311-CD-102-005

EOSDIS Core System Project

Release 4
Data Management Subsystem Database Design and Database Schema Specifications for the ECS Project

Final

March 1999

This document has not yet been approved by the Government for general use or distribution.

Raytheon Systems Company
Upper Marlboro, Maryland

Release 4
Data Management Subsystem Database Design and Database Schema Specifications for the ECS Project

Final

March 1999

Prepared Under Contract NAS5-60000
CDRL Item #050

RESPONSIBLE ENGINEER

Maureen Muganda /s/ 2/26/99
Maureen Muganda Date
EOSDIS Core System Project

SUBMITTED BY

Mary Armstrong /s/ 2/26/99
Mary Armstrong, Development Engineering Manager Date
EOSDIS Core System Project

Raytheon Systems Company
Upper Marlboro, Maryland

311-CD-102-005

This page intentionally left blank.

Preface

This document describes the database design and database schema specifications for the Data Management (DM) Subsystem. It is one of eight documents comprising the detailed database design and database schema specifications for the as-delivered ECS Subsystems. A complete list of the ten documents follows:

311-CD-102 Data Management (DM) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-103 Ingest Subsystem (INS) Database Design and Database Schema Specifications for the ECS Project

311-CD-104 Interoperability Subsystem (IOS) Database Design and Database Schema Specifications for the ECS Project

311-CD-105 Management Support Subsystem (MSS) Database Design and Database Schema Specifications for the ECS Project

311-CD-106 Planning and Data Processing Subsystem (PDPS) Database Design and Database Schema Specifications for the ECS Project

311-CD-107 Science Data Server (SDSRV) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-108 Storage Management (STMGT) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-109 Subscription Server (SUBSRV) Subsystem Database Design and Database Schema Specifications for the ECS Project

This submittal meets the milestone specified in the Contract Data Requirements List (CDRL) of NASA Contract NAS5-60000. It is a formal contract deliverable with an approval code 1. It requires Government review and approval prior to acceptance and use. This document is under ECS contractor configuration control. Once approved, contractor approved changes will be handled in accordance with Class I and Class II change control requirements described in the EOS Configuration Management Plan. Changes to this document shall be made by Document Change Notice (DCN) or by complete revision.

Entity relationship diagrams (ERDs) presented in this document have been exported directly from software tools and in some cases contain too much detail to be easily readable within hard copy page constraints. The reader is encouraged to view these diagrams in portable document format (PDF) on the ECS Data Handling System (EDHS) world wide web (WWW) site. The universal resource locator (URL) is: http://edhs1.gsfc.nasa.gov.

Any questions should be addressed to:

Data Management Office
The ECS Project Office
Raytheon Systems Company
1616 McCormick Drive
Upper Marlboro, MD 20774-5301

Abstract

This document outlines the Release 4 Drop 4PX "as-built" database design and database schema specifications for the Data Management (DM) Subsystem. It includes the entity relationship diagram (ERD), physical database table definitions, and database software listings for triggers, procedures, and scripts. The ERD describes data entities and the association between these entities used within the DM Subsystem. Other information is also included to support database installation and life cycle maintenance.

Keywords: data, database, design, specifications, configuration, installation, parameters, scripts, security, data model, replication, performance tuning, SQL server, Sybase, database security, triggers, procedures, scripts.

This page intentionally left blank.

Change Information Page

List of Effective Pages	
Page Number	**Issue**
Title	Submitted as Final
iii through xii	Submitted as Final
1-1 through 1-2	Submitted as Final
2-1 through 2-2	Submitted as Final
3-1 through 3-2	Submitted as Final
4-1 through 4-172	Submitted as Final
5-1 through 5-2	Submitted as Final
6-1 through 6-4	Submitted as Final
7-1 through 7-2	Submitted as Final
AB-1 through AB-2	Submitted as Final

Document History			
Document Number	**Status/Issue**	**Publication Date**	**CCR Number**
311-CD-102-001	Draft	January 1998	97-1755
311-CD-102-002	Draft	May 1998	98-0620
311-CD-102-003	Draft	July 1998	98-0866
311-CD-102-004	Draft	December 1998	98-1267
311-CD-102-005	Submitted as Final	March 1999	99-0166

This page intentionally left blank.

Contents

Preface

Abstract

1. Introduction

1.1 Identification 1-1

1.2 Scope 1-1

1.3 Purpose 1-1

1.4 Audience 1-1

2. Related Documents

2.1 Applicable Documents 2-1

2.2 Information Documents 2-2

3. Database Configurations

3.1 Server Configurations 3-1

3.2 Storage Device Layouts 3-1

4. Database Design

4.1 Design Overview 4-1

4.1.1 Physical Data Model Entity Relationship Diagram 4-1

4.1.2 Database Table Specifications 4-5

4.1.3 Column Specifications 4-26

4.1.4 Column Domains 4-51

4.1.5 Column Default Values 4-52

4.1.6 Referential Integrity Rules 4-52

4.1.7 Views 4-52

4.1.8 Declarative Integrity Constraints 4-53

4.1.9 Triggers 4-60

4.1.10 Stored Procedures 4-127

4.2 Flat File Usage 4-162

4.2.1 File Descriptions 4-163

4.2.2 Field Specifications 4-167

4.2.3 Domain Definitions 4-170

5. Performance and Tuning Factors

5.1 Indexes 5-1

5.2 Segments 5-1

5.3 Named Caches 5-1

6. Database Security

6.1 Approach 6-1

6.2 Initial Users 6-2

6.3 Groups 6-2

6.4 Objects 6-2

6.5 Roles 6-3

7. Scripts

7.1 Installation Scripts 7-1
7.2 De-Installation Scripts 7-1
7.3 Backup and Recovery Scripts 7-2
7.4 Miscellaneous Scripts 7-2

Figures

4-1 ERD Key 4-2
4-2 Data Management ERD 4-3
6-1 Sybase General Approach to SQL Server Security 6-1

Tables

4-1 Database Tables 4-5
4-4 Trigger Descriptions 4-60
4-5 Summary List of Stored Procedures 4-127
4-6 Flat File Descriptions 4-163
4-7 Flat File Field Specifications 4-167
5-1 Segment Descriptions 5-1
6-1 Permission Key 6-2
6-2 Object Specifications 6-3
7-1 Installation Scripts 7-1
7-2 De-Installation Scripts 7-1
7-3 Backup and Recovery Scripts 7-2
7-4 Miscellaneous Scripts and Input Data Files 7-2

Abbreviations and Acronyms

This page intentionally left blank

1. Introduction

1.1 Identification

This Data Management (DM) Subsystem Database Design and Database Schema Specifications document, Contract Data Requirement List (CDRL) Item Number 050, whose requirements are specified in Data Item Description (DID) 311/DV1, is a required deliverable under the Earth Observing System (EOS) Data and Information System (EOSDIS) Core System (ECS), Contract NAS5-60000.

1.2 Scope

The DM Subsystem Database Design and Database Schema Specifications document describes the database that supports data requirements for the DM Subsystem, Release 4 Drop 4PX.

1.3 Purpose

The purpose of the DM Subsystem Database Design and Database Schema Specifications document is to support the administrators of the DM Subsystem database throughout its life-cycle. Additionally, this document communicates the database specifications in sufficient detail to support other ongoing installation and operational activities (e.g., configuration management, data administration, system installation and maintenance).

1.4 Audience

The DM Subsystem Database Design and Database Schema Specifications document is intended to be used and maintained by the ECS maintenance and operations staff. The document is organized as follows:

Section 1 provides information regarding the identification, purpose, scope, and audience.

Section 2 provides a listing of related documents used to develop this document.

Section 3 specifies the DM Subsystem database physical architecture.

Section 4 contains an overview of the database design, including the entity relationship diagram (ERD) representing the physical data model, the database tables and columns, flat files and fields, triggers, and stored procedures.

Section 5 provides a description of database performance and tuning features, i.e., indexes, caches, and data segments for the DM Subsystem database implementation.

Section 6 provides a high level description of the preliminary security infrastructure including listings of anticipated users, groups, and permissions expected for preliminary operational use.

Section 7 provides listings of the scripts used for database installation, de-installation, backup and recovery, and other miscellaneous administration functions. This page intentionally left blank.

2. Related Documents

2.1 Applicable Documents

The following documents, including Internet links, are referenced in this document, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this volume.

305-CD-100	Release 4 Segment Design Specification for the ECS Project
920-TDG-009	DAAC Hardware Database Mapping/GSFC
920-TDN- 009	DAAC Hardware Database Mapping/NSIDC
920-TDE-009	DAAC Hardware Database Mapping/EDC
920-TDL-009	DAAC Hardware Database Mapping/LARC
920-TDS-009	DAAC Hardware Database Mapping/SMC
920-TDG-010	DAAC Database Configuration/GSFC
920-TDN-010	DAAC Database Configuration/NSIDC
920-TDE-010	DAAC Database Configuration/EDC
920-TDL-010	DAAC Database Configuration/LARC
920-TDS-010	DAAC Database Configuration/SMC
920-TDG-011	DAAC Sybase Log Mapping/GSFC
920-TDN-011	DAAC Sybase Log Mapping/NSIDC
920-TDE-011	DAAC Sybase Log Mapping/EDC
920-TDL-011	DAAC Sybase Log Mapping/LARC
920-TDS-011	DAAC Sybase Log Mapping/SMC
922-TDG-013	Disk Partitions/GSFC
922-TDN-013	Disk Partitions/NSIDC
922-TDL-013	Disk Partitions/LARC
922-TDS-013	Disk Partitions/SMC

These documents are maintained as part of the ECS baseline and available on the World Wide Web at the URL: http://cmdm.east.hitc.com/baseline. Please note that this is a partial mirror site

in that some items are not available (they are identified) since this is OPEN to all. This site may also be reached through the EDHS homepage. Scroll page to the connections line and click on the ECS Baseline Information System link.

2.2 Information Documents

The following documents, although not directly applicable, amplify or clarify the information presented in this document. These documents are not binding on this document.

313-CD-006	Release 4 CSMS/SDPS Internal ICD for the ECS Project
609-CD-003	Release 4 Operations Tools Manual for the ECS Project
611-CD-004	Release 4 Mission Operation Procedures for the ECS Project

These documents are accessible via the EDHS homepage.

3. Database Configurations

3.1 Server Configurations

The database configuration of the server varies from DAAC to DAAC based on individualized DAAC requirements and hardware availability. These DAAC-specific database configurations are detailed on the following documents:

920-TDG-009	DAAC Hardware Database Mapping/GSFC
920-TDN- 009	DAAC Hardware Database Mapping/NSIDC
920-TDE-009	DAAC Hardware Database Mapping/EDC
920-TDL-009	DAAC Hardware Database Mapping/LARC
920-TDS-009	DAAC Hardware Database Mapping/SMC
920-TDG-011	DAAC Sybase Log Mapping/GSFC
920-TDN-011	DAAC Sybase Log Mapping/NSIDC
920-TDE-011	DAAC Sybase Log Mapping/EDC
920-TDL-011	DAAC Sybase Log Mapping/LARC
920-TDS-011	DAAC Sybase Log Mapping/SMC

These documents are maintained as part of the ECS baseline and available on the world-wide web at the URL: http://cmdm.east.hitc.com/baseline. Please note that this is a partial mirror site in that some items are not available (they are identified) since this is OPEN to all. This site maye also be reached through the EDHS homepage. Scroll page to the connections line and click on the ECS Baseline Information System link.

3.2 Storage Device Layouts

Storage Device layouts, disk partitions, vary from DAAC to DAAC based on the amount of data storage expected to be needed to accommodate a particular DAAC's storage requirements. Disk partitions for the PDPS server at each DAAC is detailed in the following documents:

922-TDG-013	Disk Partitions/GSFC
922-TDN-013	Disk Partitions/NSIDC
922-TDE-013	Disk Partitions/EDC

922-TDL-013 Disk Partitions/LARC

922-TDS-013 Disk Partitions/SMC

These documents are maintained as part of the ECS baseline and available on the World Wide Web at the URL: http://cmdm.east.hitc.com/baseline. Please note that this is a partial mirror site in that some items are not available (they are identified) since this is OPEN to all. This site may also be reached through the EDHS homepage. Scroll page to the connections line and click on the ECS Baseline Information System link.

4. Database Design

4.1 Design Overview

The DM Subsystem database schema implements a majority of the persistent data requirements for the DM Subsystem. Other non-database data requirements, used for system support, are implemented in flat files—see Section 4.2 for descriptions of these flat files. The database is designed to satisfy DM Subsystem business rules while maintaining data integrity, consistency, and performance. DM Subsystem database tables are implemented using the Sybase Relational Database Management System (RDBMS). All components of the DM Subsystem database are described in the following sections; information is presented in sufficient detail to support operational needs.

4.1.1 Physical Data Model Entity Relationship Diagram

An entity relationship diagram (ERD) was developed for use as a "roadmap" to the DM Subsystem database. An ERD is a schematic of the physical data structure which illustrates the dependencies and relationships between database entities, i.e., tables. On ERDs, database entities are represented by rectangles and relationships are represented by arrows as shown by the key in Figure 4-1. Details on the syntax used by the *S-Designor Data Architect* Computer Aided Software Engineering (CASE) tool may be found in the *Powersoft: S-Designor for PowerBuilder* Reference Guide. The ERD presented in Figure 4-2 for the DM Subsystem database was produced using the *S-Designor* tool.

The ECS Conceptual Model for the Science Data Processing Segment (SDPS) was developed using an Object Oriented (OO) CASE tool. However; since Sybase implements a Relational Database Management System (RDBMS) with an Object wrapper, the syntax (model notation) is converted from OO to relational and the terminology changes—the "attribute" becomes "column" and "class" becomes "table." Since the specifications of some entities in this document are transferred from the OO Conceptual Model repository, there are many cases where the OO terminology is retained—as, for example, in the table and column names and definitions.

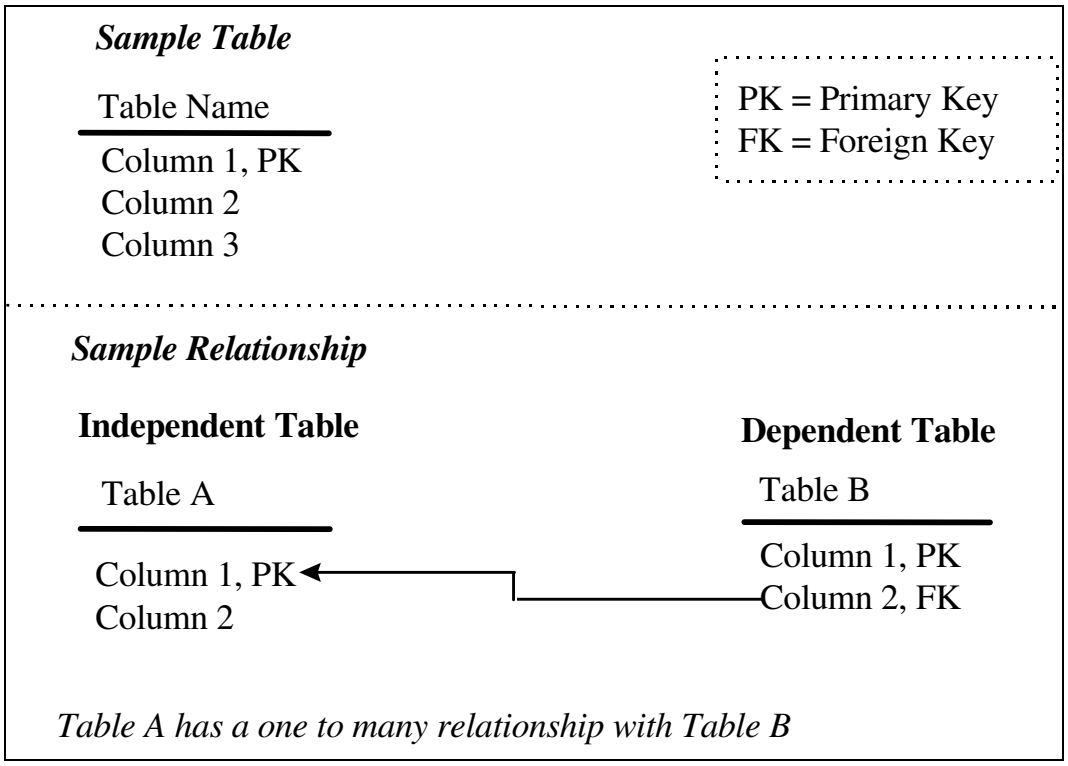

Figure 4-1. ERD Key

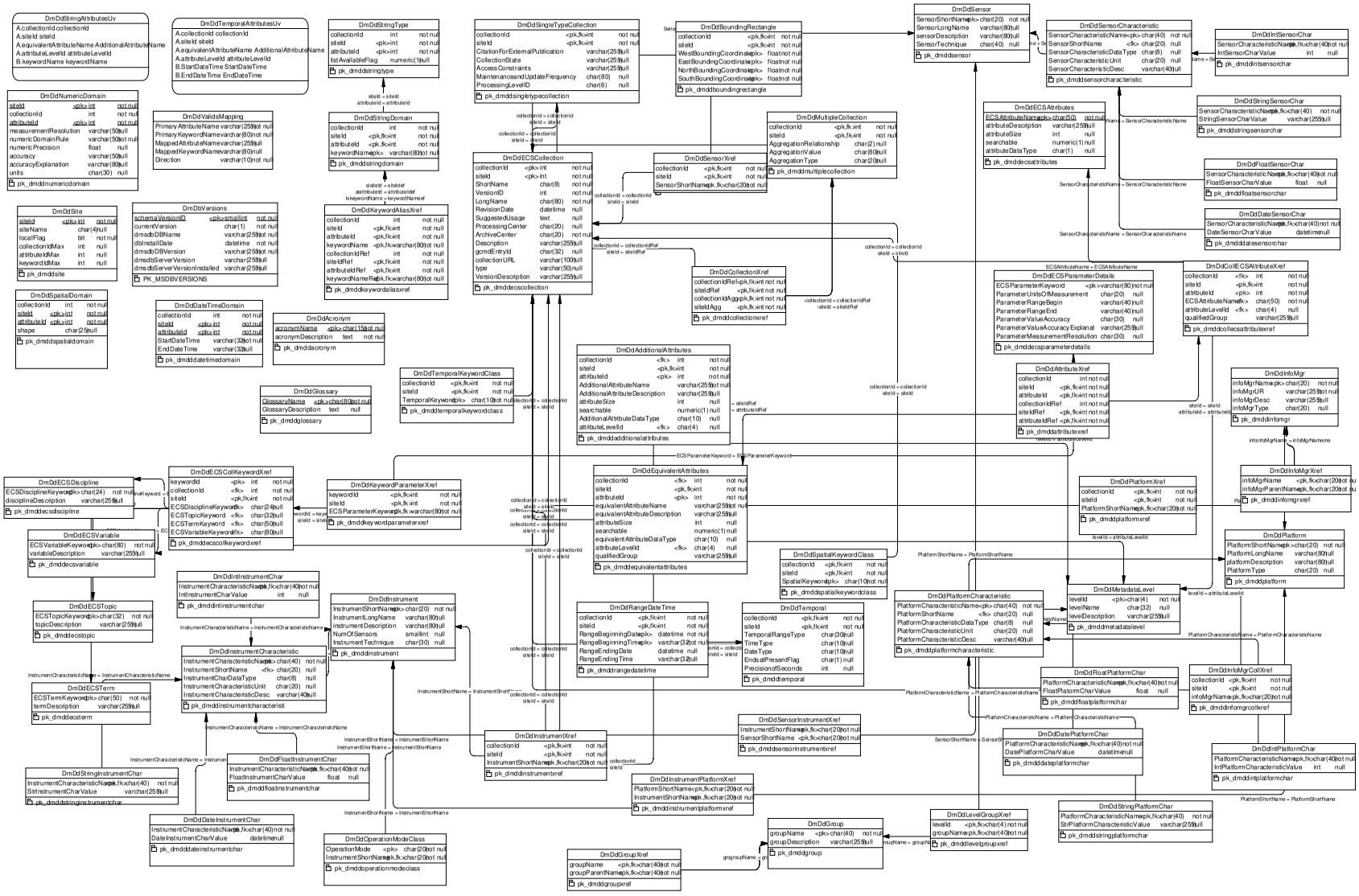

Figure 4-2. Data Management ERD

4.1.2 Database Table Specifications

Table 4-1 contains a listing of all the database tables within the DM Subsystem database. This list is presented in alphabetical order corresponding to the database tables illustrated in the ERD (reference Figure 4-2). The database tables listed immediately following Table 4-1 are presented in the same order as the table.

Table 4-1. Database Tables

Table Name	Table Name
DmDdAcronym	DmDdInstrumentXref
DmDdAdditionalAttributes	DmDdIntInstrumentChar
DmDdAttributeXref	DmDdIntPlatformChar
DmDdBoundingRectangle	DmDdIntSensorChar
DmDdCollECSAttributeXref	DmDdKeywordAliasXref
DmDdCollectionXref	DmDdKeywordParameterXref
DmDdDateInstrumentChar	DmDdLevelGroupXref
DmDdDatePlatformChar	DmDdMetadataLevel
DmDdDateSensorChar	DmDdMultipleCollection
DmDdDateTimeDomain	DmDdNumericDomain
DmDdECSAttributes	DmDdOperationModeClass
DmDdECSCollection	DmDdPlatform
DmDdECSCollKeywordXref	DmDdPlatformCharacteristic
DmDdECSDiscipline	DmDdPlatformXref
DmDdECSParameterDetails	DmDdRangeDateTime
DmDdECSTerm	DmDdSensor
DmDdECSTopic	DmDdSensorCharacteristic
DmDdECSVariable	DmDdSensorInstrumentXref
DmDdEquivalentAttributes	DmDdSensorXref
DmDdFloatInstrumentChar	DmDdSingleTypeCollection
DmDdFloatPlatformChar	DmDdSite
DmDdFloatSensorChar	DmDdSpatialDomain
DmDdGlossary	DmDdSpatialKeywordClass
DmDdGroup	DmDdStringDomain
DmDdGroupXref	DmDdStringInstrumentChar
DmDdInfoMgr	DmDdStringPlatformChar
DmDdInfoMgrCollXref	DmDdStringSensorChar
DmDdInfoMgrXref	DmDdStringType
DmDdInstrument	DmDdTemporal
DmDdInstrumentCharacteristic	DmDdTemporalKeywordClass
DmDdInstrumentPlatformXref	DmDdValidsMapping

The following report is produced by the S-Designor CASE tool and edited for format. The report provides specifications on the DM Subsystem database tables. The report is sorted in alphabetical order by table name. Specifications include the table name, a brief description of the table, and the columns comprising the table. The order of the tables presented herein correspond to the order in the list of tables in Table 4-1 above. The column information includes the column name and the column attributes, i.e., type (format of the data stored within the database), primary key indicator(s), and a mandatory indicator for determining if the column must contain data when the row exists. In some cases the content of the column specification "Type" will reference a domain value (refer to Section 4.1.4 for more information on the domain values).

Table: DmDdAcronym

Description: This table lists all the acronyms that exist within ECS.

Column List:

Name	Type	P	M
acronymDescription	text	No	No
acronymName	char(15)	Yes	Yes

Table: DmDdAdditionalAttributes

Description: This table identifies product specific attributes.

Column List:

Name	Type	P	M
AdditionalAttributeDataType	char(10)	No	No
AdditionalAttributeDescription	varchar(255)	No	No
AdditionalAttributeName	varchar(255)	No	Yes
attributeId	int	Yes	Yes
attributeLevelId	char(4)	No	No
attributeSize	int	No	No
collectionId	int	No	Yes
searchable	numeric(1,0)	No	No
siteId	int	Yes	Yes

Table: DmDdAttributeXref

Description: This is the a cross reference table among, attributes, collections and site references.

Column List:

Name	Type	P	M
attributeId	int	Yes	Yes
attributeIdRef	int	Yes	Yes
collectionId	int	No	Yes
collectionIdRef	int	No	Yes
siteId	int	Yes	Yes
siteIdRef	int	Yes	Yes

Table: DmDdBoundingRectangle

Description: This table contains area coverage for ECS collection or granules. This area coverage is expressed by latitude and longitude values in the order western, eastern, northern, and southern-most. For data sets that include a complete band of latitude around the Earth, the west coordinate = -180.0 and east coordinate = 180.0. Latitude values range from -90 to +90.

Column List:

Name	Type	P	M
collectionId	int	Yes	Yes
EastBoundingCoordinate	double precision	Yes	Yes
NorthBoundingCoordinate	double precision	Yes	Yes
siteId	int	Yes	Yes
SouthBoundingCoordinate	double precision	Yes	Yes
WestBoundingCoordinate	double precision	Yes	Yes

Table: DmDdCollECSAttributeXref

Description: This table is a cross reference between the ECS attributes and collections.

Column List:

Name	Type	P	M
attributeId	int	Yes	Yes
collectionId	int	No	Yes
ECSAttributeName	char(50)	No	Yes
siteId	int	Yes	Yes

Table: DmDdCollectionXref

Description: This is a cross reference table between the collection and sites.

Column List:

Name	Type	P	M
collectionIdAgg	int	Yes	Yes
collectionIdRef	int	Yes	Yes
siteIdAgg	int	Yes	Yes
siteIdRef	int	Yes	Yes

Table: DmDdDateInstrumentChar

Description: This holds the instrument characteristics by date.

Column List:

Name	Type	P	M
DateInstrumentCharValue	datetime	No	No
InstrumentCharacteristicName	varchar(40)	Yes	Yes

Table: DmDdDatePlatformChar

Description: This table holds the platform characteristics by date.

Column List:

Name	Type	P	M
DatePlatformCharValue	datetime	No	No
PlatformCharacteristicName	varchar(40)	Yes	Yes

Table: DmDdDateSensorChar

Description: This table holds the sensor characteristic by date.

Column List:

Name	Type	P	M
DateSensorCharValue	datetime	No	No
SensorCharacteristicName	varchar(40)	Yes	Yes

Table: DmDdDateTimeDomain

Description: This table holds the date and time domain that a collection can have.

Column List:

Name	Type	P	M
attributeId	int	Yes	Yes
collectionId	int	No	Yes
EndDateTime	datetime	No	No
siteId	int	Yes	Yes
StartDateTime	datetime	No	Yes

Table: DmDdECSAttributes

Description: This table holds the ECS attributes.

Column List:

Name	Type	P	M
attributeDataType	char(1)	No	No
attributeDescription	varchar(255)	No	No
attributeLevelId	char(4)	No	No
attributeSize	int	No	No
ECSAttributeName	char(50)	Yes	Yes
searchable	numeric(1,0)	No	No

Table: DmDdECSCollection

Description: This table provides further description of the collection to include media, revision date, usage, and processing and archive centers. It is associated with many other collection level descriptive tables.

Column List:

Name	Type	P	M
ArchiveCenter	char(20)	No	Yes
collectionId	int	Yes	Yes
collectionURL	varchar(100)	No	No
Description	varchar(255)	No	No
gcmdEntryId	char(32)	No	No
LongName	char(80)	No	No
ProcessingCenter	char(20)	No	No
RevisionDate	datetime	No	No
ShortName	char(8)	No	Yes
siteId	int	Yes	Yes
SuggestedUsage	text	No	No
VersionID	int	No	Yes
VersionDescription	varchar(255)	No	No

Table: DmDdECSCollKeywordXref

Description: This is a cross reference table between Collection and Keyword.

Column List:

Name	Type	P	M
collectionId	int	No	Yes
ECSDisciplineKeyword	char(24)	No	No
ECSTermKeyword	char(50)	No	No
ECSTopicKeyword	char(32)	No	No
ECSVariableKeyword	char(80)	No	No
keywordId	int	Yes	Yes
siteId	int	Yes	Yes

Table: DmDdECSDiscipline

Description: This table provides the discipline keyword associated with a collection.

Column List:

Name	Type	P	M
disciplineDescription	varchar(255)	No	No
ECSDisciplineKeyword	char(24)	Yes	Yes

Table: DmDdECSParameterDetails

Description: This table is used to provide further information about the physical or geophysical parameters specified in the DmDdAdditionalAttributes. It contains the units of measurement, range, accuracy, explanation, and resolution.

Column List:

Name	Type	P	M
ECSParameterKeyword	varchar(80)	Yes	Yes
ParameterMeasurementResolution	char(30)	No	No
ParameterRange	char(10)	No	No
ParameterUnitsOfMeasurement	char(20)	No	No
ParameterValueAccuracy	char(30)	No	No
ParameterValueAccuracyExplanat	varchar(255)	No	No

Table: DmDdECSTerm

Description: This table contains the term keyword(s) associated with the collection. (e.g., atmospheric temperature, precipitation, soils, sea ice).

