The critical values within this table are

Column	*Description*
Loadid	The LoadId of the step that applies the rule. This can be used to link this information to a number of auditing tables.
RuleType	The type of T-SQL statement captured
RuleSQL	The actual SQL constructed to execute the DQ Rule

To query the execution history table use the below SQL.

```
SELECT *
FROM [DQ].[RuleExecutionHistory]
```

4.6 Scenario: Tracking down issues in the Generated Code

Occasionally you may find a rule is not behaving as expected (either due to poorly defined rules or some form of bug in the code). The below are the steps that you will typically follow when troubleshooting a bad rule or bad code.

Identify the RuleAssociationCode of the mis-behaving rule. This is the code value found in the ‘RuleEntity_Association’ MDS form.

This code can be used to actual SQL code executed by DQE at runtime which is available in the DQ.RuleExecutionHistory table

```sql
SELECT * FROM [DQ].[RuleExecutionHistory]
WHERE RuleId = 64
ORDER BY RXLOgid asc
```

The above query will return all rule-specific SQL generated at runtime. The actual SQL statements can be found in the RuleSQL column whilst the RuleSQLDescription provides a short description of what the code is doing and can be used to trace where the code was generated in the stored procedure.

This will allow you to examine the physical SQL that is generated and run. If you have a requirement to go back further and examine the code that generated the rule, this information is also available and simple to access.

Using the LoadId on RuleExecutionHistory table is possible to join to either the `[Audit].[RoutineLoadHistory] table` or the base auditing table. Both tables contain a Package Name that will provide you with the name of the stored procedure that generated the code.

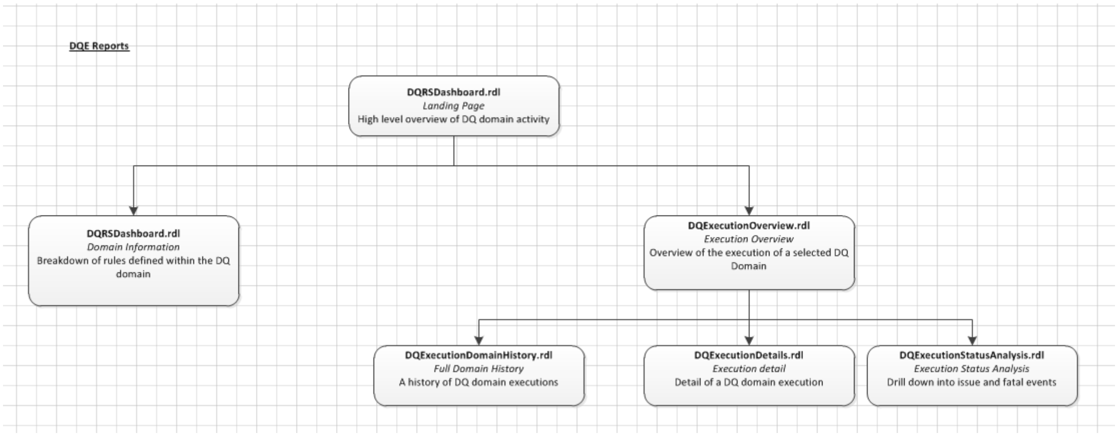
```sql
SELECT RLH.PackageName, RLH.RoutineType, *
FROM [DQ].[RuleExecutionHistory] EH
       INNER JOIN [Audit].[RoutineLoadHistory] RLH
              ON EH.LoadId = RLH.LoadId
WHERE EH.RuleId = 64
ORDER BY RXLOgid asc

SELECT RLH.PackageName, RLH.RoutineType, *
FROM [DQ].[RuleExecutionHistory] EH
       INNER JOIN [Audit].[RoutineLoad] RLH
              ON EH.LoadId = RLH.LoadId
WHERE EH.RuleId = 64
ORDER BY RXLOgid asc
```

The name of the stored procedure together with the RuleSQLDescription should provide you with all of the information needed to track down the underlying code.

5 BI Layer

A generic set of SSRS reports have been developed to report on DQE activities. The reports provide a navigable path through data produced by the DQE. The hierarchy of reports is shown below


DQE Reports
DQRSDashboard.rdl
Landing Page
High level overview of DQ domain activity
DQRSDashboard.rdl
Domain Information
Breakdown of rules defined within the DQ domain
DQExecutionOverview.rdl
Execution Overview
Overview of the execution of a selected DQ Domain
DQExecutionDomainHistory.rdl
Full Domain History
A history of DQ domain executions
DQExecutionDetails.rdl
Execution detail
Detail of a DQ domain execution
DQExecutionStatusAnalysis.rdl
Execution Status Analysis
Drill down into issue and fatal events

5.1 <u>DQRS Dashboard (DQRSDashboard.rdl)</u>

A landing page dashboard has been created to provide a number of high-level graphs. The dashboard provides summary information about

- Domain Information: The number of active DQ Rules within each DQ Domain.
- Execution Fatals: The count of errors encountered in the last execution of each DQ Domain
- Execution Information: The count of full domain executions for each DQ Domain
- Execution Information: The count of stand-alone executions for each DQ Domain

Each graph can be used to drill-down to a lower level of detail.

5.2 <u>DQ Domain Overview (DQDomainOverview.rdl)</u>

This report provides a breakdown of all of the rules configured within the selected DQ Domain.

5.3 DQ Execution Overview (DQExecutionOverview.rdl)

This report can be used to spot issues and errors in the previous executions. This report can be used to quickly identify where errors and issues are being detected and can be used to drill-into more detail.

The report also contains a summary table of recently encountered issues and errors.

5.4 DQ Execution Overview (DQExecutionOverview.rdl)

This report can be used to spot issues and errors in the previous executions. This report can be used to quickly identify where errors and issues are being detected and can be used to drill-into more detail.

The report also contains a summary table of recently encountered issues and errors.

5.5 DQ Execution Details (DQExecutionDetails.rdl)

This report can be used to view the detail of a selected execution. This report provides a list of all DQ rules executed within the domain together with any error messages.

5.6 DQ Status Analysis (DQExecutionStatusAnalysis.rdl)

This report can be used to view the details of any issues or fatal events detected during the execution of the selected DQ Domain.

5.7 DQ Execution Domain History (DQExecutionDomainHistory.rdl)

This report provides a list of all DQ domain executions at a DQ Rule level.

6 Appendix 1

DQE contains a number of different rule categories and each contains a number of rule types. Each rule behaves slightly differently and produces different types of output. In some cases the outputs simply keep a record of the number successful or failure DQ tests. In other cases, the rules will either update the column being evaluated or will write the output to a specific output or status column.

A table of all expected rule behaviours can be found in section 3 of the Technical Overview document.