311-CD-101-003

EOSDIS Core System Project

Data Distribution Server Database Design and Schema Specifications for the ECS Project

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

Draft

July 1998

Raytheon Systems Company
Upper Marlboro, Maryland

Data Distribution Server Database Design and Schema Specifications for the ECS Project

Draft

July 1998

Prepared Under Contract NAS5-60000
CDRL Item #050

RESPONSIBLE ENGINEER

Mary Armstrong /s/	7/31/98
Maureen Muganda EOSDIS Core System Project	Date

SUBMITTED BY

Paul Palmer /s/	7/31/98
Terry Fisher, EDF-Release 2 CCB Chairman EOSDIS Core System Project	Date

Raytheon Systems Company
Upper Marlboro, Maryland

This page intentionally left blank.

Preface

This document describes the data design and database specification for the Subscription Server subsystem. It is one of ten documents comprising the detailed database design specifications for each of the ECS subsystems.

The subsystem database design specifications for the as delivered system include:

311-CD-101 Data Distribution (DDIST) Subsystem Database Design and Database Schema Specifications for the ECS Project

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

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

311-CD-109 Subscription Server (DDIST) 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, and 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 tools and some cases contain too much detail to be easily readable within hard copy page constraints. The reader is encouraged to view these drawings on-line using the Portable Document Format (PDF) electronic copy available via the ECS Data Handling System (ECS) on the world-wide web at 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 "as-built" database design and database schema of the Subscription Server database including the physical layout of the database and initial installation parameters.

Keywords: data, database, design, configuration, database installation, scripts, security, data model, data dictionary, replication, performance tuning, SQL server, database security, replication, database scripts

This page intentionally left blank.

Change Information Page

List of Effective Pages	
Page Number	**Issue**
Title	Draft
iii through xii	Draft
1-1 and 1-2	Draft
2-1 and 2-2	Draft
3-1 and 3-2	Draft
4-1 though 4-64	Draft
5-1 and 5-2	Draft
6-1 and 6-2	Draft
7-1 and 7-2	Draft
8-1 and 8-2	Draft
AB-1 through AB-8	Draft

Document History			
Document Number	**Status/Issue**	**Publication Date**	**CCR Number**
311-CD-101-001	Draft	January 1998	97-1755
311-CD-101-002	Draft	May 1998	98-0620
311-CD-101-003	Draft	July 1998	98-0866

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. Data Design

4.1 Database Overview 4-1

4.1.1 Physical Data Model Entity Relationship Diagram 4-1

4.1.2 Tables 4-5

4.1.3 Columns 4-8

4.1.4 Column Domains 4-15

4.1.5 Rules 4-19

4.1.6 Defaults 4-19

4.1.7 Views....4-19
4.1.8 Integrity Constraints....4-19
4.1.9 Triggers....4-20
4.1.10 Stored Procedures....4-21
4.2 File Usage....4-64
4.2.1 Files Definitions....4-64
4.2.2 Attributes....4-64
4.2.3 Attribute Domains....4-64

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 Initial Users....6-1
6.2 Groups....6-1
6.3 Roles....6-1
6.4 Object Permissions....6-2

7. Replication

7.1 Replication is not implemented for the DDIST database....7-1

8. Scripts

8.1 Installation Scripts....8-1
8.2 De-Installation Scripts....8-1
8.3 Backup/Recovery Scripts....8-1
8.4 Miscellaneous Scripts....8-1

Figures

4-1 ERD Key 4-1
4-2 DDIST ERD 4-3

Abbreviations and Acronyms

This page is intentionally left blank.

1. Introduction

1.1 Identification

This Data Distribution Server (DDIST) Database Design and Database Schema Specification 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 DDIST Database Design and Database Schema Specification document describes the data design and database specifications to support the data requirements of Release 2 Drop 4PL DDIST software.

1.3 Purpose

The purpose of the DDIST Database Design and Database Schema Specification document is to support the maintenance of DDIST data and databases throughout the life cycle of ECS. This document communicates the database implementation in sufficient detail to support ongoing configuration management.

1.4 Audience

This document is intended to be used by ECS maintenance and operations staff. The document is organized as follows:

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

Section 2 provides a listing of the related documents, which were used as a source of information for this document.

Section 3 provides a mapping data bases to hardware components.

Section 4 contains the DDIST physical data model which is the database tables, triggers, stored procedures, and flat files.

Section 5 provides a description of database performance and tuning features such as indexes, caches, and data segments.

Section 6 provides a description of the security infrastructure used and list of the users, groups, and permissions available upon initial installation.

Section 7 Replication is not implemented for the DDIST database.

Section 8 provides a description of database and database related scripts used for installation, de-installation, backup/recovery, and other miscellaneous functions.

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	V 2.0 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-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 maye 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	Version 2.0 CSMS/SDPS Internal ICD for the ECS Project
609-CD-003	Version 2.0 Operations Tools Manual for the ECS Project
611-CD-004	Version 2.0 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

This page intentionally left blank.

4. Data Design

4.1 Database Overview

The DDIST database implements the large majority of the persistent data requirements for the DDIST subsystem. The database is designed in such a manner as to satisfy business policy while maintaining data integrity and consistency. Database tables are implemented using the Sybase Relational Database Management system (DBMS) version 11.0.2. All components of the DDIST database are described in the section which follow in sufficient detail to support maintenance needs.

4.1.1 Physical Data Model Entity Relationship Diagram

The Entity Relationship Diagram(ERD) presents a schematic depiction of the DDIST physical data model. The ERDs presented here for the DDIST database were produced using the S-Designor Data Architect Computer Aided Software Engineering (CASE) tool. ERDs represent the relationship between entities or database tables. The key for the symbols used in the ERDs follows.

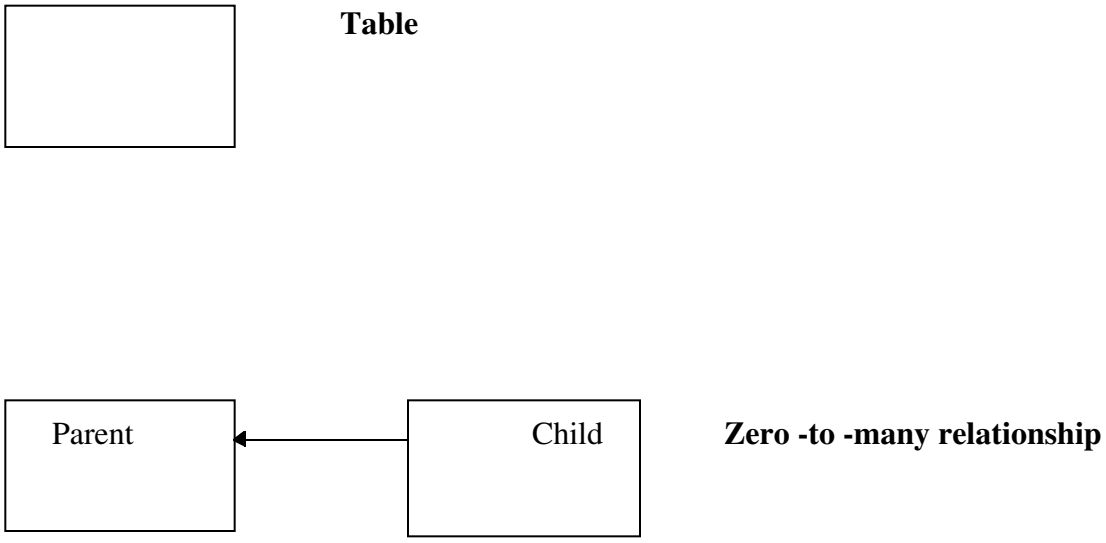

Figure 4-1. ERD Key

The ERDs for the DDIST database are shown in Figures 4-1 and 4-2.

This page intentionally left blank.

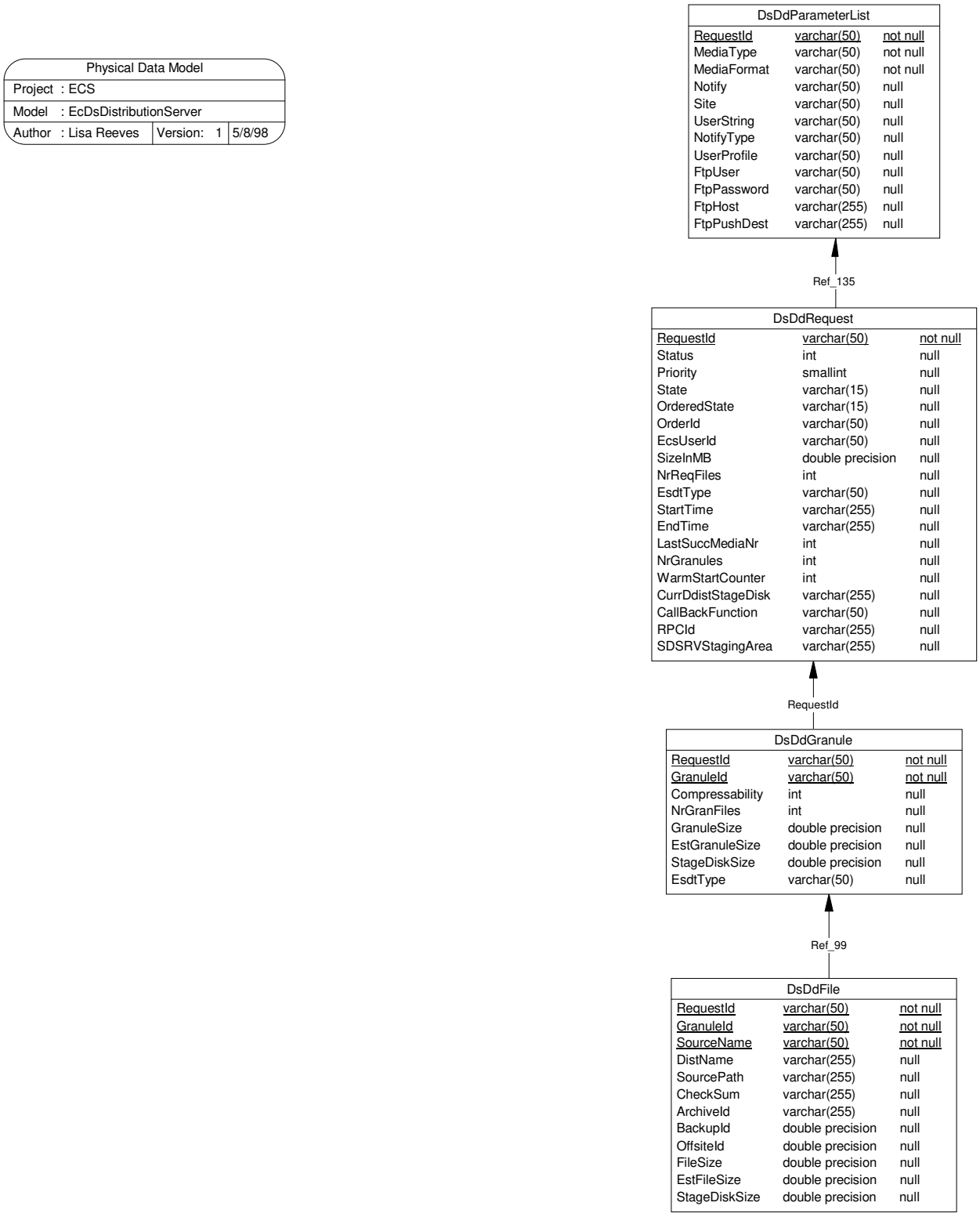

Figure 4-2. DDIST ERD

4.1.2 Tables

A listing of each the tables in the DDIST database is given here. A brief definition of each of these tables follows.

Table Name
DsDdFile
DsDdGranule
DsDdParameterList
DsDdRequest

Table: DsDdFile

Description

This table holds the distribution files currently being maintained and processed by the EcDsDistributionServer.