Column List:

Name	Type	P	M
ECSTermKeyword	char(50)	Yes	Yes
termDescription	varchar(255)	No	No

Table: DmDdECSTopic

Description: This table contains the topic keyword(s) associated with the collection.(e.g., atmospheric science, hydrosphere, land surface, ocean science).

Column List:

Name	Type	P	M
ECSTopicKeyword	char(32)	Yes	Yes
topicDescription	varchar(255)	No	No

Table: DmDdECSVariable

Description: This table contains the variable keyword(s) associated with the collection. (e.g., upper troposphere temperature, precipitable water, soil depth, albedo).

Column List:

Name	Type	P	M
ECSVariableKeyword	char(80)	Yes	Yes
variableDescription	varchar(255)	No	No

Table: DmDdEquivalentAttributes

Description: This table holds extra information for equivalent attributes found in ECS.

Column List:

Name	Type	P	M
attributeId	int	Yes	Yes
attributeSize	int	No	No
collectionId	int	No	Yes
equivalentAttributeDataType	char(10)	No	No
equivalentAttributeDescription	varchar(255)	No	No
equivalentAttributeName	varchar(255)	No	Yes
searchable	numeric(1,0)	No	No
siteId	int	Yes	Yes

Table: DmDdFloatInstrumentChar

Description: This table holds information for the instrument characteristics that have float as a data type.

Column List:

Name	Type	P	M
FloatInstrumentCharValue	float	No	No
InstrumentCharacteristicName	varchar(40)	Yes	Yes

Table: DmDdFloatPlatformChar

Description: This table holds information for the platform characteristics that have float as a data type.

Column List:

Name	Type	P	M
FloatPlatormCharValue	float	No	No
PlatformCharacteristicName	varchar(40)	Yes	Yes

Table: DmDdFloatSensorChar

Description: This table holds information for the sensor characteristics that have float as a data type.

Column List:

Name	Type	P	M
FloatSensorCharValue	Float	No	No
SensorCharacteristicName	Varchar(40)	Yes	Yes

Table: DmDdGlossary

Description: This holds the glossary information of the ECS.

Column List:

Name	Type	P	M
GlossaryDescription	Text	No	No
GlossaryName	Char(80)	Yes	Yes

Table: DmDdGroup

Description: This table holds the group names and descriptions within the ESDT.

Column List:

Name	Type	P	M
groupDescription	Varchar(255)	No	No
groupName	Char(40)	Yes	Yes

Table: DmDdGroupXref

Description: This table shows the relationship between the groups.

Column List:

Name	Type	P	M
groupName	Char(40)	Yes	Yes
groupParentName	Char(40)	Yes	Yes

Table: DmDdInfoMgr

Description: This table holds the list of information managers on each DAAC

Column List:

Name	Type	P	M
infoMgrDesc	varchar(255)	No	No
infoMgrName	char(20)	Yes	Yes
infoMgrType	char(20)	No	No
infoMgrUR	varchar(255)	No	Yes

Table: DmDdInfoMgrCollXref

Description: This table is a cross reference between InfoMgr and Collection.

Column List:

Name	Type	P	M
collectionId	int	Yes	Yes
infoMgrName	char(20)	Yes	Yes
siteId	int	Yes	Yes

Table: DmDdInfoMgrXref

Description: This table shows the relationship between the InfoMgr.

Column List:

Name	Type	P	M
infoMgrName	char(20)	Yes	Yes
infoMgrParentName	char(20)	Yes	Yes

Table: DmDdInstrument

Description: This class defines the device used to measure or record data, including direct human observation. Included in this class are defined EOS Instruments. In cases where instruments have a single sensor or the instrument and sensor are used sysnonomously (e.g. AVHRR) the both Instrument and sensor should be recorded.

Column List:

Name	Type	P	M
instrumentDescription	varchar(80)	No	No
InstrumentLongName	varchar(80)	No	No
InstrumentShortName	char(20)	Yes	Yes
InstrumentTechnique	char(30)	No	No
NumOfSensors	smallint	No	No

Table: DmDdInstrumentCharacteristic

Description: This class is used to define the characteristics of instrument specific attributes. It should not be used to define attributes of new objects.

Column List:

Name	Type	P	M
InstrumentCharacteristicDesc	varchar(40)	No	No
InstrumentCharacteristicName	varchar(40)	Yes	Yes
InstrumentCharacteristicUnit	char(20)	No	No
InstrumentCharDataType	char(6)	No	No
InstrumentShortName	char(20)	No	No

Table: DmDdInstrumentPlatformXref

Description: This is a cross reference table between instrument and platform.

Column List:

Name	Type	P	M
InstrumentShortName	char(20)	Yes	Yes
PlatformShortName	char(20)	Yes	Yes

Table: DmDdInstrumentXref

Description: This is a cross reference table between instrument and collection.

Column List:

Name	Type	P	M
collectionId	int	Yes	Yes
InstrumentShortName	char(20)	Yes	Yes
siteId	int	Yes	Yes

Table: DmDdIntInstrumentChar

Description: This table holds information for the instrument characteristics that have integer as a data type.

Column List:

Name	Type	P	M
InstrumentCharacteristicName	varchar(40)	Yes	Yes
IntInstrumentCharValue	int	No	No

Table: DmDdIntPlatformChar

Description: This table holds information for the platform characteristics that have integer as a data type.

Column List:

Name	Type	P	M
IntPlatformCharacteristicValue	int	No	No
PlatformCharacteristicName	varchar(40)	Yes	Yes

Table: DmDdIntSensorChar

Description: This table holds information for the sensor characteristics that have integer as a data type.

Column List:

Name	Type	P	M
IntSensorCharValue	int	No	No
SensorCharacteristicName	varchar(40)	Yes	Yes

Table: DmDdKeywordAliasXref

Description: This table holds the aliases for keywords.

Column List:

Name	Type	P	M
attributeId	int	Yes	Yes
attributeIdRef	int	Yes	Yes
collectionId	int	No	Yes
collectionIdRef	int	No	Yes
keywordName	varchar(80)	Yes	Yes
keywordNameRef	varchar(80)	Yes	Yes
siteId	int	Yes	Yes
siteIdRef	int	Yes	Yes

Table: DmDdKeywordParameterXref

Description: This is a cross reference table between ECS keywords parameters and the DDICT parameter keywords.

Column List:

Name	Type	P	M
ECSParameterKeyword	varchar(80)	Yes	Yes
keywordId	int	Yes	Yes
siteId	int	Yes	Yes

Table: DmDdLevelGroupXref

Description: This table is the cross reference that shows the relationship of groups and levels.

Column List:

Name	Type	P	M
groupName	char(40)	Yes	Yes
levelId	char(4)	Yes	Yes

Table: DmDdMetadataLevel

Description: This table defines the level that an attribute appears in for collections and granules.

Column List:

Name	Type	P	M
levelDescription	varchar(255)	No	No
levelId	char(4)	Yes	Yes
levelName	char(32)	No	No

Table: DmDdMultipleCollection

Description: This table holds the multiple collection information.

Column List:

Name	Type	P	M
AggregationRelationship	char(2)	No	No
AggregationType	char(20)	No	No
AggregationValue	char(80)	No	No
collectionId	int	Yes	Yes
siteId	int	Yes	Yes

Table: DmDdNumericDomain

Description: This table holds the numeric domain for a collection.

Column List:

Name	Type	P	M
accuracy	varchar(50)	No	No
accuracyExplanation	varchar(80)	No	No
attributeId	int	Yes	Yes
collectionId	int	No	Yes
measurementResolution	varchar(50)	No	No
numericDomainRule	varchar(50)	No	Yes
numericPrecision	float(8)	No	No
siteId	int	Yes	Yes
units	char(30)	No	No

Table: DmDdOperationModeClass

Description: This table contains the mode of operation for the instrument.

Column List:

Name	Type	P	M
InstrumentShortName	char(20)	No	No
OperationMode	char(20)	Yes	Yes

Table: DmDdPlatform

Description: This class describes the relevant platforms associated with the acquisition of the collection or granule. Platform types include Spacecraft, Aircraft, Vessel, Buoy, Platform, Station, Network or Human. In cases where Human is the platform type it should be of scientific relevancy to the collection. If an instrument is hand held and that is relevant to the collection of the data then PlatformType=Human. In cases where an instrument is hand-held but the human is associated with another platform then all relevant platforms should be associated with the collection. Humans can be both Platforms and Instruments (e.g. if a human is standing on the ground and makes a visual observation then: PlatformType=Human, Instrument=HumanObservation, SensorShortName=HumanVisual).

Column List:

Name	Type	P	M
platformDescription	varchar(80)	No	No
PlatformLongName	varchar(80)	No	No
PlatformShortName	char(20)	Yes	Yes
PlatformType	char(20)	No	No

Table: DmDdPlatformCharacteristic

Description: This class is used to define the characteristics of platform specific attributes. It is not intended to be used to define attributes of new objects.

Column List:

Name	Type	P	M
PlatformCharacteristicDataType	char(6)	No	No
PlatformCharacteristicDesc	varchar(40)	No	No
PlatformCharacteristicName	varchar(40)	Yes	Yes
PlatformCharacteristicUnit	char(20)	No	No
PlatformShortName	char(20)	No	No

Table: DmDdPlatformXref

Description: This is cross reference table between platform and collection.

Column List:

Name	Type	P	M
collectionId	int	Yes	Yes
PlatformShortName	char(20)	Yes	Yes
siteId	int	Yes	Yes

Table: DmDdRangeDateTime

Description: This class specifies the start and end date and time of granules and collections.

Column List:

Name	Type	P	M
collectionId	int	Yes	Yes
RangeBeginningDate	datetime	Yes	Yes
RangeBeginningTime	datetime	Yes	Yes
RangeEndingDate	datetime	No	No
RangeEndingTime	datetime	No	No
siteId	int	Yes	Yes

Table: DmDdSensor

Description: This class is used to describe sensory subcomponents of an instrument. In cases where instruments have a single sensor or the Instrument and Sensor are used synonomously (e.g. AVHRR) both the Instrument and Sensor are recorded.

Column List:

Name	Type	P	M
sensorDescription	varchar(80)	No	No
SensorLongName	varchar(80)	No	No
SensorShortName	char(20)	Yes	Yes
SensorTechnique	char(40)	No	No

Table: DmDdSensorCharacteristic

Description: This class is used to define the characteristics of sensor specific attributes. It is not intended to be used to define attributes of new objects.

Column List:

Name	Type	P	M
SensorCharacteristicDataType	char(6)	No	No
SensorCharacteristicDesc	varchar(40)	No	No
SensorCharacteristicName	varchar(40)	Yes	Yes
SensorCharacteristicUnit	char(20)	No	No

SensorShortName	char(20)	No	No

Table: DmDdSensorInstrumentXref

Description: This is a cross reference table between sensors and intruments.

Column List:

Name	Type	P	M
InstrumentShortName	char(20)	Yes	Yes
SensorShortName	char(20)	Yes	Yes

Table: DmDdSensorXref

Description: This is a cross reference between sensor and collection.

Column List:

Name	Type	P	M
collectionId	int	Yes	Yes
SensorShortName	char(20)	Yes	Yes
siteId	int	Yes	Yes

Table: DmDdSingleTypeCollection

Description: This class provides a description specific to a single, as opposed to a multitype collection; including citation of external publication, collection state, maintenance and update frequency, and access constraints. The management and development of singletype collections is the subject of other documentation. A single type collection contains a set of granules for which the dominant variation in the value of metadata attributes is in the space and time attributes.

Column List:

Name	Type	P	M
AccessConstraints	varchar(255)	No	No
CitationForExternalPublication	varchar(255)	No	No
collectionId	int	Yes	Yes
CollectionState	varchar(255)	No	No

MaintenanceandUpdateFrequency	char(80)	No	No
ProcessingLevelID	char(6)	No	Yes
siteId	int	Yes	Yes

Table: DmDdSite

Description: This table holds information about the site (e.g., site name, site id).

Column List:

Name	Type	P	M
attributeIdMax	int	No	No
collectionIdMax	int	No	No
keywordIdMax	int	No	No
localFlag	bit	No	Yes
siteId	int	Yes	Yes
siteName	char(4)	No	No

Table: DmDdSpatialDomain

Description: This table holds spatial information about a collection.

Column List:

Name	Type	P	M
attributeId	int	Yes	Yes
collectionId	int	No	Yes
shape	char(25)	No	No
siteId	int	Yes	Yes

Table: DmDdSpatialKeywordClass

Description: This class contains the spatial keywords associated with the ECS collection.

Column List:

Name	Type	P	M
collectionId	int	Yes	Yes
siteId	int	Yes	Yes
SpatialKeyword	char(10)	Yes	Yes

Table: DmDdStringDomain

Description: This table stores the values of ECS attributes for the collection.

Column List:

Name	Type	P	M
attributeId	int	Yes	Yes
collectionId	int	No	Yes
keywordName	varchar(80)	Yes	Yes
siteId	int	Yes	Yes

Table: DmDdStringInstrumentChar

Description: This table holds information for the instrument characteristics that have string as a data type.

Column List:

Name	Type	P	M
InstrumentCharacteristicName	varchar(40)	Yes	Yes
StrInstrumentCharValue	varchar(255)	No	No

Table: DmDdStringPlatformChar

Description: This table holds information for the platform characteristics that have string as a data type.

Column List:

Name	Type	P	M
PlatformCharacteristicName	varchar(40)	Yes	Yes
StrPlatformCharacteristicValue	varchar(255)	No	No

Table: DmDdStringSensorChar

Description: This table holds information for the sensor characteristics that have string as a data type.

Column List:

Name	Type	P	M
SensorCharacteristicName	varchar(40)	Yes	Yes
StringSensorCharValue	varchar(255)	No	No

Table: DmDdStringType

Description: This table stores the values of ECS attributes for the collection**.**

Column List:

Name	Type	P	M
attributeId	int	Yes	Yes
collectionId	int	No	Yes
listAvailableFlag	numeric(1,0)	No	No
siteId	int	Yes	Yes

Table: DmDdTemporal

Description: This class contains attributes which describe the basis of the time system used in other classes

Column List:

Name	Type	P	M
collectionId	int	Yes	Yes
DateType	char(10)	No	No
EndsatPresentFlag	char(1)	No	No
PrecisionofSeconds	int	No	No
siteId	int	Yes	Yes
TemporalRangeType	char(30)	No	No
TimeType	char(10)	No	No

Table: DmDdTemporalKeywordClass

Description: This class identifies the type of temporal characterization for a granule or collection.

Column List:

Name	Type	P	M
collectionId	int	Yes	Yes
siteId	int	Yes	Yes
TemporalKeyword	char(10)	Yes	Yes

Table: DmDdValidsMapping

Description: This class is used to define the attribute name, keyword and its corresponding mapping attributes, and keywords. This table gives the mapping from ECS to external (V0 or ASTER) and External (V0 or Aster) to ECS.

Column List:

Name	Type	P	M
PrimaryAttributeName	varchar(255)	No	Yes
PrimaryKeywordName	varchar(80)	No	Yes
MappedAttributeName	varchar(255)	No	Yes
MappedKeywordName	varchar(80)	No	Yes
Direction	char(10)	No	Yes

4.1.3 Column Specifications

Brief definitions of each of the columns[1] within the DM Subsystem database and their valid values, or references to other documents containing the valid values, are contained herein. Valid Values identify the permissable data content of the column where there is a finite set of acceptable values that can be defined. Other columns are formatted text, free text, or numeric.

Column: AccessConstraints

Description: Restrictions and legal prerequisites for accessing the collection. These include any

[1] The term “column” is used interchangably with the term “attribute” where it refers to a database structure. The ECS Conceptual Model was developed using Object Oriented (OO) terminology. This model is converted to a relational structure for physical implementation within Sybase. The descriptions of some "columns” are derived from the OO Conceptual Model.

access constraints applied to assure the protection of privacy or intellectual property, and any special restrictions or limitations on obtaining the collection. This is an SDSRV Subsystem column.

Note: These restrictions differ from use restrictions in that they only apply to access.

Valid Values: See 420-TP-015

Column: accuracy

Description: Describes the accuracy of a location as a numerical value. See also accuracyExplanation.

Column: accuracyExplanation

Description: Describes what the numeric value of ACCURACY represents.

Column: acronymDescription

Description: This is the acronym description.

Column: acronymName

Description: This is the acronym.

Column: AdditionalAttributeDataType

Description: This is the data type of the ParameterValue. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: AdditionalAttributeDescription

Description: This attribute contains a description of the data content identified by AdditionalAttributeName. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: AdditionalAttributeName

Description: This is the data type of AdditionalAttributeName. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: AggregationRelationship

Description: This attribute identifies the relationship between the aggregation attribute and its corresponding value. This relationship may be expressed as Boolean operations (e.g., '=, <. >, ne'). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: AggregationType

Description: This attribute will contain the criteria by which multiple type collections have been grouped. It will describe the major categorization which applies to the data therein. Possible collection groupings include: INSTRUMENT, for all collections associated with a given collecting instrument such as CERES—this is a common aggregation criteria for ECS 'datasets'; PROJECT, for all data associated with a given project that may or may not be related to a single instrument, such as FIRE—this is again a common aggregation criteria for ECS 'datasets'; PARAMETER, for all granules that reflect measurements of a single specific (or related group of specific) geophysical parameters, such as: CLOUD PROPERTIES—this is often an aggregation criteria for ECS 'products'; SUPERGRANULE, for collections of granules that the data provider wishes to be orderable as a single related grouping, such as SSM/I TIME SERIES—this is a concept adopted from MSFC use; EVENT for a predetermined/tagged set of granules that have been found to be related to a particular geophysical phenomena or event, such as MIDWEST FLOOD '93 or OZONE HOLE or MT. PINATUBO—this is a new ECS concept, also suggested by the University of Virginia Atmospheric researchers. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: AggregationValue

Description: This attribute contains the value associated with the aggregation type. An example may be EVENT (aggregation type) = MIDWEST FLOOD ’93 (aggregation value). MIDWEST FLOOD ’93 would be the value associated with the event or aggregation type. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ArchiveCenter

Description: This is the center where the collection is archived. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: attributeDataType

Description:

This is the data type of the ECS attribute.

Column: attributeDescription

Description: This holds the ECS attribute description.

Column: attributeId

Description: This is the attribute identifier. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: attributeIdMax

Description: Used to generate the next attributeId (attributeIdMax+1), last generated attributeId is stored here.

Column: attributeIdRef

Description: This is the attribute identifier reference

Column: attributeLevelId

Description: This reflects the classification of the attribute level, which defines in general terms the characteristics of the attribute.

Column: attributeSize

Description: This is the ECS attribute size.

Column: CitationForExternalPublication

Description: The recommended reference to be used when referring to this collection in publications. Its format is free text, but should include: Originator (the name of an organization or individual that developed the data set, where Editor(s)' names are followed by (ed.) and Compiler(s)' names are followed by (comp.)); Publication date (the date of publication or release of the data set); Title (the name by which document can be referenced). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: collectionId

Description: This is the unique collection identifier. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: collectionIdAgg

Description: This is the unique collection identifer for multiple collections.

Column: collectionIdMax

Description: To generate the next collectionId(collectionIdMax +1), last generated collectionId is stored here.

Column: collectionIdRef

Description: This is the Collection identifier reference.

Column: CollectionState

Description: This attribute describes the state of the collection, whether it is planned but not yet existent, partially complete due to continual additions from remotely sensed data/processing/reprocessing, or is considered a complete product/dataset. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: CollectionURL

Description: This is the collection URL.

Column: DateInstrumentCharValue

Description: This holds the date on which the instrument characteristic value was issued.

Column: DatePlatformCharValue

Description: This holds the date on which the platform characteristic value was issued.

Column: DateSensorCharValue

Description: This is the date on which the sensor characteristic value was issued.

Column: DateType

Description: This attribute specifies the type of date represented by the value in the date attributes of the temporal subclasses. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: Description

Description: This is the description of a collection.

Column: Direction

Description: This describes the direction of mapping, whether ECS to external(V0 or Aster) or external(V0 or Aster) to ECS.

Valid Values: "ECSToEquiv" The mapping is from ECS to external. "EquivToECS" The mapping is from external to ECS

Examples:

PrimaryAttribute Name	PrimaryKeyword Name	MappedAttribute Name	MappedKeyword Name	Direction
SOURCE_NAME	Air Droppable Buoy	PlatformShortName	Buoy	EquivToECS
SENSOR_NAME	ERB-Scanner	InstrumentShortName	CERES	EquivToECS
InstrumentShortName	ASTER	SOURCE_NAME	Radiometer	ECSToEquiv
PlatformType	Station	SOURCE_NAME	Ground Station	ECSToEquiv

Column: disciplineDescription

Description: This is a description for a discipline.

Column: EastBoundingCoordinate

Description: Eastern-most limit of coverage expressed in longitude.

Valid Values: [-180.0 <= East Bounding Coordinate <= 180.0]

Column: ECSAttributeName

Description: This holds the ECS attribute name.

Column: ECSDisciplineKeyword

Description: This is the keyword used to describe the general discipline area of the collection. A collection can conceivably cover several disciplines. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ECSParameterKeyword

Description: Keyword used to describe specific characteristics of a collection at a higher level of detail than provided in ECSVariableKeyword This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ECSTermKeyword

Description: Keyword used to describe the science parameter area of the collection. A collection can conceivably cover many such parameters. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ECSTopicKeyword

Description: Keyword used to describe the general topic area of the collection. A collection can conceivably cover several topics. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ECSVariableKeyword

Description: Keyword used to describe the specific science parameter content of the collection. A collection can conceivably cover many specific parameters. The keyword valid values are the lowest level physical parameter terms which are normally searched by a user; i.e. a user enters a keyword which when found may connect with one or more parameters from collections. The keywords are also the lowest level words which describe product content without being the server specific measurement (held in Parameter class). While there is a controlled list of these parameters held by GCMD, additions can be made by an as yet unspecified configuration control process. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: EndDateTime

Description: The ending date of a collection.

Column: EndsatPresentFlag

Description: This attribute will denote that a data collection which covers, temporally, a discontinuous range, currently ends at the present date. This way, the granules which comprise the data collection that are continuously being added to inventory need not update the data collection metadata for each one. Note that MODIS granules may be added several thousand times a day, making the update of the data collection metadata impractical. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: equivalentAttributeDataType

Description: This holds the data type of the equivalent attribute

Column: equivalentAttributeDescription

Description: This holds the Descriptionm of the equivalent attribute

Column: equivalentAttributeName

Description: Name of the equivalent attribute

Column: FloatInstrumentCharValue

Description: This holds the float value of the instrument.

Column: FloatPlatformCharValue

Description: This holds the float value of the platform.

Column: FloatSensorCharValue

Description: This is the float value of a sensor.

Column: gcmdEntryId

Description: The id assigned by the Global Change Master Directory fro the data collection. It is equivalent to the V0 attribute MO_ENTRY_ID.

Column: GlossaryDescription

Description: This is the glossary description.

Column: GlossaryName

Description: This is the glossary name.

Column: groupDescription

Description: This is the group description.

Column: groupName

Description: This is the group name (e.g., Spatial, SpatialDomainContainer).

Column: groupParentName

Description: This is the name of the group parent. (e.g., for Spatial--ROOT, for SpatialDomainContainer--Spatial)

Column: infoMgrDesc

Description: The description of the information manager.

Column: infoMgrName

Description: The name of the information manager.

Column: infoMgrParentName

Description: Refers to the information manager's parent name.

Column: infoMgrType

Description: States the type of information manager.

Column: infoMgrUR

Description: The information manager's Universal Reference.

Column: InstrumentCharacteristicDesc

Description: The description of the instrument attribute.

Column: InstrumentCharacteristicName

Description: The name of the instrument characteristic attribute. Instrument characteristics are instrument-specific attributes. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: InstrumentCharacteristicUnit

Description: The units of the attribute defined with InstrumentCharacteristic. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: InstrumentCharDataType

Description: The data type of the instrument characteristic/attribute defined by InstrumentCharacteristicName. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: instrumentDescription

Description: This is the instrument description.

Column: InstrumentLongName

Description: The expanded name of the primary sensory instrument. (e.g. Advanced Spaceborne Thermal Emission and Reflective Radiometer, Clouds and the Earth's Radiant Energy System, Human Observation). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: InstrumentShortName

Description: The unique identifier of an instrument. (e.g. ASTER, AVHRR-3, CERES, Human). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: InstrumentTechnique

Description: The instrument method or procedure (e.g. radiometer, manual enumeration). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: IntInstrumentCharValue

Description: This holds the integer value of the instrument.

Column: IntPlatformCharacteristicValue

Description: This holds the integer value of the platform.

Column: IntSensorCharValue

Description: This holds the integer value of the sensor.

Column: keywordId

Description: This is the keyword identifier.

Column: keywordIdMax

Description: This holds the last generated keywordId (which is the maximum keywordId) in order to generate the next keywordId.

Column: keywordName

Description: This is the keyword name for the DDICT.

Column: keywordNameRef

Description: This is the keyword name reference.

Column: levelDescription

Description: This is the Level description.

Column: levelId

Description: This is the level Identifier.

Column: levelName

Description: MetadataLevel Name

Valid Values:

Collection

Granule

Column: listAvailableFlag

Description: This is a flag used to specify if the list is available for string domain attributes.

Column: localFlag

Description: Used to specify the location of the database.

Column: LongName

Description: This data identifies the long name associated with the collection. This includes dataset name/product name. This is the reference name used in describing the scientific contents of the data collection; it is not the "id" of the data. The existing SPSO product names provide a start point. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: MaintenanceandUpdateFrequency

Description: The frequency with which changes and additions are made to the collection after the initial dataset begins to be collected/processed. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015 (MaintenanceUpdateFrequency)

Column: MappedAttributeName

Description: Describes the corresponding Mapped AttributeName for the PrimaryAttributeName

Column: MappedKeywordName

Description: Describes the corresponding value of the Mapped AttributeName

Column: measurementResolution

Description: This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: NorthBoundingCoordinate

Description: Northern-most coordinate of the limit of coverage expressed in geodetic latitude.

Valid Values: [-90.0< = North Bounding Coordinate < = 90.0], [North Bounding Coordinate => South Bounding Coordinate]

Column: numericDomainRule

Description:

Valid Values:

Column: numericPrecision

Description:

Column: NumOfSensors

Description: The number of discrete (if any) sensors on an instrument. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: OperationMode

Description: Mode of operation of the instrument. Each instrument will have 1 to n modes which may be static for the collection, or changing on a granule-by-granule basis (e.g., domains: launch, survival, initialization, safe, diagnostic, roll, tilt, standby, routine, test, calibration). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ParameterKeyWord

Description: Keyword used to describe specific characteristics of a collection at a higher level of detail than provided by ECSVariableKeyword. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ParameterMeasurementResolution

Description: This attribute will be used to identify the smallest unit increment to which the parameter value is measured.

Column: ParameterRange

Description: This attribute provides maximum and minimum value of parameter over whole collection. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ParameterUnitsOfMeasurement

Description: The standard units of measurement for a non-core attribute. AVHRR: Units of Geophysical Parameter = Units of Geophysical Parameter. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ParameterValueAccuracy

Description: An estimate of the accuracy of the assignment of attribute value (e.g., AVHRR: Measurement Error or Precision = Measurement error or precision of a data product parameter). This can be specified in percent or the units with which the parameter is measured. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ParameterValueAccuracyExplanat

Description: This defines the method used for determining the Parameter Value Accuracy that is given for this non core attribute.

Column: PlatformCharacteristicDataType

Description: The data type of the Platform Characteristic/attribute defined by the PlatformCharacteristicName. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: PlatformCharacteristicDesc

Description: Description of the Platform Characteristic attribute. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: PlatformCharacteristicName

Description: The name of the Platform Characteristic attribute. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: PlatformCharacteristicUnit

Description: Units associated with the Platform Characteristic attribute value. This is an SDSRV

Subsystem column.

Valid Values: See 420-TP-015

Column: platformDescription

Description: This is the platform description.

Column: PlatformLongName

Description: The expanded or long name of the platform associated with an instrument. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: PlatformShortName

Description: The unique platform name. (e.g. GOES-8). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: PlatformType

Description: The most relevant platform type. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: PrecisionofSeconds

Description: The precision (number of places to right of decimal point) of seconds used in measurement. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015 (PrecisionOfSeconds)

Column: PrimaryAttributeName

Description: This column describes the Attribute Name (can be ECS or other Attribute)

Column: PrimaryKeywordName

Description: Describes the value of the PrimaryAttributeName

Column: ProcessingCenter

Description: This is the name of the processing center (e.g., GSFC, JPL, SCF). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ProcessingLevelID

Description: This attribute reflects the classification of the science data processing level, which defines in general terms the characteristics of the output of the processing performed. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: RangeBeginningDate

Description: The year (and optionally month, or month and day) when the temporal coverage period being described began. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: RangeBeginningTime

Description: The first hour (and optionally minute, or minute and second) of the temporal coverage period being described.

