311-CD-103-005

EOSDIS Core System Project

Release 4
INGEST Database Design and Schema Specifications for the ECS Project

Final

March 1999

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

Raytheon Systems Company
Upper Marlboro, Maryland

Release 4
INGEST Database Design and Schema Specifications for the ECS Project

Final

March 1999

Prepared Under Contract NAS5-60000
CDRL Item #050

RESPONSIBLE ENGINEER

Maureen Muganda /s/ 2/26/99

Maureen Muganda Date
EOSDIS Core System Project

SUBMITTED BY

Mary Armstrong /s/ 2/26/99

Mary Armstrong, Development Engineering Manager Date
EOSDIS Core System Project

Raytheon Systems Company
Upper Marlboro, Maryland

311-CD-103-005

This page intentionally left blank.

Preface

This document describes the data design and database specification for the subsystem. It is one of eight 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-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 (SUBSRV) Subsystem Database Design and Database Schema Specifications for the ECS Project

This submittal meets the milestone specified in the Contract Data Requirements List (CDRL) of NASA Contract NAS5-60000. It is a formal contract deliverable with an approval code 1. It requires Government review and approval prior to acceptance and use. This document is under ECS contractor configuration control. Once approved, contractor approved changes will be handled in accordance with Class I and Class II change control requirements described in the EOS Configuration Management Plan, 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 Release 4 Drop 4PX "as-built" database design and database schema of the INGEST 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	Submitted as Final
iii through xii	Submitted as Final
1-1 and 1-2	Submitted as Final
2-1 and 2-2	Submitted as Final
3-1 and 3-110	Submitted as Final
4-1 through 4-2	Submitted as Final
5-1 through 5-4	Submitted as Final
6-1 through 6-2	Submitted as Final
A-1 through A-4	Submitted as Final
AB-1 through AB-8	Submitted as Final

Document History			
Document Number	**Status/Issue**	**Publication Date**	**CCR Number**
311-CD-103-001	Draft	January 1998	97-1755
311-CD-103-002	Draft	May 1998	98-0620
311-CD-103-003	Draft	July 1998	98-0866
311-CD-103-004	Draft	December 1998	98-1267
311-CD-103-005	Submitted as Final	March 1999	99-0166

This page intentionally left blank.

Contents

Preface

Abstract

1. Introduction

1.1 Purpose and Scope 1-1

1.2 Document Organization 1-1

2. Related Documents

2.1 Applicable Documents 2-1

2.2 Information Documents 2-2

3. Data Design

3.1 Database Overview 3-1

3.1.1 Physical Data Model Entity Relationship Diagram 3-1

3.1.2 Tables 3-2

3.1.3 Columns 3-11

3.1.4 Column Domains 3-37

3.1.5 Rules 3-38

3.1.6 Defaults 3-38

3.1.7 Views 3-39

3.1.8 Integrity Constraints 3-39

3.1.9 Triggers 3-41

3.1.10 Stored Procedures 3-43

3.2 File Usage 3-110

3.2.1 Files Definitions 3-110

3.2.2 Attributes 3-110
3.2.3 Attribute Domains 3-110

4. Performance and Tuning Factors

4.1 Indexes 4-1
4.2 Segments 4-2
4.3 Named Caches 4-2

5. Database Security

5.1 Approach 5-1
5.2 Initial Users 5-2
5.3 Groups 5-2
5.4 Roles 5-2
5.5 Object Permissions 5-3

6. Scripts

6.1 Installation Scripts 6-1
6.2 De-Installation Scripts 6-1
6.3 Backup/Recovery Scripts 6-1
6.4 Miscellaneous Scripts 6-1

Figures

3-1 ERD Key 3-1
5-1 Sybase Approach to SQL Server Security 5-1

Appendix A- Entity Relationship Diagram

Abbreviations and Acronyms

This page is intentionally left blank.

1. Introduction

1.1 Purpose and Scope

The purpose of INGEST Database Design and Database Schema Specification document is to describe the database design and schema specifications implemented to support the data requirements of Release 4 Drop 4PX INGEST CSCI.

1.2 Document Organization

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

Section 4 provides a description of database performance and tuning features such as indexes, caches, and data segments.

Section 5 provides a description of the security infrastructure used and list of the users, groups, and permissions available upon initial installation.

Section 6 provides a description of database and database related scripts used for installation, de-installation, backup/recovery, and other miscellaneous functions.

This page intentionally left blank.

2. Related Documents

2.1 Applicable Documents

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

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

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

2.2 Information Documents

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

313-CD-006	Release 4 CSMS/SDPS Internal ICD for the ECS Project
609-CD-003	Release 4 Operations Tools Manual for the ECS Project
611-CD-005	Release 4 Mission Operation Procedures for the ECS Project

These documents are accessible via the EDHS homepage.

3. Data Design

3.1 Database Overview

The INGEST database implements the large majority of the persistent data requirements for the INGEST 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 (RDBMS). All components of the INGEST database are described in the section which follow, in sufficient detail to support maintenance needs.

3.1.1 Physical Data Model Entity Relationship Diagram

The Entity Relationship Diagram(ERD) presents a schematic depiction of the INGEST physical data model. The ERDs presented here for the INGEST database were produced using the S-Designor Data Architect Computer Aided Software Engineering (CASE) tool. ERDs represent the relationship between entities or database tables. On ERDs, tables are represented by rectangles and relationships are represented as arrow (see Figure 4-1).

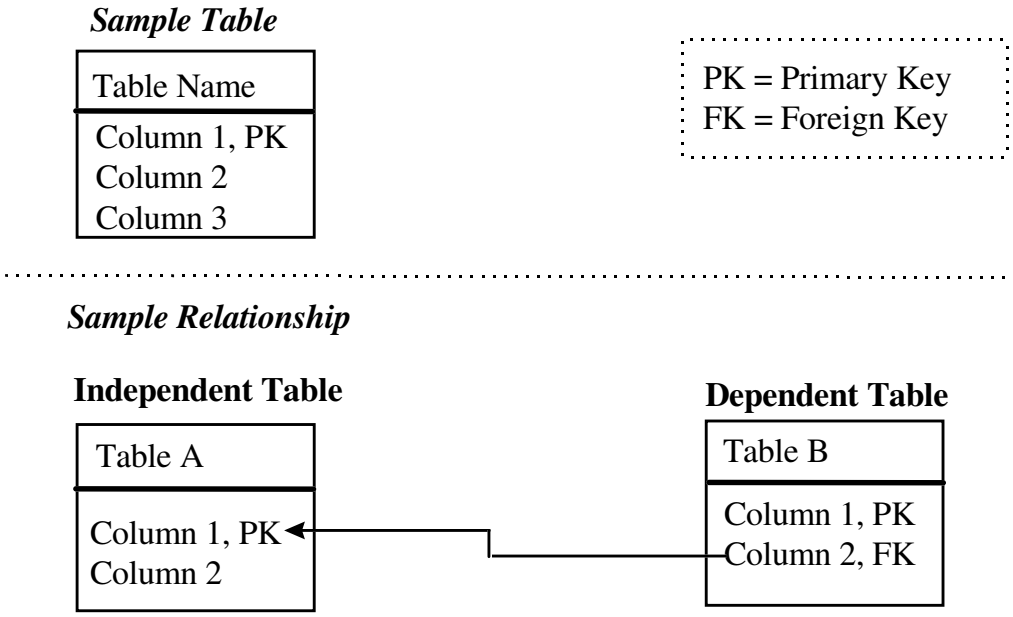

Figure 3-1. ERD Key

The ERD for the INGEST database is found in Appendix A.

3.1.2 Tables

A listing of each the tables in the INGEST database is given here. A brief definition of each of these tables follows including a listing of the columns comprising the table. The column list indicates if the column is part of the primary key for the table, that is, if the columns can be used alone or in combination with other primary key columns to uniquely identify a single row in the table. The column list also indicates whether the column is a mandatory attribute that must be included in every row.

Table Name
InCurrentDataTypeMap
InDataTypeTemplate
InEDPAdressMap
InESDTMap
InExternalDataProviderInfo
InFileTypeTemplate
InGranuleServerInfo
InMediaCheckin
InMediaType
InNextAvailableID
InRequestFileInfo
InRequestProcessData
InRequestProcessHeader
InRequestSummaryData
InRequestSummaryHeader
InSourceMCF
InSystemParameters
InValBypassPreproc
InValDataGranuleState
InValGranuleServerUR
InValIngestType
InValNotifyType
InValParameterClass
InValRequestState

Table: InCurrentDataTypeMap

Description

This table holds a data provider and the current version id.