Table abbreviation is "F" to be used as standard naming convention for stored procedures.

Column List

Name	Code	Type	P	M
ArchiveId	ARCHIVEID	varchar(255)	No	No
BackupId	BACKUPID	double precision	No	No
CheckSum	CHECKSUM	varchar(255)	No	No
DistName	DISTNAME	varchar(255)	No	No
EstFileSize	ESTFILESIZE	double precision	No	No
FileSize	FILESIZE	double precision	No	No
GranuleId	GranuleId	varchar(50)	Yes	Yes
OffsiteId	OFFSITEID	double precision	No	No
RequestId	REQUESTID	varchar(50)	Yes	Yes
SourceName	SourceName	varchar(50)	Yes	Yes
SourcePath	SOURCEPATH	varchar(255)	No	No
StageDiskSize	STAGEDISKSIZE	double precision	No	No

Table: DsDdGranule

Description

This tables holds the distribution granules currently being maintained and processed by the EcDsDistributionServer.

The table abbreviation is "G" to be used as standard naming convention for stored procedures.

Column List

Name	Code	Type	P	M
Compressability	Compressability	int	No	No
EsdtType	EsdtType	varchar(50)	No	No
EstGranuleSize	EstGranuleSize	double precision	No	No
GranuleId	GranuleId	varchar(50)	Yes	Yes
GranuleSize	GranuleSize	double precision	No	No
NrGranFiles	NrGranFiles	int	No	No
RequestId	REQUESTID	varchar(50)	Yes	Yes
StageDiskSize	StageDiskSize	double precision	No	No

Table: DsDdParameterList

Description

Table holds the GLParameter list for each request currently being maintained and processed by the EcDsDistributionServer.

This data is provided from external metadata (MCF) by SDSRV.

Request information is initiated here first.

The table's abbreviation is "PL" to be used as standard naming convention for stored procedures.

Column List

Name	Code	Type	P	M
FtpHost	FtpHost	varchar(255)	No	No
FtpPassword	FtpPassword	varchar(50)	No	No
FtpPushDest	FtpPushDest	varchar(255)	No	No
FtpUser	FtpUser	varchar(50)	No	No
MediaFormat	MediaFormat	varchar(50)	No	Yes
MediaType	MediaType	varchar(50)	No	Yes
Notify	Notify	varchar(50)	No	No
NotifyType	NotifyType	varchar(50)	No	No
RequestId	RequestId	varchar(50)	Yes	Yes
Site	Site	varchar(50)	No	No
UserProfile	UserProfile	varchar(50)	No	No
UserString	UserString	varchar(50)	No	No

Table: DsDdPriorityThread

Description

Table holds the threshold for the number of threads which can be active for each request.

The table's abbreviation is "PL" to be used as standard naming convention for stored procedures.

Column List

Name	Code	Type	P	M
ThreadName	ThreadName	varchar(12)	Yes	Yes
ThreadLimit	ThreadLimit	int	No	No

Table: DsDdRequest

Description

This table holds the distribution requests currently being maintained and processed by the EcDsSistributionServer.

This table's abbreviation is "R" to be used as std naming convention for stored procedures.

Column List

Name	Code	Type	P	M
CallBackFunction	CallBackFunction	varchar(50)	No	No
CurrDdistStageDisk	CurrDdistStageDisk	varchar(255)	No	No
EcsUserId	EcsUserId	varchar(50)	No	No
EndTime	EndTime	varchar(255)	No	No
EsdtType	EsdtType	varchar(50)	No	No
LastSuccMediaNr	LastSuccMediaNr	int	No	No
NrGranules	NrGranules	int	No	No
NrReqFiles	NrReqFiles	int	No	No
OrderedState	OrderedState	varchar(15)	No	No
OrderId	OrderId	varchar(50)	No	No
Priority	Priority	smallint	No	No
RequestId	REQUESTID	varchar(50)	Yes	Yes
RPCId	RPCId	varchar(255)	No	No
SDSRVStagingArea	SDSRVStagingArea	varchar(255)	No	No
SizeInMB	SizeInMB	double precision	No	No
StartTime	StartTime	varchar(255)	No	No
State	State	varchar(15)	No	No
Status	Status	int	No	No
WarmStartCounter	WarmStartCounter	int	No	No

Table: DsDdVersion

Description

Table holds information concerning the installation of the database.

The table's abbreviation is "PL" to be used as standard naming convention for stored procedures.

Column List

Name	Code	Type	P	M
VersionId	VersionId	int	Yes	Yes
CurrentVersion	CurrentVersion	char(1)	No	No

Name	Code	Type	P	M
DdistDBName	DdistDBName	varchar(255)	No	No
DdistDBVersion	DdistDBVersion	varchar(255)	No	No
DBInstallDate	DBInstallDate	datetime	No	No
DdistSRVRVersion	DdistSRVRVersion	varchar(255)	No	No
SRVRVersionInstalled	SRVRVersionInstalled	varchar(255)	No	No

4.1.3 Columns

Brief definitions of each of the columns present in the database tables defined above are contained herein.

Column: ARCHIVEID

Description

The Achive Id from science data server (SDSRV).

Column: BACKUPID

Column: CallBackFunction

Description

Callback fuction that should get called when distribution request are completed.

Column: CHECKSUM

Description

The Archive checksum.

Column: Compressability

Description

The compressability of the granule.

Column: CurrDdistStageDisk

Description

Current distribution staging disk.

Column: CurrentVersion

Description

This attribute holds y (yes) or n (no) if the current version is installed on the database.

Column: DBInstallDate

Description

This attribute holds the date the database was installed.

Column: DdistDBName

Description

This attribute holds the name of the database.

Column: DdistDBVersion

Description

This attribute holds the current version installed.

Column: DdistSRVRVersion

Description

This attribute holds the current version required for the server.

Column: DISTNAME

Description

The Distribution file name.

Column: EcsUserId

Description

The User ID of the user initiating request.

Column: EndTime

Description

The time that the distribution ended.

Column: EsdtType

Description

The ESDT (Earth Science Data Type) Type. A Request is of one EsdtType and can include many Granules, but all Granules associated with a Request must be of the same EsdtType.

Column: EsdtType

Description

The ESDT (Earth Science Data Type) Type. A Request is of one EsdtType and can include many Granules, but all Granules associated with a Request must be of the same EsdtType.

Column: ESTFILESIZE

Description

The estimated size of the file.

Column: EstGranuleSize

Description

The sum of the estimated size of the files in the granule.

Column: FILESIZE

Description

The size of the file.

Column: FtpHost

Description

Holds the hostname to connect to for FTP push.

Column: FtpPassword

Description

Holds the password to use for FTP push.

Column: FtpPushDest

Description

Holds the target system directory.

Column: FtpUser

Description

Holds the login to use for the FTP push.

Column: GranuleId

Description

The granule Id of the actual granule.

Column: GranuleId

Description

The Granule Id of the granule.

Column: GranuleSize

Description

The sum of the sizes of files in the granule.

Column: LastSuccMediaNr

Description

The last successful media number.

Column: MediaBlocksize

Description

The media blocksize format..

Column: MediaCapacity

Description

The capacity for media.

Column: MediaFormat

Description

The media distribution format. (i.e. FILEFORMAT, TARFORMAT...)

Column: MediaType

Description

The type of media used for a request. (i.e.: 8MM TAPE, CDRom)

Column: Notify

Description

Indicates the mail address to use for notification.

Column: NotifyType

Description

If MAIL is specified, a Distribution Notification message will be sent, either to email address of logical queue specified in the NOTIFY parameter. If LIST is specified, a Glparameter list of distributive files will be returned to the RPC caller.

Column: NrGranFiles

Description

The number of files in a granule.

Column: NrGranules

Description

The number of granules per media object.

Column: NrReqFiles

Description

The number of files in the distribution request.

Column: OFFSITEID

Description

TBS

Column: OrderedState

Description

The ordered state of a request. (i.e. Cancel, Suspend, Marked Shipped...)

Column: OrderId

Description

The OrderId for the distribution request.

Column: Priority

Description

The priority for the DistRequest object.

Column: REQUESTID

Description

The RequestID of the distribution request that come from science data server (SDSRV).

Column: REQUESTID

Description

The Request Id of the distribution request form science data server (SDSRV).

Column: REQUESTID

Description

The RequestId of the distribution request from science data server (SDSRV).

Column: RequestId

Description

The Request ID for the distribution request.

Column: RPCId

Description

For rebinding/restarting. The format is a combination of a RequestId from a system level and a TransactionId which is fixed whenever the system restarts.

Column: SDSRVStagingArea

Description

TBS

Column: SRVRVersionInstalled

Description

This attribute holds the current version installed on the server.

Column: Site

Description

Site to be notified.

Column: SizeInMB

Description

The total size of bytes in the distribution request.

Column: SourceName

Description

Input file name from science data server (SDSRV).

Column: SOURCEPATH

Description

The Staging file path.

Column: StageDiskSize

Description

The staging disk size.

Column: STAGEDISKSIZE

Description

The size of the staging disk required to hold the file.

Column: StartTime

Description

The start time of the distribution request.

Column: State

Description

The queue state of the distribution request. (i.e.: Pending, Active, Shipped,...)

Column: Status

Description

The status for the DistRequest object.

Column: ThreadLimit

Description

Holds the limit for threads.

Column: ThreadName

Description

Holds the priority level for the thread.

Column: UserProfile

Description

Holds the profile ID.

Column: UserString

Description

Free text string supplied by the user. Returned in the Distribution Email message as "UserString: <supplied string>"

Column: VersionId

Description

Thisattribute holds a number generated by the database.

Column: WarmStartCounter

Description

The warm start counter.

4.1.4 Column Domains

Domains specify the ranges of values allowed for a given table column. 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. The following domains are defined in the DDIST database (all DDIST domains are user-defined datatypes)

Domain: archiveid

Reference List

Table Code	Column Name	Column Code	Label
DsDdFile	ArchiveId	ArchiveId	

Domain: callback

Reference List

Table Code	Column Name	Column Code	Label
DsDdRequest	CallBackFunction	CallBackFunction	

Domain: checksum

Reference List

Table Code	Column Name	Column Code	Label
DsDdFile	CheckSum	CheckSum	

Domain: esdttype

Reference List

Table Code	Column Name	Column Code	Label
DsDdGranule	EsdtType	EsdtType	
DsDdRequest	EsdtType	EsdtType	

Domain: file

Reference List

Table Code	Column Name	Column Code	Label
DsDdFile	SourceName	SourceName	

Domain: granuleid

Reference List

Table Code	Column Name	Column Code	Label
DsDdFile	GranuleId	GranuleId	
DsDdGranule	GranuleId	GranuleId	

Domain: id

Reference List

Table Code	Column Name	Column Code	Label
DsDdVersion	VersionId	VersionId	

Domain: itemcount

Reference List

Table Code	Column Name	Column Code	Label
DsDdGranule	NrGranFiles	NrGranFiles	
DsDdRequest	NrGranules	NrGranules	
DsDdRequest	NrReqFiles	NrReqFiles	
DsDdRequest	WarmStartCounter	WarmStartCounter	

Domain: mediatype

Reference List

Table Code	Column Name	Column Code	Label
DsDdParameterList	MediaType	MediaType	

Domain: name

Reference List

Table Code	Column Name	Column Code	Label
DsDdVersion	DdistDBName	DdistDBName	

Domain: orderid

Reference List

Table Code	Column Name	Column Code	Label
DsDdRequest	OrderId	OrderId	

Domain: path

Reference List

Table Code	Column Name	Column Code	Label
DsDdFile	SourcePath	SourcePath	

Domain: priority

Reference List

Table Code	Column Name	Column Code	Label
DsDdRequest	Priority	Priority	

Domain: reqtime

Reference List

Table Code	Column Name	Column Code	Label
DsDdRequest	EndTime	EndTime	
DsDdRequest	StartTime	StartTime	

Domain: requestid

Reference List

Table Code	Column Name	Column Code	Label
DsDdFile	RequestId	RequestId	
DsDdGranule	RequestId	RequestId	
DsDdParameterList	RequestId	RequestId	
DsDdRequest	RequestId	RequestId	

Domain: size

Reference List

Table Code	Column Name	Column Code	Label
DsDdFile	EstFileSize	EstFileSize	
DsDdGranule	EstGranuleSize	EstGranuleSize	
DsDdFile	FileSize	FileSize	
DsDdGranule	GranuleSize	GranuleSize	
DsDdRequest	SizeInMB	SizeInMB	
DsDdFile	StageDiskSize	StageDiskSize	
DsDdGranule	StageDiskSize	StageDiskSize	

Domain: state

Reference List

Table Code	Column Name	Column Code	Label
DsDdRequest	OrderedState	OrderedState	
DsDdRequest	State	State	

Domain: status

Reference List

Table Code	Column Name	Column Code	Label
DsDdRequest	Status	Status	

Domain: thread

Reference List

Table Code	Column Name	Column Code	Label
DsDdRequest	Status	Status	

Domain: username

Reference List

Table Code	Column Name	Column Code	Label
DsDdParameterList	FtpUser	FtpUser	

Domain: version

Reference List

Table Code	Column Name	Column Code	Label
DdistDBVersion	DdistDBVersion	DdistDBVersion	
DdistDBVersion	DditSRVRVersion	DditSRVRVersion	
DdistDBVersion	SRVRVersionInstalled	SRVRVersionInstalled	

4.1.5 Rules

Sybase supports the definitions of rules. Rules provide a means for enforcing domain constraints on a given column. There are no rules defined in Sybase for the DDIST database.