Column: RangeEndingDate

Description: The last year (and optionally month, or month and day) of the temporal coverage period being described. GSFC AVHRR This date represents the end date of the latest granule contained in the product. MM/DD/YY format is product-specific for: sage_atmos_dyn, sage_atmos_comp, erbe_erpMMDDYYYY format is product-specific for: LARC_FIRE, LARC_GTE. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: RangeEndingTime

Description: The last hour (and optionally minute, or minute and second) of the temporal coverage period being described for granule or collection. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: RevisionDate

Description: Represents the date and possibly the time that this directory entry was created or the latest date and time of its modification or update. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: searchable

Description: This holds the searchable key for the ECS attribute.

Column: SensorCharacteristicDataType

Description: The data type of the Instrument Characteristic/attribute defined by InstrumentCharacteristicName. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: SensorCharacteristicDesc

Description: A description of the attribute defined by SensorCharacteristicName. (e.g. SensorCharacteristicName = SensorDevice, SensorCharacteristicDescription = Charge coupled device). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: SensorCharacteristicName

Description: The name of the Sensor Characteristic/attribute. Sensor attributes defined using SensorCharacteristicName must be a single-valued attribute of the object 'Sensor' and not the attribute of an undefined object. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: SensorCharacteristicUnit

Description: The unit of the Sensor Characteristic (e.g. nanometers). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: sensorDescription

Description: Description of a sensor.

Column: SensorLongName

Description: The generic or long name description of a sensor. (e.g. Visible-Near Infrared, Human Visual, Human Auditory). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: SensorShortName

Description: A sensor is a defined sensory sub-component of an instrument. (e.g. InstrumentShortName = ASTER, NumberofSensors = 3, SensorShortName = SWIR, SensorShortName = TIR, SensorShortName = VNIR) In cases where the Instrument has a single Sensor or the Instrument and Sensor are synonymous then both attributes should be populated. (e.g. AVHRR). Sensors cannot exist without Instruments. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: SensorTechnique

Description: The sensor technique (e.g. laser altimetry). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: shape

Description: Shape of the spatial attribute.

Column: ShortName

Description: This name will identify the short name associated with the collection or granule. This includes the ECS Technical Baseline product names (e.g., CER02, MOD12) This is the official reference name used in identifying the contents of the data collection. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: siteId

Description: This is the site identifier. This uniquely identifies the DAAC.

Column: siteIdAgg

Description: This isi the site identifier. this uniquely identifies the collection aggregation.

Column: siteIdRef

Description: This is the site identifier. This uniquely identifies the DAAC reference.

Column: siteName

Description: This holds the site name.

Column: SouthBoundingCoordinate

Description: Southern-most limit of coverage expressed in geodetic latitude.

Valid Values: [-90.0 <= South Bounding Coordinate <=90.0], South Sounding Coordinate <= North Bounding Coordinate]

Column: SpatialKeyword

Description: This attribute specifies a word or phrase which serves to summarize the spatial regions covered by the collection. It may be repeated if several regions are covered. This often occurs when a collection is described as covering some large region, and several smaller sub-regions within that region. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: StartDateTime

Description: The starting date of a collection.

Column: StringSensorCharValue

Description: This holds the string value of the sensor.

Column: StrInstrumentCharValue

Description: This holds the string value of the instrument.

Column: StrPlatformCharacteristicValue

Description: This holds the string value of the platform.

Column: SuggestedUsage

Description: This attribute describes how this collection or granule may be best used to support earth science/global change research. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: TemporalKeyword

Description: This attribute specifies a word or phrase which serves to summarize the temporal characteristics referenced in the collection (e.g., Monthly Composite, Annual Mean). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: TemporalRangeType

Description: This attribute tells the system and ultimately the end user how temporal coverage is specified for the collection, granule, or event. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: termDescription

Description: This contains the description for a term.

Column: TimeType

Description: This attribute provides the time system which the values found in temporal subclasses represent. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: TopicDescription

Description: This is the topic description.

Column: units

Description: This isi the measurement standard used for the numeric type of data.

Column: VariableDescription

Description: This is the variable description

Column: VersionDescription

Description: This is the description of the collection identified by the Version ID.

Column: VersionID

Description: This is the version identifier for the collection. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: WestBoundingCoordinate

Description: Western-most coordinate of the limit of coverage expressed in geodetic longitude.

Valid Values: [-180.0 <= West Bounding Coordinate <= 180.0]

4.1.4 Column Domains

Domains specify the ranges of values allowed for a given column within the database table. Sybase supports the definition of specific domains to further limit the format of data for a given column. Sybase domains are, in effect, user-defined data types. There are no domains currently defined within the DM Subsystem S-Designor repository.

4.1.5 Column Default Values

A default value is used to supply a value for a column when one is not defined at row insert time. Values default to null values if not specifically stated in Section 4.1.3; reference 311-CD-107-002 for those columns marked as replicated columns from the SDSRV Subsystem.

4.1.6 Referential Integrity Rules

S-Designor supports the definition of referential integrity rules. When defined within S-Designor, code may be automatically generated to support referential integrity. This code is called a *trigger*. Triggers are listed in Section 4.1.9. There are currently no referential integrity rules defined in the S-Designor repository for the DM Subsystem database.

4.1.7 Views

Sybase allows the definition of views as a means of limiting an application or users' access to data in the database. Views create a logical database table from columns found in one or more database tables. Views defined within the DM Subsystem database are listed below.

View List

Name	Description
DmDdStringAttributesUv	Tables accessed include DmDdAdditionalAttributes and DmDdStringDomain. Attributes joined include collectionId, siteId, AdditionalAttributeName, attributeLevelId, and keywordName.
DmDdTemporalAttributesUv	Tables accessed include DmDdAdditionalAttributes and DmDdDateTimeDomain. Attributes joined include collectionId, siteId, AdditionalAttributeName, attributeLevelId, StartDateTime, and EndDateTime.

View: DmDdStringAttributesUv

Code

```
create view DmDdStringAttributesUv
as
select
                A.collectionId collectionId,
                A.siteId siteId,
                A.AdditionalAttributeName AdditionalAttributeName,
                A.attributeLevelId attributeLevelId,
                B.keywordName keywordName
from
                DmDdAdditionalAttributes A, DmDdStringDomain B
where
                A.siteId = B.siteId
and             A.attributeId = B.attributeId
and             A.AdditionalAttributeDataType = "S"
```

View: DmDdTemporalAttributesUv

Code

```
create view DmDdTemporalAttributesUv
as
select
                A.collectionId collectionId,
                A.siteId siteId,
                A.AdditionalAttributeName AdditionalAttributeName,
                A.attributeLevelId attributeLevelId,
                B.StartDateTime StartDateTime,
                B.EndDateTime EndDateTime
from
                DmDdAdditionalAttributes A, DmDdDateTimeDomain B
where
                A.siteId = B.siteId
and             A.attributeId = B.attributeId
and             A.AdditionalAttributeDataType = "D"
```

4.1.8 Declarative Integrity Constraints

Sybase version 11.0.2.2 allows the enforcement of referential integrity via the use of declarative integrity constraints. Integrity constraints allow the SQL server to enforce primary and foreign key integrity checks automatically without requiring programming. Sybase is ANSI-92 compliant, therefore, its constraints support “restrict-only” operations. This means that a row can not be deleted or the key values modified if there are rows in other database tables having a foreign key dependency on that row. Cascade delete and update operations cannot be performed if a declarative integrity constraint has been used. The following information describes the declarative integrity constraints contained in the DM Subsystem database.

Dependencies on Table: DmDdCollECSAttributeXref

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdAttributeXref	siteId attributeId	siteId attributeId

Dependencies on Table: DmDdECSAttributes

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdCollECSAttributeXref	ECSAttributeName	ECSAttributeName

Dependencies on Table: DmDdECSCollection

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdBoundingRectangle	collectionId siteId	collectionId siteId
DmDdCollECSAttributeXref	collectionId siteId	collectionId siteId
DmDdEquivalentAttributes	collectionId siteId	collectionId siteId
DmDdMultipleCollection	collectionId siteId	collectionId siteId
DmDdSingleTypeCollection	collectionId siteId	collectionId siteId
DmDdSpatialKeywordClass	collectionId siteId	collectionId siteId
DmDdTemporal	collectionId siteId	collectionId siteId
DmDdTemporalKeywordClass	collectionId siteId	collectionId siteId
DmDdAdditionalAttributes	collectionId siteId	collectionId siteId
DmDdInfoMgrCollXref	collectionId siteId	collectionId siteId
DmDdInstrumentXref	collectionId siteId	collectionId siteId
DmDdPlatformXref	collectionId siteId	collectionId siteId
DmDdSensorXref	collectionId siteId	collectionId siteId

Referenced by	Primary Key	Foreign Key
DmDdECSCollKeywordXref	collectionId siteId	collectionId siteId

Dependencies on Table: DmDdECSCollKeywordXref

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdKeywordParameterXref	keywordId siteId	keywordId siteId

Dependencies on Table: DmDdECSDiscipline

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdECSCollKeywordXref	ECSDisciplineKeyword	ECSDisciplineKeyword

Dependencies on Table: DmDdECSParameterDetails

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdKeywordParameterXref	ECSParameterKeyword	ECSParameterKeyword

Dependencies on Table: DmDdECSTerm

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdECSCollKeywordXref	ECSTermKeyword	ECSTermKeyword

Dependencies on Table: DmDdECSTopic

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdECSCollKeywordXref	ECSTopicKeyword	ECSTopicKeyword

Dependencies on Table: DmDdECSVariable

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdECSCollKeywordXref	ECSVariableKeyword	ECSVariableKeyword

Dependencies on Table: DmDdEquivalentAttributes

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdAttributeXref	siteId attributeId	siteIdRef attributeIdRef

Dependencies on Table: DmDdGroup

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdGroupXref	groupName	groupName
DmDdGroupXref	groupName	groupParentName
DmDdLevelGroupXref	groupName	groupName

Dependencies on Table: DmDdInfoMgr

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdInfoMgrCollXref	infoMgrName	infoMgrName
DmDdInfoMgrXref	infoMgrName	infoMgrName
DmDdInfoMgrXref	infoMgrName	infoMgrParentName

Dependencies on Table: DmDdInstrument

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdInstrumentCharacteristic	InstrumentShortName	InstrumentShortName
DmDdInstrumentXref	InstrumentShortName	InstrumentShortName
DmDdOperationModeClass	InstrumentShortName	InstrumentShortName
DmDdSensorInstrumentXref	InstrumentShortName	InstrumentShortName
DmDdInstrumentPlatformXref	InstrumentShortName	InstrumentShortName

Dependencies on Table: DmDdInstrumentCharacteristic

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdDateInstrumentChar	InstrumentCharacteristicName	InstrumentCharacteristicName
DmDdFloatInstrumentChar	InstrumentCharacteristicName	InstrumentCharacteristicName
DmDdIntInstrumentChar	InstrumentCharacteristicName	InstrumentCharacteristicName
DmDdStringInstrumentChar	InstrumentCharacteristicName	InstrumentCharacteristicName

Dependencies on Table: DmDdMetadataLevel

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdAdditionalAttributes	levelId	attributeLevelId
DmDdECSAttributes	levelId	attributeLevelId
DmDdLevelGroupXref	levelId	levelId

Dependencies on Table: DmDdMultipleCollection

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdCollectionXref	collectionId siteId	collectionIdRef siteIdRef
DmDdCollectionXref	collectionId siteId	collectionIdAgg siteIdAgg

Dependencies on Table: DmDdPlatform

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdInstrumentPlatformXref	PlatformShortName	PlatformShortName
DmDdPlatformCharacteristic	PlatformShortName	PlatformShortName
DmDdPlatformXref	PlatformShortName	PlatformShortName

Dependencies on Table: DmDdPlatformCharacteristic

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdDatePlatformChar	PlatformCharacteristicName	PlatformCharacteristicName
DmDdFloatPlatformChar	PlatformCharacteristicName	PlatformCharacteristicName
DmDdIntPlatformChar	PlatformCharacteristicName	PlatformCharacteristicName
DmDdStringPlatformChar	PlatformCharacteristicName	PlatformCharacteristicName

Dependencies on Table: DmDdSensor

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdSensorCharacteristic	SensorShortName	SensorShortName

Referenced by	Primary Key	Foreign Key
DmDdSensorXref	SensorShortName	SensorShortName
DmDdSensorInstrumentXref	SensorShortName	SensorShortName

Dependencies on Table: DmDdSensorCharacteristic

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdDateSensorChar	SensorCharacteristicName	SensorCharacteristicName
DmDdFloatSensorChar	SensorCharacteristicName	SensorCharacteristicName
DmDdIntSensorChar	SensorCharacteristicName	SensorCharacteristicName
DmDdStringSensorChar	SensorCharacteristicName	SensorCharacteristicName

Dependencies on Table: DmDdSingleTypeCollection

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdCollectionXref	collectionId siteId	collectionIdRef siteIdRef

Dependencies on Table: DmDdStringDomain

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdKeywordAliasXref	siteId attributeId keywordName	siteId attributeId keywordName
DmDdKeywordAliasXref	siteId attributeId keywordName	siteIdRef attributeIdRef keywordNameRef

Dependencies on Table: DmDdStringType

Reference by List

Referenced by	Primary Key	Foreign Key

Referenced by	Primary Key	Foreign Key
DmDdStringDomain	siteId attributeId	siteId attributeId

Dependencies on Table: DmDdTemporal

Reference by List

Referenced by	Primary Key	Foreign Key
DmDdRangeDateTime	collectionId siteId	collectionId siteId

4.1.9 Triggers

The Sybase Database Management System (DBMS) supports the enforcement of business rules for data integrity via the use of trigger code. A trigger is defined as a set of activities or checks that are performed automatically by the DBMS whenever an update command occurs for a given database table. The term *update* is defined to identify those database commands that result in inserting (adding) a row, changing a column within an existing row, and deleting a row.

Triggers are implemented at the database table level. Table 4-4 below provides an alphabetically ordered set of triggers used to assure data integrity for the DM Subsystem database. This table includes the trigger name, the database table it is related to, and a brief description of the trigger. Table 4-4 is followed immediately by the trigger source code listings.

Table 4-4. Trigger Descriptions (1 of 6)

Trigger Name	Related Database Table	Description
ti_dmddadditionalattributes	DmDdAdditionalAttributes	/* Insert trigger */ /* Parent "DmDdECSCollection" must exist when inserting a child */ /* Parent "DmDdMetadataLevel" must exist when inserting a child */
TrigDelAddiAttribute	DmDdAdditionalAttributes	Delete trigger. Error return if dependencies exist.
ti_dmddattributexref	DmDdAttributeXref	Insert trigger /* Parent "DmDdEquivalentAttributes" must exist when inserting a child /* Parent "DmDdCollECSAttributeXref" must exist when inserting a child

Trigger Name	Related Database Table	Description
TrigUpdDmDdAttributeXref	DmDdAttributeXref	Update trigger Error return if dependencies exist
ti_dmddboundingrectangle	DmDdBoundingRectangle	Insert trigger /* Parent "DmDdECSCollection" must exist when inserting a child

Table 4-4. Trigger Descriptions (2 of 6)

Trigger Name	Related Database Table	Description
ti_dmddcollecsattributexref	DmDdCollECSAttributeXref	Insert trigger /* Parent "DmDdECSCollection" must exist when inserting a child /* Parent "DmDdECSAttributes" must exist when inserting a child
TrigDelCollECSAXref	DmDdCollECSAttributeXref	Delete trigger Error return if dependencies exist
ti_dmddcollectionxref	DmDdCollectionXref	Insert trigger /* Parent "DmDdMultipleCollection" must exist when inserting a child /* Parent "DmDdSingleTypeCollection" must exist when inserting a child /* Parent "DmDdMultipleCollection" must exist when inserting a child
ti_dmdddateinstrumentchar	DmDdDateInstrumentChar	Insert trigger /* Parent "DmDdInstrumentCharacteristic" must exist when inserting a child
ti_dmdddateplatformchar	DmDdDatePlatformChar	Insert trigger /* Parent "DmDdPlatformCharacteristic" must exist when inserting a child
ti_dmdddatesensorchar	DmDdDateSensorChar	Insert trigger /* Parent "DmDdSensorCharacteristic" must exist when inserting a child
TrigInsDmDdDateTimeDomain	DmDdDateTimeDomain	Insert trigger Error return if no refernce data exists for new record in DmDdDateTimeDomain
TrigUpdDateTimeDomain	DmDdDateTimeDomain	Update trigger Error return if no refernce data exists for updated data in DmDdDateTimeDomain
ti_dmddecsattributes	DmDdECSAttributes	Insert trigger /* Parent "DmDdMetadataLevel" must exist when inserting a child
TrigUpdDmDdECSCollection	DmDdECSCollection	Update trigger Trigger that checks for duplicate ShortName

Table 4-4. Trigger Descriptions (3 of 6)

Trigger Name	Related Database Table	Description
ti_dmddecscollkeywordxref	DmDdECSCollKeywordXref	Insert trigger /* Parent "DmDdECSCollection" must exist when inserting a child /* Parent "DmDdECSDiscipline" must exist when inserting a child /* Parent "DmDdECSTopic" must exist when inserting a child /* Parent "DmDdECSVariable" must exist when inserting a child /* Parent "DmDdECSTerm" must exist when inserting a child
ti_dmddequivalentattributes	DmDdEquivalentAttributes	Insert trigger /* Parent "DmDdECSCollection" must exist when inserting a child
TrigDelEquicAttribute	DmDdEquivalentAttributes	Delete trigger Error is returned when dependency exists
ti_dmddfloatinstrumentchar	DmDdFloatInstrumentChar	Insert trigger /* Parent "DmDdInstrumentCharacteristic" must exist when inserting a child
ti_dmddfloatplatformchar	DmDdFloatPlatformChar	Insert trigger /* Parent "DmDdPlatformCharacteristic" must exist when inserting a child
ti_dmddfloatsensorchar	DmDdFloatSensorChar	Insert trigger /* Parent "DmDdSensorCharacteristic" must exist when inserting a child in
ti_dmddgroupxref	DmDdGroupXref	Insert trigger /* Parent "DmDdGroup" must exist when inserting a child /* Parent "DmDdGroup" must exist when inserting a child
TrigInsUpdDmDdInfoMgrCollXref	DmDdInfoMgrCollXref	Insert and Update trigger Error is returned if duplicate SortName is found
TrigInsUpdDmDdInfoMgrCollXref	DmDdInfoMgrCollXref	Insert and Update trigger Error is returned if duplicate ShortName is found
ti_dmddinfomgrxref	DmDdInfoMgrXref	Insert trigger /* Parent "DmDdInfoMgr" must exist when inserting a child

Table 4-4. Trigger Descriptions (4 of 6)

Trigger Name	Related Database Table	Description
ti_dmddinstrumentcharacteristi	DmDdInstrumentCharacteristic	Insert trigger /* Parent "DmDdInstrument" must exist when inserting a child
ti_dmddinstrumentplatformxref	DmDdInstrumentPlatformXref	Insert trigger /* Parent "DmDdPlatform" must exist when inserting a child /* Parent "DmDdInstrument" must exist when inserting a child
ti_dmddinstrumentxref	DmDdInstrumentXref	Insert trigger /* Parent "DmDdInstrument" must exist when inserting a child /* Parent "DmDdECSCollection" must exist when inserting a child
ti_dmddintinstrumentchar	DmDdIntInstrumentChar	Insert trigger /* Parent "DmDdInstrumentCharacteristic" must exist when inserting a child
ti_dmddintplatformchar	DmDdIntPlatformChar	Insert trigger /* Parent "DmDdPlatformCharacteristic" must exist when inserting a child
ti_dmddintsensorchar	DmDdIntSensorChar	Insert trigger /* Parent "DmDdSensorCharacteristic" must exist when inserting a child
ti_dmddkeywordaliasxref	DmDdKeywordAliasXref	Insert trigger /* Parent "DmDdStringDomain" must exist when inserting a child /* Parent "DmDdStringDomain" must exist when inserting a child
ti_dmddkeywordparameterxref	DmDdKeywordParameterXref	Insert trigger /* Parent "DmDdECSCollKeywordXref" must exist when inserting a child /* Parent "DmDdECSParameterDetails" must exist when inserting a child
ti_dmddlevelgroupxref	DmDdLevelGroupXref	Insert trigger /* Parent "DmDdMetadataLevel" must exist when inserting a child /* Parent "DmDdGroup" must exist when inserting a child
ti_dmddmultiplecollection	DmDdMultipleCollection	Insert trigger /* Parent "DmDdECSCollection" must exist when inserting a child

Table 4-4. Trigger Descriptions (5 of 6)

Trigger Name	Related Database Table	Description
TrigInsDmDdNumDomain	DmDdNumericDomain	Insert trigger Error is returned when no refernce data exists for new record on insert
TrigUpdDmDdNumericDomain	DmDdNumericDomain	Update trigger Error is returned when no refernce data exists for updated data
ti_dmddoperationmodeclass	DmDdOperationModeClass	Insert trigger /* Parent "DmDdInstrument" must exist when inserting a child
ti_dmddplatformcharacteristic	DmDdPlatformCharacteristic	Insert trigger /* Parent "DmDdPlatform" must exist when inserting a child
ti_dmddplatformxref	DmDdPlatformXref	Insert trigger /* Parent "DmDdPlatform" must exist when inserting a child /* Parent "DmDdECSCollection" must exist when inserting a child
ti_dmddrangedatetime	DmDdRangeDateTime	Insert trigger /* Parent "DmDdTemporal" must exist when inserting a child
ti_dmddsensorcharacteristic	DmDdSensorCharacteristic	Insert trigger /* Parent "DmDdSensor" must exist when inserting a child
ti_dmddsensorinstrumentxref	DmDdSensorInstrumentXref	Insert trigger /* Parent "DmDdInstrument" must exist when inserting a child /* Parent "DmDdSensor" must exist when inserting a child
ti_dmddsensorxref	DmDdSensorXref	Insert trigger /* Parent "DmDdSensor" must exist when inserting a child /* Parent "DmDdECSCollection" must exist when inserting a child
ti_dmddsingletypecollection	DmDdSingleTypeCollection	Insert trigger /* Parent "DmDdECSCollection" must exist when inserting a child
TrigInsDmDdSpatialDomain	DmDdSpatialDomain	Insert trigger Error is returned when no refernce data exists for new record
TrigUpdDmDdSpatialDomain	DmDdSpatialDomain	Update trigger Error is returned when no refernce data exists for new record

Table 4-4. Trigger Descriptions (6 of 6)

Trigger Name	Related Database Table	Description
ti_dmddspatialkeywordclass	DmDdSpatialKeywordClass	Insert trigger /* Parent "DmDdECSCollection" must exist when inserting a child
ti_dmddstringdomain	DmDdStringDomain	Insert trigger /* Parent "DmDdStringType" must exist when inserting a child
ti_dmddstringinstrumentchar	DmDdStringInstrumentChar	Insert trigger /* Parent "DmDdInstrumentCharacteristic" must exist when inserting a child
ti_dmddstringplatformchar	DmDdStringPlatformChar	Insert trigger /* Parent "DmDdPlatformCharacteristic" must exist when inserting a child
ti_dmddstringsensorchar	DmDdStringSensorChar	Insert trigger /* Parent "DmDdSensorCharacteristic" must exist when inserting a child
TrigInsDmDdStringType	DmDdStringType	Insert trigger Error is returned when no refernce data exists for new record
TrigUpdDmDdStringType	DmDdStringType	Update trrigger Error is returned when no refernce data exists for updated data
ti_dmddtemporal	DmDdTemporal	Insert trigger /* Parent "DmDdECSCollection" must exist when inserting a child
ti_dmddtemporalkeywordclass	DmDdTemporalKeywordClass	Insert trigger /* Parent "DmDdECSCollection" must exist when inserting a child

Trigger: ti_dmddadditionalattributes

Trigger Code

```
/*  Insert trigger "ti_dmddadditionalattributes" for table "DmDdAdditionalAttributes"  */
create trigger ti_dmddadditionalattributes on DmDdAdditionalAttributes for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
```

```
    return


  /*  Parent "DmDdECSCollection" must exist when inserting a child in "DmDdAdditionalAttributes"  */
  if update(collectionId) or
    update(siteId)
  begin
    if (select count(*)
        from   DmDdECSCollection t1, inserted t2
        where  t1.collectionId = t2.collectionId
         and   t1.siteId = t2.siteId) != @numrows
       begin
         select @errno  = 30002,
              @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdAdditionalAttributes".'
         goto error
       end
  end

  /*  Parent "DmDdMetadataLevel" must exist when inserting a child in "DmDdAdditionalAttributes"  */
  if update(attributeLevelId)
  begin
    select @numnull = (select count(*)
                  from   inserted
                  where  attributeLevelId is null)
    if @numnull != @numrows
      if (select count(*)
          from   DmDdMetadataLevel t1, inserted t2
          where  t1.levelId = t2.attributeLevelId) != @numrows - @numnull
       begin
         select @errno  = 30002,
              @errmsg = 'Parent does not exist in "DmDdMetadataLevel". Cannot create child in
"DmDdAdditionalAttributes".'
         goto error
       end
  end

  return

/*  Errors handling  */
error:
  raiserror @errno @errmsg
  rollback  transaction
end
go
```

Trigger: TrigDelAddiAttribute

Trigger Code

```
--*************************************************************
--BEGIN PROLOG
```

```
--
--TRIGGER NAME:                TrigDelAddiAttribute
--
--TABLE ACCESSED:   DmDdAdditionalAttributes
--                                    DmDdStringType
--                                    DmDdDateTimeDomain
--                                    DmDdNumericDomain
--                                    DmDdSpatialDomain
--                                    DmDdAttributeXref
--                                    deleted
--
--*************************************************************

CREATE TRIGGER TrigDelAddiAttribute
ON DmDdAdditionalAttributes
FOR DELETE
AS
BEGIN

        IF (SELECT count(*) FROM deleted d, DmDdStringType s
                        WHERE d.siteId = s.siteId
                        AND d.attributeId = s.attributeId) > 0
                OR (SELECT count(*) FROM deleted d, DmDdDateTimeDomain a
                        WHERE d.siteId = a.siteId
                        AND d.attributeId = a.attributeId) > 0
                OR (SELECT count(*) FROM deleted d, DmDdNumericDomain n
                        WHERE d.siteId = n.siteId
                        AND d.attributeId = n.attributeId) > 0
                OR (SELECT count(*) FROM deleted d, DmDdSpatialDomain s
                        WHERE d.siteId = s.siteId
                        AND d.attributeId = s.attributeId) > 0
                BEGIN

                        INSERT DmDdAdditionalAttributes
                        SELECT * FROM deleted

                        RAISERROR 30515
                        "The record in DmDdAdditionalAttributes is stil refered by other data, deleting has
been aborted"
                        RETURN
                END
END
RETURN
go
```

Trigger: ti_dmddattributexref

Trigger Code

```
/*  Insert trigger "ti_dmddattributexref" for table "DmDdAttributeXref"  */
create trigger ti_dmddattributexref on DmDdAttributeXref for insert as
begin
```

```
    declare
      @numrows  int,
      @numnull  int,
      @errno    int,
      @errmsg   varchar(255)

    select  @numrows = @@rowcount
    if @numrows = 0
      return


    /*  Parent "DmDdEquivalentAttributes" must exist when inserting a child in "DmDdAttributeXref"  */
    if update(siteIdRef) or
      update(attributeIdRef)
    begin
      if (select count(*)
          from   DmDdEquivalentAttributes t1, inserted t2
          where  t1.siteId = t2.siteIdRef
           and   t1.attributeId = t2.attributeIdRef) != @numrows
         begin
           select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdEquivalentAttributes". Cannot create child in
"DmDdAttributeXref".'
           goto error
         end
    end

    /*  Parent "DmDdCollECSAttributeXref" must exist when inserting a child in "DmDdAttributeXref"  */
    if update(siteId) or
      update(attributeId)
    begin
      if (select count(*)
          from   DmDdCollECSAttributeXref t1, inserted t2
          where  t1.siteId = t2.siteId
           and   t1.attributeId = t2.attributeId) != @numrows
         begin
           select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdCollECSAttributeXref". Cannot create child in
"DmDdAttributeXref".'
           goto error
         end
    end

    return

/*  Errors handling  */
error:
    raiserror @errno @errmsg
    rollback  transaction
end
go
```