Column List

Name	Code	Type	PK	Mandatory
DataType	DataType	varchar(32)	Yes	Yes
VersionID	VersionID	varchar(16)	No	Yes

Table: InDataTypeTemplate

Description

This table is, initially, pre-populated with current, valid Earth Science Data Types (ESDTs) that Ingest is capable of ingesting. A MSAccess97 front-end interface exists to maintain the data in the table.

Column List

Name	Code	Type	PK	Mandatory
DataType	DataType	nvarchar(32)	Yes	Yes
ExpeditedDataType	ExpeditedDataType	nvarchar(32)	No	No
FileTypeTemplateKey	FileTypeTemplateKey	nvarchar(32)	No	Yes
GranuleServerURKey	GranuleServerURKey	tinyint	No	Yes
OutputDestination	OutputDestination	char(40)	No	No
PrimaryFlag	PrimaryFlag	tinyint	No	Yes
SecondaryDataType	SecondaryDataType	nvarchar(32)	No	No
StorageMgmtKey	StorageMgmtKey	varchar(30)	No	Yes
TestDataType	TestDataType	nvarchar(32)	No	No
VersionID	VersionID	varchar(16)	Yes	Yes

Table: InESDTMap

Description

This table holds the information of the Earth Science data type (ESDT) and the Client data type. It defines what a user can do with (a Class of) Granules.

Column List

Name	Code	Type	PK	Mandatory
ClientDataType	ClientDataType	varchar(15)	No	Yes
DataDescriptor	DataDescriptor	varchar(15)	No	No
ESDT	ESDT	nvarchar(32)	No	Yes
ExternalDataProvider	ExternalDataProvider	varchar(20)	Yes	Yes

Table: InExternalDataProviderInfo

Description

This table holds the information about the data provider.

Column List

Name	Code	Type	PK	Mandatory
CDSEntry	CDSEntry	Varchar(255)	No	No
CurrentRequests	CurrentRequests	Int	No	Yes
CurrentVolume	CurrentVolume	float (48)	No	Yes
DataProvMediaStorageMgmtKey	DataProvMediaStorageMgmtKey	Varchar(30)	No	No
EmailAddress	EmailAddress	Varchar(255)	No	No
ExternalDataProvider	ExternalDataProvider	Varchar(20)	Yes	Yes
FTPPassword	FTPPassword	Binary(30)	No	No
FTPPasswordSize	FTPPasswordSize	Int	No	No
FTPUsername	FTPUsername	Varchar(10)	No	No
HTMLPassword	HTMLPassword	Binary(30)	No	No
HTMLPasswordSize	HTMLPasswordSize	Int	No	No
IngestPriority	IngestPriority	Varchar(10)	No	Yes
IngestType	IngestType	Varchar(40)	No	Yes
MaximumRequests	MaximumRequests	Int	No	Yes
NotifyFTPDirectory	NotifyFTPDirectory	Varchar(255)	No	No
NotifyFTPNode	NotifyFTPNode	Varchar(255)	No	No
NotifyFTPPassword	NotifyFTPPassword	Binary(30)	No	No
NotifyFTPPasswordSize	NotifyFTPPasswordSize	Int	No	No
NotifyFTPUsername	NotifyFTPUsername	Varchar(10)	No	No
NotifyOperator	NotifyOperator	Tinyint	No	No
NotifyType	NotifyType	Varchar(10)	No	Yes
PostTransferSizeCheck	PostTransferSizeCheck	Tinyint	No	Yes
TransferFlag	TransferFlag	Tinyint	No	Yes
UUID	UUID	char(36)	No	No
VolumeThreshold	VolumeThreshold	Float	No	Yes

Table: InFileTypeTemplate

Description

This table is, initially, pre-populated with all valid Files that make up a DataType. A MSAccess97 front-end interface exists to maintain the data in the table.

Column List

Name	Code	Type	PK	Mandatory
ArchivalFlag	ArchivalFlag	char(1)	No	No
AttributeName	AttributeName	varchar(255)	No	No

Name	Code	Type	PK	Mandatory
ExtConvFileName	ExtConvFileName	varchar(48)	No	No
ExtConvType	ExtConvType	varchar(32)	No	No
FileClass	FileClass	char(3)	No	No
FileType	FileType	filetype	Yes	Yes
FileTypeTemplateKey	FileTypeTemplateKey	datatype	Yes	Yes
InternalFileType	InternalFileType	filetype	No	No
LineDelimiter	LineDelimiter	char(1)	No	No
Maximum	Maximum	tinyint	No	Yes
MetadataSpecialization	MetadataSpecialization	varchar(48)	No	No
Minimum	Minimum	tinyint	No	Yes
ParameterClassDefault	ParameterClassDefault	varchar(8)	No	No
PVSeparator	PVSeparator	char(1)	No	No
RequiredFlag	RequiredFlag	char(1)	No	Yes
ScienceSpecialization	ScienceSpecialization	varchar(48)	No	No
SourceMCF	SourceMCF	varchar(32)	No	No
StringDelimiter	StringDelimiter	char(1)	No	No

Table: InGranuleServerInfo

Description

This table holds information about the server into which granules will be ingested.

Column List

Name	Code	Type	PK	Mandatory
GranuleServerURKey	GranuleServerURKey	tinyint	Yes	Yes
TotalGranuleThreshold	TotalGranuleThreshold	int	No	Yes
VolumeThreshold	VolumeThreshold	numeric(21,0)	No	Yes

Table: InMediaCheckin

Description

This table holds information about the different types of media on which data will come in.

Column List

Name	Code	Type	PK	Mandatory
CheckinTime	CheckinTime	datetime	No	Yes
ExternalDataProvider	ExternalDataProvider	varchar(20)	No	Yes
MediaType	MediaType	varchar(10)	No	Yes
State	State	char(15)	No	Yes
VolumeID	VolumeID	varchar(40)	Yes	Yes

Table: InMediaType

Description

This table holds the valid values of the media type available that can be ingested.

Column List

Name	Code	Type	PK	Mandatory
MediaType	MediaType	varchar(10)	Yes	Yes
StackReq	StackReq	tinyint	No	Yes
StackSlotReq	StackSlotReq	tinyint	No	Yes

Table: InNextAvailableID

Description

This table holds the next available RequestID to be given.

Column List

Name	Code	Type	PK	Mandatory
NextID	NextID	int	Yes	Yes

Table: InRequestFileInfo

Description

This table holds the file information for the ingest request.

Column List

Name	Code	Type	PK	Mandatory
CompletionTime	CompletionTime	int	No	Yes
DataGranuleID	DataGranuleID	int	Yes	Yes
FileID	FileID	varchar(255)	Yes	Yes
FileNumber	FileNumber	int	No	Yes
FileSize	FileSize	int	No	Yes
FileState	FileState	char(15)	No	No
FileStatus	FileStatus	smallint	No	No
FileType	FileType	filetype	No	No
RequestID	RequestID	requestid	Yes	Yes
SourceDirectoryID	SourceDirectoryID	varchar(255)	No	Yes
TimeStamp	TimeStamp	datetime	No	No

Table: InRequestProcessData

Description

This table holds the granule information for the ingest request, this table might have different granules for a single request.

Column List

Name	Code	Type	PK	Mandatory
DataDescriptor	DataDescriptor	varchar(60)	No	No
DataGranuleID	DataGranuleID	int	Yes	Yes
DataGranuleState	DataGranuleState	varchar(20)	No	Yes
DataGranuleVolume	DataGranuleVolume	float(48)	No	No
DataType	DataType	datatype	No	Yes
DataVersion	DataVersion	int	No	No
GranuleCompleted	GranuleCompleted	smallint	No	Yes
GranuleHandle	GranuleHandle	char(36)	No	No
NodeName	NodeName	varchar(255)	No	No
ProcessingEndDateTime	ProcessingEndDateTime	datetime	No	No
ProcessingStartDateTime	ProcessingStartDateTime	datetime	No	No
RequestID	RequestID	requestid	Yes	Yes
RetryCount	RetryCount	smallint	No	No
StagingTagID	StagingTagID	varchar(255)	No	No
TimeToArchive	TimeToArchive	int	No	No
TimeToPreprocess	TimeToPreprocess	int	No	No
TimeToXfer	TimeToXfer	int	No	No
TotalFileCount	TotalFileCount	int	No	No

Table: InRequestProcessHeader

Description

This table holds information on how the request data will be processed such as the number of granules, the number of files, the path were the DAN file exits, the External Data Provider name, priority, etc. This table holds one row per RequestID.