4.1.6 Defaults

Defaults are used to supply a value for a column when one is not defined at insert time. There are no defaults defined in Sybase in the DDIST database.

4.1.7 Views

Sybase allows the definition of views as a means of limiting an application or users access to data in a table or tables. Views create a logical table from columns found in one or more tables. There are no views defined in the DDIST database.

4.1.8 Integrity Constraints

Sybase version 11.0.1 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 without automatically without requiring programming. Sybase 11 is only ANSI-92 compliant, however, therefore its constraints support "restrict-only" operations. This means that a row can not be deleted or updated if their are rows in other tables having a foreign key dependency on that row. Cascade delete and update operations can not be performed if a declarative constraint has been used. There DDIST declarative integrity constraints follow:

Dependencies on Table: DsDdGranule

Reference by List

Referenced by	Primary Key	Foreign Key
DsDdFile	RequestId GranuleId	RequestId GranuleId

Dependencies on Table: DsDdParameterList

Reference by List

Referenced by	Primary Key	Foreign Key
DsDdRequest	RequestId	RequestId

Dependencies on Table: DsDdRequest

Reference by List

Referenced by	Primary Key	Foreign Key
DsDdGranule	RequestId	RequestId

4.1.9 Triggers

Sybase supports the enforcement of business policy via the use of triggers. A trigger is best defined as set of activities or checks that should be performed automatically when ever a row is inserted, updated, or deleted from a given table. Sybase version 11.0.2 allows the definition of insert, update, and delete trigger per table. No triggers are currently defined in the DDIST database.

Table	Trigger	Description
DsDdRequest	DsDdRDeleteTrig	DsDdRequest table delete trigger

Trigger: DsDdRDeleteTrig

Trigger Code

```
-- ************************************************************
-- BEGIN PROLOG
--
-- TRIGGER NAME:    DsDdRDeleteTrig
--
-- DESCRIPTION:     DeleteTrigger on the DsDdFile table
--                  that removes all parent table references to
--                  the deleted record(s).
--
```

```
-- DATE CREATED:     1/7/98
-- CREATED BY:       Lisa Colombo
--
-- TABLES ACCESSED:  DsDdParameterList
--                   DsDdRequest
--
-- RETURNS:          Status (Success = 0)
--
-- ************************************************************
CREATE TRIGGER DsDdRDeleteTrig ON DsDdRequest
FOR DELETE AS

BEGIN

-- If no rows deleted exit trigger
  IF @@rowcount = 0
    RETURN

-- For deleted Child table, DsDdRequest, records, delete corresponding
-- Parent table records
  DELETE DsDdParameterList
  FROM DsDdParameterList pl, deleted d
  WHERE pl.RequestId = d.RequestId

  RETURN

END
go
```

4.1.10 Stored Procedures

Sybase also includes support for business policy via the use of stored procedures. Stored procedures are typically used to capture a set of activities or checks that will be performed on the database repeatedly to enforce business policy and maintain data integrity. 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 A listing of each the stored procedures in the DDIST database is given here. A brief definition of each of these stored procedures follows.

Procedure List

Name	Description
DsDdFDelete	Deletes request from the DsDdFile Table
DsDdFInsert	Insert a record into the DsDdFile table.
DsDdFSelectAll	Returns the complete row of information for all rows in the DsDdFile table.
DsDdFSelectAllFiles	Returns all files associated with a request id and granule id in the DsDdFile table.
DsDdFSelectByReqGranSrc	Returns information for a file associated with request id, granule id and source name in the DsDdFile table.
DsDdGDelete	Remove a record from the DsDdGranule Table
DsDdGInsert	Insert a record into the DsDdGranule table.
DsDdGSelectAll	Returns the complete row of information for all rows in the DsDdGranule table.
DsDdGSelectAllGranules	Returns all granules associated with a request in the DsDdGranule table.
DsDdGSelectByReqGran	Returns the granule information associated with request id and granule id in the DsDdGranule table.
DsDdGUpdate	Update a record into the DsDdGranule table.
DsDdPLDelete	Deletes request from the DsDdParameterList Table
DsDdPLInsert	Insert a record into the DsDdParameterList table.
DsDdPLSelectAll	Returns the complete row of information for all rows in the DsDdParameterList table.
DsDdPLSelectByReq	Gets information for a given RequestId
DsDdPLUpdate	Update a record into the DsDdParameterList table.
DsDdRDelete	Deletes request from the DsDdRequest Table
DsDdRInsert	Insert a record into the DsDdRequest table.
DsDdRSelectAll	Returns the complete row of information for all rows
DsDdRSelectByRequestId	Returns all information for a given RequestId.
DsDdRSelectByState	Returns the complete list of requests of a specific state in the DsDdRequest table.
DsDdRTBSelectAll	Returns the complete row of information for all rows in the DsDdRequest table.
DsDdRUpdate	Update a record into the DsDdRequest table.
DsDdRUpdateOrdState	Updates the ordered state for a given RequestId.
DsDdRUpdateState	Updates the state field in the DsDdRequest table.

Procedure: DsDdFDelete

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdFDelete
--
-- DESCRIPTION:         Deletes request from the DsDdFile Table
-- DATE CREATED:        12/03/97
-- CREATED BY:          Kenya Wright
--
-- TABLES ACCESSED:     DsDdFile
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS PARAMETER     DOMAIN
-- ---------------      ------
--    RequestId         requestid
--    GranuleId         granuleid
--    SourceName        file
--
-- OUTPUTS PARAMETER    DOMAIN
-- ----------------     ------
--    NONE
-- ************************************************************************
CREATE PROCEDURE DsDdFDelete
     @RequestId        requestid,
     @GranuleId        granuleid,
     @SourceName       file
AS
BEGIN
   DECLARE @status int

   DELETE DsDdFile
   WHERE RequestId = @RequestId
   AND GranuleId = @GranuleId
   AND SourceName = @SourceName

   SELECT @status = @@error

   IF (@status != 0)
      BEGIN
         ROLLBACK TRANSACTION
         SELECT @status = 1
         RAISERROR 20001 "DsDdFDelete: Unable to delete RequestId [%1!], GranuleId [%2!],
and SourceName [%3!].", @RequestId, @GranuleId, @SourceName
         RETURN (@status)
      END
```

END
go

Procedure: DsDdFInsert

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdFInsert
--
-- DESCRIPTION:         Insert a record into the DsDdFile table.
-- DATE CREATED: 12/03/97
-- CREATED BY:          Kenya Wright
--
-- TABLES ACCESSED:     DsDdFile
--
-- RETURNS:          Status (Success = 0)
--
-- INPUT PARAMETERS          DOMAIN
-- ----------------          ------
--   RequestId            requestid
--   GranuleId            granuleid
--   SourceName              file
--   DistName             varchar(255)
--   SourcePath           path
--   CheckSum               checksum
--   ArchiveId            archiveid
--   BackupId             varchar(255)
--   OffsiteId           varchar(255)
--   FileSize            size
--   EstFileSize           size
--   StageDiskSize           size
--
-- OUTPUT PARAMETER          DOMAIN
-- ----------------          ------
--   NONE
-- ************************************************************************
CREATE PROCEDURE DsDdFInsert
    @RequestId       requestid,
    @GranuleId       granuleid,
    @SourceName      file,
    @DistName        varchar(255),
    @SourcePath      path,
    @CheckSum        checksum,
    @ArchiveId       archiveid,
          @BackupId        varchar(255),
          @OffsiteId       varchar(255),
    @FileSize        size,
    @EstFileSize     size,
    @StageDiskSize   size
```

```
AS
BEGIN
  DECLARE @mystatus int

  SELECT @mystatus = 0

-- Error Checking to see if RequestId, GranuleId, and/or
-- SourceName is blank or null
  IF (@RequestId IS NULL OR @RequestId = ""
     OR @GranuleId IS NULL OR @GranuleId = ""
     OR @SourceName IS NULL OR @SourceName = "")
  BEGIN
    SELECT @mystatus = 1
    raiserror 20002 "DsDdFInsert:  RequestId, GranuleId, or SourceName may not be blank or
null."
    RETURN(@mystatus)
  END

-- Error Checking to see if RequestId, GranuleId and SourceName already exists
  IF EXISTS (SELECT 1 FROM DsDdFile WHERE RequestId = @RequestId
           AND GranuleId = @GranuleId
           AND SourceName = @SourceName)
    BEGIN
      SELECT @mystatus = 1
      raiserror 20003 "DsDdFInsert:  RequestId [%1!], GranuleId [%2!] and SourceName [%3!]
already exists.", @RequestId, @GranuleId, @SourceName
      RETURN(@mystatus)
    END

  BEGIN TRANSACTION
  INSERT DsDdFile (
      RequestId,
      GranuleId,
      SourceName,
      DistName,
      SourcePath,
      CheckSum,
      ArchiveId,
      BackupId,
      OffsiteId,
      FileSize,
      EstFileSize,
      StageDiskSize
     )

  VALUES (
      @RequestId,
      @GranuleId,
      @SourceName,
      @DistName,
      @SourcePath,
      @CheckSum,
```

```
        @ArchiveId,
        @BackupId,
        @OffsiteId,
        @FileSize,
        @EstFileSize,
        @StageDiskSize
           )

   SELECT @mystatus = @@error

   IF (@mystatus != 0)
      BEGIN
        SELECT @mystatus = 1
        ROLLBACK TRANSACTION
        raiserror 20004 "DsDdFInsert:  Unable to insert RequestId [%1!], GranuleId [%2!], and
SourceName [%3!].", @RequestId, @GranuleId, @SourceName
      END

   ELSE

      BEGIN
        COMMIT TRANSACTION
        RETURN(@mystatus)
      END
END
go
```

Procedure: DsDdFSelectAll

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdFSelectAll
--
-- DESCRIPTION:         Returns the complete row of information for all rows
--                      in the DsDdFile table.
-- DATE CREATED:        12/03/97
-- DEVELOPER:           Kenya Wright
--
-- TABLES ACCESSED:     DsDdFile
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS PARAMETER         DOMAIN
-- ----------------         ------
--   NONE
--
-- OUTPUTS PARAMETER        DOMAIN
-- ----------------         ------
--   RequestId              requestid
```

