[MS-DPDQS]:

Data Quality Services Data Portability Overview

Revision Summary

Date	Revision History	Revision Class	Comments
2/23/2012	1.0	New	First release
3/27/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
5/24/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
6/29/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
10/23/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
3/26/2013	1.0	None	No changes to the meaning, language, or formatting of the technical content.
6/11/2013	1.0	None	No changes to the meaning, language, or formatting of the technical content.
8/8/2013	1.0	None	No changes to the meaning, language, or formatting of the technical content.
12/5/2013	1.0	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2014	1.0	None	No changes to the meaning, language, or formatting of the technical content.
5/20/2014	1.0	None	No changes to the meaning, language, or formatting of the technical content.
5/10/2016	1.0	None	No changes to the meaning, language, or formatting of the technical content.
8/16/2017	1.0	None	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

The Data Quality Services Data Portability Overview document provides an overview of data portability scenarios that use the Microsoft SQL Server **Data Quality Services (DQS)** or Management Services client application to export data that is stored in Data Quality Services in a Data Quality Knowledge Base or is stored in a data quality project.

SQL Server Data Quality Services (DQS) is a knowledge-driven solution for creating and maintaining a **Data Quality Knowledge Base (DQKB)** that is used for performing various **data quality (DQ)** operations, such as **data cleansing** and **data matching**. (For more information, see [MSFT-DQSFAQ].)

DQS is a feature in Microsoft SQL Server 2012 that comprises a DQ server and a dedicated DQ client application. DQS also provides a Microsoft SQL Server Integration Services (SSIS) component for data correction, which delivers an integrated, easy-to-use cleansing and matching experience.

DQS enables a self-service data quality experience through a dedicated DQS client UI. A data expert with almost no database expertise can create, maintain, and execute data quality operations with minimal preparation and setup time.

DQS is a combination of two major steps:

- Building and managing knowledge, that is, **knowledge management**. (For more information, see Export DQKB Data (section 2.1).)
- Using knowledge in a **DQ project**, that is, in running a DQ project. (For more information, see Export Project Results Data (section 2.2).)

The following diagram illustrates the DQS process.

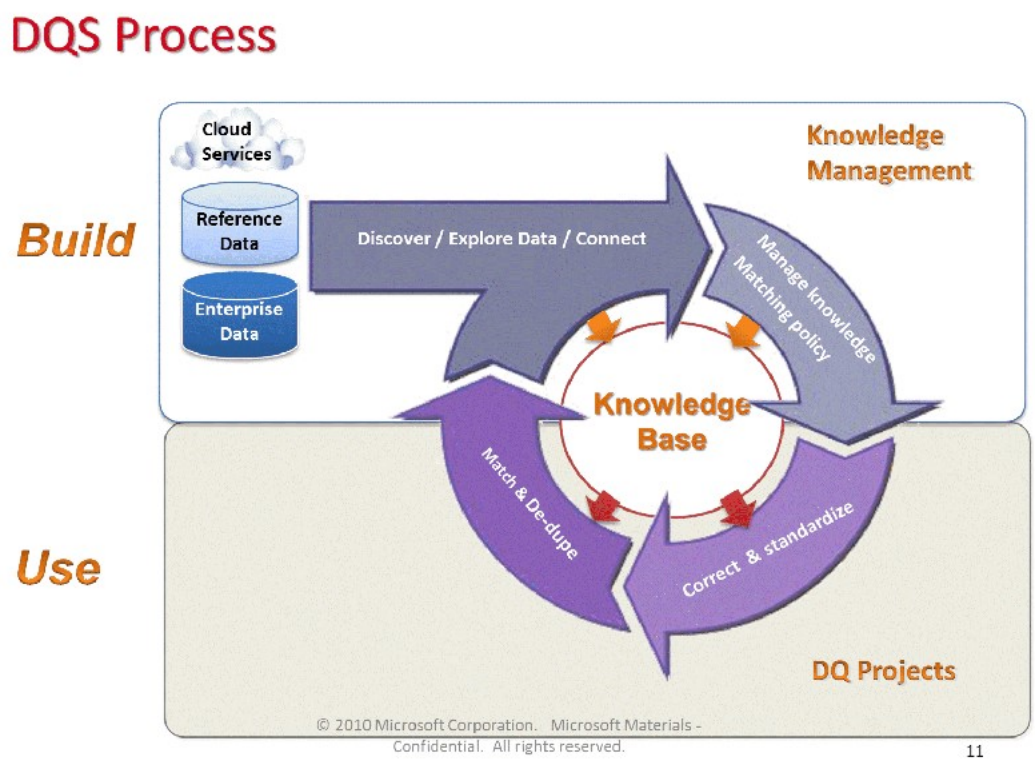

Figure 1: DQS process

This document describes the following scenarios:

- How to export data that is stored in a DQKB to a SQL Server database table. (For information about how to export to a SQL Server database table, see [MS-BCP].) This scenario includes exporting a **matching policy**, **domain** values, term-base-relations, and domain and **composite domain** rules that are named in an **XML** condition. (For more information about an XML condition, see [XML10/5].)
- How to export data that is stored in a DQ project to a SQL Server database table. This scenario includes how to use DQS and the DQ project wizard to export results from a data cleansing or data matching DQ project. (For information about how to export to a SQL Server database table, see [MS-BCP].)