Column List

Name	Code	Type	PK	Mandatory
ArchPercentComplete	ArchPercentComplete	smallint	No	Yes
CDSName	CDSName	varchar(255)	No	No
DANFileName	DANFileName	varchar(255)	No	Yes
DDNDestination	DDNDestination	int	No	No
ExpirationDateTime	ExpirationDateTime	datetime	No	No
ExpiredFlag	ExpiredFlag	smallint	No	Yes
ExternalDataProvider	ExternalDataProvider	varchar(20)	No	Yes
IngestType	IngestType	varchar(40)	No	Yes
Mission	Mission	varchar(60)	No	No
PreprocPercentComplete	PreprocPercentComplete	smallint	No	Yes
ProcessingEndDateTime	ProcessingEndDateTime	Datetime	No	No
ProcessingStartDateTime	ProcessingStartDateTime	datetime	No	No
RequestID	RequestID	requestid	Yes	Yes
RequestPriority	RequestPriority	varchar(10)	No	No
RequestStateKey	RequestStateKey	tinyint	No	No
SequenceID	SequenceID	int	No	Yes
SpecProc	SpecProc	smallint	No	Yes
TotalDataVolume	TotalDataVolume	float(48)	No	No

Name	Code	Type	PK	Mandatory
TotalFileCount	TotalFileCount	int	No	No
TotalGranuleCount	TotalGranuleCount	int	No	No
TransferFlag	TransferFlag	smallint	No	Yes
UUID	UUID	char(36)	No	No
XferPercentComplete	XferPercentComplete	smallint	No	Yes

Table: InRequestSummaryData

Description

Data is copied to this table from InRequestProcessData table when processing has been completed on an ingest request for more than the MonitorTimeForCompletedRequests in the InSystemParameters table. When the data is copied, the granule information for the ingest request is deleted from the InRequestProcessData table.

Column List

Name	Code	Type	PK	Mandatory
DataGranuleID	DataGranuleID	int	Yes	Yes
DataGranuleState	DataGranuleState	varchar(20)	No	No
DataGranuleVolume	DataGranuleVolume	float	No	No
DataType	DataType	nvarchar(32)	Yes	Yes
NodeName	NodeName	varchar(255)	No	No
ProcessingEndDateTime	ProcessingEndDateTime	datetime	No	No
ProcessingStartDateTime	ProcessingStartDateTime	datetime	No	No
RequestID	RequestID	requestid	Yes	Yes
RetryCount	RetryCount	smallint	No	No
TimeToArchive	TimeToArchive	int	No	No
TimeToPreprocess	TimeToPreprocess	int	No	No
TimeToXfer	TimeToXfer	int	No	No
TotalFileCount	TotalFileCount	int	No	No

Table: InRequestSummaryHeader

Description

Data is copied to this table from InRequestProcessHeader table when processing has been completed on an ingest request for more than the MonitorTimeForCompletedRequests in the InSystemParameters table. When the data is copied the granule information for the ingest request is deleted from the InRequestProcessHeader table.

Column List

Name	Code	Type	PK	Mandatory
DANFileName	DANFileName	varchar(255)	No	Yes
ExternalDataProvider	ExternalDataProvider	varchar(20)	No	Yes
IngestType	IngestType	varchar(40)	No	Yes
Mission	Mission	varchar(60)	No	No
ProcessingEndDateTime	ProcessingEndDateTime	datetime	No	No
ProcessingStartDateTime	ProcessingStartDateTime	datetime	No	No
RequestID	RequestID	requested	Yes	Yes
RequestPriority	RequestPriority	varchar(10)	No	No
RequestStateKey	RequestStateKey	tinyint	No	No
TimeToArchive	TimeToArchive	int	No	No
TimeToPreprocess	TimeToPreprocess	int	No	No
TimeToXfer	TimeToXfer	int	No	No
TotalDataVolume	TotalDataVolume	float(48)	No	No
TotalFileCount	TotalFileCount	int	No	No
TotalGranuleCount	TotalGranuleCount	int	No	No
TotalSuccessfulGranules	TotalSuccessfulGranules	int	No	No

Table: InSourceMCF

Description

This table is, initially, pre-populated with the valid metadata types for each FileType. It is the mapping that "points" you to the (meta)data and indicates "how" to handle the data in a standard (ODL) format.

Column List

Name	Code	Type	PK	Mandatory
CSDT	CSDT	varchar(32)	No	No
DateTimeFormat	DateTimeFormat	varchar(32)	No	No
DateTimeValueFormat	DateTimeValueFormat	varchar(32)	No	No
FieldLength	FieldLength	int	No	No
FieldLocationOffset	FieldLocationOffset	int	No	No
GroupLabel	GroupLabel	varchar(32)	No	No
ParameterClass	ParameterClass	varchar(8)	No	No
ProductSpecific	ProductSpecific	varchar(48)	No	No
SourceMCF	SourceMCF	varchar(32)	Yes	Yes
SourceParameter	SourceParameter	varchar(255)	No	No
SpecialProcessing	SpecialProcessing	varchar(8)	No	No
TargetParameter	TargetParameter	varchar(255)	No	No

Table: InSystemParameters

Description

This table holds the system parameters information. This table will always have only one row at the time.

Column List

Name	Code	Type	PK	Mandatory
CommunicationRetryCount	CommunicationRetryCount	int	No	Yes
CommunicationRetryInterval	CommunicationRetryInterval	int	No	Yes
CurrentTotalRequests	CurrentTotalRequests	int	No	Yes
CurrentTotalVolume	CurrentTotalVolume	float(48)	No	Yes
IngestFTPPassword	IngestFTPPassword	binary(30)	No	No
IngestFTPPasswordSize	IngestFTPPasswordSize	int	No	No
IngestFTPUsername	IngestFTPUsername	varchar(10)	No	No
MaximumTotalRequests	MaximumTotalRequests	int	No	Yes
MaximumTotalVolume	MaximumTotalVolume	float(48)	No	Yes
MediaStorageMgmtKey	MediaStorageMgmtKey	varchar(30)	No	Yes
MonitorTimeForCompletedRequest	MonitorTimeForCompletedRequest	int	No	Yes
ScreenUpdateInterval	ScreenUpdateInterval	int	No	Yes

Table: InValDataGranuleState

Description

This table lists all the valid values for a data granule state.

Column List

Name	Code	Type	PK	Mandatory
DataGranuleState	DataGranuleState	varchar(20)	Yes	Yes

Table: InValGranuleServerUR

Description

This table lists all the valid values for the granule server UR

Column List

Name	Code	Type	PK	Mandatory
GranuleServerUR	GranuleServerUR	varchar(40)	No	Yes
GranuleServerURKey	GranuleServerURKey	tinyint	Yes	Yes

Table: InValIngestType

Description

This table lists all the valid values for an ingest type.

Column List

Name	Code	Type	PK	Mandatory
IngestType	IngestType	varchar(40)	Yes	Yes

Table: InValNotifyType

Description

This table lists all the valid values for a notify type.

Column List

Name	Code	Type	PK	Mandatory
NotifyType	NotifyType	varchar(10)	Yes	Yes

Table: InValParameterClass

Description

Pre-populated, This table lists all the valid values for a parameter class

Column List

Name	Code	Type	PK	Mandatory
ParameterClass	ParameterClass	varchar(8)	Yes	Yes

Table: InValRequestState

Description

This table lists all the valid values for a request state.

Column List

Name	Code	Type	PK	Mandatory
RequestState	RequestState	varchar(15)	No	Yes
RequestStateKey	RequestStateKey	tinyint	Yes	Yes

3.1.3 Columns

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

Column: ArchivalFlag

Description

Boolean flag to indicate if the file is needed to be archived or not.

Valid Values: Y , N

Column: ArchPercentComplete

Description

This is the percentage of archive completed for the requested data.

Column: AttributeName

Description

The name of the metadata attribute as defined in the Core Metadata Model. Valid names/attributes are either core or product specific.

Valid Values: See 311-CD-107

Column: CDSEntry

Description

(i.e.: EcCsLandsat7 Gateway)

Column: CDSName

Description

The name of a CDS component.

Column: CheckinTime

Description

This is the date and time when the media was checked in.

Column: ClientDataType

Description

This is the client data type.

Column: CommunicationRetryCount

Description

This holds the number of times that a user retries a communication.

Column: CommunicationRetryInterval

Description

This is the interval between communication retries of a user.

Column: CompletionTime

Description

This is the calculated length of time for completion for a RequestID's (Granule) File.

Column: CSDT

Description

This is the Computer Science Data Type (CSDT). (i.e., int, float, double, short, string, LittleEndian_float, LittleEndian_Int, etc.)