```
--   GranuleId            granuleid
--   SourceName             file
--   DistName             varchar(255)
--   SourcePath           path
--   CheckSum              checksum
--   ArchiveId            archiveid
--   BackupId             varchar(255)
--   OffsiteId           varchar(255)
--   FileSize            size
--   EstFileSize          size
--   StageDiskSize          size
--
-- ************************************************************************
CREATE PROCEDURE DsDdFSelectAll
AS
BEGIN
  DECLARE @status     int

  SELECT @status = 0

  BEGIN
    SELECT RequestId,
          GranuleId,
          SourceName,
          DistName,
          SourcePath,
          CheckSum,
          ArchiveId,
          BackupId,
          OffsiteId,
          FileSize,
          EstFileSize,
          StageDiskSize
    FROM   DsDdFile

    SELECT @status = @@error

    IF (@status != 0)
      BEGIN
        SELECT @status = 1
        RAISERROR 20005 "DsDdFSelectAll:  Unable to select information of Files."
      END

   END
   RETURN(@status)
END
go
```

Procedure: DsDdFSelectAllFiles

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:       DsDdFSelectAllFiles
--
-- DESCRIPTION:                Returns all files associated with a request id
--                   and granule id in the DsDdFile table.
-- DATE CREATED:        12/10/97
-- DEVELOPER:           Kenya Wright
--
-- TABLES ACCESSED:     DsDdFile
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS PARAMETER         DOMAIN
-- ----------------         ------
--   RequestId              requestid
--   GranuleId              granuleid
--
-- OUTPUTS PARAMETER        DOMAIN
-- ----------------         ------
--   RequestId              requestid
--   GranuleId              granuleid
--   SourceName               file
--   DistName               varchar(255)
--   SourcePath             path
--   CheckSum                 checksum
--   ArchiveId              archiveid
--   BackupId               varchar(255)
--   OffsiteId              varchar(255)
--   FileSize               size
--   EstFileSize            size
--   StageDiskSize            size
--
-- ************************************************************************
CREATE PROCEDURE DsDdFSelectAllFiles
    @RequestId       requestid,
    @GranuleId       granuleid
AS
BEGIN
  DECLARE @status     int

  SELECT @status = 0

  IF NOT EXISTS (SELECT 1 FROM DsDdFile
               WHERE RequestId = @RequestId
               AND GranuleId = @GranuleId )
     BEGIN
       SELECT @status = 1
       raiserror 20006 "DsDdRequest:  The RequestId [%1!] and GranuleId [%2!] does not
exist.", @RequestId, @GranuleId
       RETURN (@status)
```

```
    END

  BEGIN
    SELECT RequestId,
          GranuleId,
          SourceName,
          DistName,
          SourcePath,
          CheckSum,
          ArchiveId,
          BackupId,
          OffsiteId,
          FileSize,
          EstFileSize,
          StageDiskSize
    FROM DsDdFile
    WHERE RequestId = @RequestId AND GranuleId = @GranuleId

    SELECT @status = @@error

    IF (@status != 0)
      BEGIN
        SELECT @status = 1
        RAISERROR 20007 "DsDdFSelectAllFiles:  Unable to select information of files."
      END

   END
   RETURN(@status)
END
go
```

Procedure: DsDdFSelectByReqGranSrc

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdFSelectByReqGranSrc
--
-- DESCRIPTION:               Returns information for a file associated with
--                  request id, granule id and source name
--                  in the DsDdFile table.
-- DATE CREATED:       12/12/97
-- DEVELOPER:          Kenya Wright
--
-- TABLES ACCESSED:    DsDdFile
--
-- RETURNS:            Status (Success = 0)
--
-- INPUTS PARAMETER        DOMAIN
-- ----------------        ------
```

```
--   RequestId            requestid
--   GranuleId            granuleid
--   SourceName             file
--
-- OUTPUTS PARAMETER       DOMAIN
-- ----------------        ------
--   RequestId            requestid
--   GranuleId            granuleid
--   SourceName             file
--   DistName              varchar(255)
--   SourcePath            path
--   CheckSum               checksum
--   ArchiveId            archiveid
--   BackupId             varchar(255)
--   OffsiteId           varchar(255)
--   FileSize            size
--   EstFileSize           size
--   StageDiskSize          size
--
-- ************************************************************************
CREATE PROCEDURE DsDdFSelectByReqGranSrc
    @RequestId       requestid,
    @GranuleId       granuleid,
    @SourceName       file
AS
BEGIN
  DECLARE @status     int

  SELECT @status = 0

  BEGIN
    SELECT RequestId,
         GranuleId,
         SourceName,
         DistName,
         SourcePath,
         CheckSum,
         ArchiveId,
         BackupId,
         OffsiteId,
         FileSize,
         EstFileSize,
         StageDiskSize
    FROM DsDdFile
    WHERE RequestId = @RequestId
    AND GranuleId = @GranuleId
    AND SourceName = @SourceName

    SELECT @status = @@error

    IF (@status != 0)
      BEGIN
```

```
        SELECT @status = 1
        RAISERROR 20008 "DsDdFSelectByReqGranSrc:  Unable to select information about
RequestId [%1!], GranuleId [%2!], and SourceName [%3!].", @RequestId, @GranuleId,
@SourceName
      END

  END
  RETURN(@status)
END
go
```

Procedure: DsDdGDelete

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdGDelete
--
-- DESCRIPTION:         Remove a record from the DsDdGranule Table
-- DATE CREATED: 12/03/97
-- CREATED BY:                Kenya Wright
--
-- TABLES ACCESSED:     DsDdGranule
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS PARAMETER     DOMAIN
-- ---------------      ------
--    RequestId         requestid
--    GranuleId         granuleid
--
-- OUTPUTS PARAMETER    DOMAIN
-- ----------------     ------
--    NONE
-- ************************************************************************
CREATE PROCEDURE DsDdGDelete
    @RequestId        requestid,
    @GranuleId        granuleid
AS
BEGIN
  DECLARE @status int

-- First delete corresponding records in Child table with foreign key constraints
  DELETE DsDdFile
  WHERE RequestId = @RequestId
  AND GranuleId = @GranuleId

-- Delete Parent table DsDdGranule record once foreign key relationships dropped
  DELETE DsDdGranule
  WHERE RequestId = @RequestId
```

```
   AND GranuleId = @GranuleId

   SELECT @status = @@error

   IF (@status != 0)
      BEGIN
         ROLLBACK TRANSACTION
         SELECT @status = 1
         RAISERROR 20011 "DsDdGDelete: Unable to delete RequestId [%1!] and GranuleId
[%2!].", @RequestId, @GranuleId
         RETURN (@status)
      END

END
go
```

Procedure: DsDdGInsert

Code

```
-- ****************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdGInsert
--
-- DESCRIPTION:         Insert a record into the DsDdGranule table.
-- DATE CREATED: 12/03/97
-- CREATED BY:          Kenya Wright
--
-- TABLES ACCESSED:     DsDdGranule
--
-- RETURNS:          Status (Success = 0)
--
-- INPUT PARAMETERS            DOMAIN
-- ----------------            ------
--   RequestId             requestid
--   GranuleId             granuleid
--   Compressability         int
--   NrGranFiles           itemcount
--   GranuleSize           size
--   EstGranuleSize          size
--   StageDiskSize           size
--   EsdtType              esdttype
--
-- OUTPUT PARAMETER            DOMAIN
-- ----------------            ------
--   NONE
-- ****************************************************************
CREATE PROCEDURE DsDdGInsert
    @RequestId       requestid,
    @GranuleId       granuleid,
    @Compressability  int,
```

```
    @NrGranFiles      itemcount,
    @GranuleSize      size,
    @EstGranuleSize   size,
    @StageDiskSize    size,
    @EsdtType         esdttype
AS
BEGIN
  DECLARE @mystatus int

  SELECT @mystatus = 0

-- Error Checking to see if RequestId and GranuleId is blank or null
  IF (@RequestId IS NULL OR @RequestId = ""
     OR @GranuleId IS NULL OR @GranuleId = "")
    BEGIN
      SELECT @mystatus = 1
      raiserror 20012 "DsDdGInsert:  RequestId and GranuleId may not be blank or null."
     RETURN(@mystatus)
    END

-- Error Checking to see if RequestId and GranuleId already exists
  IF EXISTS (SELECT 1 FROM DsDdGranule
           WHERE RequestId = @RequestId
           AND GranuleId = @GranuleId)
    BEGIN
      SELECT @mystatus = 1
      raiserror 20013 "DsDdGInsert:  RequestId [%1!] and GranuleId [%2!] already exists.",
@RequestId, @GranuleId
      RETURN(@mystatus)
    END

  BEGIN TRANSACTION
    INSERT DsDdGranule (
          RequestId,
          GranuleId,
          Compressability,
          NrGranFiles,
          GranuleSize,
          EstGranuleSize,
          StageDiskSize,
          EsdtType
         )

    VALUES (
          @RequestId,
          @GranuleId,
          @Compressability,
          @NrGranFiles,
          @GranuleSize,
          @EstGranuleSize,
          @StageDiskSize,
          @EsdtType
```

```
        )

    SELECT @mystatus = @@error

    IF (@mystatus != 0)
      BEGIN
        SELECT @mystatus = 1
        ROLLBACK TRANSACTION
        raiserror 20014 "DsDdGInsert:  Unable to insert RequestId [%1!] and GranuleId [%1!].",
@RequestId, @GranuleId
      END

    ELSE

      BEGIN
        COMMIT TRANSACTION
        RETURN(@mystatus)
      END

END
go
```

Procedure: DsDdGSelectAll

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdGSelectAll
--
-- DESCRIPTION:              Returns the complete row of information for all rows
--                  in the DsDdGranule table.
-- DATE CREATED:       12/03/97
-- CREATED BY:         Kenya Wright
--
-- TABLES ACCESSED:    DsDdGranule
--
-- RETURNS:            Status (Success = 0)
--
-- INPUTS PARAMETER        DOMAIN
-- ----------------        ------
--   NONE
--
-- OUTPUTS PARAMETER       DOMAIN
-- ----------------        ------
--   RequestId             requestid
--   GranuleId             granuleid
--   Compressability       int
--   NrGranFiles           itemcount
--   GranuleSize           size
--   EstGranuleSize        size
```

```
--   StageDiskSize          size
--   EsdtType               esdttype
--
-- ************************************************************************
CREATE PROCEDURE DsDdGSelectAll
AS
BEGIN
  DECLARE @status     int

  SELECT @status = 0

  BEGIN
    SELECT RequestId,
          GranuleId,
          Compressability,
          NrGranFiles,
          GranuleSize,
          EstGranuleSize,
          StageDiskSize,
          EsdtType
    FROM DsDdGranule

    SELECT @status = @@error

    IF (@status != 0)
      BEGIN
        SELECT @status = 1
        RAISERROR 20017 "DsDdGSelectAll:  Unable to select information about Granules."
      END

 END
 RETURN(@status)

END
go
```

Procedure: DsDdGSelectAllGranules

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:       DsDdGSelectAllGranules
--
-- DESCRIPTION:                Returns all granules associated with
--                a request in the DsDdGranule table.
-- DATE CREATED:        12/10/97
-- CREATED BY:          Kenya Wright
--
-- TABLES ACCESSED:     DsDdGranule
--
```

```
-- RETURNS:            Status (Success = 0)
--
-- INPUTS PARAMETER        DOMAIN
-- ----------------        ------
--   RequestId             requestid
--
-- OUTPUTS PARAMETER       DOMAIN
-- ----------------        ------
--   RequestId             requestid
--   GranuleId             granuleid
--   Compressability         int
--   NrGranFiles           itemcount
--   GranuleSize           size
--   EstGranuleSize          size
--   StageDiskSize         size
--   EsdtType              esdttype
--
-- ************************************************************************
CREATE PROCEDURE DsDdGSelectAllGranules
    @RequestId          requestid
AS
BEGIN
  DECLARE @status     int

  SELECT @status = 0

  IF NOT EXISTS (SELECT 1 FROM DsDdRequest WHERE
           RequestId = @RequestId )
    BEGIN
      SELECT @status = 1
      raiserror 20015 "DsDdRequest :  No Granules exist for RequestId [%1!].", @RequestId
      RETURN (@status)
    END

  BEGIN
    SELECT RequestId,
         GranuleId,
         Compressability,
         NrGranFiles,
         GranuleSize,
         EstGranuleSize,
         StageDiskSize,
         EsdtType
    FROM DsDdGranule
    WHERE RequestId = @RequestId

    SELECT @status = @@error

    IF (@status != 0)
      BEGIN
        SELECT @status = 1
        RAISERROR 20016 "DsDdGSelectAllGranules:  Unable to select information of
```

```
Granules for RequestId [%1!].", @RequestId
      END

    END
    RETURN(@status)
END
go
```