1.1 Glossary

This document uses the following terms:

composite domain: A structure that is composed of a set of domains that share the same subject area. Following are some examples of composite domains: Name: Composed of first name, middle name, and family name; Address: Composed of street, city, state, postal code, and country.

data cleansing: An information scrap-and-rework process that corrects data errors in a collection of data to bring the quality of the data to an acceptable level to meet the information customers' needs. Data cleansing is the act of detecting and correcting or removing corrupt or inaccurate records from a recordset, table, or database.

data matching: A way to compare data so that similar, but slightly different, records can be aligned. Data matching can use "fuzzy logic" to find duplicates in the data. For example, data matching often recognizes that "Bob" and "Robert" might be the same individual. Data matching might be able to detect household connections or find links between husband and wife at the same address.

data quality (DQ): The degree to which the data is suitable for use in the required business processes. The quality of data can be defined, measured, and managed through various data quality metrics, such as completeness, conformity, consistency, accuracy, and duplication. Data quality is achieved through people, technology, and processes.

Data Quality Knowledge Base (DQKB): A repository of metadata that is created by the user or by the Data Quality Services (DQS) platform to improve the quality of data. A DQKB stores all the knowledge that is related to a specific type of data source. For example, one DQKB can handle information on an organization's customer database, while another DQKB handles an employee database.

data quality project (DQ project): A means of using a DQKB to improve the quality of source data by performing data cleansing and/or data matching activities and then exporting the resultant data to a SQL Server database or a comma-separated value (.csv) file.

Data Quality Services (DQS): A knowledge-driven solution for creating and maintaining a DQKB that is used to perform various data quality operations, such as data cleansing and data matching.

data steward: A business user, information worker, or IT professional who improves the quality of data and manages the organization's data quality processes and tasks. The data steward is responsible for improving the reusability, accessibility, and quality of the organization's data assets. The data steward's responsibilities include approving business naming standards,

developing consistent data definitions, determining data aliases and derivations, documenting the business rules of the corporation, and monitoring the quality of the data.

domain: A capture of the data semantics. Example domains include email address, gender, and state.

domain value: A term that is approved by the user as a valid domain value. This term is a word or compound word that is used in a specific context.

knowledge management: The conscious and systematic facilitation of knowledge creation or development, diffusion or transfer, safeguarding, and use at the individual, team, and organizational level.

matching policy: The matching rules that are used to perform data deduplication. The matching-policy process enables matching rules to be created and fine-tuned based on matching results and profiling data. The process also adds the policy to the knowledge base.

term-based relations: A correction to a term that is part of a value in a DQS domain. A term-based relation enables multiple values that are identical except for the spelling of a common part of them to be considered as identical synonyms. For example, a term-based relation might change the term "Inc." to "Incorporated" for every occurrence of the term "Inc." in the domain. In this example, instances of "Contoso, Inc." are changed to "Contoso, Incorporated", and the two values are considered to be exact synonyms.

XML: The Extensible Markup Language, as described in [XML1.0].

XML schema definition (XSD): The World Wide Web Consortium (W3C) standard language that is used in defining XML schemas. Schemas are useful for enforcing structure and constraining the types of data that can be used validly within other XML documents. XML schema definition refers to the fully specified and currently recommended standard for use in authoring XML schemas.

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

[MS-BCP] Microsoft Corporation, "Bulk Copy Format".

[MS-DPBCP] Microsoft Corporation, "Bulk Copy Utility Data Portability Overview".

[MSFT-DQSFAQ] Microsoft Corporation, "Data Quality Services (DQS) FAQ", http://social.technet.microsoft.com/wiki/contents/articles/3919.data-quality-services-dqs-faq.aspx

[XML10/5] Bray, T., Paoli, J., Sperberg-McQueen, C.M., et al., Eds., "Extensible Markup Language (XML) 1.0 (Fifth Edition)", W3C Recommendation, November 2008, http://www.w3.org/TR/2008/REC-xml-20081126/

2 Data Portability Scenarios

The following two data portability scenarios describe how a **data steward** can export data that is stored in a **Data Quality Knowledge Base (DQKB)**. The first scenario describes how to export a DQKB from a SQL query via the Microsoft SQL Server Management Studio client application. Whereas, the second scenario uses the capability that is available in the **Data Quality Services (DQS)** client application to export **data quality (DQ) project** results to a SQL Server database table.

2.1 Export DQKB Data

This scenario describes how a **data steward** can export data that is stored in a **Data Quality Knowledge Base (DQKB)**.

2.1.1 Data Description

The following user data can be stored in a **Data Quality Knowledge Base (DQKB)**:

- **Domain values**
- **Domain term-based relations**
- Domain and **composite domain** rules
- **Matching policies**

2.1.2 Format and Protocol Summary

The following table provides a comprehensive list of the formats and protocols that are used in this data portability scenario.

Protocol or format name	Description	Reference
SQL Server database table	A SQL Server database table	[MS-BCP]
XML	Extensible Markup Language	[XML10/5]

2.1.3 Data Portability Methodology

Export DQKB Data

A **data steward** has created a **Data Quality Knowledge Base (DQKB)** with **domains**, **composite domains**, and potentially, a **matching policy**. The data steward wants to export the DQKB content. To do this, the data steward accesses SQL Server Management Studio, executes a set of simple queries, and then views or exports the data, or both.

The procedures in this section describe how to export DQKB data in this scenario.