Trigger Code

```
--*************************************************************
--BEGIN PROLOG
--
--TRIGGER NAME:              TrigUpdDmDdAttributeXref
--
--TABLE ACCESSED:   DmDdCollECSAttributeXref
--                                  DmDdAttributeXref
--                                  inserted
--                                  deleted
--
--*************************************************************

CREATE TRIGGER TrigUpdDmDdAttributeXref
ON DmDdAttributeXref
FOR UPDATE
AS
BEGIN
        DECLARE @attributeIdIns             attIdType,
                        @siteIdIns          int,
                        @attributeIdDel     attIdType,
                        @siteIdDel          int,
                        @attributeIdRefIns  attIdType,
                        @siteIdRefIns   int,
                        @attributeIdRefDel  attIdType,
                        @siteIdRefDel       int,
                        @error                      int

BEGIN
        DECLARE updAttributeXrefCS CURSOR FOR
                SELECT i.siteId, i.attributeId,  d.siteId, d.attributeId,
                        i.siteIdRef, i.attributeIdRef,  d.siteIdRef, d.attributeIdRef
                FROM inserted i, deleted d
                WHERE i.siteId *= d.siteId
                AND i.attributeId *= d.attributeId

        OPEN updAttributeXrefCS

        FETCH updAttributeXrefCS INTO
                @siteIdIns,
                @attributeIdIns,
                @siteIdDel,
                @attributeIdDel,
                @siteIdRefIns,
                @attributeIdRefIns,
                @siteIdRefDel,
                @attributeIdRefDel
```

```
WHILE @@sqlstatus != 2
BEGIN
        SELECT @error = 0
        IF @siteIdDel = null AND @attributeIdDel = null
                AND @siteIdRefDel = null AND @attributeIdRefDel = null
        BEGIN

                IF (select count(*) FROM DmDdCollECSAttributeXref
                        WHERE siteId = @siteIdIns
                        AND attributeId = @attributeIdIns) = 0
                AND (SELECT count(*) FROM DmDdEquivalentAttributes
                        WHERE siteId = @siteIdIns
                        AND attributeId = @attributeIdIns) = 0
                BEGIN
                        SELECT @error = 1
                END

                IF @error = 0
                BEGIN
                        IF (select count(*) FROM DmDdCollECSAttributeXref
                                WHERE siteId = @siteIdRefIns
                                AND attributeId = @attributeIdRefIns) = 0
                        AND (SELECT count(*) FROM DmDdEquivalentAttributes
                                WHERE siteId = @siteIdRefIns
                                AND attributeId = @attributeIdRefIns) = 0
                        BEGIN
                                SELECT @error = 1
                        END
                END

                IF @error = 1
                BEGIN
                        DELETE DmDdAttributeXref
                        FROM DmDdAttributeXref s, inserted i
                        WHERE s.siteId = i.siteId
                        AND s.attributeId = i.attributeId

                        INSERT DmDdAttributeXref
                        SELECT * from deleted

                        RAISERROR 30515
                "No refernce data exists for updated data in DmDdAttributeXref, updating has
been aborted"
                        RETURN
                END

        END /* if @siteIdDel = null and @attributeIdDel = null */

        FETCH updAttributeXrefCS INTO
                        @siteIdIns,
                        @attributeIdIns,
                        @siteIdDel,
```

```
                                @attributeIdDel,
                                @siteIdRefIns,
                                @attributeIdRefIns,
                                @siteIdRefDel,
                                @attributeIdRefDel


        END /* WHILE */
END
END
RETURN
go
```

Trigger: ti_dmddboundingrectangle

Trigger Code

```
/*  Insert trigger "ti_dmddboundingrectangle" for table "DmDdBoundingRectangle"  */
create trigger ti_dmddboundingrectangle on DmDdBoundingRectangle for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdECSCollection" must exist when inserting a child in "DmDdBoundingRectangle"  */
   if update(collectionId) or
     update(siteId)
   begin
     if (select count(*)
         from   DmDdECSCollection t1, inserted t2
         where  t1.collectionId = t2.collectionId
          and   t1.siteId = t2.siteId) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdBoundingRectangle".'
          goto error
        end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
```

```
   rollback  transaction
end
go
```

Trigger: ti_dmddcollecsattributexref

Trigger Code

```
/*  Insert trigger "ti_dmddcollecsattributexref" for table "DmDdCollECSAttributeXref"  */
create trigger ti_dmddcollecsattributexref on DmDdCollECSAttributeXref for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdECSCollection" must exist when inserting a child in "DmDdCollECSAttributeXref"  */
   if update(collectionId) or
     update(siteId)
   begin
     if (select count(*)
         from   DmDdECSCollection t1, inserted t2
         where  t1.collectionId = t2.collectionId
          and   t1.siteId = t2.siteId) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdCollECSAttributeXref".'
          goto error
        end
   end

   /*  Parent "DmDdECSAttributes" must exist when inserting a child in "DmDdCollECSAttributeXref"  */
   if update(ECSAttributeName)
   begin
     if (select count(*)
         from   DmDdECSAttributes t1, inserted t2
         where  t1.ECSAttributeName = t2.ECSAttributeName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdECSAttributes". Cannot create child in
"DmDdCollECSAttributeXref".'
          goto error
        end
   end
```

```
   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: TrigDelCollECSAXref

Trigger Code

```
--*************************************************************
--BEGIN PROLOG
--
--TRIGGER NAME:                TrigDelCollECSAttriXref
--
--TABLE ACCESSED:   DmDdCollECSAttributeXref
--                                    DmDdStringType
--                                    DmDdDateTimeDomain
--                                    DmDdNumericDomain
--                                    DmDdSpatialDomain
--                                    DmDdAttributeXref
--                                    deleted
--
--*************************************************************

CREATE TRIGGER TrigDelCollECSAXref
ON DmDdCollECSAttributeXref
FOR DELETE
AS
BEGIN

        IF (SELECT count(*) FROM deleted d, DmDdStringType s
                        WHERE d.siteId = s.siteId
                        AND d.attributeId = s.attributeId) > 0
                OR (SELECT count(*) FROM deleted d, DmDdDateTimeDomain a
                        WHERE d.siteId = a.siteId
                        AND d.attributeId = a.attributeId) > 0
                OR (SELECT count(*) FROM deleted d, DmDdNumericDomain n
                        WHERE d.siteId = n.siteId
                        AND d.attributeId = n.attributeId) > 0
                OR (SELECT count(*) FROM deleted d, DmDdSpatialDomain s
                        WHERE d.siteId = s.siteId
                        AND d.attributeId = s.attributeId) > 0
                OR (SELECT count(*) FROM deleted d, DmDdAttributeXref s
                        WHERE d.siteId = s.siteId
                        AND d.attributeId = s.attributeId
                        OR d.siteId = s.siteIdRef
                        AND d.attributeId = s.attributeIdRef) > 0
                BEGIN
```

```
                        INSERT DmDdCollECSAttributeXref
                        SELECT * FROM deleted

                        RAISERROR 30515
                        "The record in DmDdCollECSAttributeXref is stil refered by other data, deleting
has been aborted"
                        RETURN
                    END
END
RETURN
go
```

Trigger: ti_dmddcollectionxref

Trigger Code

```
/*  Insert trigger "ti_dmddcollectionxref" for table "DmDdCollectionXref"  */
create trigger ti_dmddcollectionxref on DmDdCollectionXref for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdMultipleCollection" must exist when inserting a child in "DmDdCollectionXref"  */
   if update(collectionIdRef) or
     update(siteIdRef)
   begin
     if (select count(*)
         from   DmDdMultipleCollection t1, inserted t2
         where  t1.collectionId = t2.collectionIdRef
          and   t1.siteId = t2.siteIdRef) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdMultipleCollection". Cannot create child in
"DmDdCollectionXref".'
          goto error
        end
   end

   /*  Parent "DmDdSingleTypeCollection" must exist when inserting a child in "DmDdCollectionXref"  */
   if update(collectionIdRef) or
     update(siteIdRef)
   begin
     if (select count(*)
```

```
         from   DmDdSingleTypeCollection t1, inserted t2
         where  t1.collectionId = t2.collectionIdRef
          and   t1.siteId = t2.siteIdRef) != @numrows
        begin
          select @errno  = 30002,
               @errmsg = 'Parent does not exist in "DmDdSingleTypeCollection". Cannot create child in
"DmDdCollectionXref".'
          goto error
        end
   end

   /*  Parent "DmDdMultipleCollection" must exist when inserting a child in "DmDdCollectionXref"  */
   if update(collectionIdAgg) or
     update(siteIdAgg)
   begin
     if (select count(*)
         from   DmDdMultipleCollection t1, inserted t2
         where  t1.collectionId = t2.collectionIdAgg
          and   t1.siteId = t2.siteIdAgg) != @numrows
        begin
          select @errno  = 30002,
               @errmsg = 'Parent does not exist in "DmDdMultipleCollection". Cannot create child in
"DmDdCollectionXref".'
          goto error
        end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmdddateinstrumentchar

Trigger Code

```
/*  Insert trigger "ti_dmdddateinstrumentchar" for table "DmDdDateInstrumentChar"  */
create trigger ti_dmdddateinstrumentchar on DmDdDateInstrumentChar for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return
```

```
   /*  Parent "DmDdInstrumentCharacteristic" must exist when inserting a child in
"DmDdDateInstrumentChar"  */
   if update(InstrumentCharacteristicName)
   begin
      if (select count(*)
          from   DmDdInstrumentCharacteristic t1, inserted t2
          where  t1.InstrumentCharacteristicName = t2.InstrumentCharacteristicName) != @numrows
         begin
            select @errno  = 30002,
                  @errmsg = 'Parent does not exist in "DmDdInstrumentCharacteristic". Cannot create child in
"DmDdDateInstrumentChar".'
            goto error
         end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmdddateplatformchar

Trigger Code

```
/*  Insert trigger "ti_dmdddateplatformchar" for table "DmDdDatePlatformChar"  */
create trigger ti_dmdddateplatformchar on DmDdDatePlatformChar for insert as
begin
   declare
      @numrows  int,
      @numnull  int,
      @errno    int,
      @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
      return


   /*  Parent "DmDdPlatformCharacteristic" must exist when inserting a child in "DmDdDatePlatformChar"
*/
   if update(PlatformCharacteristicName)
   begin
      if (select count(*)
          from   DmDdPlatformCharacteristic t1, inserted t2
          where  t1.PlatformCharacteristicName = t2.PlatformCharacteristicName) != @numrows
         begin
```

```
        select @errno  = 30002,
               @errmsg = 'Parent does not exist in "DmDdPlatformCharacteristic". Cannot create child in
"DmDdDatePlatformChar".'
        goto error
      end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmdddatesensorchar

Trigger Code

```
/*  Insert trigger "ti_dmdddatesensorchar" for table "DmDdDateSensorChar"  */
create trigger ti_dmdddatesensorchar on DmDdDateSensorChar for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdSensorCharacteristic" must exist when inserting a child in "DmDdDateSensorChar"  */
   if update(SensorCharacteristicName)
   begin
     if (select count(*)
         from   DmDdSensorCharacteristic t1, inserted t2
         where  t1.SensorCharacteristicName = t2.SensorCharacteristicName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdSensorCharacteristic". Cannot create child in
"DmDdDateSensorChar".'
          goto error
        end
   end
```

```
   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: TrigInsDmDdDateTimeDomain

Trigger Code

```
--*************************************************************
--BEGIN PROLOG
--
--TRIGGER NAME:              TrigInsDmDdDateTimeDomain
--
--TABLE ACCESSED:   DmDdCollECSAttributeXref
--                                    DmDdAdditionalAttributes
--                                    DmDdDateTimeDomain
--                                    inserted
--
--*************************************************************

CREATE TRIGGER TrigInsDmDdDateTimeDomain
ON DmDdDateTimeDomain
FOR INSERT
AS
BEGIN
        DECLARE @attributeId attIdType,
                        @siteId                 int,
                        @hasIt                  int

BEGIN
        DECLARE insDateTimeDomainCS CURSOR FOR
                SELECT siteId, attributeId
                FROM inserted

        OPEN insDateTimeDomainCS

        FETCH insDateTimeDomainCS INTO
                @siteId,
                @attributeId

        WHILE @@sqlstatus != 2
        BEGIN
                EXEC ProcCheckAttriExist @siteId = @siteId,
                                                        @attributeId = @attributeId,
                                                        @hasIt = @hasIt output
                IF @hasIt = 0
                BEGIN
```

```
                        DELETE DmDdDateTimeDomain
                        FROM DmDdDateTimeDomain s, inserted i
                        WHERE s.siteId = i.siteId
                        AND s.attributeId = i.attributeId

                        RAISERROR 30515
                        "No refernce data exists for new record in DmDdDateTimeDomain, inserting has
been aborted"
                        RETURN
                END

                FETCH insDateTimeDomainCS INTO
                        @siteId,
                        @attributeId
        END /* WHILE */
END
END
RETURN
go
```

Trigger: TrigUpdDateTimeDomain

Trigger Code

```
--*************************************************************
--BEGIN PROLOG
--
--TRIGGER NAME:               TrigUpdDmDdDateTimeDomain
--
--TABLE ACCESSED:   DmDdCollECSAttributeXref
--                                  DmDdDateTimeDomain
--                                  inserted
--                                  deleted
--
--*************************************************************

CREATE TRIGGER TrigUpdDateTimeDomain
ON DmDdDateTimeDomain
FOR UPDATE
AS
BEGIN
        DECLARE @attributeIdIns             attIdType,
                @siteIdIns                  int,
                @attributeIdDel             attIdType,
                @siteIdDel                  int

BEGIN
        DECLARE updDateTimeDomainCS CURSOR FOR
                SELECT i.siteId, i.attributeId,  d.siteId, d.attributeId
                FROM inserted i, deleted d
                WHERE i.siteId *= d.siteId
```

```
                AND i.attributeId *= d.attributeId

        OPEN updDateTimeDomainCS

        FETCH updDateTimeDomainCS INTO
                @siteIdIns,
                @attributeIdIns,
                @siteIdDel,
                @attributeIdDel

        WHILE @@sqlstatus != 2
        BEGIN
                IF @siteIdDel = null AND @attributeIdDel = null
                BEGIN

                        IF (select count(*) FROM DmDdCollECSAttributeXref
                                WHERE siteId = @siteIdIns
                                AND attributeId = @attributeIdIns) = 0
                        AND (SELECT count(*) FROM DmDdAdditionalAttributes
                                WHERE siteId = @siteIdIns
                                AND attributeId = @attributeIdIns) = 0
                        AND (SELECT count(*) FROM DmDdEquivalentAttributes
                                WHERE siteId = @siteIdIns
                                AND attributeId = @attributeIdIns) = 0
                        BEGIN
                                DELETE DmDdDateTimeDomain
                                FROM DmDdDateTimeDomain s, inserted i
                                WHERE s.siteId = i.siteId
                                AND s.attributeId = i.attributeId

                                INSERT DmDdDateTimeDomain
                                SELECT * from deleted

                                RAISERROR 30515
                        "No refernce data exists for updated data in DmDdDateTimeDomain, updating
has been aborted"
                                RETURN
                        END

                END /* if @siteIdDel = null and @attributeIdDel = null */

                FETCH updDateTimeDomainCS INTO
                                @siteIdIns,
                                @attributeIdIns,
                                @siteIdDel,
                                @attributeIdDel

        END /* WHILE */
END
END
RETURN
go
```

Trigger: ti_dmddecsattributes

Trigger Code

```
/*  Insert trigger "ti_dmddecsattributes" for table "DmDdECSAttributes"  */
create trigger ti_dmddecsattributes on DmDdECSAttributes for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdMetadataLevel" must exist when inserting a child in "DmDdECSAttributes"  */
   if update(attributeLevelId)
   begin
     select @numnull = (select count(*)
                        from   inserted
                        where  attributeLevelId is null)
     if @numnull != @numrows
       if (select count(*)
           from   DmDdMetadataLevel t1, inserted t2
           where  t1.levelId = t2.attributeLevelId) != @numrows - @numnull
       begin
         select @errno  = 30002,
                @errmsg = 'Parent does not exist in "DmDdMetadataLevel". Cannot create child in
"DmDdECSAttributes".'
         goto error
       end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: TrigUpdDmDdECSCollection

Trigger Code

```
--//////////////////////////////////////////////////////////
-- Name: TrigUpdDmDdECSCollection
--
```

```
-- Description:
--
-- Trigger that check duplicate ShortName
--
--////////////////////////////////////////////////////////

create trigger TrigUpdDmDdECSCollection on DmDdECSCollection
for update as
begin

        if @@rowcount = 0
                return

        declare shortName_crsr cursor for
                select i.infoMgrName, i.collectionId, i.siteId, c.ShortName
                from inserted c, DmDdInfoMgrCollXref i, DmDdInfoMgr m
                where i.collectionId = c.collectionId
                and i.siteId = c.siteId
                and i.infoMgrName = m.infoMgrName
                and m.infoMgrType = "SDSRV"

        open shortName_crsr

        declare
                @infoMgrName char(20),
                @collectionId    int,
                @siteId                 int,
                @shortName              char(20)

        fetch shortName_crsr into
                @infoMgrName,
                @collectionId,
                @siteId,
                @shortName

        while @@sqlstatus != 2
        begin

                if(select count(*)
                        from DmDdInfoMgrCollXref i, DmDdECSCollection c
                        where i.collectionId = c.collectionId
                        and i.siteId = c.siteId
                        and i.infoMgrName = @infoMgrName
                        and c.ShortName = @shortName
                        and (i.collectionId != @collectionId or i.siteId != @siteId)
                        ) > 0
                begin

                                raiserror 25100 "Duplicate SortName found, update failed"
                                rollback transaction
                                return
                end -- if(select count(*)
```

```
                fetch shortName_crsr into
                @infoMgrName,
                @collectionId,
                @siteId,
                @shortName

        end --while
        close shortName_crsr
end
return
go
```

Trigger: ti_dmddecscollkeywordxref

Trigger Code

```
/*  Insert trigger "ti_dmddecscollkeywordxref" for table "DmDdECSCollKeywordXref"  */
create trigger ti_dmddecscollkeywordxref on DmDdECSCollKeywordXref for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdECSCollection" must exist when inserting a child in "DmDdECSCollKeywordXref"  */
   if update(collectionId) or
     update(siteId)
   begin
     if (select count(*)
         from   DmDdECSCollection t1, inserted t2
         where  t1.collectionId = t2.collectionId
          and   t1.siteId = t2.siteId) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdECSCollKeywordXref".'
          goto error
        end
   end

   /*  Parent "DmDdECSDiscipline" must exist when inserting a child in "DmDdECSCollKeywordXref"  */
   if update(ECSDisciplineKeyword)
   begin
     select @numnull = (select count(*)
                from   inserted
```

```
                    where  ECSDisciplineKeyword is null)
      if @numnull != @numrows
        if (select count(*)
            from   DmDdECSDiscipline t1, inserted t2
            where  t1.ECSDisciplineKeyword = t2.ECSDisciplineKeyword) != @numrows - @numnull
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdECSDiscipline". Cannot create child in
"DmDdECSCollKeywordXref".'
          goto error
        end
    end

    /*  Parent "DmDdECSTopic" must exist when inserting a child in "DmDdECSCollKeywordXref"  */
    if update(ECSTopicKeyword)
    begin
      select @numnull = (select count(*)
                    from   inserted
                    where  ECSTopicKeyword is null)
      if @numnull != @numrows
        if (select count(*)
            from   DmDdECSTopic t1, inserted t2
            where  t1.ECSTopicKeyword = t2.ECSTopicKeyword) != @numrows - @numnull
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdECSTopic". Cannot create child in
"DmDdECSCollKeywordXref".'
          goto error
        end
    end

    /*  Parent "DmDdECSVariable" must exist when inserting a child in "DmDdECSCollKeywordXref"  */
    if update(ECSVariableKeyword)
    begin
      select @numnull = (select count(*)
                    from   inserted
                    where  ECSVariableKeyword is null)
      if @numnull != @numrows
        if (select count(*)
            from   DmDdECSVariable t1, inserted t2
            where  t1.ECSVariableKeyword = t2.ECSVariableKeyword) != @numrows - @numnull
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdECSVariable". Cannot create child in
"DmDdECSCollKeywordXref".'
          goto error
        end
    end

    /*  Parent "DmDdECSTerm" must exist when inserting a child in "DmDdECSCollKeywordXref"  */
    if update(ECSTermKeyword)
    begin
```

```
    select @numnull = (select count(*)
                       from   inserted
                       where  ECSTermKeyword is null)
    if @numnull != @numrows
      if (select count(*)
          from   DmDdECSTerm t1, inserted t2
          where  t1.ECSTermKeyword = t2.ECSTermKeyword) != @numrows - @numnull
      begin
        select @errno  = 30002,
               @errmsg = 'Parent does not exist in "DmDdECSTerm". Cannot create child in
"DmDdECSCollKeywordXref".'
        goto error
      end
  end

  return

/*  Errors handling  */
error:
  raiserror @errno @errmsg
  rollback  transaction
end
go
```

Trigger: ti_dmddequivalentattributes

Trigger Code

```
/*  Insert trigger "ti_dmddequivalentattributes" for table "DmDdEquivalentAttributes"  */
create trigger ti_dmddequivalentattributes on DmDdEquivalentAttributes for insert as
begin
  declare
    @numrows  int,
    @numnull  int,
    @errno    int,
    @errmsg   varchar(255)

  select  @numrows = @@rowcount
  if @numrows = 0
    return


  /*  Parent "DmDdECSCollection" must exist when inserting a child in "DmDdEquivalentAttributes"  */
  if update(collectionId) or
    update(siteId)
  begin
    if (select count(*)
        from   DmDdECSCollection t1, inserted t2
        where  t1.collectionId = t2.collectionId
         and   t1.siteId = t2.siteId) != @numrows
      begin
        select @errno  = 30002,
               @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
```

```
"DmDdEquivalentAttributes".'
            goto error
         end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: TrigDelEquicAttribute

Trigger Code

```
--*************************************************************
--BEGIN PROLOG
--
--TRIGGER NAME:                TrigDelEquicAttribute
--
--TABLE ACCESSED:   DmDdAdditionalAttributes
--                                        DmDdStringType
--                                        DmDdDateTimeDomain
--                                        DmDdNumericDomain
--                                        DmDdSpatialDomain
--                                        DmDdAttributeXref
--                                        deleted
--
--*************************************************************

CREATE TRIGGER TrigDelEquicAttribute
ON DmDdEquivalentAttributes
FOR DELETE
AS
BEGIN

          IF (SELECT count(*) FROM deleted d, DmDdStringType s
                        WHERE d.siteId = s.siteId
                        AND d.attributeId = s.attributeId) > 0
                  OR (SELECT count(*) FROM deleted d, DmDdDateTimeDomain a
                        WHERE d.siteId = a.siteId
                        AND d.attributeId = a.attributeId) > 0
```

```
                OR (SELECT count(*) FROM deleted d, DmDdNumericDomain n
                        WHERE d.siteId = n.siteId
                        AND d.attributeId = n.attributeId) > 0
                OR (SELECT count(*) FROM deleted d, DmDdSpatialDomain s
                        WHERE d.siteId = s.siteId
                        AND d.attributeId = s.attributeId) > 0
                OR (SELECT count(*) FROM deleted d, DmDdAttributeXref s
                        WHERE d.siteId = s.siteId
                        AND d.attributeId = s.attributeId
                        OR d.siteId = s.siteIdRef
                        AND d.attributeId = s.attributeIdRef) > 0
                BEGIN

                        INSERT DmDdEquivalentAttributes
                        SELECT * FROM deleted

                        RAISERROR 30515
                        "The record in DmDdEquivalentAttributes is stil refered by other data, deleting
has been aborted"
                        RETURN
                END
END
RETURN
go
```

Trigger: ti_dmddfloatinstrumentchar

Trigger Code

```
/*  Insert trigger "ti_dmddfloatinstrumentchar" for table "DmDdFloatInstrumentChar"  */
create trigger ti_dmddfloatinstrumentchar on DmDdFloatInstrumentChar for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdInstrumentCharacteristic" must exist when inserting a child in
"DmDdFloatInstrumentChar"  */
   if update(InstrumentCharacteristicName)
   begin
     if (select count(*)
         from   DmDdInstrumentCharacteristic t1, inserted t2
         where  t1.InstrumentCharacteristicName = t2.InstrumentCharacteristicName) != @numrows
        begin
          select @errno  = 30002,
```

```
            @errmsg = 'Parent does not exist in "DmDdInstrumentCharacteristic". Cannot create child in
"DmDdFloatInstrumentChar".'
            goto error
          end
    end

    return

/*  Errors handling  */
error:
    raiserror @errno @errmsg
    rollback  transaction
end
go
```

Trigger: ti_dmddfloatplatformchar

Trigger Code

```
/*  Insert trigger "ti_dmddfloatplatformchar" for table "DmDdFloatPlatformChar"  */
create trigger ti_dmddfloatplatformchar on DmDdFloatPlatformChar for insert as
begin
    declare
      @numrows  int,
      @numnull  int,
      @errno    int,
      @errmsg   varchar(255)

    select  @numrows = @@rowcount
    if @numrows = 0
      return


    /*  Parent "DmDdPlatformCharacteristic" must exist when inserting a child in "DmDdFloatPlatformChar"
*/
    if update(PlatformCharacteristicName)
    begin
      if (select count(*)
          from   DmDdPlatformCharacteristic t1, inserted t2
          where  t1.PlatformCharacteristicName = t2.PlatformCharacteristicName) != @numrows
        begin
          select @errno  = 30002,
              @errmsg = 'Parent does not exist in "DmDdPlatformCharacteristic". Cannot create child in
"DmDdFloatPlatformChar".'
          goto error
        end
    end

    return

/*  Errors handling  */
error:
```

```
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddfloatsensorchar

Trigger Code

```
/*  Insert trigger "ti_dmddfloatsensorchar" for table "DmDdFloatSensorChar"  */
create trigger ti_dmddfloatsensorchar on DmDdFloatSensorChar for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdSensorCharacteristic" must exist when inserting a child in "DmDdFloatSensorChar"  */
   if update(SensorCharacteristicName)
   begin
     if (select count(*)
         from   DmDdSensorCharacteristic t1, inserted t2
         where  t1.SensorCharacteristicName = t2.SensorCharacteristicName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdSensorCharacteristic". Cannot create child in
"DmDdFloatSensorChar".'
          goto error
        end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddgroupxref

Trigger Code

```
/*  Insert trigger "ti_dmddgroupxref" for table "DmDdGroupXref"  */
create trigger ti_dmddgroupxref on DmDdGroupXref for insert as
begin
```

```
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdGroup" must exist when inserting a child in "DmDdGroupXref"  */
   if update(groupName)
   begin
     if (select count(*)
         from   DmDdGroup t1, inserted t2
         where  t1.groupName = t2.groupName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdGroup". Cannot create child in
"DmDdGroupXref".'
          goto error
        end
   end

   /*  Parent "DmDdGroup" must exist when inserting a child in "DmDdGroupXref"  */
   if update(groupParentName)
   begin
     if (select count(*)
         from   DmDdGroup t1, inserted t2
         where  t1.groupName = t2.groupParentName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdGroup". Cannot create child in
"DmDdGroupXref".'
          goto error
        end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: TrigInsUpdDmDdInfoMgrCollXref

Trigger Code

```
--//////////////////////////////////////////////////////
-- Name: TrigInsUpdDmDdInfoMgrCollXref
--
-- Description:
--
-- Trigger that check duplicate ShortName
--
--//////////////////////////////////////////////////////

create trigger TrigInsUpdDmDdInfoMgrCollXref on DmDdInfoMgrCollXref
for insert, update as
begin

        if @@rowcount = 0
                return

        declare shortName_crsr cursor for
                select i.infoMgrName, i.collectionId, i.siteId, c.ShortName
                from inserted i, DmDdECSCollection c, DmDdInfoMgr m
                where i.collectionId = c.collectionId
                and i.siteId = c.siteId
                and i.infoMgrName = m.infoMgrName
                and m.infoMgrType = "SDSRV"

        open shortName_crsr

        declare
                @infoMgrName char(20),
                @collectionId    int,
                @siteId                 int,
                @shortName              char(20)

        fetch shortName_crsr into
                @infoMgrName,
                @collectionId,
                @siteId,
                @shortName

        while @@sqlstatus != 2
        begin

                if(select count(*)
                        from DmDdInfoMgrCollXref i, DmDdECSCollection c
                        where i.collectionId = c.collectionId
                        and i.siteId = c.siteId
                        and i.infoMgrName = @infoMgrName
                        and c.ShortName = @shortName
                        and (i.collectionId != @collectionId or i.siteId != @siteId)
                        ) > 0
                begin

                        if (select count(*) from deleted) > 0 --update
```

```
                        begin
                                raiserror 25100 "Duplicate SortName found, update failed"
                                rollback transaction
                                return
                        end
                        else
                        begin
                                raiserror 25100 "Duplicate SortName found, inserting records in
DmDdInfoMgrCollXref and DmDdECSCollection have been deleted"
                                declare delColl_crsr cursor for
                                        select collectionId, siteId
                                        from inserted
                                open delColl_crsr
                                while @@sqlstatus != 2
                                begin
                                        fetch delColl_crsr into
                                        @collectionId,
                                        @siteId

                                        delete DmDdInfoMgrCollXref
                                        where collectionId = @collectionId
                                        and siteId = @siteId

                                        delete DmDdECSCollection
                                        where collectionId = @collectionId
                                        and siteId = @siteId

                                        fetch delColl_crsr into
                                        @collectionId,
                                        @siteId
                                end
                                close delColl_crsr
                                return
                        end --if (select count(*) from deleted) > 0
                end -- if(select count(*)

                fetch shortName_crsr into
                @infoMgrName,
                @collectionId,
                @siteId,
                @shortName

        end --while
        close shortName_crsr
end
return
go
```