Procedure: DsDdGSelectByReqGran

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdGSelectByReqGran
--
-- DESCRIPTION:              Returns the granule information associated with
--                  request id and granule id in the DsDdGranule table.
-- DATE CREATED:       12/12/97
-- CREATED BY:         Kenya Wright
--
-- TABLES ACCESSED:    DsDdGranule
--
-- RETURNS:            Status (Success = 0)
--
-- INPUTS PARAMETER        DOMAIN
-- ----------------        ------
--   RequestId             requestid
--   GranuleId             granuleid
--
-- OUTPUTS PARAMETER       DOMAIN
-- ----------------        ------
--   RequestId             requestid
--   GranuleId             granuleid
--   Compressability       int
--   NrGranFiles           itemcount
--   GranuleSize           size
--   EstGranuleSize        size
--   StageDiskSize         size
--   EsdtType              esdttype
--
-- ************************************************************************
CREATE PROCEDURE DsDdGSelectByReqGran
    @RequestId        requestid,
    @GranuleId        granuleid
AS
BEGIN
  DECLARE @status     int

  SELECT @status = 0
```

```
  BEGIN
    SELECT RequestId,
          GranuleId,
          Compressability,
          NrGranFiles,
          GranuleSize,
          EstGranuleSize,
          StageDiskSize,
          EsdtType
    FROM DsDdGranule
    WHERE RequestId = @RequestId
    AND GranuleId = @GranuleId

    SELECT @status = @@error

    IF (@status != 0)
      BEGIN
        SELECT @status = 1
        RAISERROR 20018 "DsDdGSelectByReqGran:  Unable to select information on
RequestId [%1!] and GranuleId [%2!].", @RequestId, @GranuleId
      END

  END
  RETURN(@status)
END
go
```

Procedure: DsDdGUpdate

Code

```
-- ****************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdGUpdate
--
-- DESCRIPTION:         Update a record into the DsDdGranule table.
-- DATE CREATED: 12/05/97
-- CREATED BY:          Kenya Wright
--
-- TABLES ACCESSED:     DsDdGranule
--
-- RETURNS:          Status (Success = 0)
--
-- INPUT PARAMETERS            DOMAIN
-- ----------------            ------
--   RequestId              requestid
--   GranuleId              granuleid
--   Compressability           int
--   NrGranFiles            itemcount
--   GranuleSize            size
--   EstGranuleSize           size
```

```
--   StageDiskSize            size
--   EsdtType                 esdttype
--
-- OUTPUT PARAMETER             DOMAIN
-- ----------------                  ------
--   NONE
-- ****************************************************************
CREATE PROCEDURE DsDdGUpdate
    @RequestId       requestid,
    @GranuleId       granuleid,
    @Compressability  int,
    @NrGranFiles      itemcount,
    @GranuleSize      size,
    @EstGranuleSize   size,
    @StageDiskSize    size,
    @EsdtType        esdttype
AS
BEGIN
  DECLARE @mystatus int

  SELECT @mystatus = 0

  IF NOT EXISTS (SELECT 1 FROM DsDdGranule
              WHERE RequestId = @RequestId
              AND GranuleId = @GranuleId )
    BEGIN
      SELECT @mystatus = 1
      raiserror 20019 "DsDdGUpdate :  The RequestId [%1!] and GranuleId [%2!] you are
attempting to update does not exist.", @RequestId, @GranuleId
      RETURN (@mystatus)
    END

  BEGIN TRANSACTION
    UPDATE DsDdGranule
    SET Compressability = @Compressability,
       NrGranFiles     = @NrGranFiles,
       GranuleSize     = @GranuleSize,
       EstGranuleSize  = @EstGranuleSize,
       StageDiskSize   = @StageDiskSize,
       EsdtType        = @EsdtType
    WHERE RequestId = @RequestId
    AND GranuleId = @GranuleId

    SELECT @mystatus = @@error

    IF (@mystatus != 0)
      BEGIN
        SELECT @mystatus = 1
        ROLLBACK TRANSACTION
        raiserror 20020 "DsDdGUpdate:  Unable to update RequestId [%1!] and GranuleId
[%2!].", @RequestId, @GranuleId
      END
```

```
    ELSE

      BEGIN
        COMMIT TRANSACTION
        RETURN(@mystatus)
      END
END
go
```

Procedure: DsDdPLDelete

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdPLDelete
--
-- DESCRIPTION:         Deletes request from the DsDdParameterList Table
-- DATE CREATED: 12/03/97
-- CREATED BY:                Kenya Wright
--
-- TABLES ACCESSED:     DsDdParameterList
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS PARAMETER     DOMAIN
-- ---------------      ------
--    RequestId         requestid
--
-- OUTPUTS PARAMETER    DOMAIN
-- -----------------    ------
--    NONE
-- ************************************************************************
CREATE PROCEDURE DsDdPLDelete
    @RequestId        requestid
AS
BEGIN
  DECLARE @status int

-- First delete corresponding GrandChild table records with foreign key
-- constraints
  DELETE DsDdFile
  WHERE RequestId = @RequestId

  DELETE DsDdGranule
  WHERE RequestId = @RequestId

-- Delete corresponding Child table records with foreign key constraints
  DELETE DsDdRequest
  WHERE RequestId = @RequestId
```

```
-- Delete Parent table records with foreign key relationships dropped
  DELETE DsDdParameterList
  WHERE RequestId = @RequestId

  SELECT @status = @@error

  IF (@status != 0)
    BEGIN
      ROLLBACK TRANSACTION
      SELECT @status = 1
      RAISERROR 20021 "DsDdPLDelete: Unable to delete RequestId [%1!] from
DsDdParameterList table.", @RequestId
      RETURN (@status)
    END

END
go
```

Procedure: DsDdPLInsert

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdPLInsert
--
-- DESCRIPTION:         Insert a record into the DsDdParameterList table.
-- DATE CREATED: 12/02/97
-- CREATED BY:          Kenya Wright
--
-- TABLES ACCESSED:     DsDdParameterList
--
-- RETURNS:          Status (Success = 0)
--
-- INPUT PARAMETERS          DOMAIN
-- ----------------          ------
--   RequestId            requestid
--   MediaType              mediatype
--   MediaFormat             varchar(50)
--   Notify               varchar(50)
--   Site                 varchar(50)
--   UserString             varchar(50)
--   NotifyType              varchar(50)
--   UserProfile            varchar(50)
--   FtpUser              username
--   FtpPassword             varchar(50)
--   FtpHost              varchar(50)
--   FtpPushDest             varchar(50)
--
-- OUTPUT PARAMETER          DOMAIN
```

```
-- ---------------                    ------
--   NONE
-- ************************************************************************
CREATE PROCEDURE DsDdPLInsert
    @RequestId       requestid,
    @MediaType       mediatype,
    @MediaFormat     varchar(50),
    @Notify          varchar(50),
    @Site            varchar(50),
    @UserString      varchar(50),
    @NotifyType      varchar(50),
    @UserProfile     varchar(50),
    @FtpUser         username,
    @FtpPassword     varchar(50),
    @FtpHost         varchar(50),
    @FtpPushDest     varchar(50)
AS
BEGIN
  DECLARE @mystatus int

  SELECT @mystatus = 0

-- Error Checking to see if RequestId is blank or null
  IF (@RequestId IS NULL OR @RequestId = "")
    BEGIN
      SELECT @mystatus = 1
      raiserror 35023 "DsDdPLInsert:  RequestId may not be blank or null."
      RETURN(@mystatus)
    END

-- Error Checking to see if RequestId already exists
  IF EXISTS (SELECT 1 FROM DsDdParameterList
          WHERE RequestId = @RequestId)
    BEGIN
      SELECT @mystatus = 1
      raiserror 20022 "DsDdPLInsert:  RequestId [%1!] already exists.", @RequestId
      RETURN(@mystatus)
    END

  BEGIN TRANSACTION
  INSERT DsDdParameterList (
      RequestId,
      MediaType,
      MediaFormat,
      Notify,
      Site,
      UserString,
      NotifyType,
      UserProfile,
      FtpUser,
      FtpPassword,
      FtpHost,
```

```
      FtpPushDest
     )

  VALUES (
     @RequestId,
     @MediaType,
     @MediaFormat,
     @Notify,
     @Site,
     @UserString,
     @NotifyType,
     @UserProfile,
     @FtpUser,
     @FtpPassword,
     @FtpHost,
     @FtpPushDest
        )

  SELECT @mystatus = @@error

  IF (@mystatus != 0)
    BEGIN
      SELECT @mystatus = 1
      ROLLBACK TRANSACTION
      raiserror 20023 "DsDdPLInsert:  Unable to insert RequestId [%1!]."
    END

  ELSE

    BEGIN
      COMMIT TRANSACTION
      RETURN(@mystatus)
    END
END
go
```

Procedure: DsDdPLSelectAll

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdPLSelectAll
--
-- DESCRIPTION:              Returns the complete row of information for all rows
--                  in the DsDdParameterList table.
-- DATE CREATED:       12/03/97
-- CREATED BY:         Kenya Wright
--
-- TABLES ACCESSED:    DsDdParameterList
--
```

```
-- RETURNS:            Status (Success = 0)
--
-- INPUTS PARAMETER        DOMAIN
-- ----------------        ------
--   NONE
--
-- OUTPUTS PARAMETER       DOMAIN
-- ----------------        ------
--   RequestId             requestid
--   MediaType               mediatype
--   MediaFormat             varchar(50)
--   Notify                varchar(50)
--   Site                  varchar(50)
--   UserString              varchar(50)
--   NotifyType              varchar(50)
--   UserProfile             varchar(50)
--   FtpUser               username
--   FtpPassword             varchar(50)
--   FtpHost               varchar(50)
--   FtpPushDest             varchar(50)
--
-- ************************************************************************
CREATE PROCEDURE DsDdPLSelectAll
AS
BEGIN
  DECLARE @status     int

  SELECT @status = 0

  BEGIN
    SELECT RequestId,
         MediaType,
         MediaFormat,
         Notify,
         Site,
         UserString,
         NotifyType,
         UserProfile,
         FtpUser,
         FtpPassword,
         FtpHost,
         FtpPushDest
    FROM   DsDdParameterList

    SELECT @status = @@error

    IF (@status != 0)
      BEGIN
        SELECT @status = 1
        RAISERROR 20024 "DsDdPLSelectAll:  Unable to select DsDdParameterList records."
      END
```

```
  END
  RETURN(@status)
END
go
```

Procedure: DsDdPLSelectByReq

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdPLSelectByReq
--
-- DESCRIPTION:             Gets information for a given RequestId
-- DATE CREATED:       12/17/97
-- CREATED BY:         Lisa Colombo
--
-- TABLES ACCESSED:    DsDdParameterList
--
-- RETURNS:            Status (Success = 0)
--
-- INPUTS PARAMETER        DOMAIN
-- ----------------        ------
--  RequestId              requestid
--
-- OUTPUTS PARAMETER       DOMAIN
-- ----------------        ------
--   RequestId             requestid
--   MediaType               mediatype
--   MediaFormat             varchar(50)
--   Notify                varchar(50)
--   Site                varchar(50)
--   UserString              varchar(50)
--   NotifyType              varchar(50)
--   UserProfile             varchar(50)
--   FtpUser               username
--   FtpPassword             varchar(50)
--   FtpHost               varchar(50)
--   FtpPushDest             varchar(50)
--
-- ************************************************************************
CREATE PROCEDURE DsDdPLSelectByReq
  @RequestId         requestid
AS
BEGIN
  DECLARE @status     int