Valid Values: See 311-CD-107

Column: CurrentRequests

Description

Keeps a running total of the number of requests currently in the system for an External Data Provider.

Column: CurrentTotalRequests

Description

This is the total requests that are current in the system.

Column: CurrentTotalVolume

Description

This is the total volume of ingested data for all RequestIDs that are currently in the system.

Column: CurrentVolume

Description

A running total of the volume of RequestIDs currently in the system for an External Data Provider.

Column: DANFileName

Description

File name of Data Availability Notice (DAN). Contains DAN information for a single granule.

Column: DataDescriptor

Description

This is the data descriptor.

Column: DataGranuleID

Description

This is the data granule identifier.

Column: DataGranuleState

Description

This is the state of a data granule.

Valid Values:

ArchErr, Archived, Cancelled, New, PreprocErr, Preprocessed, Terminated, Transferred, XferErr.

Column: DataGranuleVolume

Description

Total data volume to be ingested for a data granule in an ingest request. The total data volume for the data granule is determined by summing the data volumes for the files comprising the data granule.

Column: DataProvMediaStorageMgmtKey

Description

This is the data provider media storage management key.

Column: DataType

Description

This holds primary ESDT short-name of a ECS data type that is handled by a particular data server. (i.e.,AM1 L0, SAGEIII L0, Radat ALT L0, Landsat7 L0R, SeaWinds,Ancillary, etc.)

Valid Values: See column ShortName in 311-CD-107

Column: DataVersion

Description

This holds the data version.

Column: DateTimeFormat

Description

This is the date time format that the file contains; required for standard handling by Science Data Server (i.e., yy-mm-dd - hh:ii:ss, yyyy-mm-ddThh:ii:ss.ssss, etc.)

Column: DateTimeValueFormat

Description

This is the value of the date time format. (i.e., %06.0f, etc.)

Column: DDNDestination

Description

The destination of a given Data Delivery Notice (DDN).

Column: EmailAddress

Description

This is the email address of the external data provider.

Column: ESDT

Description

The short name of an Earth Science Data Type. An ECS type that Science Data Server holds.

Valid Values: See column Shortname in 311-CD-107

Column: ExpeditedDataType

Description

This is the name of the expedited data type.

Column: ExpirationDateTime

Description

Date/time by which the corresponding ingest request must be completed (i.e., archive insertion complete and response returned to the External Data Provider).

Column: ExpiredFlag

Description

A boolean flag indicating whether data has passed the expiration date.

Column: ExtConvFileName

Description

This holds the provider's external file name to be converted.

Column: ExtConvType

Description

This is the external file type to be converted

Valid Values:

Script, SharedObject

Column: ExternalDataProvider

Description

This is the name of the External data provider.

Column: FieldLength

Description

This is the field length of the file.

Column: FieldLocationOffset

Description

This is the integer location offset for the field.

Column: FileClass

Description

This holds the 3-letter acronym name class of the file.,(i.e., TEX, HDF).

Column: FileID

Description

The unique identifier of the ingest file.

Column: FileNumber

Description

This is the file number indicator.

Column: FileSize

Description

This attribute represents the size of the individual file.

Column: FileState

Description

This is the file state.

Column: FileStatus

Description

Final error status for the ingest processing of a data granule.

Column: FileType

Description

This holds the valid file type of the ingest file

Valid Values:

SCIENCE
METADATA
BROWSE
NATIVE
HTML
TEXT
PDF
POSTSCRIPT
RTF
DANFILE
DOCUMENT
DATA
IMAGE1
IMAGE2
IMAGE3
IMAGE4
IMAGE5
IMAGE6
IMAGE7
IMAGE8
CALIBRATION
MSCD
PCD
ALGORITHM

Column: FileTypeTemplateKey

Description

The unique grouping of all related FileTypes.

Column: FTPPassword

Description

This is the FTP user password used by the external data provider. This is used to access the External Data Provider's system.

Column: FTPPasswordSize

Description

This is the FTP user password size used by the external data provider. This is used to access the External Data Provider's system.

Column: FTPUsername

Description

This is the FTP user name used by the external data provider. This is used to access the External Data Provider's system.

Column: GranuleCompleted

Description

This indicates if the granule was completed.

Column: GranuleHandle

Description

This is the name of the granule handle.

Column: GranuleServerUR

Description

This is the CDS name of the granule server.

Valid Values

InGranServer
InGranServer0
InGranServer1
InGranServer2
InGranServertest
InGranServer5
InGranServer6
InGranServer7
InGranServer8
InGranServersgi
InGranServer0sgi
EcInGran0
EcInGran1
EcInGran2
EcInGran3
EcInGran4
EcInGran5
EcInGran6
EcInGran7
EcInGran8
EcInGran9
EcInGran10
EcInGran11
EcInGran12
EcInGran13
EcInGran14
EcInGran0sgi
EcInGran1sgi
EcInGran2sgi
EcInGran3sgi
EcInGran4sgi
EcInGran5sgi
EcInGran

Column: GranuleServerURKey

Description

This holds the Granule ServerID that is mapped to a specific Granule Server's name.

Column: GroupLabel

Description

This is a label for a group that a file contains.

Valid Values:

BEGIN- END

START- STOP

DATASTART- DATAEND

SUBSTART- SUBSTOP

STARTDATETIME- ENDDATETIME

Column: HTMLPassword

Description

This is the HTML user password used by the external data provider.

Column: HTMLPasswordSize

Description

This is the HTML user password size used by the external data provider.

Column: IngestFTPPassword

Description

This is the ingest FTP user password.

Column: IngestFTPPasswordSize

Description

This is the ingest FTP user password size.

Column: IngestFTPUsername

Description

This is the ingest FTP user name.

Column: IngestPriority

Description

This is the ingest priority assigned to a request.

Valid Values:

Normal,

High,

Low

Column: IngestType

Description

This is the valid type of Ingest for processing a RequestID.

Valid Values:

Auto

Interactive

Media

Polling_w/DR
Polling_wo/DR

Column: InternalFileType

Description

This attribute holds the type of internal file.

Valid Values:

METADATA,

SCIENCE,

BROWSE

Column: LineDelimiter

Description

This attribute wil define the symbol used to indicate the end of a parameter-value metadata statement.

Column: Maximum

Description

The minimum fields in the file.

Column: MaximumRequests

Description

These are the maximum requests available for the external data provider.

Column: MaximumTotalRequests

Description

These are the maximum requests that the system can hold at a time.

Column: MaximumTotalVolume

Description

This is the maximum volume of data that the system can hold at a time.

Column: MediaStorageMgmtKey

Description

A System's Parameter, this is the valid storage (<HWCI>_<mode>) management key of where the data will be stored (i.e.: HWCi2_TS1)

Column: MediaType

Description

This is the name of the media type.

Valid Values:

4mm Tape
8mm Tape
D3 Tape

Column: MetadataSpecialization

Description

This attribute holds the specialization of the metadata in the file.

Valid Values:

ODLMetadata,

EDOSMetadata

Column: Minimum

Description

The maximum fields in the file.

Column: Mission

Description

This is the name of the mission from which the data had come. (i.e., AM-1)

Column: MonitorTimeForCompletedRequest

Description

This is the length of time a request is held before the request information can be moved to the archive (InRequestSummary) tables.

Column: NextID

Description

Automatically generated in a sequential order, this provides the unique RequestID.

Column: NodeName

Description

This holds the path were the data granule exit.

Column: NotifyFTPDirectory

Description

This is the directory where the notify FTP exists.

Column: NotifyFTPNode

Description

This is the path where the notify FTP exists.

Column: NotifyFTPPassword

Description

This is the notify FTP password.

Column: NotifyFTPPasswordSize

Description

This is the notify FTP password size.

Column: NotifyFTPUsername

Description

This the notify FTP user name.

Column: NotifyOperator

Description

This is number for the operator to be notified.

Column: NotifyType

Description

This is the valid Notify Type.

Valid Values:

Buffer
EDOS
PVL

Column: OutputDestination

Description

This is the name of a subsystem where the output data will be put in. (i.e., SDSRV)

Column: ParameterClass

Description

This is the Parameter Class of the file .

Valid Values:

OBJ
PV
TOOLKIT

Column: ParameterClassDefault

Description

This is the default for a parameter class.

Column: PostTransferSizeCheck

Description

The size of the file after transfer is complete.

Column: PreprocPercentComplete

Description

This is the percentage of pre processing completed.

Column: PrimaryFlag

Description

This is the flag for the primary data type.