Trigger: TrigInsUpdDmDdInfoMgrCollXref

Trigger Code

```
--//////////////////////////////////////////////////////
-- Name: TrigInsUpdDmDdInfoMgrCollXref
--
-- Description:
--
-- Trigger that check duplicate ShortName
--
--//////////////////////////////////////////////////////

create trigger TrigInsUpdDmDdInfoMgrCollXref on DmDdInfoMgrCollXref
for insert, update as
begin

        if @@rowcount = 0
                return

        declare shortName_crsr cursor for
                select i.infoMgrName, i.collectionId, i.siteId, c.ShortName
                from inserted i, DmDdECSCollection c, DmDdInfoMgr m
                where i.collectionId = c.collectionId
                and i.siteId = c.siteId
                and i.infoMgrName = m.infoMgrName
                and m.infoMgrType = "SDSRV"

        open shortName_crsr

        declare
                @infoMgrName char(20),
                @collectionId    int,
                @siteId                 int,
                @shortName              char(20)

        fetch shortName_crsr into
                @infoMgrName,
                @collectionId,
                @siteId,
                @shortName

        while @@sqlstatus != 2
        begin

                if(select count(*)
                        from DmDdInfoMgrCollXref i, DmDdECSCollection c
                        where i.collectionId = c.collectionId
                        and i.siteId = c.siteId
                        and i.infoMgrName = @infoMgrName
                        and c.ShortName = @shortName
                        and (i.collectionId != @collectionId or i.siteId != @siteId)
                        ) > 0
                begin
```

```
                        if (select count(*) from deleted) > 0 --update
                        begin
                                raiserror 25100 "Duplicate SortName found, update failed"
                                rollback transaction
                                return
                        end
                        else
                        begin
                                raiserror 25100 "Duplicate SortName found, inserting records in
DmDdInfoMgrCollXref and DmDdECSCollection have been deleted"
                                declare delColl_crsr cursor for
                                        select collectionId, siteId
                                        from inserted
                                open delColl_crsr
                                while @@sqlstatus != 2
                                begin
                                        fetch delColl_crsr into
                                        @collectionId,
                                        @siteId

                                        delete DmDdInfoMgrCollXref
                                        where collectionId = @collectionId
                                        and siteId = @siteId

                                        delete DmDdECSCollection
                                        where collectionId = @collectionId
                                        and siteId = @siteId

                                        fetch delColl_crsr into
                                        @collectionId,
                                        @siteId
                                end
                                close delColl_crsr
                                return
                        end --if (select count(*) from deleted) > 0
                end -- if(select count(*)

                fetch shortName_crsr into
                @infoMgrName,
                @collectionId,
                @siteId,
                @shortName

        end --while
        close shortName_crsr
end
return
go
```

Trigger: ti_dmddinfomgrxref

Trigger Code

```
/*  Insert trigger "ti_dmddinfomgrxref" for table "DmDdInfoMgrXref"  */
create trigger ti_dmddinfomgrxref on DmDdInfoMgrXref for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdInfoMgr" must exist when inserting a child in "DmDdInfoMgrXref"  */
   if update(infoMgrName)
   begin
     if (select count(*)
         from   DmDdInfoMgr t1, inserted t2
         where  t1.infoMgrName = t2.infoMgrName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdInfoMgr". Cannot create child in
"DmDdInfoMgrXref".'
          goto error
        end
   end

   /*  Parent "DmDdInfoMgr" must exist when inserting a child in "DmDdInfoMgrXref"  */
   if update(infoMgrParentName)
   begin
     if (select count(*)
         from   DmDdInfoMgr t1, inserted t2
         where  t1.infoMgrName = t2.infoMgrParentName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdInfoMgr". Cannot create child in
"DmDdInfoMgrXref".'
          goto error
        end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddinstrumentcharacteristi

Trigger Code

```
/*  Insert trigger "ti_dmddinstrumentcharacteristi" for table "DmDdInstrumentCharacteristic"  */
create trigger ti_dmddinstrumentcharacteristi on DmDdInstrumentCharacteristic for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdInstrument" must exist when inserting a child in "DmDdInstrumentCharacteristic"  */
   if update(InstrumentShortName)
   begin
     select @numnull = (select count(*)
                        from   inserted
                        where  InstrumentShortName is null)
     if @numnull != @numrows
       if (select count(*)
           from   DmDdInstrument t1, inserted t2
           where  t1.InstrumentShortName = t2.InstrumentShortName) != @numrows - @numnull
       begin
         select @errno  = 30002,
              @errmsg = 'Parent does not exist in "DmDdInstrument". Cannot create child in
"DmDdInstrumentCharacteristic".'
         goto error
       end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddinstrumentplatformxref

Trigger Code

```
/*  Insert trigger "ti_dmddinstrumentplatformxref" for table "DmDdInstrumentPlatformXref"  */
create trigger ti_dmddinstrumentplatformxref on DmDdInstrumentPlatformXref for insert as
begin
```

```
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdPlatform" must exist when inserting a child in "DmDdInstrumentPlatformXref"  */
   if update(PlatformShortName)
   begin
     if (select count(*)
         from   DmDdPlatform t1, inserted t2
         where  t1.PlatformShortName = t2.PlatformShortName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdPlatform". Cannot create child in
"DmDdInstrumentPlatformXref".'
          goto error
        end
   end

   /*  Parent "DmDdInstrument" must exist when inserting a child in "DmDdInstrumentPlatformXref"  */
   if update(InstrumentShortName)
   begin
     if (select count(*)
         from   DmDdInstrument t1, inserted t2
         where  t1.InstrumentShortName = t2.InstrumentShortName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdInstrument". Cannot create child in
"DmDdInstrumentPlatformXref".'
          goto error
        end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddinstrumentxref

Trigger Code

```
/*  Insert trigger "ti_dmddinstrumentxref" for table "DmDdInstrumentXref"  */
create trigger ti_dmddinstrumentxref on DmDdInstrumentXref for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdInstrument" must exist when inserting a child in "DmDdInstrumentXref"  */
   if update(InstrumentShortName)
   begin
     if (select count(*)
         from   DmDdInstrument t1, inserted t2
         where  t1.InstrumentShortName = t2.InstrumentShortName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdInstrument". Cannot create child in
"DmDdInstrumentXref".'
          goto error
        end
   end

   /*  Parent "DmDdECSCollection" must exist when inserting a child in "DmDdInstrumentXref"  */
   if update(collectionId) or
     update(siteId)
   begin
     if (select count(*)
         from   DmDdECSCollection t1, inserted t2
         where  t1.collectionId = t2.collectionId
          and   t1.siteId = t2.siteId) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdInstrumentXref".'
          goto error
        end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddintinstrumentchar

Trigger Code

```
/*  Insert trigger "ti_dmddintinstrumentchar" for table "DmDdIntInstrumentChar"  */
create trigger ti_dmddintinstrumentchar on DmDdIntInstrumentChar for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdInstrumentCharacteristic" must exist when inserting a child in
"DmDdIntInstrumentChar"  */
   if update(InstrumentCharacteristicName)
   begin
     if (select count(*)
         from   DmDdInstrumentCharacteristic t1, inserted t2
         where  t1.InstrumentCharacteristicName = t2.InstrumentCharacteristicName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdInstrumentCharacteristic". Cannot create child in
"DmDdIntInstrumentChar".'
          goto error
        end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddintplatformchar

Trigger Code

```
/*  Insert trigger "ti_dmddintplatformchar" for table "DmDdIntPlatformChar"  */
create trigger ti_dmddintplatformchar on DmDdIntPlatformChar for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
```

```
    @errno     int,
    @errmsg    varchar(255)

  select  @numrows = @@rowcount
  if @numrows = 0
    return


  /*  Parent "DmDdPlatformCharacteristic" must exist when inserting a child in "DmDdIntPlatformChar"  */
  if update(PlatformCharacteristicName)
  begin
    if (select count(*)
        from   DmDdPlatformCharacteristic t1, inserted t2
        where  t1.PlatformCharacteristicName = t2.PlatformCharacteristicName) != @numrows
       begin
         select @errno  = 30002,
              @errmsg = 'Parent does not exist in "DmDdPlatformCharacteristic". Cannot create child in
"DmDdIntPlatformChar".'
         goto error
       end
  end

  return

/*  Errors handling  */
error:
  raiserror @errno @errmsg
  rollback  transaction
end
go
```

Trigger: ti_dmddintsensorchar

Trigger Code

```
/*  Insert trigger "ti_dmddintsensorchar" for table "DmDdIntSensorChar"  */
create trigger ti_dmddintsensorchar on DmDdIntSensorChar for insert as
begin
  declare
    @numrows  int,
    @numnull  int,
    @errno    int,
    @errmsg   varchar(255)

  select  @numrows = @@rowcount
  if @numrows = 0
    return


  /*  Parent "DmDdSensorCharacteristic" must exist when inserting a child in "DmDdIntSensorChar"  */
  if update(SensorCharacteristicName)
  begin
    if (select count(*)
```

```
         from   DmDdSensorCharacteristic t1, inserted t2
         where  t1.SensorCharacteristicName = t2.SensorCharacteristicName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdSensorCharacteristic". Cannot create child in
"DmDdIntSensorChar".'
          goto error
        end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddkeywordaliasxref

Trigger Code

```
/*  Insert trigger "ti_dmddkeywordaliasxref" for table "DmDdKeywordAliasXref"  */
create trigger ti_dmddkeywordaliasxref on DmDdKeywordAliasXref for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdStringDomain" must exist when inserting a child in "DmDdKeywordAliasXref"  */
   if update(siteId) or
     update(attributeId) or
     update(keywordName)
   begin
     if (select count(*)
         from   DmDdStringDomain t1, inserted t2
         where  t1.siteId = t2.siteId
          and   t1.attributeId = t2.attributeId
          and   t1.keywordName = t2.keywordName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdStringDomain". Cannot create child in
"DmDdKeywordAliasXref".'
```

```
            goto error
         end
   end

   /*  Parent "DmDdStringDomain" must exist when inserting a child in "DmDdKeywordAliasXref"  */
   if update(siteIdRef) or
      update(attributeIdRef) or
      update(keywordNameRef)
   begin
      if (select count(*)
          from   DmDdStringDomain t1, inserted t2
          where  t1.siteId = t2.siteIdRef
           and   t1.attributeId = t2.attributeIdRef
           and   t1.keywordName = t2.keywordNameRef) != @numrows
         begin
            select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdStringDomain". Cannot create child in
"DmDdKeywordAliasXref".'
            goto error
         end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddkeywordparameterxref

Trigger Code

```
/*  Insert trigger "ti_dmddkeywordparameterxref" for table "DmDdKeywordParameterXref"  */
create trigger ti_dmddkeywordparameterxref on DmDdKeywordParameterXref for insert as
begin
   declare
      @numrows  int,
      @numnull  int,
      @errno    int,
      @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
      return


   /*  Parent "DmDdECSCollKeywordXref" must exist when inserting a child in
"DmDdKeywordParameterXref"  */
   if update(keywordId) or
```

```
      update(siteId)
   begin
      if (select count(*)
          from   DmDdECSCollKeywordXref t1, inserted t2
          where  t1.keywordId = t2.keywordId
           and   t1.siteId = t2.siteId) != @numrows
         begin
           select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdECSCollKeywordXref". Cannot create child in
"DmDdKeywordParameterXref".'
           goto error
         end
   end

   /*  Parent "DmDdECSParameterDetails" must exist when inserting a child in
"DmDdKeywordParameterXref"  */
   if update(ECSParameterKeyword)
   begin
      if (select count(*)
          from   DmDdECSParameterDetails t1, inserted t2
          where  t1.ECSParameterKeyword = t2.ECSParameterKeyword) != @numrows
         begin
           select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdECSParameterDetails". Cannot create child in
"DmDdKeywordParameterXref".'
           goto error
         end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddlevelgroupxref

Trigger Code

```
/*  Insert trigger "ti_dmddlevelgroupxref" for table "DmDdLevelGroupXref"  */
create trigger ti_dmddlevelgroupxref on DmDdLevelGroupXref for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
```

```
    if @numrows = 0
      return


    /*  Parent "DmDdMetadataLevel" must exist when inserting a child in "DmDdLevelGroupXref"  */
    if update(levelId)
    begin
      if (select count(*)
          from   DmDdMetadataLevel t1, inserted t2
          where  t1.levelId = t2.levelId) != @numrows
         begin
           select @errno  = 30002,
                  @errmsg = 'Parent does not exist in "DmDdMetadataLevel". Cannot create child in
"DmDdLevelGroupXref".'
           goto error
         end
    end

    /*  Parent "DmDdGroup" must exist when inserting a child in "DmDdLevelGroupXref"  */
    if update(groupName)
    begin
      if (select count(*)
          from   DmDdGroup t1, inserted t2
          where  t1.groupName = t2.groupName) != @numrows
         begin
           select @errno  = 30002,
                  @errmsg = 'Parent does not exist in "DmDdGroup". Cannot create child in
"DmDdLevelGroupXref".'
           goto error
         end
    end

    return

/*  Errors handling  */
error:
    raiserror @errno @errmsg
    rollback  transaction
end
go
```

Trigger: ti_dmddmultiplecollection

Trigger Code

```
/*  Insert trigger "ti_dmddmultiplecollection" for table "DmDdMultipleCollection"  */
create trigger ti_dmddmultiplecollection on DmDdMultipleCollection for insert as
begin
    declare
      @numrows  int,
      @numnull  int,
      @errno    int,
```

```
    @errmsg   varchar(255)

  select  @numrows = @@rowcount
  if @numrows = 0
    return


  /*  Parent "DmDdECSCollection" must exist when inserting a child in "DmDdMultipleCollection"  */
  if update(collectionId) or
    update(siteId)
  begin
    if (select count(*)
        from   DmDdECSCollection t1, inserted t2
        where  t1.collectionId = t2.collectionId
         and   t1.siteId = t2.siteId) != @numrows
       begin
         select @errno  = 30002,
              @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdMultipleCollection".'
         goto error
       end
  end

  return

/*  Errors handling  */
error:
  raiserror @errno @errmsg
  rollback  transaction
end
go
```

Trigger: TrigInsDmDdNumDomain

Trigger Code

```
--*************************************************************
--BEGIN PROLOG
--
--TRIGGER NAME:                TrigInsDmDdNumericDomain
--
--TABLE ACCESSED:   DmDdCollECSAttributeXref
--                                      DmDdAdditionalAttributes
--                                      DmDdNumericDomain
--                                      inserted
--
--*************************************************************

CREATE TRIGGER TrigInsDmDdNumDomain
ON DmDdNumericDomain
FOR INSERT
AS
```

```
BEGIN
        DECLARE @attributeId attIdType,
                        @siteId                 int,
                        @hasIt                  int

BEGIN
        DECLARE insNumericDomainCS CURSOR FOR
                SELECT siteId, attributeId
                FROM inserted

        OPEN insNumericDomainCS

        FETCH insNumericDomainCS INTO
                @siteId,
                @attributeId

        WHILE @@sqlstatus != 2
        BEGIN
                EXEC ProcCheckAttriExist @siteId = @siteId,
                                                                @attributeId = @attributeId,
                                                                @hasIt = @hasIt output
                IF @hasIt = 0
                BEGIN
                        DELETE DmDdNumericDomain
                        FROM DmDdNumericDomain s, inserted i
                        WHERE s.siteId = i.siteId
                        AND s.attributeId = i.attributeId

                        RAISERROR 30515
                        "No refernce data exists for new record in DmDdNumericDomain, inserting has
been aborted"
                        RETURN
                END

                FETCH insNumericDomainCS INTO
                        @siteId,
                        @attributeId
        END /* WHILE */
END
END
RETURN
go
```

Trigger: TrigUpdDmDdNumericDomain

Trigger Code

```
--**************************************************************
--BEGIN PROLOG
--
--TRIGGER NAME:                 TrigUpdDmDdNumericDomain
--
```

```
--TABLE ACCESSED:   DmDdCollECSAttributeXref
--                                  DmDdNumericDomain
--                                  inserted
--                                  deleted
--
--************************************************************

CREATE TRIGGER TrigUpdDmDdNumericDomain
ON DmDdNumericDomain
FOR UPDATE
AS
BEGIN
        DECLARE @attributeIdIns             attIdType,
                        @siteIdIns          int,
                        @attributeIdDel     attIdType,
                        @siteIdDel          int

BEGIN
        DECLARE updNumericDomainCS CURSOR FOR
                SELECT i.siteId, i.attributeId,  d.siteId, d.attributeId
                FROM inserted i, deleted d
                WHERE i.siteId *= d.siteId
                AND i.attributeId *= d.attributeId

        OPEN updNumericDomainCS

        FETCH updNumericDomainCS INTO
                @siteIdIns,
                @attributeIdIns,
                @siteIdDel,
                @attributeIdDel

        WHILE @@sqlstatus != 2
        BEGIN
                IF @siteIdDel = null AND @attributeIdDel = null
                BEGIN

                        IF (select count(*) FROM DmDdCollECSAttributeXref
                                WHERE siteId = @siteIdIns
                                AND attributeId = @attributeIdIns) = 0
                        AND (SELECT count(*) FROM DmDdAdditionalAttributes
                                WHERE siteId = @siteIdIns
                                AND attributeId = @attributeIdIns) = 0
                        AND (SELECT count(*) FROM DmDdEquivalentAttributes
                                WHERE siteId = @siteIdIns
                                AND attributeId = @attributeIdIns) = 0
                        BEGIN
                                DELETE DmDdNumericDomain
                                FROM DmDdNumericDomain s, inserted i
                                WHERE s.siteId = i.siteId
                                AND s.attributeId = i.attributeId
```

```
                                INSERT DmDdNumericDomain
                                SELECT * from deleted

                                RAISERROR 30515
                        "No refernce data exists for updated data in DmDdNumericDomain, updating has
been aborted"
                                RETURN
                        END

                END /* if @siteIdDel = null and @attributeIdDel = null */

                FETCH updNumericDomainCS INTO
                                @siteIdIns,
                                @attributeIdIns,
                                @siteIdDel,
                                @attributeIdDel

        END /* WHILE */
END
END
RETURN
go
```

Trigger: ti_dmddoperationmodeclass

Trigger Code

```
/*  Insert trigger "ti_dmddoperationmodeclass" for table "DmDdOperationModeClass"  */
create trigger ti_dmddoperationmodeclass on DmDdOperationModeClass for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdInstrument" must exist when inserting a child in "DmDdOperationModeClass"  */
   if update(InstrumentShortName)
   begin
     select @numnull = (select count(*)
                 from   inserted
                 where  InstrumentShortName is null)
     if @numnull != @numrows
       if (select count(*)
          from   DmDdInstrument t1, inserted t2
          where  t1.InstrumentShortName = t2.InstrumentShortName) != @numrows - @numnull
       begin
```

```
            select @errno  = 30002,
                   @errmsg = 'Parent does not exist in "DmDdInstrument". Cannot create child in
"DmDdOperationModeClass".'
            goto error
          end
    end

    return

/*  Errors handling  */
error:
    raiserror @errno @errmsg
    rollback  transaction
end
go
```

Trigger: ti_dmddplatformcharacteristic

Trigger Code

```
/*  Insert trigger "ti_dmddplatformcharacteristic" for table "DmDdPlatformCharacteristic"  */
create trigger ti_dmddplatformcharacteristic on DmDdPlatformCharacteristic for insert as
begin
    declare
      @numrows  int,
      @numnull  int,
      @errno    int,
      @errmsg   varchar(255)

    select  @numrows = @@rowcount
    if @numrows = 0
      return


    /*  Parent "DmDdPlatform" must exist when inserting a child in "DmDdPlatformCharacteristic"  */
    if update(PlatformShortName)
    begin
      select @numnull = (select count(*)
                         from   inserted
                         where  PlatformShortName is null)
      if @numnull != @numrows
        if (select count(*)
            from   DmDdPlatform t1, inserted t2
            where  t1.PlatformShortName = t2.PlatformShortName) != @numrows - @numnull
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdPlatform". Cannot create child in
"DmDdPlatformCharacteristic".'
```

```
            goto error
         end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddplatformxref

Trigger Code

```
/*  Insert trigger "ti_dmddplatformxref" for table "DmDdPlatformXref"  */
create trigger ti_dmddplatformxref on DmDdPlatformXref for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdPlatform" must exist when inserting a child in "DmDdPlatformXref"  */
   if update(PlatformShortName)
   begin
     if (select count(*)
         from   DmDdPlatform t1, inserted t2
         where  t1.PlatformShortName = t2.PlatformShortName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdPlatform". Cannot create child in
"DmDdPlatformXref".'
          goto error
        end
   end

   /*  Parent "DmDdECSCollection" must exist when inserting a child in "DmDdPlatformXref"  */
   if update(collectionId) or
     update(siteId)
   begin
     if (select count(*)
         from   DmDdECSCollection t1, inserted t2
         where  t1.collectionId = t2.collectionId
```

```
          and   t1.siteId = t2.siteId) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdPlatformXref".'
          goto error
        end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddrangedatetime

Trigger Code

```
/*  Insert trigger "ti_dmddrangedatetime" for table "DmDdRangeDateTime"  */
create trigger ti_dmddrangedatetime on DmDdRangeDateTime for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdTemporal" must exist when inserting a child in "DmDdRangeDateTime"  */
   if update(collectionId) or
     update(siteId)
   begin
     if (select count(*)
         from   DmDdTemporal t1, inserted t2
         where  t1.collectionId = t2.collectionId
          and   t1.siteId = t2.siteId) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdTemporal". Cannot create child in
"DmDdRangeDateTime".'
          goto error
```

```
        end
    end

    return

/*  Errors handling  */
error:
    raiserror @errno @errmsg
    rollback  transaction
end
go
```

Trigger: ti_dmddsensorcharacteristic

Trigger Code

```
/*  Insert trigger "ti_dmddsensorcharacteristic" for table "DmDdSensorCharacteristic"  */
create trigger ti_dmddsensorcharacteristic on DmDdSensorCharacteristic for insert as
begin
    declare
      @numrows  int,
      @numnull  int,
      @errno    int,
      @errmsg   varchar(255)

    select  @numrows = @@rowcount
    if @numrows = 0
      return


    /*  Parent "DmDdSensor" must exist when inserting a child in "DmDdSensorCharacteristic"  */
    if update(SensorShortName)
    begin
      select @numnull = (select count(*)
                         from   inserted
                         where  SensorShortName is null)
      if @numnull != @numrows
        if (select count(*)
            from   DmDdSensor t1, inserted t2
            where  t1.SensorShortName = t2.SensorShortName) != @numrows - @numnull
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdSensor". Cannot create child in
"DmDdSensorCharacteristic".'
          goto error
        end
    end

    return
```

```
/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddsensorinstrumentxref

Trigger Code

```
/*  Insert trigger "ti_dmddsensorinstrumentxref" for table "DmDdSensorInstrumentXref"  */
create trigger ti_dmddsensorinstrumentxref on DmDdSensorInstrumentXref for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdInstrument" must exist when inserting a child in "DmDdSensorInstrumentXref"  */
   if update(InstrumentShortName)
   begin
     if (select count(*)
         from   DmDdInstrument t1, inserted t2
         where  t1.InstrumentShortName = t2.InstrumentShortName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdInstrument". Cannot create child in
"DmDdSensorInstrumentXref".'
          goto error
        end
   end

   /*  Parent "DmDdSensor" must exist when inserting a child in "DmDdSensorInstrumentXref"  */
   if update(SensorShortName)
   begin
     if (select count(*)
         from   DmDdSensor t1, inserted t2
         where  t1.SensorShortName = t2.SensorShortName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdSensor". Cannot create child in
"DmDdSensorInstrumentXref".'
          goto error
        end
   end
```

```
   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddsensorxref

Trigger Code

```
/*  Insert trigger "ti_dmddsensorxref" for table "DmDdSensorXref"  */
create trigger ti_dmddsensorxref on DmDdSensorXref for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdSensor" must exist when inserting a child in "DmDdSensorXref"  */
   if update(SensorShortName)
   begin
     if (select count(*)
         from   DmDdSensor t1, inserted t2
         where  t1.SensorShortName = t2.SensorShortName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdSensor". Cannot create child in
"DmDdSensorXref".'
          goto error
        end
   end

   /*  Parent "DmDdECSCollection" must exist when inserting a child in "DmDdSensorXref"  */
   if update(collectionId) or
     update(siteId)
   begin
     if (select count(*)
         from   DmDdECSCollection t1, inserted t2
         where  t1.collectionId = t2.collectionId
          and   t1.siteId = t2.siteId) != @numrows
        begin
          select @errno  = 30002,
```

```
            @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdSensorXref".'
          goto error
        end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddsingletypecollection

Trigger Code

```
/*  Insert trigger "ti_dmddsingletypecollection" for table "DmDdSingleTypeCollection"  */
create trigger ti_dmddsingletypecollection on DmDdSingleTypeCollection for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdECSCollection" must exist when inserting a child in "DmDdSingleTypeCollection"  */
   if update(collectionId) or
     update(siteId)
   begin
     if (select count(*)
         from   DmDdECSCollection t1, inserted t2
         where  t1.collectionId = t2.collectionId
          and   t1.siteId = t2.siteId) != @numrows
        begin
          select @errno  = 30002,
               @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdSingleTypeCollection".'
          goto error
        end
   end

   return

/*  Errors handling  */
```

```
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: TrigInsDmDdSpatialDomain

Trigger Code

```
--*************************************************************
--BEGIN PROLOG
--
--TRIGGER NAME:              TrigInsDmDdSpatialDomain
--
--TABLE ACCESSED:   DmDdCollECSAttributeXref
--                                  DmDdAdditionalAttributes
--                                  DmDdSpatialDomain
--                                  inserted
--
--*************************************************************

CREATE TRIGGER TrigInsDmDdSpatialDomain
ON DmDdSpatialDomain
FOR INSERT
AS
BEGIN
        DECLARE @attributeId attIdType,
                        @siteId                 int,
                        @hasIt                  int

BEGIN
        DECLARE insSpatialDomainCS CURSOR FOR
                SELECT siteId, attributeId
                FROM inserted

        OPEN insSpatialDomainCS

        FETCH insSpatialDomainCS INTO
                @siteId,
                @attributeId

        WHILE @@sqlstatus != 2
        BEGIN
                EXEC ProcCheckAttriExist @siteId = @siteId,
                                                                @attributeId = @attributeId,
                                                                @hasIt = @hasIt output
                IF @hasIt = 0
                BEGIN
                        DELETE DmDdSpatialDomain
                        FROM DmDdSpatialDomain s, inserted i
                        WHERE s.siteId = i.siteId
```

```
                        AND s.attributeId = i.attributeId

                        RAISERROR 30515
                        "No refernce data exists for new record in DmDdSpatialDomain, inserting has
been aborted"
                        RETURN
                END

                FETCH insSpatialDomainCS INTO
                        @siteId,
                        @attributeId
        END /* WHILE */
END
END
RETURN
go
```

Trigger: TrigUpdDmDdSpatialDomain

Trigger Code

```
--*************************************************************
--BEGIN PROLOG
--
--TRIGGER NAME:                 TrigUpdDmDdSpatialDomain
--
--TABLE ACCESSED:   DmDdCollECSAttributeXref
--                                      DmDdSpatialDomain
--                                      inserted
--                                      deleted
--
--*************************************************************

CREATE TRIGGER TrigUpdDmDdSpatialDomain
ON DmDdSpatialDomain
FOR UPDATE
AS
BEGIN
        DECLARE @attributeIdIns             attIdType,
                        @siteIdIns              int,
                        @attributeIdDel         attIdType,
                        @siteIdDel              int

BEGIN
        DECLARE updSpatialDomainCS CURSOR FOR
                SELECT i.siteId, i.attributeId,  d.siteId, d.attributeId
                FROM inserted i, deleted d
                WHERE i.siteId *= d.siteId
```

```
                AND i.attributeId *= d.attributeId

        OPEN updSpatialDomainCS

        FETCH updSpatialDomainCS INTO
                @siteIdIns,
                @attributeIdIns,
                @siteIdDel,
                @attributeIdDel

        WHILE @@sqlstatus != 2
        BEGIN
                IF @siteIdDel = null AND @attributeIdDel = null
                BEGIN

                        IF (select count(*) FROM DmDdCollECSAttributeXref
                                WHERE siteId = @siteIdIns
                                AND attributeId = @attributeIdIns) = 0
                        AND (SELECT count(*) FROM DmDdAdditionalAttributes
                                WHERE siteId = @siteIdIns
                                AND attributeId = @attributeIdIns) = 0
                        AND (SELECT count(*) FROM DmDdEquivalentAttributes
                                WHERE siteId = @siteIdIns
                                AND attributeId = @attributeIdIns) = 0
                        BEGIN
                                DELETE DmDdSpatialDomain
                                FROM DmDdSpatialDomain s, inserted i
                                WHERE s.siteId = i.siteId
                                AND s.attributeId = i.attributeId

                                INSERT DmDdSpatialDomain
                                SELECT * from deleted

                                RAISERROR 30515
                        "No refernce data exists for updated data in DmDdSpatialDomain, updating has
been aborted"
                                RETURN
                        END

                END /* if @siteIdDel = null and @attributeIdDel = null */

                FETCH updSpatialDomainCS INTO
                                @siteIdIns,
                                @attributeIdIns,
                                @siteIdDel,
                                @attributeIdDel

        END /* WHILE */
END
END
RETURN
go
```