Note For more information about how to export data from SQL Server, see [MS-DPBCP].

To Run the Get DQKB Scheme Name Query

To access data that is stored in a DQKB, the DQKB scheme name is first required to be obtained by following these steps:

1. Open the SQL Server Management Studio client application.

2. To find the exact name of the DQKB to be exported, open the DQS client, click **Open Knowledge Base**, and then locate the full name of the DQKB.

3. Execute the following query for getting the knowledge base scheme name. Before running the script, replace all occurrences of *<DQKB_Name>* with the exact name of the DQKB that was found in step 2.

4. Copy the DQKB scheme name from the query results.

   Following is the script for getting the DQKB scheme name.

```
SELECT '[KnowledgeBase' + cast(ID as nvarchar) + ']'
FROM [DQS_MAIN].[dbo].[A_KNOWLEDGEBASE]
WHERE TYPE = 1 and NAME = '<DQKB_Name>'
```

To Run the Get DQKB Domain Values Query

- In SQL Server Management Studio, execute the following Get DQKB Domain Values query. Before running the script, replace all occurrences of *<DQKB_ID>* with the DQKB scheme name. To find the name, execute the DQKB Scheme Name query that is described earlier in this scenario.

  Following is the script for getting the DQKB domain values.

```
SELECT DOMAIN.[NAME] as DOMAIN_NAME,
   VALUE.[VALUE],
   VALUE TYPE.VALUE as TYPE,
   CORRECTION.[TO_TERM_VALUE] as CORRECTION_VALUE,
   LEADING.VALUE as LEADING_VALUE
FROM [DQS_MAIN].<DQKB_ID>.[V_B_TERM] VALUE
  LEFT JOIN [DQS MAIN].<DQKB ID>.[V B TERM] LEADING
ON VALUE.LEADING_EXTENSION_ID = LEADING.TERM_EXTENSION_ID
  LEFT JOIN [DQS MAIN].<DQKB ID>.[V B INDEX LEXICON EXTENSION RELATION] CORRECTION
ON VALUE.TERM_EXTENSION_ID = CORRECTION.FROM_TERM_EXTENSION_ID AND CORRECTION.STATUS
in(3,4)
  LEFT JOIN [DQS_MAIN].<DQKB_ID>.[B_DATA_SERVICE_FIELD] DOMAIN
ON VALUE.FIELD ID = DOMAIN.ID
  LEFT JOIN [DQS MAIN].[dbo].[A CODE MEMBER] VALUE TYPE
ON VALUE_TYPE.[KEY] = VALUE.STATUS AND VALUE_TYPE.GROUP_ID=118
WHERE VALUE.TERM_ID>0
```

For a detailed explanation of the query content, see Appendix A: DQKB Queries Content (section 3).

To Run the Get DQKB Term-Based Relations Query

- In SQL Server Management Studio, execute the following Get DQKB Term-Based Relations query. Before running the script, replace all occurrences of *<DQKB_ID>* with the DQKB scheme name. To find the name, execute the DQKB Scheme Name query that is described earlier in this scenario.

  Following is the script for getting DKQB **term-based relations**.

```
SELECT DOMAIN.NAME as DOMAIN_NAME,
   REL.[FROM VALUE] as VALUE,
   REL.[TO VALUE] as CORRECT TO
FROM [DQS_MAIN].<DQKB_ID>.[B_DOMAIN_TERM_BASED_RELATION] REL
LEFT JOIN [DQS_MAIN].<DQKB_ID>.[B_DATA_SERVICE_FIELD] DOMAIN
ON REL.DOMAIN_ID = DOMAIN.ID
```

For a detailed explanation of the query content, see Appendix A: DQKB Queries Content (section 3).

To Run the Get DQKB Domain and Composite Domain Rules Query

- In SQL Server Management Studio, execute the following Get DQKB Domain and Composite Domain Rules query. Before running the script, replace all occurrences of *<DQKB_ID>* with the DQKB scheme name. To find the name, execute the Get DQKB Scheme Name query that is described earlier in this scenario.

  Following is the script for getting DKQB domain rules.

```
SELECT DOMAIN.NAME as 'DOMAIN/CD_NAME'
  ,CASE WHEN [FUNCTIONAL_TYPE] = 1 THEN 'Domain' ELSE 'Composite Domain' END as TYPE
  ,DOMAIN_RULE.[NAME] as RULE_NAME
  ,DOMAIN RULE.[DESCRIPTION] as RULE DESCRIPTION
  ,[CONDITION]
  ,CASE WHEN [STATUS]=1 THEN 'Active' ELSE 'Not Active' END as STATUS
FROM [DQS_MAIN].<DQKB_ID>.[B_RULE] DOMAIN_RULE
  INNER JOIN [DQS_MAIN].<DQKB_ID>.B_DATA_SERVICE_FIELD DOMAIN
ON DOMAIN_RULE.DOMAIN_ID = DOMAIN.ID
WHERE DOMAIN_RULE.[IS_RULE_COPY] = 0
```

For a detailed explanation of the query content and the XML scheme, see Appendix A: DQKB Queries Content (section 3).

To Run the Get DQKB Matching Policy Query

- In SQL Server Management Studio, execute the following Get DQKB Matching Policy query. Before running the script, replace all occurrences of *<DQKB_ID>* with the DQKB scheme name. To find the name, execute the Get DQKB Scheme Name query that is described earlier in this scenario.