Column: ProcessingEndDateTime

Description

This is the processing end date and time for an ingest of a data granule.

Column: ProcessingStartDateTime

Description

This is the processing start date and time for the ingest of a data granule.

Column: ProductSpecific

Description

This is the granule's product specification.

Column: PVSeparator

Description

This attribute will define the separator symbol used in between the parameter-values (i.e., =).

Column: RequestID

Description

This is the request identifier automatically generated from the InNextAvailableID table.

Column: RequestPriority

Description

The information that determines the order in which an ingest request will be processed relative to other ingest request waiting to be processed. This priority is provided by the InExternalDataProviderInfo for each external data provider.

Valid Values:

Normal,
High,
Low

Column: RequestState

Description

This is the set of valid state for a Request.

Valid Values:

Active

Suspending

Suspended

SettingPriority

Canceling

Resuming

Successful

Cancelled

Failed

Partial_Failure

Terminated

Column: RequestStateKey

Description

This is the valid identification integer for the request state.

Valid Values:

1 - Active

2 - Suspending

3 - Suspended

4 - SettingPriority

5 - Canceling

6 - Resuming,

7 - Successful

8 - Cancelled

9 - Failed

10 - Partial_Failure

11 - Terminated

Column: RequiredFlag

Description

This is a flag that is set if the file is to be required for a granule.

Column: RetryCount

Description

This holds the retry counts for the data granule.

Column: ScienceSpecialization

Description

This attribute holds the specialization of the science data in the file.

Valid Values:

See ECSTopicKeyword, ECSTermKeyword, ECSVariableKeyword, ECSParameterKeyword in 311-CD-107-001

Column: ScreenUpdateInterval

Description

A System's Parameter, this is the length of time between updates to the GUI screens.

Column: SecondaryDataType

Description

This holds the secondary ESDT short-name of a DataType that is handled by a particular Data Server.

Valid Values

See Shortname in 311-CD-107

Column: SequenceID

Description

This is the sequence identifier of the request.

Column: ServerType

Description

Specifies either Science Data Server (Sdsrv) or Document Server (Dcsrv) to connect to using the SdsrvUR.

Column: SourceDirectoryID

Description

The source directories where the files can be found.

Column: SourceMCF

Description

This is the acronym name of the source Metadata Configuration File (MCF). (i.e., AST_L1,L7OR1,TRMMEPH1,ODL11,etc.)

Column: SourceParameter

Description

This is how the metadata type is named when it comes in. (mapping).

Column: SpecialProcessing

Description

This row is to specify if the request to be processed is special.

Column: SpecProc

Description

This is the number of special processes.

Column: StackReq

Description

This holds the stack required for the media type.

Column: StackSlotReq

Description

This holds the stack slot required for the media type.

Column: StagingTagID

Description

This is an identifier to indicate the staging of the data granule.

Column: State

Description

Current state of External Data Provider's request processing (i.e.: Active, Suspended)

Column: StorageMgmtKey

Description

This holds the valid Storage Management Key (<HWCI>) of where the data will be stored (i.e.: DRP1, SGI, HWCI1, ICL1).

Column: StringDelimiter

Description

This attribute will define the symbol used to indicate the end of a parameter-value metadata string.

Column: TargetParameter

Description

This is how the metadata type is to be named when it goes out (mapping).

Column: TestDataType

Description

This is the name of the test data type.

Column: TimeStamp

Description

This is the current date and time of when a File's State is completed.

Column: TimeToArchive

Description

Time from submit of archive request to Data Server to receipt of completion status (success or fail).

Column: TimeToArchive

Description

This is the date and time to archive the data.

Column: TimeToPreprocess

Description

Time from start of preprocessing of granule to time of completion (success or fail) of preprocessing.

Column: TimeToXfer

Description

Time from start of transfer for 1st file in granule to time of receipt of status (success or fail) for last file in granule.

Column: TotalDataVolume

Description

This is the total data volume of the granule.

Column: TotalFileCount

Description

This is the total number of files for the request.

Column: TotalGranuleCount

Description

This is the total number of granules for the request.

Column: TotalGranuleThreshold

Description

This is the total number of data granules that a server can hold.

Column: TotalSuccessfulGranules

Description

This is the total number of data granule successful ingested.

Column: TransferFlag

Description

This is a flag to indicate the transfer of data.

Column: UUID

Description

The user id.

Column: VersionID

Description

This holds the version identifier of the data.

Column: VolumeID

Description

This is the volume identifier for the media check in.

Column: VolumeThreshold

Description

This is the volume limit available for the External Data Provider.

Column: XferPercentComplete

Description

This is the percentage transfer of granule complete

3.1.4 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 given column. Sybase domains are, in effect, user-defined data types. The domains defined in the INGEST database are given here as well as an indication of the columns to which they apply.

Domain: boolean

Reference List

Table Code	Column Name	Column Code	Label
InFileTypeTemplate	ArchivalFlag	ArchivalFlag	
InFileTypeTemplate	RequiredFlag	RequiredFlag	

Domain: datatype

Reference List

Table Code	Column Name	Column Code	Label
InDataTypeTemplate	DataType	DataType	

Table Code	Column Name	Column Code	Label
InRequestProcessData	DataType	DataType	
InRequestSummaryData	DataType	DataType	
InESDTMap	ESDT	ESDT	
InDataTypeTemplate	ExpeditedDataType	ExpeditedDataType	
InDataTypeTemplate	FileTypeTemplateKey	FileTypeTemplateKey	
InFileTypeTemplate	FileTypeTemplateKey	FileTypeTemplateKey	
InDataTypeTemplate	SecondaryDataType	SecondaryDataType	
InDataTypeTemplate	TestDataType	TestDataType	

Domain: filetype

Reference List

Table Code	Column Name	Column Code	Label
InFileTypeTemplate	FileType	FileType	
InRequestFileInfo	FileType	FileType	
InFileTypeTemplate	InternalFileType	InternalFileType	

Domain: requestid

Reference List

Table Code	Column Name	Column Code	Label
InNextAvailableID	NextID	NextID	
InRequestFileInfo	RequestID	RequestID	
InRequestProcessData	RequestID	RequestID	
InRequestProcessHeader	RequestID	RequestID	
InRequestSummaryData	RequestID	RequestID	
InRequestSummaryHeader	RequestID	RequestID	

Domain: userref

3.1.5 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 INGEST database.

3.1.6 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 for the INGEST database.

3.1.7 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 Sybase for the INGEST database.

3.1.8 Integrity Constraints

Sybase 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. All declarative integrity constraints defined in the INGEST database are described herein.

Dependencies on Table: InCurrentDataTypeMap

Reference by List

Referenced by	Primary Key	Foreign Key
InDataTypeTemplate	DataType, VersionID	

Dependencies on Table: InDataTypeTemplate

Reference by List

Referenced by	Primary Key	Foreign Key
InESDTMap	DataType	ESDT
InRequestProcessData	DataType	DataType

Dependencies on Table: InExternalDataProviderInfo

Reference by List

Referenced by	Primary Key	Foreign Key
InESDTMap	ExternalDataProvider	ExternalDataProvider
InMediaCheckin	ExternalDataProvider	ExternalDataProvider
InRequestProcessHeader	ExternalDataProvider	ExternalDataProvider

Dependencies on Table: InMediaType

Reference by List

Referenced by	Primary Key	Foreign Key
InMediaCheckin	MediaType	MediaType

Dependencies on Table: InRequestProcessHeader

Reference by List

Referenced by	Primary Key	Foreign Key
InRequestProcessData	RequestID	RequestID

Dependencies on Table: InValDataGranuleState

Reference by List

Referenced by	Primary Key	Foreign Key
InRequestProcessData	DataGranuleState	DataGranuleState

Dependencies on Table: InValGranuleServerUR

Reference by List

Referenced by	Primary Key	Foreign Key
InGranuleServerInfo	GranuleServerURKey	GranuleServerURKey

Dependencies on Table: InValIngestType

Reference by List

Referenced by	Primary Key	Foreign Key
InExternalDataProviderInfo	IngestType	IngestType
InRequestProcessHeader	IngestType	IngestType

Dependencies on Table: InValNotifyType

Reference by List

Referenced by	Primary Key	Foreign Key
InExternalDataProviderInfo	NotifyType	NotifyType

Dependencies on Table: InValParameterClass

Reference by List

Referenced by	Primary Key	Foreign Key
InFileTypeTemplate	ParameterClass	ParameterClassDefault
InSourceMCF	ParameterClass	ParameterClass

Dependencies on Table: InValRequestState

Reference by List

Referenced by	Primary Key	Foreign Key
InRequestProcessHeader	RequestStateKey	RequestStateKey

3.1.9 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 by Sybase whenever a row is inserted, updated, or deleted from a given table. Sybase allows the definition of insert, update, and delete triggers for each table. A listing of each the triggers in the INGEST database is given here. A brief definition of each of these triggers follows.