Trigger: ti_dmddspatialkeywordclass

Trigger Code

```
/*  Insert trigger "ti_dmddspatialkeywordclass" for table "DmDdSpatialKeywordClass"  */
create trigger ti_dmddspatialkeywordclass on DmDdSpatialKeywordClass for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdECSCollection" must exist when inserting a child in "DmDdSpatialKeywordClass"  */
   if update(collectionId) or
     update(siteId)
   begin
     if (select count(*)
         from   DmDdECSCollection t1, inserted t2
         where  t1.collectionId = t2.collectionId
          and   t1.siteId = t2.siteId) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdSpatialKeywordClass".'
          goto error
        end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddstringdomain

Trigger Code

```
/*  Insert trigger "ti_dmddstringdomain" for table "DmDdStringDomain"  */
create trigger ti_dmddstringdomain on DmDdStringDomain for insert as
begin
   declare
     @numrows  int,
```

```
    @numnull  int,
    @errno    int,
    @errmsg   varchar(255)

  select  @numrows = @@rowcount
  if @numrows = 0
    return


  /*  Parent "DmDdStringType" must exist when inserting a child in "DmDdStringDomain"  */
  if update(siteId) or
    update(attributeId)
  begin
    if (select count(*)
        from   DmDdStringType t1, inserted t2
        where  t1.siteId = t2.siteId
         and   t1.attributeId = t2.attributeId) != @numrows
       begin
         select @errno  = 30002,
              @errmsg = 'Parent does not exist in "DmDdStringType". Cannot create child in
"DmDdStringDomain".'
         goto error
       end
  end

  return

/*  Errors handling  */
error:
  raiserror @errno @errmsg
  rollback  transaction
end
go
```

Trigger: ti_dmddstringinstrumentchar

Trigger Code

```
/*  Insert trigger "ti_dmddstringinstrumentchar" for table "DmDdStringInstrumentChar"  */
create trigger ti_dmddstringinstrumentchar on DmDdStringInstrumentChar for insert as
begin
  declare
    @numrows  int,
    @numnull  int,
    @errno    int,
    @errmsg   varchar(255)

  select  @numrows = @@rowcount
  if @numrows = 0
    return


  /*  Parent "DmDdInstrumentCharacteristic" must exist when inserting a child in
```

```
"DmDdStringInstrumentChar"  */
   if update(InstrumentCharacteristicName)
   begin
      if (select count(*)
          from   DmDdInstrumentCharacteristic t1, inserted t2
          where  t1.InstrumentCharacteristicName = t2.InstrumentCharacteristicName) != @numrows
         begin
            select @errno  = 30002,
                  @errmsg = 'Parent does not exist in "DmDdInstrumentCharacteristic". Cannot create child in
"DmDdStringInstrumentChar".'
            goto error
         end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddstringplatformchar

Trigger Code

```
/*  Insert trigger "ti_dmddstringplatformchar" for table "DmDdStringPlatformChar"  */
create trigger ti_dmddstringplatformchar on DmDdStringPlatformChar for insert as
begin
   declare
      @numrows  int,
      @numnull  int,
      @errno    int,
      @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
      return


   /*  Parent "DmDdPlatformCharacteristic" must exist when inserting a child in "DmDdStringPlatformChar"
*/
   if update(PlatformCharacteristicName)
   begin
      if (select count(*)
          from   DmDdPlatformCharacteristic t1, inserted t2
          where  t1.PlatformCharacteristicName = t2.PlatformCharacteristicName) != @numrows
         begin
            select @errno  = 30002,
                  @errmsg = 'Parent does not exist in "DmDdPlatformCharacteristic". Cannot create child in
"DmDdStringPlatformChar".'
            goto error
```

```
        end
    end

    return

/*  Errors handling  */
error:
    raiserror @errno @errmsg
    rollback  transaction
end
go
```

Trigger: ti_dmddstringsensorchar

Trigger Code

```
/*  Insert trigger "ti_dmddstringsensorchar" for table "DmDdStringSensorChar"  */
create trigger ti_dmddstringsensorchar on DmDdStringSensorChar for insert as
begin
    declare
      @numrows  int,
      @numnull  int,
      @errno    int,
      @errmsg   varchar(255)

    select  @numrows = @@rowcount
    if @numrows = 0
      return


    /*  Parent "DmDdSensorCharacteristic" must exist when inserting a child in "DmDdStringSensorChar"
*/
    if update(SensorCharacteristicName)
    begin
      if (select count(*)
          from   DmDdSensorCharacteristic t1, inserted t2
          where  t1.SensorCharacteristicName = t2.SensorCharacteristicName) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdSensorCharacteristic". Cannot create child in
"DmDdStringSensorChar".'
          goto error
        end
    end

    return

/*  Errors handling  */
error:
    raiserror @errno @errmsg
    rollback  transaction
end
```

go

Trigger: TrigInsDmDdStringType

Trigger Code

```
--*************************************************************
--BEGIN PROLOG
--
--TRIGGER NAME:              TrigInsDmDdStringType
--
--TABLE ACCESSED:   DmDdCollECSAttributeXref
--                                  DmDdAdditionalAttributes
--                                  DmDdStringType
--                                  inserted
--
--*************************************************************

CREATE TRIGGER TrigInsDmDdStringType
ON DmDdStringType
FOR INSERT
AS
BEGIN
        DECLARE @attributeId attIdType,
                        @siteId                 int,
                        @hasIt                  int

BEGIN
        DECLARE insStringTypeCS CURSOR FOR
                SELECT siteId, attributeId
                FROM inserted

        OPEN insStringTypeCS

        FETCH insStringTypeCS INTO
                @siteId,
                @attributeId

        WHILE @@sqlstatus != 2
        BEGIN
                EXEC ProcCheckAttriExist @siteId = @siteId,
                                                                @attributeId = @attributeId,
                                                                @hasIt = @hasIt output
                IF @hasIt = 0
                BEGIN
                        DELETE DmDdStringType
                        FROM DmDdStringType s, inserted i
                        WHERE s.siteId = i.siteId
                        AND s.attributeId = i.attributeId

                        RAISERROR 30515
                        "No refernce data exists for new record in DmDdStringType, inserting has been
```

```
aborted"
                        RETURN
                END

                FETCH insStringTypeCS INTO
                        @siteId,
                        @attributeId
        END /* WHILE */
END
END
RETURN
go
```

Trigger: TrigUpdDmDdStringType

Trigger Code

```
--*************************************************************
--BEGIN PROLOG
--
--TRIGGER NAME:                 TrigUpdDmDdStringType
--
--TABLE ACCESSED:   DmDdCollECSAttributeXref
--                                      DmDdStringType
--                                      inserted
--                                      deleted
--
--*************************************************************

CREATE TRIGGER TrigUpdDmDdStringType
ON DmDdStringType
FOR UPDATE
AS
BEGIN
        DECLARE @attributeIdIns                 attIdType,
                        @siteIdIns              int,
                        @attributeIdDel         attIdType,
                        @siteIdDel              int

BEGIN
        DECLARE updStringTypeCS CURSOR FOR
                SELECT i.siteId, i.attributeId,  d.siteId, d.attributeId
                FROM inserted i, deleted d
                WHERE i.siteId *= d.siteId
                AND i.attributeId *= d.attributeId

        OPEN updStringTypeCS

        FETCH updStringTypeCS INTO
                @siteIdIns,
                @attributeIdIns,
```

```
                @siteIdDel,
                @attributeIdDel

        WHILE @@sqlstatus != 2
        BEGIN
                IF @siteIdDel = null AND @attributeIdDel = null
                BEGIN

                        IF (select count(*) FROM DmDdCollECSAttributeXref
                                WHERE siteId = @siteIdIns
                                AND attributeId = @attributeIdIns) = 0
                        AND (SELECT count(*) FROM DmDdAdditionalAttributes
                                WHERE siteId = @siteIdIns
                                AND attributeId = @attributeIdIns) = 0
                        AND (SELECT count(*) FROM DmDdEquivalentAttributes
                                WHERE siteId = @siteIdIns
                                AND attributeId = @attributeIdIns) = 0
                        BEGIN
                                DELETE DmDdStringType
                                FROM DmDdStringType s, inserted i
                                WHERE s.siteId = i.siteId
                                AND s.attributeId = i.attributeId

                                INSERT DmDdStringType
                                SELECT * from deleted

                                RAISERROR 30515
                        "No refernce data exists for updated data in DmDdStringType, updating has been
aborted"
                                RETURN
                        END

                END /* if @siteIdDel = null and @attributeIdDel = null */

                FETCH updStringTypeCS INTO
                                @siteIdIns,
                                @attributeIdIns,
                                @siteIdDel,
                                @attributeIdDel

        END /* WHILE */
END
END
RETURN
go
```

Trigger: ti_dmddtemporal

Trigger Code

```
/*  Insert trigger "ti_dmddtemporal" for table "DmDdTemporal"  */
create trigger ti_dmddtemporal on DmDdTemporal for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
   if @numrows = 0
     return


   /*  Parent "DmDdECSCollection" must exist when inserting a child in "DmDdTemporal"  */
   if update(collectionId) or
     update(siteId)
   begin
     if (select count(*)
         from   DmDdECSCollection t1, inserted t2
         where  t1.collectionId = t2.collectionId
          and   t1.siteId = t2.siteId) != @numrows
        begin
          select @errno  = 30002,
                 @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdTemporal".'
          goto error
        end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

Trigger: ti_dmddtemporalkeywordclass

Trigger Code

```
/*  Insert trigger "ti_dmddtemporalkeywordclass" for table "DmDdTemporalKeywordClass"  */
create trigger ti_dmddtemporalkeywordclass on DmDdTemporalKeywordClass for insert as
begin
   declare
     @numrows  int,
     @numnull  int,
     @errno    int,
     @errmsg   varchar(255)

   select  @numrows = @@rowcount
```

```
   if @numrows = 0
      return


   /*  Parent "DmDdECSCollection" must exist when inserting a child in "DmDdTemporalKeywordClass"  */
   if update(collectionId) or
      update(siteId)
   begin
      if (select count(*)
          from   DmDdECSCollection t1, inserted t2
          where  t1.collectionId = t2.collectionId
           and   t1.siteId = t2.siteId) != @numrows
         begin
            select @errno  = 30002,
                  @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdTemporalKeywordClass".'
            goto error
         end
   end

   return

/*  Errors handling  */
error:
   raiserror @errno @errmsg
   rollback  transaction
end
go
```

4.1.10 Stored Procedures

In addition to triggers, the Sybase RDBMS supports the implementation of stored procedures for the purpose of enforcing business rules that ensure data integrity within the database. Stored procedures are parsed and compiled SQL code that reside in the database and may be called by name by an application, trigger or another stored procedure. Table 4-5 contains a summary listing of the stored procedures in the DM Subsystem database including the procedure name and a brief definition of the purpose of the procedure. Following this table is a listing of each stored procedure within the DM Subsystem database.

Table 4-5. Summary List of Stored Procedures (1 of 2)

Procedure Name	Description
ProcCheckAttriExist	Check for duplication of siteId and attributeId in DmDdCollECSAttributeXref or DmDdAdditionalAttributes or DmDdEquivalentAttributes If duplicated return an error code.
ProcCleanUpMappings	Used to clean up all mappings in the Data Dictionary
ProcDeleteAdditionalAttributes	Delete all related rows in all referenced tables -- when deleting a DmDdAdditionalAttributes row

Table 4-5. Summary List of Stored Procedures (2 of 2)

Procedure Name	Description
ProcDeleteAttributeRelated	Delete all related rows in all referenced tables -- for deleting a DmDdCollECSAttributeXref row -- or deleting a DmDdAdditionalAttributes row -- or deleting a DmDdEquivalentAttributes row
ProcDeleteCollECSAttributeXref	Delete all related rows in all referenced tables -- when deleting a DmDdCollECSAttributeXref row
ProcDeleteCollection	Delete all related rows in all referenced tables -- when deleting a DmDdECSCollection row
ProcDeleteECSAttributes	Delete all related rows in all referenceed tables -- when deleting a DmDdECSAttributes row
ProcDeleteEquivalentAttributes	Delete all related rows in all referenced tables -- when deleting a DmDdEquivalentAttributes row
ProcDeleteInfoMgr	Delete all related rows in all referenced tables -- when deleting a DmDdInfoMgr row
ProcDeleteInstrument	Delete all related rows in all referenced tables -- when deleting a DmDdECSInstrument row
ProcDeletePlatform	Delete all related rows in all referenced tables -- when deleting a DmDdECSPlatform row
ProcDeleteSensor	Delete all related rows in all referenced tables -- when deleting a DmDdECSSensor row

Procedure: ProcCheckAttriExist

Code

```
CREATE PROC ProcCheckAttriExist
        (@siteId                int,
         @attributeId    attIdType,
         @hasIt                 int             output)
AS
BEGIN
        IF (SELECT count(*) FROM DmDdCollECSAttributeXref
                        WHERE siteId = @siteId
                        AND attributeId = @attributeId) = 0
        BEGIN
                IF (SELECT count(*) FROM DmDdAdditionalAttributes
                        WHERE siteId = @siteId
                        AND attributeId = @attributeId) = 0
                BEGIN
                        IF (SELECT count(*) FROM DmDdEquivalentAttributes
                                WHERE siteId = @siteId
                                AND attributeId = @attributeId) = 0
                        BEGIN
                                SELECT @hasIt = 0
                                RETURN
                        END
                END
        END

        SELECT @hasIt = 1
```

```
        RETURN

END
go
```

Procedure: ProcCleanUpMappings

```
--/////////////////////////////////////////////////////////
-- Name : ProcCleanUpMappings
--
-- Description:
--
-- Delete all related records in all referenceing tables for mapping
--
-- Implementation Notes:
--/////////////////////////////////////////////////////////

CREATE PROC ProcCleanUpMappings
AS

BEGIN TRAN

        DECLARE @error int

        DELETE FROM DmDdKeywordAliasXref
        SELECT @error = @@error
        IF @error != 0
        BEGIN
                raiserror 30515
    "An error occured when delete DmDdKeywordAliasXref"
    ROLLBACK TRANSACTION
    RETURN
  END

        DELETE FROM DmDdAttributeXref
        SELECT @error = @@error
        IF @error != 0
        BEGIN
                raiserror 30515
    "An error occured when delete DmDdAttributeXref"
    ROLLBACK TRANSACTION
    RETURN
  END

        DELETE DmDdStringDomain
        FROM DmDdStringDomain, DmDdCollECSAttributeXref, DmDdInfoMgr,
```

```
                DmDdInfoMgrCollXref
        WHERE DmDdInfoMgr.infoMgrType LIKE "%IMS"
        AND DmDdInfoMgr.infoMgrName = DmDdInfoMgrCollXref.infoMgrName
        AND DmDdInfoMgrCollXref.collectionId = DmDdCollECSAttributeXref.collectionId
        AND DmDdInfoMgrCollXref.siteId = DmDdCollECSAttributeXref.siteId
        AND DmDdStringDomain.siteId = DmDdCollECSAttributeXref.siteId
        AND DmDdStringDomain.attributeId = DmDdCollECSAttributeXref.attributeId
        SELECT @error = @@error
        IF @error != 0
        BEGIN
                raiserror 30515
    "An error occured when delete DmDdStringDomain - for ECS dummy attr"
    ROLLBACK TRANSACTION
    RETURN
END

        DELETE DmDdStringType
        FROM DmDdStringType, DmDdCollECSAttributeXref, DmDdInfoMgr,
                DmDdInfoMgrCollXref
        WHERE DmDdInfoMgr.infoMgrType LIKE "%IMS"
        AND DmDdInfoMgr.infoMgrName = DmDdInfoMgrCollXref.infoMgrName
        AND DmDdInfoMgrCollXref.collectionId = DmDdCollECSAttributeXref.collectionId
        AND DmDdInfoMgrCollXref.siteId = DmDdCollECSAttributeXref.siteId
        AND DmDdStringType.siteId = DmDdCollECSAttributeXref.siteId
        AND DmDdStringType.attributeId = DmDdCollECSAttributeXref.attributeId
        SELECT @error = @@error
        IF @error != 0
        BEGIN
                raiserror 30515
    "An error occured when delete DmDdStringType - for ECS dummy attr"
    ROLLBACK TRANSACTION
    RETURN
END

        DELETE DmDdStringDomain
        FROM DmDdStringDomain, DmDdEquivalentAttributes, DmDdInfoMgr,
            DmDdInfoMgrCollXref
        WHERE DmDdInfoMgr.infoMgrType LIKE "%SDSRV%"
        AND DmDdInfoMgr.infoMgrName = DmDdInfoMgrCollXref.infoMgrName
        AND DmDdInfoMgrCollXref.collectionId = DmDdEquivalentAttributes.collectionId
        AND DmDdInfoMgrCollXref.siteId = DmDdEquivalentAttributes.siteId
        AND DmDdStringDomain.siteId = DmDdEquivalentAttributes.siteId
        AND DmDdStringDomain.attributeId = DmDdEquivalentAttributes.attributeId
        SELECT @error = @@error
        IF @error != 0
```

```
        BEGIN
                raiserror 30515
    "An error occured when delete DmDdStringDomain - for Equiv dummy attr"
    ROLLBACK TRANSACTION
    RETURN
END

        DELETE DmDdStringType
        FROM DmDdStringType, DmDdEquivalentAttributes, DmDdInfoMgr,
            DmDdInfoMgrCollXref
        WHERE DmDdInfoMgr.infoMgrType LIKE "%SDSRV%"
        AND DmDdInfoMgr.infoMgrName = DmDdInfoMgrCollXref.infoMgrName
        AND DmDdInfoMgrCollXref.collectionId = DmDdEquivalentAttributes.collectionId
        AND DmDdInfoMgrCollXref.siteId = DmDdEquivalentAttributes.siteId
        AND DmDdStringType.siteId = DmDdEquivalentAttributes.siteId
        AND DmDdStringType.attributeId = DmDdEquivalentAttributes.attributeId
        SELECT @error = @@error
        IF @error != 0
        BEGIN
                raiserror 30515
    "An error occured when delete DmDdStringType - for Equiv dummy attr"
    ROLLBACK TRANSACTION
    RETURN
END

        DELETE DmDdNumericDomain
        FROM DmDdNumericDomain, DmDdCollECSAttributeXref, DmDdInfoMgr,
                DmDdInfoMgrCollXref
        WHERE DmDdInfoMgr.infoMgrType LIKE "%IMS"
        AND DmDdInfoMgr.infoMgrName = DmDdInfoMgrCollXref.infoMgrName
        AND DmDdInfoMgrCollXref.collectionId = DmDdCollECSAttributeXref.collectionId
        AND DmDdInfoMgrCollXref.siteId = DmDdCollECSAttributeXref.siteId
        AND DmDdNumericDomain.siteId = DmDdCollECSAttributeXref.siteId
        AND DmDdNumericDomain.attributeId = DmDdCollECSAttributeXref.attributeId
        SELECT @error = @@error
        IF @error != 0
        BEGIN
                raiserror 30515
    "An error occured when delete DmDdNumericDomain - for ECS dummy attr"
    ROLLBACK TRANSACTION
    RETURN
END

        DELETE DmDdNumericDomain
        FROM DmDdNumericDomain, DmDdEquivalentAttributes, DmDdInfoMgr,
```

```
        DmDdInfoMgrCollXref
    WHERE DmDdInfoMgr.infoMgrType LIKE "%SDSRV%"
    AND DmDdInfoMgr.infoMgrName = DmDdInfoMgrCollXref.infoMgrName
    AND DmDdInfoMgrCollXref.collectionId = DmDdEquivalentAttributes.collectionId
    AND DmDdInfoMgrCollXref.siteId = DmDdEquivalentAttributes.siteId
    AND DmDdNumericDomain.siteId = DmDdEquivalentAttributes.siteId
    AND DmDdNumericDomain.attributeId = DmDdEquivalentAttributes.attributeId
    SELECT @error = @@error
    IF @error != 0
    BEGIN
            raiserror 30515
  "An error occured when delete DmDdNumericDomain - for Equiv dummy attr"
  ROLLBACK TRANSACTION
  RETURN
END

    DELETE DmDdSpatialDomain
    FROM DmDdSpatialDomain, DmDdCollECSAttributeXref, DmDdInfoMgr,
          DmDdInfoMgrCollXref
    WHERE DmDdInfoMgr.infoMgrType LIKE "%IMS"
    AND DmDdInfoMgr.infoMgrName = DmDdInfoMgrCollXref.infoMgrName
    AND DmDdInfoMgrCollXref.collectionId = DmDdCollECSAttributeXref.collectionId
    AND DmDdInfoMgrCollXref.siteId = DmDdCollECSAttributeXref.siteId
    AND DmDdSpatialDomain.siteId = DmDdCollECSAttributeXref.siteId
    AND DmDdSpatialDomain.attributeId = DmDdCollECSAttributeXref.attributeId
    SELECT @error = @@error
    IF @error != 0
    BEGIN
            raiserror 30515
  "An error occured when delete DmDdSpatialDomain - for ECS dummy attr"
  ROLLBACK TRANSACTION
  RETURN
END

    DELETE DmDdSpatialDomain
    FROM DmDdSpatialDomain, DmDdEquivalentAttributes, DmDdInfoMgr,
        DmDdInfoMgrCollXref
    WHERE DmDdInfoMgr.infoMgrType LIKE "%SDSRV%"
    AND DmDdInfoMgr.infoMgrName = DmDdInfoMgrCollXref.infoMgrName
    AND DmDdInfoMgrCollXref.collectionId = DmDdEquivalentAttributes.collectionId
    AND DmDdInfoMgrCollXref.siteId = DmDdEquivalentAttributes.siteId
    AND DmDdSpatialDomain.siteId = DmDdEquivalentAttributes.siteId
    AND DmDdSpatialDomain.attributeId = DmDdEquivalentAttributes.attributeId
    SELECT @error = @@error
    IF @error != 0
```

```
        BEGIN
                raiserror 30515
    "An error occured when delete DmDdSpatialDomain - for Equiv dummy attr"
    ROLLBACK TRANSACTION
    RETURN
END

        DELETE DmDdDateTimeDomain
        FROM DmDdDateTimeDomain, DmDdCollECSAttributeXref, DmDdInfoMgr,
                DmDdInfoMgrCollXref
        WHERE DmDdInfoMgr.infoMgrType LIKE "%IMS"
        AND DmDdInfoMgr.infoMgrName = DmDdInfoMgrCollXref.infoMgrName
        AND DmDdInfoMgrCollXref.collectionId = DmDdCollECSAttributeXref.collectionId
        AND DmDdInfoMgrCollXref.siteId = DmDdCollECSAttributeXref.siteId
        AND DmDdDateTimeDomain.siteId = DmDdCollECSAttributeXref.siteId
        AND DmDdDateTimeDomain.attributeId = DmDdCollECSAttributeXref.attributeId
        SELECT @error = @@error
        IF @error != 0
        BEGIN
                raiserror 30515
    "An error occured when delete DmDdDateTimeDomain - for ECS dummy attr"
    ROLLBACK TRANSACTION
    RETURN
END

        DELETE DmDdDateTimeDomain
        FROM DmDdDateTimeDomain, DmDdEquivalentAttributes, DmDdInfoMgr,
            DmDdInfoMgrCollXref
        WHERE DmDdInfoMgr.infoMgrType LIKE "%SDSRV%"
        AND DmDdInfoMgr.infoMgrName = DmDdInfoMgrCollXref.infoMgrName
        AND DmDdInfoMgrCollXref.collectionId = DmDdEquivalentAttributes.collectionId
        AND DmDdInfoMgrCollXref.siteId = DmDdEquivalentAttributes.siteId
        AND DmDdDateTimeDomain.siteId = DmDdEquivalentAttributes.siteId
        AND DmDdDateTimeDomain.attributeId = DmDdEquivalentAttributes.attributeId
        SELECT @error = @@error
        IF @error != 0
        BEGIN
                raiserror 30515
    "An error occured when delete DmDdDateTimeDomain - for Equiv dummy attr"
    ROLLBACK TRANSACTION
    RETURN
END

        DELETE DmDdCollECSAttributeXref
        FROM DmDdCollECSAttributeXref, DmDdInfoMgr, DmDdInfoMgrCollXref
```

```
        WHERE DmDdInfoMgr.infoMgrType LIKE "%IMS"
        AND DmDdInfoMgr.infoMgrName = DmDdInfoMgrCollXref.infoMgrName
        AND DmDdInfoMgrCollXref.collectionId = DmDdCollECSAttributeXref.collectionId
        AND DmDdInfoMgrCollXref.siteId = DmDdCollECSAttributeXref.siteId
        SELECT @error = @@error
        IF @error != 0
        BEGIN
                raiserror 30515
    "An error occured when delete DmDdCollECSAttributeXref"
    ROLLBACK TRANSACTION
    RETURN
  END

        DELETE DmDdEquivalentAttributes
        FROM DmDdEquivalentAttributes, DmDdInfoMgr, DmDdInfoMgrCollXref
        WHERE DmDdInfoMgr.infoMgrType LIKE "%SDSRV"
        AND DmDdInfoMgr.infoMgrName = DmDdInfoMgrCollXref.infoMgrName
        AND DmDdInfoMgrCollXref.collectionId = DmDdEquivalentAttributes.collectionId
        AND DmDdInfoMgrCollXref.siteId = DmDdEquivalentAttributes.siteId
        SELECT @error = @@error
        IF @error != 0
        BEGIN
                raiserror 30515
    "An error occured when delete DmDdEquivalentAttributes"
    ROLLBACK TRANSACTION
    RETURN
  END

COMMIT TRAN
RETURN
go
```

Procedure: ProcDeleteAdditionalAttributes

Code

```
--////////////////////////////////////////////////////////
-- Name : ProcDeleteAdditionalAttributes
--
-- Description:
--
-- Delete all related records in all referenceing tables
-- when delete a DmDdAdditionalAttributes record
--
-- Implementation Notes:
--////////////////////////////////////////////////////////

create proc ProcDeleteAdditionalAttributes(
```

```
        @attributeId      int,
        @siteId           int,
        @error   int        output)
as
        BEGIN TRANSACTION

        IF (SELECT count(*) FROM DmDdAdditionalAttributes
           WHERE attributeId = @attributeId
                AND siteId = @siteId) = 0
        BEGIN
      raiserror 30515
      "No such AdditionalAttributes exists"
                ROLLBACK TRANSACTION
                RETURN
        END

        else

        BEGIN  --START DELETE

                EXEC ProcDeleteAttributeRelated @attributeId = @attributeId,
                                @siteId = @siteId,
                                @error = @error output
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete Attribute Related"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdAdditionalAttributes
--
                DELETE DmDdAdditionalAttributes
             WHERE attributeId = @attributeId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdAdditionalAttributes"
                        ROLLBACK TRANSACTION
                        RETURN
                END
        END  -- end of START DELETE

        COMMIT TRANSACTION

RETURN
go
```

Procedure: ProcDeleteAttributeRelated

Code

```
--/////////////////////////////////////////////////////////
-- Name : ProcDeleteAttributeRelated
--
-- Description:
--
-- Delete all related records in all referenceing tables
-- for delete a DmDdCollECSAttributeXref record
-- or delete a DmDdAdditionalAttributes record
-- or delete a DmDdEquivalentAttributes record
--
-- Implementation Notes:
--/////////////////////////////////////////////////////////

create proc ProcDeleteAttributeRelated(
        @attributeId    int,
        @siteId         int,
        @error  int       output)
as
--
--DmDdAttributeXref
--
                DELETE DmDdAttributeXref
            WHERE attributeId = @attributeId
                AND siteId = @siteId
            OR attributeIdRef = @attributeId
                AND siteIdRef = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdAttributeXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdSpatialDomain
--
                DELETE DmDdSpatialDomain
            WHERE attributeId = @attributeId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdSpatialDomain"
                        ROLLBACK TRANSACTION
                        RETURN
```