  Following is the script for getting DKQB matching rules.

```
SELECT MATCHING RULE.[NAME]as RULE NAME,
   MATCHING_RULE.[DESCRIPTION],
   MATCHING_RULE.[ORDER] as RULE_ORDER,
   MATCHING_RULE.[MIN_MATCHING_SCORE],
   DOMAIN.NAME as DOMAIN NAME,
   ELM_TYPE.VALUE as DOMAIN_ELEMENT_TYPE,
   ELM.[DOMAIN WEIGHT],
   ELM.[NUMBER_SIMILARITY_DIFFERENCE],
   ELM.[NUMBER_SIMILARITY_PERCENTAGE],
   ELM.[DATE_SIMILARITY_DAYS],
   ELM.[DATE SIMILARITY MONTHS],
   ELM.[DATE_SIMILARITY_YEARS]
FROM [DQS MAIN].<DQKB ID>.[B MATCHING RULE DOMAIN ELEMENT] ELM
  LEFT JOIN [DQS_MAIN].<DQKB_ID>.[B_MATCHING_RULE] MATCHING_RULE
ON MATCHING_RULE.ID = ELM.MATCHING_RULE_ID
  LEFT JOIN [DQS_MAIN].<DQKB_ID>.[B_DATA_SERVICE_FIELD] DOMAIN
ON ELM.DOMAIN ID = DOMAIN.ID
  LEFT JOIN [DQS_MAIN].[dbo].[A_CODE_MEMBER] ELM_TYPE
ON ELM_TYPE.[KEY] = ELM.DOMAIN_ELEMENT_TYPE AND ELM_TYPE.GROUP_ID=138
WHERE ORIGINAL_RULE_ID is null
```

For a detailed explanation of the query content, see Appendix A: DQKB Queries Content (section 3).

2.1.3.1 Preconditions

Read-access permission to the DQS_main database is required to run the queries that are used in this scenario.

The **Data Quality Knowledge Base (DQKB)** is required to be published before the DQKB tables are exported. Unpublished data is not exported.

2.1.3.2 Versioning

This data portability scenario has no special versioning requirements.

2.1.3.3 Error Handling

This data portability scenario has no special error handling requirements.

2.1.3.4 Coherency Requirements

This data portability scenario has no special coherency requirements.

2.1.3.5 Additional Considerations

This data portability scenario has no additional considerations.

2.2 Export Project Results Data

This scenario describes how a **data steward** can export **DQ project** data to a SQL Server database table via **Data Quality Services (DQS)**. The table contains the output results for a **data cleansing** or **data matching** project.

2.2.1 Data Description

The exported project SQL Server database table contains all the data that is captured during the **data quality (DQ)** cleansing or matching project.

The file contains cleansing or matching output results, original source, cleansing or matching reason, status (such as correct or corrected), and confidence (which is the level of certainty for a correction).

2.2.2 Format and Protocol Summary

The following table provides a comprehensive list of the formats and protocols that are used in this data portability scenario.

Protocol or format name	Description	Reference
SQL Server database table	A SQL Server database table	[MS-BCP]

2.2.3 Data Portability Methodology

Export Project Results Data

A **data steward** has run a **data cleansing** or **data matching** project on a source table. After completing the project, the data steward wants to export the cleansing or matching output results. On the results screen, the data steward sees that the cleansing project result can be exported to a SQL Server database table. The data steward selects a database name, enters a table name, and then clicks **Export**.

Note For more information about how to export data from SQL Server, see [MS-DPBCP].

To Export the Results of a Cleansing Project

1. Open the Data Quality Services (DQS) client application.
2. Click **Open Data Quality Projects**, and then select the project to export.

3. In the DQ project wizard, click **Next** until the **View and export cleansing results** page is displayed.

4. In the **Export cleansing results** section (shown here), select the appropriate options for exporting the DQ project, and then click **Export** to export the cleansing project output results data to a standard table that contains the fields that were selected for export. (Following the figure are descriptions of the options that are available for exporting the DQ cleansing project output results data.)

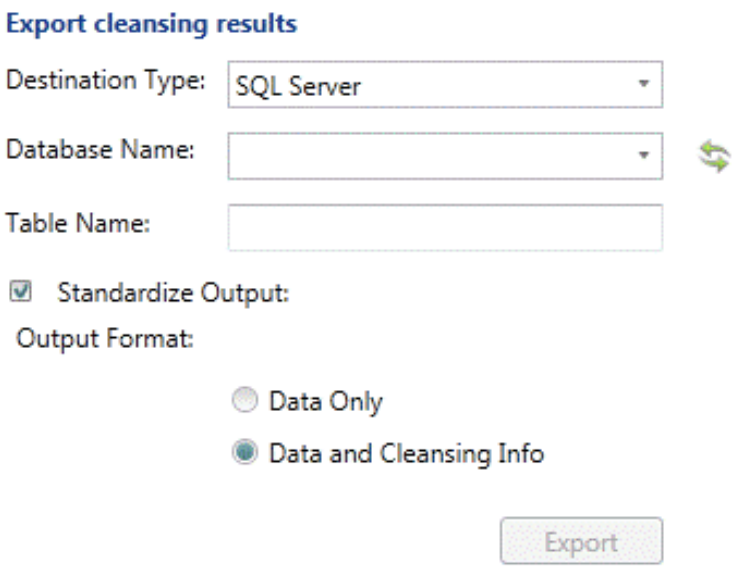

Figure 2: Export cleansing results options

- **Standardize Output:** An option to export the results based on the domain-specified format output property: Upper Case, Lower Case, or Capitalize.
- **Output Format:**
- **Data Only:** The table will have the same structure as the data source, but the table will include only the cleansing output data.
- **Data and Cleansing Info:** The table will include all the columns—such as source, output, reason, status, and confidence—that appear on the interactive cleansing screen for each domain.