Table Code	Trigger	Description
InFileTypeTemplate	InFTTInsertTrg	Referential Integrity Enforcement
InFileTypeTemplate	InFTTInsertTrg	Referential Integrity Enforcement
InRequestFileInfo	InRFIInsertTrg	Referential Integrity Enforcement
InSourceMCF	InSMCFDeleteTrg	Referential Integrity Enforcement

Trigger: InRFIInsertTrg

Trigger Code

```
--**************************************************************************
-- BEGIN PROLOG
--
-- TRIGGER NAME:    InRFIInsertTrg
--
-- DESCRIPTION:     Insert trigger on InRequestFileInfo table.
--
-- DATE CREATED:    1997
-- CREATED BY:      Nishi
--
-- TABLES ACCESSED:  InRequestProcessData
--                   InRequestFileInfo
--
-- RETURNS:         Status (Success = 0 )
--
--**************************************************************************
CREATE TRIGGER InRFIInsertTrg ON InRequestFileInfo
FOR Insert AS

BEGIN
  DECLARE  @status              int

/* If no rows affected, exit trigger */

  IF @@rowcount = 0
    RETURN

/* Initialize @status variable to 0 to represent success.  Any other */
/* value will represent a failure/error or warning status. */

  SELECT @status = 0
```

```
/* Trigger fired by insert */

  IF((SELECT COUNT(inserted.RequestID)
    FROM inserted, InRequestProcessData
    WHERE inserted.RequestID = InRequestProcessData.RequestID
    AND inserted.DataGranuleID = InRequestProcessData.DataGranuleID)
            = 0 )

      BEGIN
        SELECT @status = 1
        ROLLBACK TRANSACTION
        RAISERROR 73300 "InRFIInsertTrg:  Invalid Request ID and DataGranuleID.  A record must exist
in the InRequestProcessData table with that Request ID and DataGranuleID."
        RETURN
      END
END
Go
```

Trigger: InSMCFDeleteTrg

```
Trigger Code
--//////////////////////////////////////////////////////////
--  BEGIN PROLOG
--
--  TRIGGER NAME:     InSMCFDeleteTrg
--
--  DESCRIPTION:      Delete trigger on InSourceMCF table.
--
--  DATE CREATED:     1197
--  CREATED BY:       Nishi
--
--  TABLES ACCESSED:  InFileTypeTemplate
--                    InSourceMCF
--
--  RETURNS:          (Status (Success = 0)
--
--**********************************************************************
CREATE TRIGGER InSMCFDeleteTrg ON InSourceMCF
FOR Delete AS

BEGIN
  DECLARE  @status            int

/* If no rows affected, exit trigger */
  IF @@rowcount = 0
    RETURN

/* Initialize @status variable to 0 to represent success.  Any other */
/* value will represent a failure/error or warning status. */
```

```
   SELECT @status = 0

/* Trigger fired by delete */

   IF((SELECT COUNT(*) FROM deleted, InFileTypeTemplate ftt
      WHERE deleted.SourceMCF = ftt.SourceMCF) != 0 )

      BEGIN
         SELECT @status = 1
         ROLLBACK TRANSACTION
         RAISERROR 74400 "InSMCFDeleteTrg:  Rollback Transaction:  Referential Integrity violated.
Cannot delete the InSourceMCF record because the SourceMCF is referenced by the InFileTypeTemplate
table."
         RETURN
      END
END
go
```

Trigger: InFTTInsertTrg

Trigger Code

```
CREATE TRIGGER InFTTInsertTrg ON InFileTypeTemplate FOR INSERT, UPDATE
AS
   print "Insert/Update Trigger Kicking in ..."
   IF (select SourceMCF from inserted)="NULL"
   BEGIN
     print "SPECIAL CASE : NULL SourceMCF in record. Allowed."
     return
   END
   IF( (SELECT COUNT(*)
        FROM   inserted, InSourceMCF
        WHERE  inserted.SourceMCF     = InSourceMCF.SourceMCF)
           =0)

              BEGIN
              ROLLBACK TRIGGER WITH RAISERROR 30000
              END
go
```

Trigger: InRFIInsertTrg

3.1.10 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 complied 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 INGEST database is given here. A brief definition of each of these stored procedures follows.

Procedure List

Name	Description
InCDTInsert	Inserts records into the InCurrentDataTypeMap table.
InCDTSelectByType	Selects datatypes from InCurrentDataTypeMap table.
InDTTUpdGSURkey	Update Value of Granule Server UR Key
InEDPIUpdate	Update all columns of the InExternalDataProviderInfo table for one ExternalDataProvider
InEDPIUpdCurr	Update the CurrentVolume and CurrentRequests columns of the InExternalDataProviderInfo table for one ExternalDataProvider and the CurrentTotalVolume and CurrentTotalRequests columns of the InSystemParameters table.
InGetID	To get the next available number for a new ingest request and increment the number for the next request
InGSIUpdate	Update the values of VolumeThreshold and TotalGranuleThreshold in the InGranuleServerInfo table.
InRFIInsert	Insert a record into the InRequestFileInfo table.
InRFIUpdGranuleStatus	Update the value of InRequestFileInfo.FileStatus for all files for a given Granule.
InRFIUpdState	Update the value of InRequestFileInfo.FileState
InRFIUpdStateStatus	Update the value of InRequestFileInfo.FileState and InRequestFileInfo.FileStatus
InRFIUpdStatus	Update the value of InRequestFileInfo.FileStatus
InRPDInsert	Insert a record into the InRequestProcessData table.
InRPDRecov	To determine if a granule was completed and if not, to set its RetryCount and/or the FileStatus for the associated files appropriately.
InRPDSetGranComp	Update the value of InRequestProcessData.GranuleCompleted to 1
InRPDUpdEndTime	Update the value of InRequestProcessData.ProcessingEndDateTime
InRPDUpdHandle	Update the value of InRequestProcessData.ProcessingEndDateTime
InRPDUpdStaging	Update the value of InRequestProcessData.StagingTagID.
InRPDUpdStagReq	Update the value of InRequestProcessData.StagingTagID.
InRPDUpdStartTime	Record ProcessingStartDate and Time for a given Request.
InRPDUpdStateArch	Update the value of InRequestProcessData.DataGranuleStateand the appropriate time based on the value the state.
InRPDUpdStateDefault	Update the value of InRequestProcessData.DataGranuleState and the appropriate time based on the value the state.

Name	Description
InCDTInsert	Inserts records into the InCurrentDataTypeMap table.
InRPDUpdStatePreproc	Update request state to “pre-processing”.
InRPDUpdStateXfer	Update request state to “transferring”.
InRPHColdStart	Deletes all records from the InRequestProcessHeader, InRequestProcessData, and InRequestFileInfo tables.
InRPHDelCompReq	Delete and archive completed requests.
InRPHInsert	Insert a record into the InRequestProcessHeader table.
InRPHUpdDDNDest	Update Data Delivery Notice (DDN) destination information for a given Request.
InRPHUpdEndTime	Record Request .Processing End Date and Time
InRPHUpdExpired	Set status of Request to “expired”.
InRPHUpdFailReq	Set status of Request to “failed”.
InRPHUpdPercentCompleteArch	Provide update to Archival percentage complete status
InRPHUpdPercentCompletePre	Provide update “pre-processing” percentage complete status.
InRPHUpdPercentCompleteXfer	Provide update “transfer” percentage complete status.
InRPHUpdPriority	Change the priority of a given Request.
InRPHUpdPriorityDataProv	Change the priority of all requests received from a given Etxernal Data Provider.
InRPHUpdState	Set the Request state for a given Request as indicated.
InRPHUpdStateChng	Set the Request state for a given Request as indicated.
InRPHWarmStart	Removes all Request from the database that have an invalid request state.
InSPUpdAll	Updates system parameters found in the InSystemParameters table.