```
        END
--
--DmDdNumericDomain
--
        DELETE DmDdNumericDomain
    WHERE attributeId = @attributeId
        AND siteId = @siteId

        SELECT @error = @@error
        IF @error != 0
        BEGIN
            raiserror 30515
            "An error occured when delete DmDdNumericDomain"
            ROLLBACK TRANSACTION
            RETURN
        END
--
--DmDdDateTimeDomain
--
        DELETE DmDdDateTimeDomain
    WHERE attributeId = @attributeId
        AND siteId = @siteId

        SELECT @error = @@error
        IF @error != 0
        BEGIN
            raiserror 30515
            "An error occured when delete DmDdDateTimeDomain"
            ROLLBACK TRANSACTION
            RETURN
        END
--
--DmDdKeywordAliasXref
--
        DELETE DmDdKeywordAliasXref
    WHERE attributeId = @attributeId
        AND siteId = @siteId
    OR attributeIdRef = @attributeId
        AND siteIdRef = @siteId

        SELECT @error = @@error
        IF @error != 0
        BEGIN
            raiserror 30515
            "An error occured when delete DmDdKeywordAliasXref"
            ROLLBACK TRANSACTION
            RETURN
        END
--
--DmDdStringDomain
--
        DELETE DmDdStringDomain
```

```
            WHERE attributeId = @attributeId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdStringDomain"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdStringType
--
                DELETE DmDdStringType
            WHERE attributeId = @attributeId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdStringType"
                        ROLLBACK TRANSACTION
                        RETURN
                END
RETURN
go
```

Procedure: ProcDeleteCollECSAttributeXref

Code

```
--/////////////////////////////////////////////////////////
-- Name : ProcDeleteCollECSAttributeXref
--
-- Description:
--
-- Delete all related records in all referenceing tables
-- when delete a DmDdCollECSAttributeXref record
--
-- Implementation Notes:
--/////////////////////////////////////////////////////////

create proc ProcDeleteCollECSAttributeXref(
        @attributeId    int,
        @siteId         int,
        @error  int       output)
as
        BEGIN TRANSACTION

        IF (SELECT count(*) FROM DmDdCollECSAttributeXref
           WHERE attributeId = @attributeId
```

```
                AND siteId = @siteId) = 0
        BEGIN
    raiserror 30515
    "No such CollECSAttributeXref exists"
                ROLLBACK TRANSACTION
                RETURN
        END

        else

        BEGIN  --START DELETE

                EXEC ProcDeleteAttributeRelated @attributeId = @attributeId,
                                @siteId = @siteId,
                                @error = @error output
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete Attribute Related"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdCollECSAttributeXref
--
                DELETE DmDdCollECSAttributeXref
          WHERE attributeId = @attributeId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdCollECSAttributeXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
        END  -- end of START DELETE

        COMMIT TRANSACTION

RETURN
go
```

Procedure: ProcDeleteCollection

Code

```
--////////////////////////////////////////////////////
-- Name : ProcDeleteCollection
--
-- Description:
--
```

```
-- Delete all related records in all referenceing tables
-- when delete a DmDdECSCollection record
--
-- Implementation Notes:
--////////////////////////////////////////////////////

create proc ProcDeleteCollection(
        @collectionId    int,
        @siteId          int,
        @error   int       output)
as
        BEGIN TRANSACTION

        IF (SELECT count(*) FROM DmDdECSCollection
           WHERE collectionId = @collectionId
                AND siteId = @siteId) = 0
        BEGIN
      raiserror 30515
      "No such collection exists"
                ROLLBACK TRANSACTION
                RETURN
        END

        else

        BEGIN  --START DELETE
--
--DmDdTemporalKeywordClass
--
                DELETE DmDdTemporalKeywordClass
                WHERE collectionId = @collectionId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdTemporalKeywordClass"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdSpatialKeywordClass
--
                DELETE DmDdSpatialKeywordClass
                WHERE collectionId = @collectionId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
```

```
                        "An error occured when delete DmDdSpatialKeywordClass"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdCollectionXref
--
                DELETE DmDdCollectionXref
                WHERE collectionIdRef = @collectionId
                AND siteIdRef = @siteId
                OR collectionIdAgg = @collectionId
                AND siteIdAgg = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdCollectionXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdSingleTypeCollection
--
                DELETE DmDdSingleTypeCollection
                WHERE collectionId = @collectionId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdSingleTypeCollection"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdMultipleCollection
--
                DELETE DmDdMultipleCollection
                WHERE collectionId = @collectionId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdMultipleCollection"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
```

```
--DmDdKeywordParameterXref
--
                DELETE DmDdKeywordParameterXref
                WHERE keywordId =
                        (SELECT keywordId FROM DmDdECSCollKeywordXref
                         WHERE collectionId = @collectionId
                         AND siteId = @siteId
                        )
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdKeywordParameterXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdECSCollKeywordXref
--
                DELETE DmDdECSCollKeywordXref
                WHERE collectionId = @collectionId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdECSCollKeywordXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdSensorXref
--
                DELETE DmDdSensorXref
                WHERE collectionId = @collectionId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdSensorXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdInstrumentXref
--
                DELETE DmDdInstrumentXref
```

```
                WHERE collectionId = @collectionId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdInstrumentXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdPlatformXref
--
                DELETE DmDdPlatformXref
                WHERE collectionId = @collectionId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdPlatformXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdSpatialDomain
--
                DELETE DmDdSpatialDomain
                WHERE collectionId = @collectionId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdSpatialDomain"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdNumericDomain
--
                DELETE DmDdNumericDomain
                WHERE collectionId = @collectionId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
```

```
                "An error occured when delete DmDdNumericDomain"
                ROLLBACK TRANSACTION
                RETURN
        END
--
--DmDdDateTimeDomain
--
        DELETE DmDdDateTimeDomain
        WHERE collectionId = @collectionId
        AND siteId = @siteId

        SELECT @error = @@error
        IF @error != 0
        BEGIN
                raiserror 30515
                "An error occured when delete DmDdDateTimeDomain"
                ROLLBACK TRANSACTION
                RETURN
        END
--
--DmDdKeywordAliasXref
--
        DELETE DmDdKeywordAliasXref
        WHERE collectionId = @collectionId
        AND siteId = @siteId
        OR collectionIdRef = @collectionId
        AND siteIdRef = @siteId

        SELECT @error = @@error
        IF @error != 0
        BEGIN
                raiserror 30515
                "An error occured when delete DmDdKeywordAliasXref"
                ROLLBACK TRANSACTION
                RETURN
        END
--
--DmDdStringDomain
--
        DELETE DmDdStringDomain
        WHERE collectionId = @collectionId
        AND siteId = @siteId

        SELECT @error = @@error
        IF @error != 0
        BEGIN
                raiserror 30515
                "An error occured when delete DmDdStringDomain"
                ROLLBACK TRANSACTION
                RETURN
        END
--
```

```
--DmDdStringType
--
                DELETE DmDdStringType
                WHERE collectionId = @collectionId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdStringType"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdAttributeXref
--
                DELETE DmDdAttributeXref
                WHERE collectionId = @collectionId
                AND siteId = @siteId
                OR collectionIdRef = @collectionId
                AND siteIdRef = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdAttributeXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdCollECSAttributeXref
--
                DELETE DmDdCollECSAttributeXref
                WHERE collectionId = @collectionId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdCollECSAttributeXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdAdditionalAttributes
--
                DELETE DmDdAdditionalAttributes
                WHERE collectionId = @collectionId
                AND siteId = @siteId
```

```
		SELECT @error = @@error
		IF @error != 0
		BEGIN
			raiserror 30515
			"An error occured when delete DmDdAdditionalAttributes"
			ROLLBACK TRANSACTION
			RETURN
		END
--
--DmDdEquivalentAttributes
--
		DELETE DmDdEquivalentAttributes
		WHERE collectionId = @collectionId
		AND siteId = @siteId

		SELECT @error = @@error
		IF @error != 0
		BEGIN
			raiserror 30515
			"An error occured when delete DmDdEquivalentAttributes"
			ROLLBACK TRANSACTION
			RETURN
		END
--
--DmDdInfoMgrCollXref
--
		DELETE DmDdInfoMgrCollXref
		WHERE collectionId = @collectionId
		AND siteId = @siteId

		SELECT @error = @@error
		IF @error != 0
		BEGIN
			raiserror 30515
			"An error occured when delete DmDdInfoMgrCollXref"
			ROLLBACK TRANSACTION
			RETURN
		END
--
--DmDdBoundingRectangle
--
		DELETE DmDdBoundingRectangle
		WHERE collectionId = @collectionId
		AND siteId = @siteId

		SELECT @error = @@error
		IF @error != 0
		BEGIN
			raiserror 30515
			"An error occured when delete DmDdBoundingRectangle"
			ROLLBACK TRANSACTION
```

```
                        RETURN
                END
--
--DmDdRangeDateTime
--
                DELETE DmDdRangeDateTime
                WHERE collectionId = @collectionId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdRangeDateTime"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdTemporal
--
                DELETE DmDdTemporal
                WHERE collectionId = @collectionId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdTemporal"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdECSCollection
--
                DELETE DmDdECSCollection
                WHERE collectionId = @collectionId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdECSCollection"
                        ROLLBACK TRANSACTION
                        RETURN
                END
        END  -- end of START DELETE

        COMMIT TRANSACTION
```

```
RETURN
go
```

Procedure: ProcDeleteECSAttributes

Code

```
--//////////////////////////////////////////////////////
-- Name : ProcDeleteECSAttributes
--
-- Description:
--
-- Delete all related records in all referenceing tables
-- when delete a DmDdECSAttributes record
--
-- Implementation Notes:
--//////////////////////////////////////////////////////

create proc ProcDeleteECSAttributes(
        @attributeName          char(32),
        @error   int        output)
as
        BEGIN TRANSACTION

        IF (SELECT count(*) FROM DmDdECSAttributes
                WHERE ECSAttributeName = @attributeName ) = 0
        BEGIN
      raiserror 30515
      "No such ECSAttribute exists"
                ROLLBACK TRANSACTION
                RETURN
        END

        else

        BEGIN  --START DELETE

        declare axref_cursor cursor for
                SELECT attributeId, siteId
                FROM DmDdCollECSAttributeXref
                WHERE ECSAttributeName = @attributeName

        open axref_cursor

        declare @attributeId      int,
                        @siteId                 int

        fetch axref_cursor into
                @attributeId,
                @siteId

        while @@sqlstatus != 2
        begin
```

```
                EXEC ProcDeleteCollECSAttributeXref
                                @attributeId = @attributeId,
                                @siteId = @siteId,
                                @error = @error output

                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete CollECSAttributeXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END

                fetch axref_cursor into
                @attributeId,
                @siteId

        end

        DELETE  DmDdECSAttributes
        WHERE ECSAttributeName = @attributeName

        SELECT @error = @@error
        IF @error != 0
        BEGIN
                raiserror 30515
                "An error occured when delete DmDdECSAttributes"
                ROLLBACK TRANSACTION
                RETURN
        END

        END  -- end of START DELETE

        COMMIT TRANSACTION

RETURN
go
```

Procedure: ProcDeleteEquivalentAttributes

Code

```
--////////////////////////////////////////////////////////
-- Name : ProcDeleteEquivalentAttributes
--
-- Description:
--
-- Delete all related records in all referenceing tables
-- when delete a DmDdEquivalentAttributes record
--
-- Implementation Notes:
--////////////////////////////////////////////////////////
```

```
create proc ProcDeleteEquivalentAttributes(
        @attributeId    int,
        @siteId         int,
        @error  int       output)
as
        BEGIN TRANSACTION

        IF (SELECT count(*) FROM DmDdEquivalentAttributes
           WHERE attributeId = @attributeId
                AND siteId = @siteId) = 0
        BEGIN
      raiserror 30515
      "No such EquivalentAttributes exists"
                ROLLBACK TRANSACTION
                RETURN
        END

        else

        BEGIN  --START DELETE

                EXEC ProcDeleteAttributeRelated @attributeId = @attributeId,
                                @siteId = @siteId,
                                @error = @error output
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete Attribute Related"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdEquivalentAttributes
--
                DELETE DmDdEquivalentAttributes
            WHERE attributeId = @attributeId
                AND siteId = @siteId

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdEquivalentAttributes"
                        ROLLBACK TRANSACTION
                        RETURN
                END
        END  -- end of START DELETE

        COMMIT TRANSACTION
```

```
RETURN
go
```

Procedure: ProcDeleteInfoMgr

Code

```
--//////////////////////////////////////////////////////
-- Name : ProcDeleteInfoMgr
--
-- Description:
--
-- Delete all related records in all referenceing tables
-- when delete a DmDdInfoMgr record
--
-- Implementation Notes:
--//////////////////////////////////////////////////////

create proc ProcDeleteInfoMgr(
        @infoMgrName char(20),
        @error                          int         output)
as
        BEGIN TRANSACTION

        IF (SELECT count(*) FROM DmDdInfoMgr
           WHERE infoMgrName = @infoMgrName
                ) = 0
        BEGIN
      raiserror 30515
      "No such InfoMgr exists"
                ROLLBACK TRANSACTION
                RETURN
        END

        else

        BEGIN  --START DELETE
--
--DmDdInfoMgrCollXref
--
                DELETE DmDdInfoMgrCollXref
           WHERE infoMgrName = @infoMgrName

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdInfoMgrCollXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdInfoMgrXref
```

```
--
                DELETE DmDdInfoMgrXref
            WHERE infoMgrName = @infoMgrName
            OR infoMgrParentName = @infoMgrName

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdInfoMgrXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdInfoMgr
--
                DELETE DmDdInfoMgr
            WHERE infoMgrName = @infoMgrName

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdInfoMgr"
                        ROLLBACK TRANSACTION
                        RETURN
                END
        END  -- end of START DELETE

        COMMIT TRANSACTION

RETURN
go
```

Procedure: ProcDeleteInstrument

Code

```
--//////////////////////////////////////////////////////
-- Name : ProcDeleteInstrument
--
-- Description:
--
-- Delete all related records in all referenceing tables
-- when delete a DmDdECSInstrument record
--
-- Implementation Notes:
--//////////////////////////////////////////////////////

create proc ProcDeleteInstrument(
        @instrumentShortNamechar(20),
        @error                          int         output)
as
```

```
        BEGIN TRANSACTION

        IF (SELECT count(*) FROM DmDdInstrument
           WHERE InstrumentShortName = @instrumentShortName
                ) = 0
        BEGIN
      raiserror 30515
      "No such Instrument exists"
                ROLLBACK TRANSACTION
                RETURN
        END

        else

        BEGIN  --START DELETE
--
--DmDdInstrumentXref
--
                DELETE DmDdInstrumentXref
           WHERE InstrumentShortName = @instrumentShortName

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdInstrumentXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdInstrumentPlatformXref
--
                DELETE DmDdInstrumentPlatformXref
           WHERE InstrumentShortName = @instrumentShortName

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdInstrumentPlatformXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdOperationModeClass
--
                DELETE DmDdOperationModeClass
           WHERE InstrumentShortName = @instrumentShortName

                SELECT @error = @@error
                IF @error != 0
                BEGIN
```

```
                raiserror 30515
                "An error occured when delete DmDdOperationModeClass"
                ROLLBACK TRANSACTION
                RETURN
        END
--
--DmDdSensorInstrumentXref
--
        DELETE DmDdSensorInstrumentXref
    WHERE InstrumentShortName = @instrumentShortName

        SELECT @error = @@error
        IF @error != 0
        BEGIN
                raiserror 30515
                "An error occured when delete DmDdSensorInstrumentXref"
                ROLLBACK TRANSACTION
                RETURN
        END
--
--DmDdIntInstrumentChar
--
        DELETE DmDdIntInstrumentChar
        WHERE InstrumentCharacteristicName in
                (SELECT InstrumentCharacteristicName
                 FROM DmDdInstrumentCharacteristic
         WHERE InstrumentShortName = @instrumentShortName)

        SELECT @error = @@error
        IF @error != 0
        BEGIN
                raiserror 30515
                "An error occured when delete DmDdIntInstrumentChar"
                ROLLBACK TRANSACTION
                RETURN
        END
--
--DmDdFloatInstrumentChar
--
        DELETE DmDdFloatInstrumentChar
        WHERE InstrumentCharacteristicName in
                (SELECT InstrumentCharacteristicName
                 FROM DmDdInstrumentCharacteristic
         WHERE InstrumentShortName = @instrumentShortName)

        SELECT @error = @@error
        IF @error != 0
        BEGIN
                raiserror 30515
                "An error occured when delete DmDdFloatInstrumentChar"
                ROLLBACK TRANSACTION
                RETURN
```

```
                END
--
--DmDdDateInstrumentChar
--
                DELETE DmDdDateInstrumentChar
                WHERE InstrumentCharacteristicName in
                        (SELECT InstrumentCharacteristicName
                         FROM DmDdInstrumentCharacteristic
                 WHERE InstrumentShortName = @instrumentShortName)

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdDateInstrumentChar"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdStringInstrumentChar
--
                DELETE DmDdStringInstrumentChar
                WHERE InstrumentCharacteristicName in
                        (SELECT InstrumentCharacteristicName
                         FROM DmDdInstrumentCharacteristic
                 WHERE InstrumentShortName = @instrumentShortName)

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdStringInstrumentChar"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdInstrumentCharacteristic
--
                DELETE DmDdInstrumentCharacteristic
            WHERE InstrumentShortName = @instrumentShortName

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdInstrumentCharacteristic"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdInstrument
--
```

```
                DELETE DmDdInstrument
            WHERE InstrumentShortName = @instrumentShortName

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdInstrument"
                        ROLLBACK TRANSACTION
                        RETURN
                END
        END  -- end of START DELETE

        COMMIT TRANSACTION

RETURN
go
```

Procedure: ProcDeletePlatform

Code

```
--////////////////////////////////////////////////////////
-- Name : ProcDeletePlatform
--
-- Description:
--
-- Delete all related records in all referenceing tables
-- when delete a DmDdECSPlatform record
--
-- Implementation Notes:
--////////////////////////////////////////////////////////

create proc ProcDeletePlatform(
        @platformShortName    char(20),
        @error                int          output)
as
        BEGIN TRANSACTION

        IF (SELECT count(*) FROM DmDdPlatform
           WHERE PlatformShortName = @platformShortName
                ) = 0
        BEGIN
      raiserror 30515
      "No such Platform exists"
                ROLLBACK TRANSACTION
                RETURN
        END

        else
```

```
        BEGIN  --START DELETE
--
--DmDdPlatformXref
--
                DELETE DmDdPlatformXref
            WHERE PlatformShortName = @platformShortName

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdPlatformXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdInstrumentPlatformXref
--
                DELETE DmDdInstrumentPlatformXref
            WHERE PlatformShortName = @platformShortName

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdInstrumentPlatformXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdIntPlatformChar
--
                DELETE DmDdIntPlatformChar
                WHERE PlatformCharacteristicName in
                        (SELECT PlatformCharacteristicName
                         FROM DmDdPlatformCharacteristic
                 WHERE PlatformShortName = @platformShortName)

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdIntPlatformChar"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdFloatPlatformChar
--
                DELETE DmDdFloatPlatformChar
                WHERE PlatformCharacteristicName in
```

```
                        (SELECT PlatformCharacteristicName
                         FROM DmDdPlatformCharacteristic
                 WHERE PlatformShortName = @platformShortName)

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdFloatPlatformChar"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdDatePlatformChar
--
                DELETE DmDdDatePlatformChar
                WHERE PlatformCharacteristicName in
                        (SELECT PlatformCharacteristicName
                         FROM DmDdPlatformCharacteristic
                 WHERE PlatformShortName = @platformShortName)

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdDatePlatformChar"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdStringPlatformChar
--
                DELETE DmDdStringPlatformChar
                WHERE PlatformCharacteristicName in
                        (SELECT PlatformCharacteristicName
                         FROM DmDdPlatformCharacteristic
                 WHERE PlatformShortName = @platformShortName)

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdStringPlatformChar"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdPlatformCharacteristic
--
                DELETE DmDdPlatformCharacteristic
            WHERE PlatformShortName = @platformShortName
```

```
                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdPlatformCharacteristic"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdPlatform
--
                DELETE DmDdPlatform
            WHERE PlatformShortName = @platformShortName

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdPlatform"
                        ROLLBACK TRANSACTION
                        RETURN
                END
        END  -- end of START DELETE

        COMMIT TRANSACTION

RETURN
go
```

Procedure: ProcDeleteSensor

Code

```
--//////////////////////////////////////////////////////
-- Name : ProcDeleteSensor
--
-- Description:
--
-- Delete all related records in all referenceing tables
-- when delete a DmDdECSSensor record
--
-- Implementation Notes:
--//////////////////////////////////////////////////////

create proc ProcDeleteSensor(
        @sensorShortName      char(20),
        @error                          int         output)
as
        BEGIN TRANSACTION

        IF (SELECT count(*) FROM DmDdSensor
           WHERE SensorShortName = @sensorShortName
                ) = 0
```

```
        BEGIN
      raiserror 30515
      "No such Sensor exists"
                ROLLBACK TRANSACTION
                RETURN
        END

        else

        BEGIN  --START DELETE
--
--DmDdSensorXref
--
                DELETE DmDdSensorXref
            WHERE SensorShortName = @sensorShortName

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdSensorXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdSensorInstrumentXref
--
                DELETE DmDdSensorInstrumentXref
            WHERE SensorShortName = @sensorShortName

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdSensorInstrumentXref"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdIntSensorChar
--
                DELETE DmDdIntSensorChar
                WHERE SensorCharacteristicName in
                        (SELECT SensorCharacteristicName
                         FROM DmDdSensorCharacteristic
                 WHERE SensorShortName = @sensorShortName)

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdIntSensorChar"
```

```
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdFloatSensorChar
--
                DELETE DmDdFloatSensorChar
                WHERE SensorCharacteristicName in
                        (SELECT SensorCharacteristicName
                         FROM DmDdSensorCharacteristic
                 WHERE SensorShortName = @sensorShortName)

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdFloatSensorChar"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdDateSensorChar
--
                DELETE DmDdDateSensorChar
                WHERE SensorCharacteristicName in
                        (SELECT SensorCharacteristicName
                         FROM DmDdSensorCharacteristic
                 WHERE SensorShortName = @sensorShortName)

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdDateSensorChar"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdStringSensorChar
--
                DELETE DmDdStringSensorChar
                WHERE SensorCharacteristicName in
                        (SELECT SensorCharacteristicName
                         FROM DmDdSensorCharacteristic
                 WHERE SensorShortName = @sensorShortName)

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdStringSensorChar"
                        ROLLBACK TRANSACTION
```

```
                        RETURN
                END
--
--DmDdSensorCharacteristic
--
                DELETE DmDdSensorCharacteristic
            WHERE SensorShortName = @sensorShortName

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdSensorCharacteristic"
                        ROLLBACK TRANSACTION
                        RETURN
                END
--
--DmDdSensor
--
                DELETE DmDdSensor
            WHERE SensorShortName = @sensorShortName

                SELECT @error = @@error
                IF @error != 0
                BEGIN
                        raiserror 30515
                        "An error occured when delete DmDdSensor"
                        ROLLBACK TRANSACTION
                        RETURN
                END
        END  -- end of START DELETE