To Export the Results of a Matching Project

1. Open the Data Quality Services (DQS) client application.
2. Click **Open Data Quality Projects**, and then select the project to export.
3. In the DQ project wizard, click **Next** until the **Specify the export content and destination** page is displayed.
4. On the **Specify the export content and destination** page, specify the formats in which to export the output results to a SQL Server database table. Choose one or both of the two format

options, and then click **Export** to export the matching project output data to a standard table in the format that was selected for the export. (Following the figure are descriptions of the options that are available for exporting the matching project output results data.)

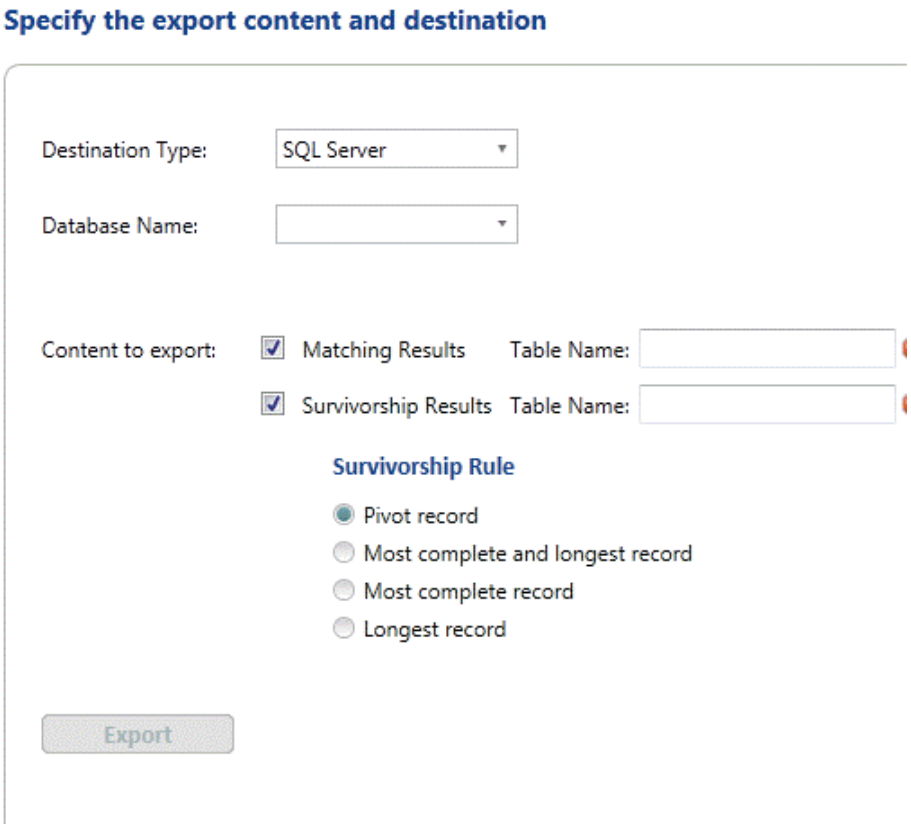

Figure 3: Export matching results options

- **Matching Results:** The table that the Matching Results checkbox generates contains the same data as the source file and contains the following additional columns:
  - **CLUSTER_ID:** An ID that is shared by all the records in the cluster.
  - **RECORD_ID:** A unique identifier for each record.
  - **MATCHING_RULE:** The name of the matching rule that was used to find the match. The value in the MATCHING_RULE column is NULL if the record is a pivot record.
  - **SCORE:** The matching score between the record and the cluster's pivot. The value in the SCORE column is NULL if the record is a pivot record.
  - **PIVOT_MARK:** A value that denotes which record is the pivot. The value in the PIVOT_MARK column is the string "Pivot" if the record is a pivot record. Otherwise, the value in the PIVOT_MARK column is an empty string.
- **Survivorship Results:** The table that the Survivorship Results checkbox generates contains the same data as the source file, but the table does not contain any duplicate records. Duplicate records are removed based on the selected Survivorship Rule.
- **Survivorship Rule:** The rule that is selected determines for each matching cluster which of the records is the surviving record. All other records are considered to be duplicates and are deleted from the survivorship results list. The Survivorship Rule includes the following options:

- **Pivot record (the default):** Identifies the surviving record as the initial pivot record chosen arbitrarily by DQS.
- **Most complete and longest record:** Identifies the surviving record as the record that has the largest number of populated fields and the largest number of terms in each field. All source fields are checked, including those fields that were not mapped to a domain on the Map page.
- **Most complete record:** Identifies the surviving record as the one that has the largest number of populated fields. A populated field contains at least one value (string, numeric, or both). All source fields are checked, including those fields that were not mapped to a domain on the Map page.
- **Longest record:** Identifies the surviving record as the one that has the largest number of terms in its source field. To determine the length of each record, DQS verifies the length of the terms in all source fields, including those fields that were not mapped to a domain on the Map page.