Procedure: InCDTInsert

Code

```
--
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      InCDTInsert
-- DESCRIPTION:         Insert a record into the InCurrentDataTypeMap table.
-- TABLES ACCESSED:     InCurrentDataTypeMap
-- CREATED:             8/31/98/Trina Scurry
-- RETURNS:             Status (Success = 0)
-- INPUTS:      PARAMETER               DOMAIN
-- -------      ---------               ------
--              DataType                varchar(32)
--              VersionID               varchar(16)

-- OUTPUTS:     PARAMETER               DOMAIN
-- --------     ---------               ------
--              NONE
-- --//////////////////////////////////////////////////////
CREATE PROCEDURE InCDTInsert
```

```
    @DataType           varchar(32),
    @VersionID          varchar(16)
AS
BEGIN
    DECLARE @status int
    SELECT @status = 0

    BEGIN TRANSACTION
    INSERT InCurrentDataTypeMap(
        DataType,
        VersionID
    )
    VALUES (
        @DataType,
        @VersionID
    )
    SELECT @status = @@error

    IF (@status !=0)
      BEGIN
        SELECT @status = 1
        ROLLBACK TRANSACTION
        RAISERROR 71210 "InCDTInsert:  Insert of datatype [%1!], [%2!] failed.", @DataType,
@VersionID
      END
      ELSE
    BEGIN
      COMMIT TRANSACTION
      RETURN(@status)
    END
END
go
```

Procedure: InCDTSelectByType

Code

```
-- --////////////////////////////////////////////////////////*
-- BEGIN PROLOG
--
-- PROCEDURE NAME:     InCDTSelectByType
-- DESCRIPTION:        Selects data types from the InCurrentDataTypeMap table.
-- TABLES ACCESSED:    InCurrentDataTypeMap
-- CREATED:            8/31/98/Trina Scurry
-- RETURNS:            Status (Success = 0)
-- INPUTS:     PARAMETER           DOMAIN
-- -------     ---------           ------
--             DataType            datatype,
--             VersionID           varchar(16)
-- OUTPUTS:    PARAMETER           DOMAIN
-- --------    ---------           ------
--             NONE
-- --////////////////////////////////////////////////////////
```

```
CREATE PROCEDURE InCDTSelectByType
      @DataType            datatype
AS
BEGIN
      DECLARE @status int

      BEGIN TRANSACTION
      SELECT * FROM InCurrentDataTypeMap
            WHERE DataType = @DataType

      SELECT @status = @@error

   IF (@status !=0)
      BEGIN
      ROLLBACK TRANSACTION
      RETURN (@status)
   END
   ELSE
   BEGIN
         COMMIT TRANSACTION
   END

   RETURN(@status)

END
Go
```

Procedure: InDTTUpdGSURkey

Code

```
-- --////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME: ..... InDTTUpdGSURkey
--
-- DESCRIPTION:                Update the value of
--                InDataTypeTemplate.GranuleServerURKey
--
-- TABLES ACCESSED:InDataTypeTemplate, InValGranuleServerUR
--
-- RETURNS:            Status (Success = 0)
--
-- INPUTS:      PARAMETER            DOMAIN
-- -------     ---------                  ------
--
--                DataType                datatype
--                GranuleServerUR                varchar(40)
--
--
-- OUTPUTS:   PARAMETER            DOMAIN
-- --------    ---------                  ------
```

```
--
--                  NONE
--
--////////////////////////////////////////////////////

CREATE PROCEDURE InDTTUpdGSURkey
        @DataType           datatype,
        @GranuleServerUR    varchar(40)
AS
BEGIN
        DECLARE @status             int
      DECLARE @rows               int
      DECLARE @GranuleServerURKey tinyint

      -- First make sure that the GranuleServerUR is a valid one.
      SELECT @GranuleServerURKey = GranuleServerURKey
      FROM   InValGranuleServerUR
      WHERE  GranuleServerUR = @GranuleServerUR

      select @rows = @@rowcount
      select @status = @@error

      IF (@rows = 0)
      BEGIN
            select @status = 71100
            RAISERROR 71100
            "GranuleServerUR %1! not in InValGranuleServerUR table",
            @GranuleServerUR
            RETURN(@status)
      END

         BEGIN TRANSACTION
         UPDATE         InDataTypeTemplate
         SET    GranuleServerURKey = @GranuleServerURKey
         WHERE          DataType         = @DataType

         SELECT @status = @@error

         IF (@status != 0)
         BEGIN
                  ROLLBACK TRANSACTION
                  RETURN (@status)
         END
         ELSE
         BEGIN
                  COMMIT TRANSACTION
                  RETURN(@status)
         END

END
go
```

Procedure: InEDPIUpdate

Code

```
--//////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    InEDPIUpdate
--
-- DESCRIPTION:       Update all columns of the InExternalDataProviderInfo
--                    table for one ExternalDataProvider.
--
-- TABLES ACCESSED:   InExternalDataProviderInfo
--
-- RETURNS:           Status (Success = 0)
--
-- INPUTS:      PARAMETER                    DOMAIN
-- -------      ---------                    ------
--
--             ExternalDataProvider          varchar(20),
--             IngestType                    varchar(40),
--             IngestPriority                varchar(10),
--             VolumeThreshold               float(48),
--             MaximumRequests               int,
--             PostTransferSizeCheck         tinyint,
--             NotifyOperator                tinyint,
--             TransferFlag                  tinyint,
--             FTPUsername                   varchar(10),
--             EmailAddress                  varchar(255),
--             CDSEntry                      varchar(255),
--             UUID                          char(36),
--             NotifyType                    varchar(10),
--             NotifyFTPNode                 varchar(255),
--             NotifyFTPDirectory            varchar(255),
--             NotifyFTPUsername             varchar(10),
--             NotifyFTPPassword             varbinary(30),
--             NotifyFTPPasswordSize         int,
--                     HTMLPassword                  varbinary(30),
--                     HTMLPasswordSize              int,
--             FTPPassword                   varbinary(30),
--                     FTPPasswordSize               int
----
--
-- OUTPUTS:     PARAMETER                    DOMAIN
-- --------     ---------                    ------
--
--             NONE
--
-- --//////////////////////////////////////////////////////

CREATE PROCEDURE InEDPIUpdate
      @ExternalDataProvider          varchar(20),
      @IngestType                    varchar(40),
      @IngestPriority                varchar(10),
      @VolumeThreshold               float(48),
```

```
    @MaximumRequests            int,
    @PostTransferSizeCheck      tinyint,
    @NotifyOperator             tinyint,
    @TransferFlag               tinyint,
    @FTPUsername                varchar(10),
    @EmailAddress               varchar(255),
    @CDSEntry                   varchar(255),
    @UUID                       char(36),
    @NotifyType                 varchar(10),
    @NotifyFTPNode              varchar(255),
    @NotifyFTPDirectory         varchar(255),
    @NotifyFTPUsername          varchar(10),
    @NotifyFTPPassword          varbinary(30),
    @NotifyFTPPasswordSize      int,
      @HTMLPassword                     varbinary(30),
      @HTMLPasswordSize             int,
    @FTPPassword                varbinary(30),
      @FTPPasswordSize              int


AS
BEGIN
    DECLARE @status int

    BEGIN TRANSACTION
      UPDATE        InExternalDataProviderInfo
    SET IngestType              = @IngestType,
        IngestPriority          = @IngestPriority,
      VolumeThreshold           = @VolumeThreshold,
      MaximumRequests           = @MaximumRequests,
      FTPUsername               = @FTPUsername,
      PostTransferSizeCheck     = @PostTransferSizeCheck,
      NotifyOperator            = @NotifyOperator,
      TransferFlag              = @TransferFlag,
      EmailAddress              = @EmailAddress,
      CDSEntry                  = @CDSEntry,
      UUID                      = @UUID,
      NotifyType                = @NotifyType,
      NotifyFTPNode             = @NotifyFTPNode,
      NotifyFTPDirectory        = @NotifyFTPDirectory,
      NotifyFTPUsername         = @NotifyFTPUsername,
      NotifyFTPPassword         = @NotifyFTPPassword,
      NotifyFTPPasswordSize     = @NotifyFTPPasswordSize,
        FTPPassword                     = @FTPPassword,
        FTPPasswordSize         = @FTPPasswordSize,
        HTMLPassword            = @HTMLPassword,
        HTMLPasswordSize        = @HTMLPasswordSize

    WHERE ExternalDataProvider    = @ExternalDataProvider

        SELECT @status = @@error

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

```
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
                RETURN(@status)
        END
END
go
```

Procedure: InEDPIUpdCurr

Code

```
-- --//////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    InEDPIUpdCurr
--
-- DESCRIPTION:       Update the CurrentVolume and CurrentRequests
--                    columns of the InExternalDataProviderInfo
--                    table for one ExternalDataProvider and the
--                    CurrentTotalVolume and CurrentTotalRequests columns of
--                    the InSystemParameters table.
--
-- TABLES ACCESSED:   InExternalDataProviderInfo, InSystemParameters
--
-- RETURNS:           Status (Success = 0)
--
-- INPUTS:      PARAMETER                   DOMAIN
-- -------      ---------                   ------
--
--              ExternalDataProvider         varchar(20),
--              VolumeChange                float(48),
--              RequestChange               int
--
--
-- OUTPUTS:     PARAMETER                   DOMAIN
-- --------     ---------                   ------
--
--              NONE
--
--//////////////////////////////////////////////////////