        COMMIT TRANSACTION

RETURN
go
```

4.2 Flat File Usage

A flat file is an operating system file that is read and written serially. It is generally independent with respect to other files that exist. There are cases when the implementation of certain persistent data is better suited to a flat file than to a database (e.g., system configuration data, external interface data, log files). DM Subsystem file usage is detailed in this section via file, field, and domain definitions.

4.2.1 File Descriptions

A summary listing of the files in the DM Subsystem is given in Table 4-6 together with a brief description of the file usage. The use of these files will allow the user to change the database schema without recompiling the code. After a change, initialization must be run again.

Table 4-6. Flat File Descriptions (1 of 5)

File Name	Description
DmDdAddAttrFmt.dat	Additional Attribute Format. Used to process an Additional Attribute – verify attributes and reference to another table containing the format of the attribute.
DmDdAddAttrUpdFmt.dat	Additional Attribute Format
DmDdAttrGrpFmt.dat	Insert attribute Group format - Used to verify attributes and make sure we have the reuqired ones for an ECS Collection to insert into the database.
DmDdBoundRecFmt.dat	Insert Bounding Rectangle Format
DmDdCollKeyFmt.dat	Insert collectionkeyword format
DmDdCommand.dat	Command – Used to determine the command currently executed and reference to another table containing the format of the command.
DmDdDateTimeDmnFmt.dat	Date Time Domain Format
DmDdDateTimeDmnupdFmt.dat	Date Time Domain Update Format
DmDdDelRelCollFmt.dat	Release Collection Format
DmDdDelUnRelCollFmt.dat	Release Collection Format
DmDdDeleteAcronymFmt.dat	Delete Acronym Format - Used to verify necessary parts of the command message are present and to retrieve the table(s) name to be updated. Also stores data used to determine whether an argument is “required” or not and a reference to another table containing the required attributes for a given argument.
DmDdDeleteAttrGroupFmt.dat	Delete Attribute Group Format
DmDdDeleteBoundingRecFmt.dat	Delete Bounding Rectangle Format
DmDdDeleteCollFmt.dat	Delete Collection Format
DmDdDeleteCollKeyFmt.dat	Delete Collection Keywords Format
DmDdDeleteDataOrigFmt.dat	Delete Data Originator Format
DmDdDeleteEcsFmt.dat	Delete EcsAttribute Format
DmDdDeleteEquivAttrFmt.dat	Delete Equivalent Attributes Xref
DmDdDeleteGlossaryFmt.dat	Delete Glossary Format
DmDdDeleteIPXrefFmt.dat	Delete Instrument Platform Cross Reference Format

Table 4-6. Flat File Descriptions (2 of 5)

File Name	Description
DmDdDeleteInfoFmt.dat	Delete Info Manager Format
DmDdDeleteInfoMgrToCollXrefFmt.dat	Delete Info Manager to Coll Xref Format
DmDdDeleteInstXrefFmt.dat	Delete Instrument Xref Format
DmDdDeleteKeywordAliasFmt.dat	Delete Keyword Alias Format
DmDdDeleteKeywordFmt.dat	Delete Keyword Format
DmDdDeleteMetaLevelFmt.dat	Delete Attribute Group Format
DmDdDeletePlatXrefFmt.dat	Delete Platform Xref Format
DmDdDeleteSIXrefFmt.dat	Delete SensorInstrument Cross Ref Format
DmDdDeleteSpatialKeyFmt.dat	Delete Spatial Keyword Format
DmDdDeleteTemporalKeyFmt.dat	Delete Temporal Keyword Format
DmDdDeleteValidsMappingFmt.dat	Delete Valids Mapping Format
DmDdDiscTopFmt.dat	Discipline Topic Parameter Format
DmDdDiscTopicUpdFmt.dat	Update Discipline Topic Parameter Format
DmDdDisciplineFmt.dat	Discipline Description Format
DmDdEcsAttrFmt.dat	ECS Attribute Format
DmDdEcsAttrUpdFmt.dat	ECS Attribute Format for update
DmDdEcsCollFmt.dat	ECS Collection Format
DmDdEcsCollUpdFmt.dat	ECS Collection Format for update
DmDdEcsParameterFmt.dat	ECS Parameter Format
DmDdEcsParameterUpdFmt.dat	ECS Parameter Format for update
DmDdEquivAttrFmt.dat	Equivalent Attribute Format
DmDdEquivListFmt.dat	Equivalent Attribute List Format
DmDdEquivXrefFmt.dat	Insert Equivalent Attributes Xref Format
DmDdIPXrefFmt.dat	Insert Instrument Platform Xref Format
DmDdInfoFmt.dat	Insert Info Manager Format
DmDdInfoMgrToCollXrefFmt.dat	Insert Info Mgr To Coll Cross Reference Format
DmDdInfoParentFmt.dat	Insert Info Manager Parent Format
DmDdInfoUpdFmt.dat	Update Info Manager Format
DmDdInsRelCollFmt.dat	Release Collection Format
DmDdInsertAcronymFmt.dat	Insert Acronym Format
DmDdInsertAcronymFmt2.dat	Insert Acronym Format 2
DmDdInsertAttrFmt.dat	Insert Attributes Format
DmDdInsertAttrGroupFmt.dat	Insert Attribute Group Format
DmDdInsertBoundingRecFmt.dat	Insert Bounding Rectangle
DmDdInsertCollFmt.dat	Insert Collection Format
DmDdInsertCollKeywordFmt.dat	Insert Collection Keyword Value
DmDdInsertDataOrigFmt.dat	Insert Data Originator Format
DmDdInsertEcsFmt.dat	Insert EcsAttribute Format
DmDdInsertEcsFmt2.dat	Insert EcsAttribute Format 2
DmDdInsertEquivAttrFmt.dat	Insert Equivalent Attributes Xref
DmDdInsertGlossaryFmt.dat	Insert Glossary Format
DmDdInsertGlossaryFmt2.dat	Insert Glossary Format
DmDdInsertIPXrefFmt.dat	Insert Instrument Platform Cross Reference Format

Table 4-6. Flat File Descriptions (3 of 5)

File Name	Description
DmDdInsertInfoFmt.dat	Insert Info Manager Format
DmDdInsertInfoFmt2.dat	Insert Info Manager Format
DmDdInsertInfoMgrToCollXrefFmt.dat	Insert Info Manager to Coll Xref Format
DmDdInsertInstXrefFmt.dat	Insert Instrument Xref Format
DmDdInsertKeywordAliasFmt.dat	Insert Keyword Alias Format
DmDdInsertKeywordFmt.dat	Insert Keyword Format
DmDdInsertMetaLevelFmt.dat	Insert Attribute Group Format
DmDdInsertPlatXrefFmt.dat	Insert Platform Xref Format
DmDdInsertSensInsXrefFmt.dat	Insert SensorInstrument Cross Ref Format
DmDdInsertSensXrefFmt.dat	Insert SensorCross Ref Format
DmDdInsertSiteFmt.dat	Insert Site Format
DmDdInsertSiteFmt2.dat	Insert Stie Format 2
DmDdInsertSpatialKeyFmt.dat	Insert Spatial Keyword Format
DmDdInsertTemporalKeyFmt.dat	Insert Temporal Keyword Format
DmDdInsertValidsMappingFmt.dat	Insert Valids Mapping Format
DmDdInstCharFmt.dat	Insert Instrument Characteristic Format
DmDdInstCharUpdFmt.dat	Insert Instrument Characteristic Format for update
DmDdInstDateFmt.dat	Instrument Char Date Format
DmDdInstDescFmt.dat	Instrument Description Format
DmDdInstFloatFmt.dat	Instrument Char Float Format
DmDdInstFmt.dat	Instrument Format
DmDdInstIntFmt.dat	Instrument Char Int Format
DmDdInstOperFmt.dat	Instrument Operation Mode Format
DmDdInstStringFmt.dat	Instrument Char String Format
DmDdInstUpdFmt.dat	Instrument format for update
DmDdInstXrfeFmt.dat	Insert Instrument Corss Reference Fromat
DmDdJoinablePairs.dat	Joinable Pairs table: Allows the user to decipher a join list for a specified search
DmDdKeywordAliasFmt.dat	Insert Keyword Alias Format
DmDdMetaLevelFmt.dat	Insert Attrbute Group Format
DmDdMultiTypeCollFmt.dat	Multiple Type Collection Format
DmDdNumericDmnFmt.dat	Numeric Domain Format
DmDdNumericDmnUpdFmt.dat	Numeric Domain Update Format
DmDdParamDescFmt.dat	Parameter Description Format
DmDdPlatCharFmt.dat	Platform Characteristic Format
DmDdPlatCharUpdFmt	Platform Characteristic Format for update
DmDdPlatDateFmt.dat	Platform Char Date Format
DmDdPlatDescFmt.dat	Platform Description Format
DmDdPlatDescUpdFmt.dat	Platform description Format for update
DmDdPlatFloatFmt.dat	Platform Char Float Format

Table 4-6. Flat File Descriptions (4 of 5)

File Name	Description
DmDdPlatFmt.dat	Platform Format
DmDdPlatIntFmt.dat	Platform Char Int Format
DmDdPlatStringFmt.dat	Platform Char String Format
DmDdPlatUpdFmt.dat	Platform Format for update
DmDdPlatXrefFmt.dat	Insert Platform Cross Reference Format
DmDdRangeDateTimeFmt.dat	Range Date Time Format
DmDdRangeDateTimeUpdFmt.dat	Range Date Time Format for update
DmDdRelCollFmt.dat	Release Collection Format
DmDdSIXrefFmt.dat	Insert Sensor Instrument Xref Format
DmDdSkeyFmt.dat	Insert Spatial keyword Format
DmDdSensCharFmt.dat	Sensor Characteristic Format
DmDdSensCharUpdFmt	Sensor Characteristic Format for update
DmDdSensDateFmt.dat	Sensor Char Date Format
DmDdSensDescFmt.dat	Sensor Description Format
DmDdSensDescUpdFmt.dat	Sensor description Format for update
DmDdSensFloatFmt.dat	Sensor Char Float Format
DmDdSensFmt.dat	Sensor Format
DmDdSensIntFmt.dat	Sensor Char Int Format
DmDdSensStringFmt.dat	Sensor Char String Format
DmDdSensUpdFmt.dat	Sensor Format for update
DmDdSensXrefFmt.dat	Insert Sensor Cross Reference Format
DmDdSingleTypeCollFmt.dat	Single Type Collection Format
DmDdSpatDmnFmt.dat	Spatial Domain Format
DmDdSpatKeyFmt.dat	Spatial Key Format for update
DmDdStringDmnFmt.dat	String Domain Format
DmDdStringDmnUpdFmt.dat	String Domain Format for Update
DmDdTkeyFmt.dat	Insert temporal Keyword Fromat
DmDdTableArgInfo.dat	Used to specify if the Collection Id, Site Id, attribute Id, and Keyword Id are required
DmDdTableNameList.dat	List of format file names under different specified levels.
DmDdTableNameXref.dat	Cross-reference table: - Serves two purposes: - 1) allows the user to determine the database argument name based on the command argument and table names - 2) Allows the user to determine which if the coomand arguments/db
DmDdTempFmt.dat	Temporal Format
DmDdTempKeyFmt.dat	Temporal Keyword Format
DmDdTempKeyUpdFmt.dat	Temporal Keyword Format for Update
DmDdTempUpdFmt.dat	Temporal Format for Update

Table 4-6. Flat File Descriptions (5 of 5)

File Name	Description
DmDdtermDescFmt.dat	Term Description Format
DmDdTopicDescFmt.dat	Topic Description Format
DmDdUpdateAcronymFmt.dat	Update Acronym Format
DmDdUpdateAttrGroupFmt.dat	Update Attribute Group Format
DmDdUpdateAttrGroupFmt2.dat	Update attribute Group 2 Format
DmDdUpdateCollFmt.dat	Update Collection Format
DmDdUpdateDataOrigFmt.dat	Update Data Originator Format
DmDdUpdateEcsFmt.dat	Update ECS Attrbute Format
DmDdUpdateEcsFmt2.dat	Update ECS Attrbute Format 2
DmDdUpdateGlossaryFmt.dat	Update Glossary Format
DmDdUpdateInfoFmt.dat	Update Info Manager Format
DmDdUpdateKeywordFmt.dat	Update keyword Format
DmDdUpdateMetaLevelFmt.dat	Update Attribute Group Format
DmDdUpdateMetaLevelFmt2.dat	Update Attribute Group Format 2
DmDdValidsMappingFmt.dat	Insert Attribute Group Fromat
DmDdVarDescFmt.dat	Variable Description Format

4.2.2 Field Specifications

Brief specifications of the fields present within the DM Subsystem flat files are contained in Table 4-7. Files with the same fields are grouped together alphabetically.

Table 4-7. Flat File Field Specifications

File Name	Field Name	Description
DmDdDelRelCollFmt.dat DmDdDelUnRelCollFmt.dat DmDdDeleteAcronymFmt.dat DmDdDeleteAttrGroupFmt.dat DmDdDeleteBoundingRecFmt.dat	Attribute (Arg) Name	Argument names associated with the actual database argument names via the cross-reference file: DmDdTableNameXref.dat
DmDdDeletCollFmt.dat DmDdDeleteCollKeyFmt.dat	Database Table Name	Name of the related database table
DmDdDeleteDataOrigFmt.dat DmDdDeleteEcsFmt.dat	Req	Specify if the argument is required
DmDdDeleteEquivAttrFmt.dat DmDdDeleteGlossaryFmt.dat	Mult	Specify if multiple instance allowed
DmDdDeleteIPXrefFmt.dat DmDdDeleteInfoFmt.dat	Case	Grouping Information
DmDdDeleteInfoMgrToCollXrefFmt.dat	Grp	Group number that this entry is in; Can be any number greater than 0

File Name	Field Name	Description
DmDdDeleteInstXrefFmt.dat		
DmDdDeleteKeywordAliasFmt.dat	Format Table	The table which specifies the argument format
DmDdDeleteKeywordFmt.dat		
DmDdDeleteMetaLevelFmt.dat		
DmDdDeletePlatXrefFmt.dat		Cross Reference table which needs t
DmDdDeleteSIXrefFmt.dat	X-ref Db Table	be updated
DmDdDeleteSpatialKeyFmt.dat		
DmDdDeleteTemporalKeyFmt.dat		Name of attribute field which needs t
DmDdDeleteValidsMappingFmt.dat	Attribute Name	be verified
DmDdEquivXrefFmt.dat		
DmDdInsertAcronymFmt.dat		
DmDdInsertAcronymFmt2.dat		
DmDdInsertAttrFmt.dat		
DmDdInsertAttrGroupFmt.dat		
DmDdInsertBoundingRecFmt.dat		
DmDdInsertCollFmt.dat		
DmDdInsertCollKeywordFmt.dat		
DmDdInsertDataOrigFmt.dat		
DmDdInsertEcsFmt.dat		
DmDdInsertEquivAttrFmt.dat		
DmDdInsertGlossaryFmt.dat		
DmDdInsertIPXrefFmt.dat		
DmDdInsertInfoFmt.dat		
DmDdInsertInfoMgrToCollXrefFmt.dat		
DmDdInsertInstXrefFmt.dat		
DmDdInsertKeywordAliasFmt.dat		
DmDdInsertKeywordFmt.dat		
DmDdInsertMetaLevelFmt.dat		
DmDdInsertPlatXrefFmt.dat		
DmDdInsertSIXrefFmt.dat		
DmDdInsertSpatialKeyFmt.dat		
DmDdInsertTemporalKeyFmt.dat		
DmDdInsertValidsMappingFmt.dat		
DmDdKeywordAliasFmt.dat		
DmDdUpdateAcronymFmt.dat		
DmDdUpdateAttrGroupFmt.dat		
DmDdUpdateCollFmt.dat		
DmDdUpdateDataOrigFmt.dat		
DmDdUpdateEcsFmt.dat		
DmDdUpdateGlossaryFmt.dat		
DmDdUpdateInfoFmt.dat		
DmDdUpdateKeywordFmt.dat		
DmDdUpdateMetaLevelFmt.dat		

File Name	Field Name	Description
DmDdAddAttrFmt.dat DmDdAddAttrUpdFmt.dat DmDdAttrGrpFmt.dat DmDdBoundRecFmt.dat DmDdCollKeyFmt.dat	Attribute (Arg) Name	Argument names associated with the actual database argument names via the cross-reference file: DmDdTableNameXref.dat
DmDdCommand.dat	Database Table Name	Name of the related database table
DmDdDateTimeDmnFmt.dat DmDdDateTimeDmnupdFmt.dat DmDdDiscTopFmt.dat	Req	Specify if the argument is required
DmDdDiscTopicUpdFmt.dat DmDdDisciplineFmt.dat	Mult	Specify if multiple instance allowed
DmDdEcsAttrFmt.dat DmDdEcsAttrUpdFmt.dat	Case	Grouping Information
DmDdEcsCollFmt.dat DmDdEcsCollUpdFmt.dat	Grp	Group number that this entry is in; Can be any number greater than 0
DmDdEcsParameterFmt.dat DmDdEcsParameterUpdFmt.dat DmDdEquivAttrFmt.dat DmDdEquivListFmt.dat DmDdEquivXrefFmt.dat DmDdIPXrefFmt.dat DmDdInfoFmt.dat DmDdInfoMgrToCollXrefFmt.dat DmDdInfoParentFmt.dat DmDdInfoUpdFmt.dat DmDdInsertAcronymFmt2.dat DmDdInsertSiteFmt2.dat DmDdInstCharFmt.dat DmDdInstCharUpdFmt.dat DmDdInstDateFmt.dat DmDdInstDescFmt.dat DmDdInstFloatFmt.dat DmDdInstFmt.dat DmDdInstIntFmt.dat DmDdInstOperFmt.dat DmDdInstStringFmt.dat DmDdInstUpdFmt.dat DmDdInstXrfeFmt.dat DmDdMetaLevelFmt.dat DmDdMultiTypeCollFmt.dat DmDdNumericDmnFmt.dat DmDdNumericDmnUpdFmt.dat DmDdParamDescFmt.dat DmDdPlatCharFmt.dat DmDdPlatCharUpdFmt DmDdPlatDateFmt.dat DmDdPlatDescFmt.dat DmDdPlatDescUpdFmt.dat	Format Table	The table which specifies the argument format

File Name	Field Name	Description
DmDdCommand.dat	Command	Name of the command used to determine the command currently being executed and reference to another table containing the format fc the command
	Format Table	Format file name
	Special	Used to specifiy the type of special cases: None, replace, NewList, NewList/Replace
DmDdJoinablePairs.dat	Table Name	Name of the Database Table 1
	Db Argument	Attribute Name in Table 1
	Table Name	Name of the Database Table 2
	Db Argument	Attribute Name in Table 2
DmDdTableArgInfo.dat	Table	Name of the Database Table
	Collection Id	Specify if Collection Id is required
	Site Id	Specify if Siteid is required
	Attribute Id	Specify if attribute id is required
	Keyword Id	Specify of keyword id is required
DmDdTableNameList.dat	LEVEL 1	Tells initializer the level for this file is
	LEVEL 2	Tells initializer the level for this file is
	LEVEL 3	Tells initializer the level for this file is
DmDdTableNameXref.dat	Table	Name of the Database table
	Command Argument	The command argument name
	DB Argument	The database argument name
	Primary	Specify if the argument is the primary key

4.2.3 Domain Definitions

Domain definitions specify the data type and valid content of fields within a file (e.g., specific values for a limited set of data, ranges of numeric data, units of measure for applicable data).

This information is generally used by software to edit incoming data for validity prior to writing or changing data within the file. Use of domain values in updating (adding and changing) records within files preserves the integrity of the data within the file.

To create flat files for the DM subsystem, the following format rules should be followed:

- Table format rules:

1) Blank lines are allowed.
2) Comments are designated by "##" as the first non-whitespace character.
3) The level must be the first non-comment/non-blank line in the file.
4) The lines must not be longer than the maximum prescribed in the software (currently 150).
5) Preceding/trailing blanks will be stripped off.
6) If tables fields are added/deleted, then need to be adfded/deleted from the initialize functions.
7) If the Format Table is specified for level 3, then the Database Table must also be specified.
8) The first 7 characters of the file name determine (for level 2 tables) what type of command it is:
   - DmDdIns … = ins
   - DmDdUpd … = Update
   - DmDddel … = Delete.

- Argument/Parametr Names

They are specified by the user. They are associated with the actual database argument names via the cross-reference file: DmDdTableNameXref.dat.

- Level

Tells the initializer the level for this file (1 to 3), 1 being the highest.

- Req

Is this argument required? (Yes/No/-

"-" indicates that this is in a group, so look to the group case to find out the required status.

- Mult

Are multiple instance (more than 1) allowed? Yes/No.

- Format

The table which specifies the argument format. Special processing is hard-coded inside process message.

- X-Ref

Specifies the cross-reference table which needs to be updated.

-Attr Name

Name of attribute field which needs to be verified.

Special Case: To handle a two-table join, this field can be set (for delete only) tp JOIN1 or JOIN2. JOIN1 indicates the primary table that will have rows deleted where JOIN1 table arg name = xyz and the JOIN2 table arg name = abc (xyz and abc supplied in GL Parameterlist). The JOIN1 entry should be specified first in this table.

- Case

0=No grouping

1= It is not required that any be present; more than one cannot be present e.g. 0 or 1 present.

2= It is not required that any be present; More than one can be present e.g. 0 or more present.

3 = It is required that one be present; Exactly one and only one must be present e.g 1 present.

4 = It is required that one be present; More than one can be rpesent e.g.>=1 present.

9 = SPECIAL processing for these fields

- Grp/Group

Can be any number greater than 0; It is applicable only within this table. It is ignored for Case 0. Groups can have two or more entries.

5. Performance and Tuning Factors

5.1 Indexes

An index provides a means of locating a row in a database table based on the value of a specific column(s), without having to scan all data in the table. When properly implemented, indexes can significantly decrease the time it takes to retrieve data, thereby increasing performance. Sybase allows the definition of two types of indexes, clustered and non-clustered.

In a clustered index, the rows in a database table are physically stored in sequence-determined by the index. Clustered indexes are particularly useful, when the data is frequently retrieved in sequential order. Only one clustered index may be defined per table.

Non-clustered indexes differ from their clustered counterpart, in that, data is not physically stored in sorted order—newly added rows are stored at the end of the related database table.

There are no indexes currently defined for the DM Subsystem database.

5.2 Segments

Sybase supports the declaration of segments. A segment is a named pointer to a storage device(s). Segments are used to physically allocate a database object to a particular storage device. Segments defined for the DM Subsystem and all other subsystem databases are described in Table 5-1.

Table 5-1. Segment Descriptions

Segment Name	Description
default	Tables, indexes, and objects
logsegment	SYSLOGS, Transaction Logs
systemsegment	System tables and indexes

5.3 Named Caches

A cache is a block of memory that is used by Sybase to retain and manage data pages that are currently being processed. A *named cache* is a block of memory that is named and used by the DBMS to store these frequently accessed data pages. Assigning a database table to named cache causes accessed pages to be loaded into memory and retained. The named cache does not need to be allocated to accommodate the entire database table since the DBMS manages the cache according to use. Named caches greatly increase performance by eliminating the time associated for disk input and output (I/O). There are no named caches that are currently defined for the DM Subsystem database.

This page intentionally left blank.

6. Database Security

6.1 Approach

The database security discussed within this section is bounded to security implementation within the Sybase SQL Server DBMS. A Sybase general approach to security is adopted as illustrated in Figure 6-1.

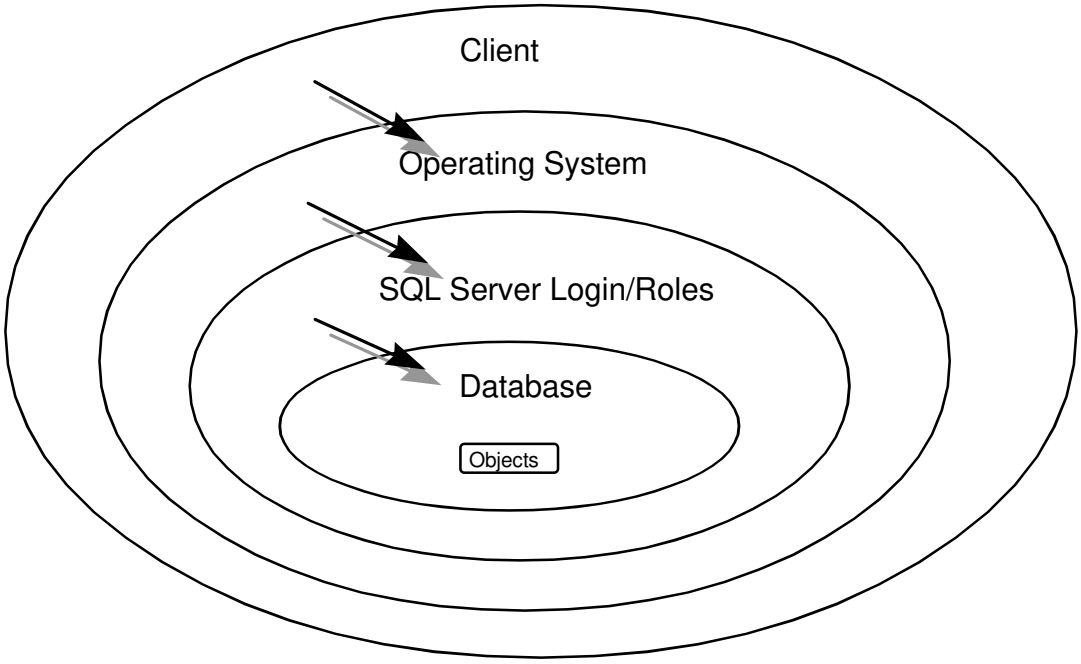

Figure 6-1. Sybase General Approach to SQL Server Security[2]

The client (user) requires a SQL Server login to access the DBMS. The System Administrator may grant or revoke login by various roles (e.g., sa, sso, oper). Roles are defined in Section 6.5. The login is assigned to a user with certain related permissions for gaining access to particular objects (e.g., database tables, views, commands) within the database. Descriptions of ECS user and system administrator permissions follows.

[2] Reference Sybase Student Guide: *Advanced SQL Server Administration*.

6.2 Initial Users

After initial installation users will have been assigned permissions for access to the DM Subsystem database. Users include scientists and other interested persons, operations and administrative personnel, and software. The level of access is limited to that associated with their assigned group, objects, and/or role. A definition of each if these groups, objects, and roles is given in the following Sections.

6.3 Groups

Groups are a means of logically associating users with similar data access needs. Once a group has been defined, object and command permissions can be granted to that group. A user who is member of a group inherits all of the permissions granted to that group. No groups have been initially defined in the DM Subsystem "EcDmDictService" default database. Object Permissions are set within the installation scripts of the DM Subsystem for each object/user access.

6.4 Objects

Permissions must be assigned for the user to gain access to the objects within the database. Examples of objects associated with the database include database tables, views, commands. The initial object permissions for the DM Subsystem database are identified herein.

Permissions are identified in Table 6-1. A specification of the user permissions is contained in Table 6-2.

Table 6-1. Permission Key

Permission	Description
A	All
S	Select
I	Insert
U	Update
D	Delete
E	Execute

Table 6-2. Object Specifications

Group	SYBASE LOGIN	Permissions Granted					
		A	S	I	U	D	E
No groups	EcDmDimServer		Y				
No groups	EcDmLimServer		Y				
No groups	EcDmDictServer	Y					
No groups	EcDmV0ToEcsGateway		Y				
No groups	EcDmEcsToV0Gateway		Y				
No groups	EcDmAsterToEcsGateway		Y				
No groups	EcDmEcsToAsterGateway		Y				
No groups	EcDmLmClient		Y				

6.5 Roles

Roles were introduced in Sybase 10 to allow a structured means for granting users the permissions needed to perform standard database administration activities and also provide a means for easily identifying such users. There are six pre-defined roles that may be assigned to a user. A definition of each of these roles follows as well as a description of the types of activities that may be performed by each role.

System Administrator (*sa_role*): This role is used to grant a specific user permissions needed to perform standard system administrator duties including:

- installing SQL server and specific SQL server modules
- managing the allocation of physical storage
- tuning configuration parameters
- creating databases

Site Security Officer (*sso_role*): This role is used to grant a specific user the permissions needed to maintain SQL server security including:

- adding server logins
- administrating passwords
- managing the audit system
- granting users all roles except the *sa_role*

Operator (*oper_role*): This role is used to grant a specific user the permissions needed to perform standard functions for the database including:

- dumping transactions and databases

- loading transactions and databases

Navigator (*navigator_role*): This role is used to grant a specific user the permissions needed to manage the navigation server.

Replication (*replication_role*): This role is used to grant a specific user the permissions needed to manage the replication server.

Sybase Technical Support (*sybase_ts_role*): This role is used to grant a specific user the permissions needed to execute *database consistency checker (dbcc)*, a Sybase supplied utility supporting commands that are normally outside of the realm of routine system administrator activities.

7. Scripts

The scripts identified in this section may be found in the directory named /ecs/formal/DM/DDICT/src/database.

7.1 Installation Scripts

Scripts used to support installation of the DM Subsystem database are listed in Table 8-1.

Table 7-1. Installation Scripts

Script File	Description
DmDdCreateAllRun.csh	Script that is used to install the DM database
DbRun.csh	Script that is used to install the DM database
DmDdAdditionalAttributesDel.sp	SQL file to create the initial DM database (1 of 13)
DmDdAddUser.sql	SQL file to create the initial DM database (2 of 13)
DmDdCollDel.sp	SQL file to create the initial DM database (3 of 13)
DmDdCollECSAttributeXrefDel.sp	SQL file to create the initial DM database (4 of 13)
DmDdCreateAll.sql	SQL file to create the initial DM database (5 of 13)
DmDdCreateConstraint.sql	SQL file to create the initial DM database (6 of 13)
DmDdECSAttributesDel.sp	SQL file to create the initial DM database (7 of 13)
DmDdEquivalentAttributesDel.sp	SQL file to create the initial DM database (8 of 13)
DmDdGrantUser.sql	SQL file to create the initial DM database (9 of 13)
DmDdInfoMgrDel.sp	SQL file to create the initial DM database (10 of 13)
DmDdInstrumentDel.sp	SQL file to create the initial DM database (11 of 13)
DmDdPlatformDel.sp	SQL file to create the initial DM database (12 of 13)
DmDdSensorDel.sp	SQL file to create the initial DM database (13 of 13)

7.2 De-Installation Scripts

Scripts used to support de-installation of the DM Subsystem database are listed in Table 8-2.

Table 7-2. De-Installation Scripts

Script File	Description
DmDdDropAllRun.csh	Script used for de-installation of the DM database
DmDdDropAll.sql	SQL code to perform de-installation of the DM database

7.3 Backup and Recovery Scripts

Scripts used to perform backup and recovery of the DM Subsystem database are listed in Table 7-3.

Table 7-3. Backup and Recovery Scripts

Script File	Description
bcp.copyin.csh	Script to restore the data into the database
bcp.copyout.csh	Script to backup the data from the database

7.4 Miscellaneous Scripts

Miscellaneous scripts and input data files that are applicable to the DM Subsystem database are listed in Table 7-4.

Table 7-4. Miscellaneous Scripts and Input Data Files

Script or Data File	Description
bcp.initdata.csh	Script used to populat initial data (.dat files)
DmDdECSAttributes.dat	A flat data file for populating the initial data (1 of 13)
DmDdECSAttributesShapes.dat	A flat data file for populating the initial data (2 of 13)
DmDdECSDiscipline.dat	A flat data file for populating the initial data (3 of 13)
DmDdECSTerm.dat	A flat data file for populating the initial data (4 of 13)
DmDdECSTopic.dat	A flat data file for populating the initial data (5 of 13)
DmDdECSVariable.dat	A flat data file for populating the initial data (6 of 13)
DmDdInfoMgr.dat	A flat data file for populating the initial data (7 of 13)
DmDdInfoMgrXref.dat	A flat data file for populating the initial data (8 of 13)
DmDdInstrument.dat	A flat data file for populating the initial data (9 of 13)
DmDdMetadataLevel.dat	A flat data file for populating the initial data (10 of 13)
DmDdPlatform.dat	A flat data file for populating the initial data (11 of 13)
DmDdSensor.dat	A flat data file for populating the initial data (12 of 13)
DmDdSensorSDSRV.dat	A flat data file for populating the initial data (13 of 13)
DmDdInsertValidsMappingFmt.dat	Used to support update of the DDICT

Abbreviations and Acronyms

ACL	Access Control List
ANSI	American National Standards Institute
CASE	Computer-Aided Software Engineering
CCB	Change Control Board (Raytheon Convention)
	Configuration Control Board (NASA Convention)
CCR	configuration change request
CD	contractual delivery 214-001
CDRL	Contract Data Requirements List
CSCI	computer software configuration item
CSMS	Communications and Systems Management Segment (ECS)
DAAC	Distributed Active Archive Center
DBMS	database management system
DCN	document change notice
DDICT	Data Dictionary CSCI
DDIST	Data Distribution Services CSCI
DID	data item description
DM	Data Management CSCI
DmDd	data management data dictionary
ECS	EOSDIS Core System
EDC	EROS Data Center
EDHS	ECS Data Handling System
EOS	Earth Observing System
EOSDIS	Earth Observing System Data and Information System
ERD	entity relationship diagram
GSFC	Goddard Space Flight Center
ID	identification
INS	Ingest Subsystem

IOS	Interoperability Subsystem
I/O	Input and Output
LaRC	Langley Research Center (DAAC)
MIME	multipurpose internet mail extension
MSS	Management Support Subsystem
NAS	National Academy of Science
NASA	National Aeronautics and Space Administration
NSIDC	National Snow and Ice Data Center (DAAC)
OO	Object Oriented
PDF	portable document format
PDPS	Planning and Data Processing Subsystem
RDBMS	Relational Database Management System
SDPS	Science Data Processing Segment (ECS)
SDSRV	Science Data Server CSCI
SMC	System Management Center (ECS)
SQL	structured query language
sql	structured query language file designator
STD	Software Test Document
STMGT	Storage Management CSCI
SUBSRV	Subscription Server
TBS	to be supplied
TD	technical document
TP	technical paper
UR	Universal Reference
URL	Universal Resource Locator
VATC	Verification and Acceptance Testing Center
WWW	World-Wide Web