2.2.3.1 Preconditions

The user is required to be authorized to access the **data quality (DQ) project**, and the project is required to pass the cleanse/match analyzing step.

2.2.3.2 Versioning

This data portability scenario has no special versioning requirements.

2.2.3.3 Error Handling

This data portability scenario has no special error handling requirements.

2.2.3.4 Coherency Requirements

This data portability scenario has no special coherency requirements.

2.2.3.5 Additional Considerations

This data portability scenario has no additional considerations.

3 Appendix A: DQKB Queries Content

Get DQKB Domain Values Query

The Get DQKB Domain Values query returns a list of values that are stored in the **domains** of the selected **Data Quality Knowledge Base (DQKB)**. This query returns the following output columns.

DOMAIN_NAME	VALUE	TYPE	CORRECTION_VALUE	LEADING_VALUE
Home Arena	Minute Maid Park	Correct	NULL	NULL
Home Arena	Madison Square Garden	Correct	NULL	NULL
Home Arena	Oriole Park at Camden Yards	Correct	NULL	NULL
Home Arena	Turner Field	Correct	NULL	NULL
Home Arena	Comerica Park	Correct	NULL	NULL
Home Arena	Pepsi C.	Correct	NULL	Pepsi Center
Home Arena	Pepsi Cntr	Correct	NULL	Pepsi Center
Home Arena	Amercian Airlines Center	Error	American Airlines Center	American Airlines Center
Home Arena	Giants Stadium	Correct	NULL	NULL
Home Arena	American Airlines Center	Correct	NULL	American Airlines Center
Sports Type	Hockey	Correct	NULL	Hockey
Sports Type	NFL	Correct	NULL	Football

Figure 4: Output columns of the Get DQKB Domain Values query

Following are descriptions of these columns:

- **DOMAIN_NAME:** The name of the domain in which the **domain value** is stored.
- **VALUE:** The DQKB domain values that were added to the domain.
- **TYPE:** The status of the VALUE field data, such as Correct, Error, or Invalid.
- **CORRECTION_VALUE:** The value that the original value is changed to if the TYPE field contains Error or Invalid.
- **LEADING_VALUE:** The synonym that the original value (VALUE column) will be changed to.

Get DQKB Term-Based Relations Query

The Get DQKB Term-Based Relations query contains a list of the **term-based relation** values that are stored in the domains of the selected DQKB. This query returns the following output columns.

DOMAIN_NAME	VALUE	CORRECT_TO
Team Name	LA	Los Angeles
Team Name	NY	New York
Home Arena	C.	Center
Home Arena	G.	Garden
Home Arena	S.	Stadium
Home Arena	P.	Park
Home Arena	A.	Arena
Home Arena	Cntr	Center
Home Arena	Ar	Arena

Figure 5: Output columns of the Get DQKB Term-Based Relations query

Following are descriptions of these columns:

- **DOMAIN_NAME:** The name of the domain in which the term-based relation is stored.
- **VALUE:** The term-based relation term that is replaced in the selected domain.
- **CORRECT_TO:** The term-based relation term that the VALUE field data is replaced with.

Get DQKB Domain and Composite Domain Rules Query

The Get DQKB Domain and Composite Domain Rules query contains a list of rules that are stored in the domains and **composite domains** of the selected DQKB. This query returns the following output columns.

DOMAIN/CD_NAME	TYPE	RULE_NAME	RULE_DESCRIPTION	CONDITION	STATUS
Team Type CD	Composite Domain	San Francisco Giants		<Microsoft.Ssdqs.DomainRules.Define.Condition.DB...	Active
Team Type CD	Composite Domain	New York Giants		<Microsoft.Ssdqs.DomainRules.Define.Condition.DB...	Active
Account ID	Domain	Account Validation	Length and Type validation	<Microsoft.Ssdqs.DomainRules.Define.Condition.DB...	Active

Figure 6: Output columns of the Get DQKB Domain and Composite Domain Rules query

Following are descriptions of these columns:

- **DOMAIN/CD_NAME:** The name of the domain or composite domain that the rule is a part of.
- **TYPE:** The type of the rule. The type value is either Domain or Composite Domain.
- **RULE_NAME:** The name of the rule.
- **DESCRIPTION:** The description of the rule.
- **CONDITION:** An **XML** file that contains the rule structure. See the following **XSD** that governs the rule scheme.
- **STATUS:** The status of the rule. The status value is either Active or Inactive.

The rule's XML condition (CONDITION field data) that is generated by **Data Quality Services (DQS)** conforms to the following XSD.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:drs="http://microsoft.com/schemas/SQL/DQS/DomainRules"
           targetNamespace="http://microsoft.com/schemas/SQL/DQS/DomainRules">

    <xs:element name="Microsoft.Ssdqs.DomainRules.Define.Condition.DBStorageCondition"
type="drs:DBStorageCondition" />

    <xs:complexType name="DBStorageCondition">
        <xs:choice minOccurs="1" maxOccurs="1">
            <xs:group ref="drs:AnyDomainRuleGroup" minOccurs="1" maxOccurs="1" />
            <xs:element
name="Microsoft.Ssdqs.DomainRules.Define.Condition.CrossDomain.CrossDomainRule"
type="drs:CrossDomainRule" />
        </xs:choice>
    </xs:complexType>