  SELECT @status = 0

  BEGIN
    SELECT RequestId,
```

```
            MediaType,
            MediaFormat,
            Notify,
            Site,
            UserString,
            NotifyType,
            UserProfile,
            FtpUser,
            FtpPassword,
            FtpHost,
            FtpPushDest
        FROM DsDdParameterList
        WHERE RequestId = @RequestId

        SELECT @status = @@error

        IF (@status != 0)
          BEGIN
            SELECT @status = 1
            RAISERROR 20025 "DsDdPLSelectByReq:  Unable to select DsDdParameterList
information for Requestid [%1!].", @RequestId
          END

    END
    RETURN(@status)
END
go
```

Procedure: DsDdPLUpdate

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdPLUpdate
--
-- DESCRIPTION:         Update a record into the DsDdParameterList table.
-- DATE CREATED: 12/05/97
-- CREATED BY:          Kenya Wright
--
-- TABLES ACCESSED:     DsDdParameterList
--
-- RETURNS:          Status (Success = 0)
--
-- INPUT PARAMETERS            DOMAIN
-- ----------------            ------
--   RequestId              requestid
--   MediaType                mediatype
--   MediaFormat               varchar(50)
--   Notify                 varchar(50)
--   Site                  varchar(50)
```

```
--   UserString              varchar(50)
--   NotifyType              varchar(50)
--   UserProfile             varchar(50)
--   FtpUser                 username
--   FtpPassword             varchar(50)
--   FtpHost                 varchar(50)
--   FtpPushDest             varchar(50)
--
-- OUTPUT PARAMETER             DOMAIN
-- ---------------                  ------
--   NONE
-- ************************************************************************
CREATE PROCEDURE DsDdPLUpdate
    @RequestId       requestid,
    @MediaType       mediatype,
    @MediaFormat     varchar(50),
    @Notify          varchar(50),
    @Site            varchar(50),
    @UserString      varchar(50),
    @NotifyType      varchar(50),
    @UserProfile     varchar(50),
    @FtpUser         username,
    @FtpPassword     varchar(50),
    @FtpHost         varchar(50),
    @FtpPushDest     varchar(50)
AS
BEGIN
  DECLARE @mystatus int

  SELECT @mystatus = 0

  IF NOT EXISTS (SELECT 1 FROM DsDdParameterList
            WHERE RequestId = @RequestId )
    BEGIN
      SELECT @mystatus = 1
      raiserror 20026 "DsDdPLUpdate :  The RequestId [%1!] you are attempting to update does
not exist.", @RequestId
      RETURN (@mystatus)
    END

  BEGIN TRANSACTION
  UPDATE DsDdParameterList
  SET MediaType   = @MediaType,
      MediaFormat = @MediaFormat,
      Notify      = @Notify,
      Site        = @Site,
      UserString  = @UserString,
      NotifyType  = @NotifyType,
      UserProfile = @UserProfile,
      FtpUser     = @FtpUser,
      FtpPassword = @FtpPassword,
      FtpHost     = @FtpHost,
```

```
    FtpPushDest = @FtpPushDest
  WHERE RequestId = @RequestId

  SELECT @mystatus = @@error

  IF (@mystatus != 0)
    BEGIN
      SELECT @mystatus = 1
      ROLLBACK TRANSACTION
      raiserror 20027 "DsDdPLUpdate:  Unable to update RequestId [%1!]."
    END

  ELSE

    BEGIN
      COMMIT TRANSACTION
      RETURN(@mystatus)
    END
END
go
```

Procedure: DsDdRDelete

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdRDelete
--
-- DESCRIPTION:         Deletes request from the DsDdRequest Table
-- DATE CREATED: 12/03/97
-- CREATED BY:                Kenya Wright
--
-- TABLES ACCESSED:     DsDdRequest
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS PARAMETER     DOMAIN
-- ----------------     ------
--    RequestId         requestid
--
-- OUTPUTS PARAMETER    DOMAIN
-- -----------------    ------
--    NONE
-- ************************************************************************
CREATE PROCEDURE DsDdRDelete
    @RequestId        requestid
AS
BEGIN
  DECLARE @status int
```

```
-- First delete corresponding GrandChild table records with foreign
-- key constraints
   DELETE DsDdFile
   WHERE RequestId = @RequestId

-- Delete corresponding Child table records with foreign key constraints
   DELETE DsDdGranule
   WHERE RequestId = @RequestId

-- Delete Parent table record with foreign key relationships dropped
   DELETE DsDdRequest
   WHERE RequestId = @RequestId

   SELECT @status = @@error

   IF (@status != 0)
      BEGIN
         ROLLBACK TRANSACTION
         SELECT @status = 1
         RAISERROR 20028 "DsDdRDelete: Unable to delete RequestId [%1!].", @RequestId
         RETURN (@status)
      END

END
go
```

Procedure: DsDdRInsert

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdRInsert
--
-- DESCRIPTION:         Insert a record into the DsDdRequest table.
-- DATE CREATED: 12/03/97
-- CREATED BY:          Kenya Wright
--
-- TABLES ACCESSED:     DsDdRequest
--
-- RETURNS:          Status (Success = 0)
--
-- INPUT PARAMETERS            DOMAIN
-- ----------------            ------
--   RequestId              requestid
--   Status                 status
--   Priority               priority
--   State                  state
--   OrderedState           state
--   OrderId                orderid
--   EcsUserId              varchar(50)
```

```
--   SizeInMB                size
--   NrReqFiles              itemcount
--   EsdtType                esdttype
--   StartTime               reqtime
--   EndTime                 reqtime
--   LastSuccMediaNr         int
--   NrGranules              itemcount
--   WarmStartCounter        itemcount
--   CurrDdistStageDisk      varchar(255)
--   CallBackFunction        callback
--   RPCId                   varchar(255)
--   SDSRVStageArea          varchar(255)
--
-- OUTPUT PARAMETER              DOMAIN
-- ----------------              ------
--   NONE
-- ************************************************************************
CREATE PROCEDURE DsDdRInsert
    @RequestId          requestid,
    @Status             status,
    @Priority           priority,
    @State              state,
    @OrderedState       state,
    @OrderId            orderid,
    @EcsUserId          varchar(50),
    @SizeInMB           size,
    @NrReqFiles         itemcount,
    @EsdtType           esdttype,
    @StartTime          reqtime,
    @EndTime            reqtime,
    @LastSuccMediaNr    int,
    @NrGranules         itemcount,
    @WarmStartCounter   itemcount,
    @CurrDdistStageDisk varchar(255),
    @CallBackFunction   callback,
    @RPCId              varchar(255),
    @SDSRVStageArea     varchar(255)
AS
BEGIN
  DECLARE @mystatus int

  SELECT @mystatus = 0

-- Error Checking to see if RequestId is blank or null
  IF (@RequestId IS NULL OR @RequestId = "")
    BEGIN
      SELECT @mystatus = 1
      raiserror 20029 "DsDdRInsert:  RequestId may not be blank or null."
      RETURN(@mystatus)
    END
-- Error Checking to see if RequestId already exists
  IF EXISTS (SELECT 1 FROM DsDdRequest WHERE RequestId = @RequestId)
```

```
    BEGIN
      SELECT @mystatus = 1
      raiserror 20030 "DsDdRInsert:  RequestId [%1!] already exists.", @RequestId
      RETURN(@mystatus)
    END

  BEGIN TRANSACTION
  INSERT DsDdRequest (
        RequestId,
        Status,
        Priority,
        State,
        OrderedState,
        OrderId,
        EcsUserId,
        SizeInMB,
        NrReqFiles,
        EsdtType,
        StartTime,
        EndTime,
        LastSuccMediaNr,
        NrGranules,
        WarmStartCounter,
        CurrDdistStageDisk,
        CallBackFunction,
        RPCId,
        SDSRVStageArea
       )

  VALUES (
        @RequestId,
        @Status,
        @Priority,
        @State,
        @OrderedState,
        @OrderId,
        @EcsUserId,
        @SizeInMB,
        @NrReqFiles,
        @EsdtType,
        @StartTime,
        @EndTime,
        @LastSuccMediaNr,
        @NrGranules,
        @WarmStartCounter,
        @CurrDdistStageDisk,
        @CallBackFunction,
        @RPCId,
        @SDSRVStageArea
       )

  SELECT @mystatus = @@error
```

```
  IF (@mystatus != 0)
    BEGIN
      SELECT @mystatus = 1
      ROLLBACK TRANSACTION
      raiserror 20031 "DsDdRInsert:  Unable to insert RequestId [%1!].", @RequestId
    END

  ELSE

    BEGIN
      COMMIT TRANSACTION
      RETURN(@mystatus)
    END
END
go
```

Procedure: DsDdRSelectAll

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdRSelectAll
--
-- DESCRIPTION:              Returns the complete row of information for all rows
--                  in the DsDdRequest table.
-- DATE CREATED:       12/03/97
-- CREATED BY:         Kenya Wright
--
-- TABLES ACCESSED:    DsDdRequest
--
-- RETURNS:            Status (Success = 0)
--
-- INPUTS PARAMETER        DOMAIN
-- ----------------        ------
--   NONE
--
-- OUTPUTS PARAMETER       DOMAIN
-- ----------------        ------
--   RequestId             requestid
--   Status                status
--   Priority              priority
--   State                 state
--   OrderedState          state
--   OrderId               orderid
--   EcsUserId             varchar(50)
--   SizeInMB              size
--   NrReqFiles            itemcount
--   EsdtType              esdttype
--   StartTime             reqtime
```

```
--   EndTime              reqtime
--   LastSuccMediaNr      int
--   NrGranules           itemcount
--   WarmStartCounter     itemcount
--   CurrDdistStageDisk   varchar(255)
--   CallBackFunction     callback
--   RPCId                varchar(255)
--   SDSRVStageArea       varchar(255)
--
-- ************************************************************************
CREATE PROCEDURE DsDdRSelectAll
AS
BEGIN
    DECLARE @status    int

    SELECT @status = 0

    BEGIN
      SELECT RequestId,
           Status,
           Priority,
           State,
           OrderedState,
           OrderId,
           EcsUserId,
           SizeInMB,
           NrReqFiles,
           EsdtType,
           StartTime,
           EndTime,
           LastSuccMediaNr,
           NrGranules,
           WarmStartCounter,
           CurrDdistStageDisk,
           CallBackFunction,
           RPCId,
           SDSRVStageArea
      FROM   DsDdRequest

      SELECT @status = @@error

      IF (@status != 0)
        BEGIN
          SELECT @status = 1
          RAISERROR 20032 "DsDdRSelectAll:  Unable to select information about Request."
        END