    <xs:complexType name="ComplexRuleCondition">
        <xs:sequence>
            <xs:element name="Conditions">
                <xs:complexType>
                    <xs:sequence>
```

```
                        <xs:element minOccurs="2" maxOccurs="unbounded" name="Condition">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="PrefixOperation">
                                        <xs:simpleType>
                                            <xs:restriction base="xs:string">
                                                <xs:enumeration value="None"/>
                                                <xs:enumeration value="And"/>
                                                <xs:enumeration value="Or"/>
                                            </xs:restriction>
                                        </xs:simpleType>
                                    </xs:element>
                                    <xs:group ref="drs:AnyDomainRuleGroup" minOccurs="1"
maxOccurs="1" />
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AnyDomainRuleType">
        <xs:group ref="drs:AnyDomainRuleGroup"/>
    </xs:complexType>

    <xs:group name="AnyDomainRuleGroup">
        <xs:choice>
            <xs:element
name="Microsoft.Ssdqs.DomainRules.Define.Condition.ComplexRuleCondition"
type="drs:ComplexRuleCondition" />
            <xs:element
name="Microsoft.Ssdqs.DomainRules.Define.Condition.IsEmptyRuleCondition"
type="drs:IsEmptyRuleCondition" />
            <xs:element
name="Microsoft.Ssdqs.DomainRules.Define.Condition.RegularExpressionRuleCondition"
type="drs:RegularExpressionRuleCondition" />
            <xs:element
name="Microsoft.Ssdqs.DomainRules.Define.Condition.PatternRuleCondition"
type="drs:PatternRuleCondition" />
            <xs:element
name="Microsoft.Ssdqs.DomainRules.Define.Condition.InListOfValuesRuleCondition"
type="drs:InListOfValuesRuleCondition" />
            <xs:element
name="Microsoft.Ssdqs.DomainRules.Define.Condition.DataTypeRuleCondition"
type="drs:DataTypeRuleCondition" />
            <xs:element
name="Microsoft.Ssdqs.DomainRules.Define.Condition.StringValueRuleCondition"
type="drs:StringValueRuleCondition" />
            <xs:element
name="Microsoft.Ssdqs.DomainRules.Define.Condition.LengthRuleCondition"
type="drs:LengthRuleCondition" />
            <xs:element
name="Microsoft.Ssdqs.DomainRules.Define.Condition.NumericRuleCondition"
type="drs:NumericRuleCondition" />
            <xs:element name="Microsoft.Ssdqs.DomainRules.Define.Condition.DateRuleCondition"
type="drs:DateRuleCondition" />
        </xs:choice>
    </xs:group>

    <xs:complexType name="IsEmptyRuleCondition">
        <xs:sequence>
            <xs:element name="ConditionType">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="IsValueEmpty"/>
                        <xs:enumeration value="IsValueNotEmpty"/>
                    </xs:restriction>
```

```
                </xs:simpleType>
            </xs:element>
            <xs:element name="DataFieldType">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="String"/>
                        <xs:enumeration value="Date"/>
                        <xs:enumeration value="Integer"/>
                        <xs:enumeration value="Decimal"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="RegularExpressionRuleCondition">
        <xs:sequence>
            <xs:element name="ConditionType">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="MatchesRegularExpression"/>
                        <xs:enumeration value="DoesNotMatchRegularExpression"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="RegularExpression" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="PatternRuleCondition">
        <xs:sequence>
            <xs:element name="ConditionType">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="EqualsPattern"/>
                        <xs:enumeration value="ContainsPattern"/>
                        <xs:enumeration value="DoesNotEqualPattern"/>
                        <xs:enumeration value="DoesNotContainPattern"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="RegularExpression" />
            <xs:element name="MatchingSample" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="InListOfValuesRuleCondition">
        <xs:sequence>
            <xs:element name="ConditionType">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="ValueIsInList"/>
                        <xs:enumeration value="ValueIsNotInList"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="Parameter">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="List">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element minOccurs="2" maxOccurs="unbounded"
name="Value" type="xs:string" />
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
```

```
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="DataTypeRuleCondition">
    <xs:sequence>
        <xs:element name="ConditionType">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="IsNumericValue"/>
                    <xs:enumeration value="IsDateValue"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="StringValueRuleCondition">
    <xs:sequence>
        <xs:element name="ConditionType">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="ValueEqualsTo"/>
                    <xs:enumeration value="ValueNotEqualsTo"/>
                    <xs:enumeration value="ValueBeginsWith"/>
                    <xs:enumeration value="ValueEndsWith"/>
                    <xs:enumeration value="ValueContains"/>
                    <xs:enumeration value="ValueDoesNotContain"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="Parameter" type="xs:string" />
    </xs:sequence>
</xs:complexType>

<xs:complexType name="LengthRuleCondition">
    <xs:sequence>
        <xs:element name="ConditionType">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="LengthEqualsTo"/>
                    <xs:enumeration value="LengthIsGreaterThanOrEqualsTo"/>
                    <xs:enumeration value="LengthIsLessThanOrEqualsTo"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="Parameter" type="xs:unsignedInt" />
    </xs:sequence>
</xs:complexType>

<xs:complexType name="NumericRuleCondition">
    <xs:sequence>
        <xs:element name="ConditionType">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="ValueIsGreaterThan"/>
                    <xs:enumeration value="ValueIsLessThan"/>
                    <xs:enumeration value="ValueIsGreaterThanOrEqualsTo"/>
                    <xs:enumeration value="ValueIsLessThanOrEqualsTo"/>
                    <xs:enumeration value="ValueIsEqualTo"/>
                    <xs:enumeration value="ValueIsNotEqualTo"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="Parameter" type="xs:double" />
    </xs:sequence>
</xs:complexType>

<xs:complexType name="DateRuleCondition">
```

```
        <xs:sequence>
            <xs:element name="ConditionType">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="ValueIsGreaterThan"/>
                        <xs:enumeration value="ValueIsLessThan"/>
                        <xs:enumeration value="ValueIsGreaterThanOrEqualsTo"/>
                        <xs:enumeration value="ValueIsLessThanOrEqualsTo"/>
                        <xs:enumeration value="ValueIsEqualTo"/>
                        <xs:enumeration value="ValueIsNotEqualTo"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="Parameter" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="CrossDomainRule">
        <xs:sequence>
            <xs:element name="Condition">
                <xs:complexType>
                    <xs:choice minOccurs="1" maxOccurs="1">
                        <xs:element
name="Microsoft.Ssdqs.DomainRules.Define.Condition.CrossDomain.CrossDomainRuleCondition"
type="drs:CrossDomainRuleCondition" />
                        <xs:element
name="Microsoft.Ssdqs.DomainRules.Define.Condition.CrossDomain.ComplexCrossDomainRuleConditio
n" type="drs:ComplexCrossDomainRuleCondition" />
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <xs:element name="Conclusions">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element minOccurs="1" maxOccurs="unbounded" name="Conclusion">
                            <xs:complexType>
                                <xs:sequence minOccurs="1" maxOccurs="1">
                                    <xs:element
name="Microsoft.Ssdqs.DomainRules.Define.Condition.CrossDomain.CrossDomainRuleCondition"
type="drs:CrossDomainRuleCondition" />
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ComplexCrossDomainRuleCondition">
        <xs:sequence>
            <xs:element name="Conditions">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element minOccurs="1" maxOccurs="unbounded" name="Condition">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="PrefixOperation">
                                        <xs:simpleType>
                                            <xs:restriction base="xs:string">
                                                <xs:enumeration value="None"/>
                                                <xs:enumeration value="And"/>
                                                <xs:enumeration value="Or"/>
                                            </xs:restriction>
                                        </xs:simpleType>
                                    </xs:element>
                                    <xs:element
name="Microsoft.Ssdqs.DomainRules.Define.Condition.CrossDomain.CrossDomainRuleCondition"
type="drs:CrossDomainRuleCondition" />
```

```
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="CrossDomainRuleCondition">
        <xs:sequence>
            <xs:element name="DomainId" type="xs:unsignedInt" />
            <xs:element name="Condition" type="drs:AnyDomainRuleType" />
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Get DQKB Matching Policy Query

The Get DQKB Matching Policy query contains the matching policy of the selected DQKB. Each policy is comprised of one or more matching rules. This query returns the following output columns.

RULE_NAME	DESCRIPTION	RULE_ORDER	MIN_MATCHING_SCORE	DOMAIN_NAME	DOMAIN_ELEMENT_TYPE	DOMAIN_WEIGHT	NUMBER_SIMILARITY_DIFFERENCE	NU
Dates + detailes + Name		1	80	Birth Date	Prerequisite	0	NULL	NU
Dates + detailes + Name		1	80	Hire Date	Exact	10	NULL	NU
Dates + detailes + Name		1	80	Employee Name	Similar	30	NULL	NU
Dates + detailes + Name		1	80	Job Title	Similar	20	NULL	NU
Dates + detailes + Name		1	80	Gender	Similar	40	NULL	NU

Figure 7: Output columns of the Get DQKB Matching Policy query

Following are descriptions of these columns:

- **RULE_NAME:** The name of the matching rule.
- **DESCRIPTION:** The description of the matching rule.
- **RULE_ORDER:** The order of execution of the rule within the policy (for example, the first rule will have a value of "1").
- **MIN_MATCHING_SCORE:** The threshold at or above which two records are considered to be a match.
- **DOMAIN_NAME:** The name of the domain or composite domain that is used in the rule.
- **DOMAIN_ELEMENT_TYPE:** The type of contribution that the domain will have in the matching analysis, such as Prerequisite, Similar, or Exact.
- **DOMAIN_WEIGHT:** A numeric weight that determines the contribution of the domain to the overall matching score.
- **NUMBER_SIMILARITY_DIFFERENCE:** An integer tolerance that can be added to a numeric domain only if the DOMAIN_ELEMENT_TYPE field contains Similar.
- **NUMBER_SIMILARITY_PERCENTAGE:** A percentage tolerance that can be added to a numeric domain only if the DOMAIN_ELEMENT_TYPE field contains Similar.
- **DATE_SIMILARITY_DAYS:** An integer tolerance that represents day tolerance and that can be added to a date domain only if the DOMAIN_ELEMENT_TYPE field contains Similar.
- **DATE_SIMILARITY_MONTHS:** An integer tolerance that represents month tolerance and that can be added to a date domain only if the DOMAIN_ELEMENT_TYPE field contains Similar.

- **DATE_SIMILARITY_YEARS:** An integer tolerance that represents year tolerance and that can be added to a date domain only if the DOMAIN_ELEMENT_TYPE field contains Similar.

4 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

5 Index