    END
    RETURN(@status)
END
go
```

Procedure: DsDdRSelectByRequestId

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:       DsDdRSelectByRequestId
--
-- DESCRIPTION:              Returns all information for a given RequestId.
-- DATE CREATED:        12/11/97
-- CREATED BY:          Kenya Wright
--
-- TABLES ACCESSED:     DsDdRequest
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS PARAMETER         DOMAIN
-- ----------------         ------
--   RequestId              requestid
--
-- OUTPUTS PARAMETER        DOMAIN
-- ----------------         ------
--   RequestId              requestid
--   Status                 status
--   Priority               priority
--   State                  state
--   OrderedState           state
--   OrderId                orderid
--   EcsUserId              varchar(50)
--   SizeInMB               size
--   NrReqFiles             itemcount
--   EsdtType               esdttype
--   StartTime              reqtime
--   EndTime                reqtime
--   LastSuccMediaNr        int
--   NrGranules             itemcount
--   WarmStartCounter       itemcount
--   CurrDdistStageDisk     varchar(255)
--   CallBackFunction       callback
--   RPCId                  varchar(255)
--   SDSRVStageArea         varchar(255)
--
-- ************************************************************************
CREATE PROCEDURE DsDdRSelectByRequestId
  @RequestId         requestid
AS
BEGIN
  DECLARE @status     int

  SELECT @status = 0
```

```
  BEGIN
    SELECT RequestId,
          Status,
          Priority,
          State,
          OrderedState,
          OrderId,
          EcsUserId,
          SizeInMB,
          NrReqFiles,
          EsdtType,
          StartTime,
          EndTime,
          LastSuccMediaNr,
          NrGranules,
          WarmStartCounter,
          CurrDdistStageDisk,
          CallBackFunction,
          RPCId,
          SDSRVStageArea
     FROM DsDdRequest
     WHERE RequestId = @RequestId

    SELECT @status = @@error

    IF (@status != 0)
      BEGIN
        SELECT @status = 1
        RAISERROR 20033 "DsDdRSelectByRequestId:  Unable to select information about
RequestId [%1!].", @RequestId
      END

  END
  RETURN(@status)
END
go
```

Procedure: DsDdRSelectByState

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdRSelectByState
--
-- DESCRIPTION:               Returns the complete list of requests of a
--                  specific state in the DsDdRequest table.
-- DATE CREATED:       12/08/97
-- DEVELOPER:          Kenya Wright
--
-- TABLES ACCESSED:    DsDdRequest
```

```
--
-- RETURNS:            Status (Success = 0)
--
-- INPUTS PARAMETER        DOMAIN
-- ----------------        ------
--   State                state
--
-- OUTPUTS PARAMETER       DOMAIN
-- ----------------        ------
--   RequestId             requestid
--   Status               status
--   Priority             priority
--   State                state
--   OrderedState          state
--   OrderId              orderid
--   EcsUserId             varchar(50)
--   SizeInMB              size
--   NrReqFiles            itemcount
--   EsdtType             esdttype
--   StartTime            reqtime
--   EndTime              reqtime
--   LastSuccMediaNr       int
--   NrGranules           itemcount
--   WarmStartCounter      itemcount
--   CurrDdistStageDisk    varchar(255)
--   CallBackFunction      callback
--   RPCId                varchar(255)
--   SDSRVStageArea        varchar(255)
--
-- ****************************************************************************
CREATE PROCEDURE DsDdRSelectByState
  @State            state
AS
BEGIN
  DECLARE @status    int

  SELECT @status     = 0

-- Error checking to see if State requested exists
  IF NOT EXISTS (SELECT 1 FROM DsDdRequest WHERE State = @State)
    BEGIN
      SELECT @status = 1
      RAISERROR 20034 "DsDdRSelectByState:  State [%1!] does not exist.", @State
      RETURN (@status)
    END

  BEGIN
    SELECT RequestId,
           Status,
           Priority,
           State,
           OrderedState,
```

```
            OrderId,
            EcsUserId,
            SizeInMB,
            NrReqFiles,
            EsdtType,
            StartTime,
            EndTime,
            LastSuccMediaNr,
            NrGranules,
            WarmStartCounter,
            CurrDdistStageDisk,
            CallBackFunction,
            RPCId,
            SDSRVStageArea
        FROM DsDdRequest
        WHERE State = @State

        SELECT @status = @@error

        IF (@status !=0)
          BEGIN
            SELECT @status = 1
            RAISERROR 20034 "DsDdRSelectByState:  Unable to select information with State
[%1!].", @State
          END

    END
    RETURN(@status)
END
go
```

Procedure: DsDdRTBSelectAll

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdRTBSelectAll
--
-- DESCRIPTION:               Returns the complete row of information for all rows
--                    in the DsDdRequest table.
-- DATE CREATED:        12/03/97
-- CREATED BY:          Kenya Wright
--
-- TABLES ACCESSED:     DsDdRequest
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS PARAMETER         DOMAIN
-- ----------------         ------
--   NONE
```

```
--
-- OUTPUTS PARAMETER       DOMAIN
-- ----------------        ------
--   RequestId              requestid
--   Status                 status
--   Priority               priority
--   State                  state
--   OrderedState           state
--   OrderId                orderid
--   EcsUserId              varchar(50)
--   SizeInMB               size
--   NrReqFiles             itemcount
--   EsdtType               esdttype
--   StartTime              reqtime
--   EndTime                reqtime
--   LastSuccMediaNr        int
--   NrGranules             itemcount
--   WarmStartCounter       itemcount
--   CurrDdistStageDisk     varchar(255)
--   CallBackFunction       callback
--   RPCId                  varchar(255)
--   SDSRVStageArea         varchar(255)
--
-- ************************************************************************
CREATE PROCEDURE DsDdRTBSelectAll
AS
BEGIN
    DECLARE @status     int

    SELECT @status = 0

    BEGIN
      SELECT RequestId,
            Status,
            Priority,
            State,
            OrderedState,
            OrderId,
            EcsUserId,
            SizeInMB,
            NrReqFiles,
            EsdtType,
            StartTime,
            EndTime,
            LastSuccMediaNr,
            NrGranules,
            WarmStartCounter,
            CurrDdistStageDisk,
            CallBackFunction,
            RPCId,
            SDSRVStageArea
      FROM   DsDdRequestTB
```

```
        SELECT @status = @@error

        IF (@status != 0)
          BEGIN
            SELECT @status = 1
            RAISERROR 20032 "DsDdRTBSelectAll:  Unable to select information about
Request."
          END

      END
      RETURN(@status)
END
go
```

Procedure: DsDdRUpdate

Code

```
-- ************************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdRUpdate
--
-- DESCRIPTION:         Update a record into the DsDdRequest table.
-- DATE CREATED: 12/05/97
-- CREATED BY:          Kenya Wright
--
-- TABLES ACCESSED:     DsDdRequest
--
-- RETURNS:          Status (Success = 0)
--
-- INPUT PARAMETERS            DOMAIN
-- ----------------            ------
--   RequestId              requestid
--   Status                 status
--   Priority               priority
--   State                  state
--   OrderedState           state
--   OrderId                orderid
--   EcsUserId              varchar(50)
--   SizeInMB               size
--   NrReqFiles             itemcount
--   EsdtType               esdttype
--   StartTime              reqtime
--   EndTime                reqtime
--   LastSuccMediaNr        int
--   NrGranules             itemcount
--   WarmStartCounter       itemcount
--   CurrDdistStageDisk     varchar(255)
--   CallBackFunction       callback
--   RPCId                  varchar(255)
```

```
--   SDSRVStageArea            varchar(255)
--
-- OUTPUT PARAMETER              DOMAIN
-- ----------------              ------
--   NONE
-- ****************************************************************
CREATE PROCEDURE DsDdRUpdate
    @RequestId          requestid,
    @Status             status,
    @Priority           priority,
    @State              state,
    @OrderedState       state,
    @OrderId            orderid,
    @EcsUserId          varchar(50),
    @SizeInMB           size,
    @NrReqFiles         itemcount,
    @EsdtType           esdttype,
    @StartTime          reqtime,
    @EndTime            reqtime,
    @LastSuccMediaNr    int,
    @NrGranules         itemcount,
    @WarmStartCounter   itemcount,
    @CurrDdistStageDisk varchar(255),
    @CallBackFunction   callback,
    @RPCId              varchar(255),
    @SDSRVStageArea     varchar(255)
AS
BEGIN
  DECLARE @mystatus int

  SELECT @mystatus = 0

  IF NOT EXISTS (SELECT 1 FROM DsDdRequest
            WHERE RequestId = @RequestId )
    BEGIN
      SELECT @mystatus = 1
      raiserror 20035 "DsDdRUpdate :  The RequestId [%1!] you are attempting to update does
not exist .", @RequestId
      RETURN (@mystatus)
    END

  BEGIN TRANSACTION
  UPDATE DsDdRequest
  SET Status            = @Status,
     Priority           = @Priority,
     State              = @State,
     OrderedState       = @OrderedState,
     OrderId            = @OrderId,
     EcsUserId          = @EcsUserId,
     SizeInMB           = @SizeInMB,
     NrReqFiles         = @NrReqFiles,
     EsdtType           = @EsdtType,
```

```
      StartTime          = @StartTime,
      EndTime            = @EndTime,
      LastSuccMediaNr    = @LastSuccMediaNr,
      NrGranules         = @NrGranules,
      WarmStartCounter   = @WarmStartCounter,
      CurrDdistStageDisk = @CurrDdistStageDisk,
      CallBackFunction   = @CallBackFunction,
      RPCId              = @RPCId,
      SDSRVStageArea     = @SDSRVStageArea
    WHERE RequestId = @RequestId

    SELECT @mystatus = @@error

    IF (@mystatus != 0)
      BEGIN
        SELECT @mystatus = 1
        ROLLBACK TRANSACTION
        raiserror 20036 "DsDdRUpdate:  Unable to update RequestId [%1!]."
      END

    ELSE

      BEGIN
        COMMIT TRANSACTION
        RETURN(@mystatus)
      END

END
go
```

Procedure: DsDdRUpdateOrdState

Code

```
-- ****************************************************************
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      DsDdRUpdateOrdState
--
-- DESCRIPTION:         Updates the ordered state for a given RequestId.
-- DATE CREATED: 12/12/97
-- CREATED BY:          Kenya Wright
--
-- TABLES ACCESSED:     DsDdRequest
--
-- RETURNS:          Status (Success = 0)
--
-- INPUT PARAMETERS          DOMAIN
-- ----------------          ------
--   RequestId             requestid
--   OrderedState            state
--
```

```
-- OUTPUT PARAMETER            DOMAIN
-- ----------------            ------
--   NONE
-- ***************************************************************
CREATE PROCEDURE DsDdRUpdateOrdState
   @RequestId         requestid,
   @OrderedState      state
AS
BEGIN
   DECLARE @mystatus int

   SELECT @mystatus = 0

   IF NOT EXISTS (SELECT 1 FROM DsDdRequest
                  WHERE RequestId = @RequestId )
      BEGIN
         SELECT @mystatus = 1
         raiserror 20037 "DsDdRUpdateOrdState:  The RequestId [%1!] you are attempting to
update does not exist.", @RequestId
         RETURN (@mystatus)
      END

   BEGIN TRANSACTION
   UPDATE DsDdRequest
   SET OrderedState = @OrderedState
   WHERE RequestId = @RequestId

   SELECT @mystatus = @@error

   IF (@mystatus != 0)
      BEGIN
         SELECT @mystatus = 1
         ROLLBACK TRANSACTION
         raiserror 20038 "DsDdRUpdateOrdState:  Unable to update RequestId [%1!]."
      END

   ELSE

      BEGIN
         COMMIT TRANSACTION
         RETURN(@mystatus)
      END

END
go
```

Procedure: DsDdRUpdateState

Code

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

```
--
-- PROCEDURE NAME:      DsDdRUpdateState
--
-- DESCRIPTION:         Updates the state field in the DsDdRequest table.
-- DATE CREATED: 12/12/97
-- CREATED BY:          Kenya Wright
--
-- TABLES ACCESSED:     DsDdRequest
--
-- RETURNS:          Status (Success = 0)
--
-- INPUT PARAMETERS          DOMAIN
-- ----------------          ------
--   RequestId            requestid
--   State                state
--
-- OUTPUT PARAMETER          DOMAIN
-- ----------------          ------
--   NONE
-- ************************************************************************
CREATE PROCEDURE DsDdRUpdateState
  @RequestId       requestid,
  @State           state
AS
BEGIN
  DECLARE @mystatus int

  SELECT @mystatus = 0

  IF NOT EXISTS (SELECT 1 FROM DsDdRequest
            WHERE RequestId = @RequestId )
    BEGIN
      SELECT @mystatus = 1
      raiserror 20039 "DsDdRUpdateState:  The RequestId [%1!] you are attempting to update
does not exist.", @RequestId
      RETURN (@mystatus)
    END

  BEGIN TRANSACTION
  UPDATE DsDdRequest
  SET State    = @State
  WHERE RequestId = @RequestId

  SELECT @mystatus = @@error

  IF (@mystatus != 0)
    BEGIN
      SELECT @mystatus = 1
      ROLLBACK TRANSACTION
      raiserror 20040 "DsDdRUpdateState:  Unable to update RequestId [%1!]."
    END
```

```
  ELSE

    BEGIN
      COMMIT TRANSACTION
      RETURN(@mystatus)
    END

END
go
```

4.2 File Usage

There are cases when the implementation of a persistent data requirement is better suited to a flat file than to a database table. A typical example of such data is system configuration information. System configuration information is fairly static and usually has no explicit relationship to other data in the enterprise. Another common use of files in ECS is as an interface mechanism between ECS and the external world. FIILSUB file usage is detailed in this section via file definitions, attribute definitions, and attribute domain definitions.

4.2.1 Files Definitions

A listing of each the files in the DDIST database is given here. A brief definition of each of these files follows.

TBS

4.2.2 Attributes

Brief definitions of each of the attributes present in the files defined above are contained herein.

TBS

4.2.3 Attribute Domains

Domains represent the ranges of valid values allowed for a given file attribute. Attributes domains for each of the attributes defined above are given here.

TBS

5. Performance and Tuning Factors

5.1 Indexes

An index provides a means of locating a row in a table based on the value of specific a columns, without having to scan each row in the table. If used appropriately, indexes can significantly increase data retrieval. Sybase allows the definition of two types of indexes, clustered and non-clustered. In a clustered index, the rows in a tables are physically stored in the sort order determined by the index. Clustered indexes are particularly useful, when the data is frequently retrieved in order. Non-clustered indexes differ from their clustered counterpart, in that data is not physically stored in sort order. Only one clustered index may be defined per table. All of the indexes defined against tables in the DDIST database are described herein.

Index List

Table Code	Index Code	P	F	U	C
DsDdFile	pk_dsddfile	Yes	No	Yes	Yes
DsDdGranule	pk_dsddgranule	Yes	No	Yes	Yes
DsDdParameterList	pk_dsddparameterlist	Yes	No	Yes	Yes
DsDdRequest	pk_dsddrequest	Yes	Yes	Yes	Yes

5.2 Segments

Sybase supports the definition of segments. A segment is a named pointer to a storage device or devices. Segments are used to manually place database objects onto particular storage devices. All segments explicitly defined in the DDIST database are described herein.

The are no explicitly defined segments for the DDIST database.

5.3 Named Caches

A cache is a block of memory that is used by Sybase to house data pages that are currently being accessed. A named cache is a named block of memory that the SQL server can use to house frequently accessed tables. Assigning a table to cache causes it to be loaded into memory. This greatly increases performance by eliminating the time expense normally associated with disk i/o. Named caches used in the DDIST databases are described herein.

The are no named caches for the subscription server database.

This page intentionally left blank.

6. Database Security

6.1 Initial Users

Upon initial installation the following users will have access to DDIST database. The level of access is limited to that associated with their assigned group and/or role. A complete definition of each if these groups and roles is given below.

EcDsDdistGui

EcDsDistributionServer

6.2 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. Several group have been defined in the DDIST database upon initial installation. A definition of each of these groups is contained herein.

TBS

6.3 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 a 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 to 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 sa_role

Operator (oper_role) - This role is used to grant a specific user the permissions needed to manage backup and recovery of 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 perform database consistency checker (dbcc), a sybase supplied utility, commands that are considered outside of the realm of normal system administrator activities.

6.4 Object Permissions

Group Permissions

Group	Object	Grant				
TBS						

7. Replication

Replication is not implemented for the DDIST database.

This page intentionally left blank.

8. Scripts

8.1 Installation Scripts

Any scripts used to support installation of the DDIST database are described herein. These files are found in the directory /ecs/formal/CSS/DOF/src/SUBSCRIPTION/sybase

Script File	Description
TBS	

8.2 De-Installation Scripts

Any scripts used to support de-installation of the DDIST database are described herein.

Script File	Description
TBS	

8.3 Backup/Recovery Scripts

Any scripts used to facilitate backup or recovery of the DDIST database are described herein.

Script File	Description
TBS	

8.4 Miscellaneous Scripts

Miscellaneous scripts applicable to the DDIST database are described herein.

Script File	Description
TBS	

This page intentionally left blank.

Abbreviations and Acronyms

ACL	Access Control List
ACMHW	Access and Control Management HWCI
ADC	affiliated data center
ADSHW	Advertising Server HWCI
ADSRV	Advertising Service CSCI
AI&T	algorithm integration and test
AITHW	Algorithm Integration and Test HWCI
AITTL	Algorithm Integration and Test CSCI
AM-1	EOS AM Project spacecraft 1, morning spacecraft series -- ASTER, CERES, MISR, MODIS and MOPITT instruments
ANSI	American National Standards Institute
API	application program (or programming) interface
APID	application's process ID
AQAHW	Algorithm QA HWCI
ASCII	American Standard Code for Information Exchange
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer (formerly ITIR)
AVHRR	Advanced Very High-Resolution Radiometer
BER	bit error rate
BUFR	binary universal format for representation of data
CASE	Computer Aided Software Engineering
CCSDS	Consultative Committee for Space Data Systems
CD	contractual delivery 214-001
CD-ROM	compact disk -- read only memory
CDR	Critical Design Review
CDRL	contract data requirements list
CERES	Clouds and Earth's Radiant Energy System

CI	configuration item
COTS	commercial off-the-shelf (hardware or software)
CPU	central processing unit
CSCI	computer software configuration item
CSDT	Computer Science Data Type
CSMS	Communications and Systems Management Segment (ECS)
CSS	Communications Subsystem
DAAC	Distributed Active Archive Center
DAN	data availability notice
DAO	Data Assimilation Office
DAR	data acquisition request
DAS	data availability schedule
DBMS	Database Management System
DDICT	Data Dictionary CSCI
DDIST	Data Distribution Services CSCI
DDSRV	Document Data Server CSCI
DESKT	Desktop CSCI
DID	data item description
DIM	distributed information manager (SDPS)
DIMGR	Distributed Information Manager CSCI
DIPHW	Distribution and Ingest Peripheral Management HWCI
DMGHW	Data Management HWCI
DMS	Data Management Subsystem
DMWG	Data Management Working Group
DP	Data Provider
DPR	data processing request
DPREP	Science Data Preprocessing CSCI
DPS	Data Processing Subsystem
DRPHW	Data Repository HWCI

DSS	Data Server Subsystem
ECS	EOSDIS Core System
EDC	EROS Data Center
EDHS	ECS Data Handling System
EDOS	EOS Data and Operations System
EOS	Earth Observing System
EOS-AM	EOS Morning Crossing (Descending) Mission -- see AM-1
EOSDIS	Earth Observing System Data and Information System
EROS	Earth Resources Observation System
ESDIS	Earth Science Data and Information System (GSFC)
ESDT	Earth science data types
ESN	EOSDIS Science Network (ECS)
FDDI	fiber distributed data interface
FDF	flight dynamics facility
FDFEPHEM	FDF-generated definitive orbit data
FGDC	Federal Geographic Data Commuittee
FK	Foreign Key
FOO	Flight of Opportunity
FOS	Flight Operations Segment (ECS)
GB	gigabyte (10^9)
GNU	(recursive acronym: "GNU's Not Unix"); a project supported by the Free Software Foundation dedicated to the delivery of free software
GPCP	Global Precipitation Climatology Project
GPCP	Global Precipitation Climatology Project
GPI	GOES Precipitation Index
GRIB	GRid In Binary
GSFC	Goddard Space Flight Center
GTWAY	Version 0 Interoperability Gateway CSCI
GUI	graphic user interface

GV	ground validation
HDF	hierarchical data format
HDF-EOS	an EOS proposed standard for a specialized HDF data format
HIPPI	high performance parallel interface
HMI	human machine interface
HTML	HyperText Markup Language
HTTP	Hypertext Transport Protocol
HWCI	hardware configuration item
I&T	integration and test
I/F	interface
I/O	input/output
ICD	interface control document
ICLHW	Ingest Client HWCI
ID	identification
IDE	Interactive Development Environments
IDG	Infrastructure Development Group
IDR	Incremental Design Review
IERS	International Earth Rotation Service
IMS	Information Management System (obsolete ECS element name)
INGST	Ingest Services CSCI
IOS	Interoperability Subsystem
IP	international partners
IR-1	Interim Release 1
IRD	interface requirements document
ISO	International Standards Organization
ISS	Internetworking Subsystem
IV&V	independent verification and validation
JPL	Jet Propulsion Laboratory
L0-L4	Level 0 (zero) through Level 4

LaRC	Langley Research Center (DAAC)
LIM	local information manager (SDPS)
LIMGR	Local Information Manager CSCI
LIS	Lightning Imaging Sensor
LSM	local system management (ECS)
MB	megabyte (10^6)
MDT	mean downtime
MDT	mean downtime
MFLOPS	mega (millions of) floating-point operations (10^6) per second
MISR	Multi-Angle Imaging SpectroRadiometer
MODIS	Moderate-Resolution Imaging Spectrometer
MOPITT	Measurements of Pollution in the Troposphere
MSFC	Marshall Space Flight Center
MSS	Management Support Subsystem
MTBF	mean time between failure
MTPE	Mission to Planet Earth
MTTR	mean time to restore
N/A	not applicable
NAS	National Academy of Science
NASA	National Aeronautics and Space Administration
NESDIS	National Environmental Satellite Data and Information Service
NMC	National Meteorological Center (NOAA)
NOAA	National Oceanic and Atmospheric Administration
NSIDC	National Snow and Ice Data Center (DAAC)
O/A	orbit/altitude
ODC	other data center
OSI	Open System Interconnect
PDPS	Planning and Data Processing Subsystem
PDR	Preliminary Design Review

PDS	production data set
PGE	Product Generation Executive
PGS	Product Generation System (obsolete ECS element name) (ASTER)
PK	Primary Key
PLANG	Production Planning CSCI
PLNHW	Planning HWCI
PLS	Planning Subsystem
POSIX	Portable Operating System Interface for Computer Environments
PR	Precipitation Radar (TRMM)
PRONG	Processing CSCI
QA	quality assurance
RMA	reliability, maintainability, availability
RTF	rich text format
SAA	satellite active archive
SAGE	Stratospheric Aerosol and Gas Experiment
SCF	Science Computing Facility
SDP	Science Data Processing
SDPF	Sensor Data Processing Facility (GSFC)
SDPS	Science Data Processing Segment (ECS)
SDPTK	SDP Toolkit CSCI
SDSRV	Science Data Server CSCI
SeaWIFS II	Sea-Viewing Wide Field-of-View Sensor II
SFDU	Standard Format Data Unit
SMC	System Management Center (ECS)
SPRHW	Science Processing HWCI
SRS	software requirements specification
SSM/I	Special Sensor for Microwave/Imaging (DMSP)
SST	sea surface temperature
STMGMT	Storage Management

STMGT	Storage Management Software CSCI
DDIST	Subscription Server
TMI	TRMM Microwave Image
TOMS	Total Ozone Mapping Spectrometer
TONS	TDRS On-board Navigational System
TRMM	Tropical Rainfall Measuring Mission (joint US-Japan)
TSDIS	TRMM Science Data and Information System
USNO	US Naval Observatory
UT	universal time
UTC	universal time code
V0	Version 0
VIRS	Visible Infrared Scanner (TRMM)
WAIS	Wide Area Information Server
WKBCH	Workbench CSCI
WKSHW	Working Storage HWCI
WWW	World-Wide Web

This page intentionally left blank.