CREATE PROCEDURE InEDPIUpdCurr
        @ExternalDataProvider         varchar(20),
        @VolumeChange                 float(48),
        @RequestChange                int
AS
BEGIN
        DECLARE @status int
        DECLARE @rows   int

        BEGIN TRANSACTION
            UPDATE          InExternalDataProviderInfo
```

```
    SET         CurrentVolume           = CurrentVolume + @VolumeChange,
        CurrentRequests         = CurrentRequests + @RequestChange
    WHERE   ExternalDataProvider    = @ExternalDataProvider

    SELECT @rows = @@rowcount
        SELECT @status = @@error

    IF (@status != 0)
    BEGIN
        ROLLBACK TRANSACTION
        RETURN(@status)
    END

    IF (@rows = 0)
    BEGIN
        select @status = 72000
        RAISERROR 72000 "External Data Provider %1! not in InExternalDataProviderInfo table",
@ExternalDataProvider
        ROLLBACK TRANSACTION
        RETURN(@status)
    END

        UPDATE          InSystemParameters
    SET         CurrentTotalVolume  = CurrentTotalVolume + @VolumeChange,
        CurrentTotalRequests    = CurrentTotalRequests + @RequestChange

        SELECT @status = @@error

    IF (@status != 0)
    BEGIN
        ROLLBACK TRANSACTION
        RETURN(@status)
    END

    COMMIT TRANSACTION
        RETURN(@status)

END
go
```

Procedure: InGetID

Code

```
-- --////////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:InGetID
--
-- DESCRIPTION:  To get the next availble number for a new ingest request
--                  and increment the number for the next request.
--
-- TABLES ACCESSED:    InNextAvailableID
--
--
```

```
-- RETURNS:            Status (Success = 0)
--
-- INPUTS:      PARAMETER              DOMAIN
-- -------      ---------              ------
--
--                    NONE
--
--
-- OUTPUTS:     PARAMETER              DOMAIN
-- --------     ---------              ------
--
--                    RequestID                    int
--
--////////////////////////////////////////////////////////

CREATE PROCEDURE InGetID
@IDnumber int = 0 output
AS
BEGIN
        DECLARE @status int

        -- Retrieve the next available id number for the specified table

        BEGIN TRANSACTION

        -- Lock the table by performing an update
        UPDATE InNextAvailableID
        SET    NextID =  NextID

        SELECT @IDnumber    = NextID
        FROM   InNextAvailableID

        -- Increment and update the id number for the specified table
        UPDATE InNextAvailableID
        SET    NextID =  NextID + 1

        SELECT @status = @@error

        IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
                RETURN(@status)
        END
END
go
```

Procedure: InGSIUpdate

Code

```
--//////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    InGSIUpdate
--
-- DESCRIPTION:       Update the values of VolumeThreshold and
--                    TotalGranuleThreshold in the InGranuleServerInfo
--                    table.
--
-- TABLES ACCESSED:   InGranuleServerInfo, InGranuleServerUR
--
-- RETURNS:           Status (Success = 0)
--
-- INPUTS:      PARAMETER                    DOMAIN
-- -------      ---------                    ------
--
--              GranuleServerUR              varchar(40),
--              VolumeThreshold              numeric(21, 0),
--              TotalGranuleThreshold        int
--
--
-- OUTPUTS:     PARAMETER                    DOMAIN
-- --------     ---------                    ------
--
--              NONE
----//////////////////////////////////////////////////////

CREATE PROCEDURE InGSIUpdate
      @GranuleServerUR              varchar(40),
      @VolumeThreshold              numeric(21, 0),
      @TotalGranuleThreshold        int
AS
BEGIN
      DECLARE @status int

      BEGIN TRANSACTION
         UPDATE          InGranuleServerInfo
      SET VolumeThreshold              = @VolumeThreshold,
         TotalGranuleThreshold         = @TotalGranuleThreshold
      FROM InGranuleServerInfo t1, InValGranuleServerUR t2
      WHERE t2.GranuleServerUR    = @GranuleServerUR
      AND   t1.GranuleServerURKey = t2.GranuleServerURKey

         SELECT @status = @@error

      IF (@status != 0)
      BEGIN
            ROLLBACK TRANSACTION
            RETURN (@status)
      END
      ELSE
```

```
    BEGIN
        COMMIT TRANSACTION
        RETURN(@status)
    END

END
go
```

Procedure: InRFIInsert

Code

```
-- *--//////////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:InRFIInsert
--
-- DESCRIPTION:                 Insert a record into the InRequestFileInfo table.
--
-- TABLES ACCESSED:InRequestFileInfo
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:      PARAMETER               DOMAIN
-- -------      ---------               ------
--
--                   RequestID               requestid,
--                   DataGranuleID           int,
--                   FileID                  varchar(255),
--                   FileSize        int,
--           FileType            filetype,
--                   SourceDirectoryID       varchar(255)
--
-- OUTPUTS:     PARAMETER               DOMAIN
-- --------     ---------               ------
--
--                   NONE
--
--//////////////////////////////////////////////////////////

CREATE PROCEDURE InRFIInsert
        @RequestID              requestid,
        @DataGranuleID                  int,
        @FileID                         varchar(255),
        @FileSize               int,
    @FileType           filetype,
        @SourceDirectoryID      varchar(255),
        @FileNumber             int
AS
BEGIN
    DECLARE @status int

        BEGIN TRANSACTION

        INSERT InRequestFileInfo (
```

```
                RequestID,
                DataGranuleID,
                FileID,
                FileSize,
          FileType,
                SourceDirectoryID,
                FileNumber
                )
        VALUES (
                @RequestID,
                @DataGranuleID,
                @FileID,
                @FileSize,
          @FileType,
                @SourceDirectoryID,
                @FileNumber
        )

        SELECT @status = @@error
        IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN(@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
                RETURN (@status)
        END
END
go
```

Procedure: InRFIUpdGranuleStatus

Code

```
-- ***--//////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:InRFIUpdGranuleStatus
--
-- DESCRIPTION:                 Update the value of InRequestFileInfo.FileStatus
--                 for all files for a given Granule.
--
-- TABLES ACCESSED:InRequestFileInfo
--
-- RETURNS:              Status (Success = 0)
--
-- INPUTS:       PARAMETER              DOMAIN
-- -------      ---------                  ------
--
--                 RequestID            int,
--                 DataGranuleID          int,
--                 FileStatus           smallint
--
```

```
--
-- OUTPUTS:    PARAMETER               DOMAIN
-- --------     ---------               ------
--
--                  NONE
--
-- --////////////////////////////////////////////////////
CREATE PROCEDURE InRFIUpdGranuleStatus
        @RequestID                  int,
        @DataGranuleID              int,
        @FileStatus                 smallint
AS
BEGIN
        DECLARE @status int

        BEGIN TRANSACTION

        UPDATE          InRequestFileInfo
        SET     FileStatus      = @FileStatus
        WHERE           RequestID       = @RequestID
        AND     DataGranuleID   = @DataGranuleID

        SELECT @status = @@error

        IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
                RETURN(@status)
        END
END
go
```

Procedure: InRFIUpdState

Code

```
-- --////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:InRFIUpdState
--
-- DESCRIPTION:                Update the value of InRequestFileInfo.FileState
--
-- TABLES ACCESSED:InRequestFileInfo
--
-- RETURNS:            Status (Success = 0)
--
-- INPUTS:     PARAMETER               DOMAIN
-- -------     ---------               ------
```

```
--
--                  RequestID           int,
--                  CompletionTime      int,
--                  DataGranuleID       int,
--                  FileID              varchar(255),
--                  FileState           char(15),
--                  TimeStamp           datetime
--
-- OUTPUTS:   PARAMETER            DOMAIN
-- --------   ---------            ------
--
--                  NONE
--
-- *--//////////////////////////////////////////////////////

CREATE PROCEDURE InRFIUpdState
        @RequestID              int,
        @CompletionTime         int,
        @DataGranuleID          int,
        @FileID                 varchar(255),
        @FileState              char(15),
    @TimeStamp                  datetime

AS
BEGIN
        DECLARE @status int

        BEGIN TRANSACTION

        UPDATE      InRequestFileInfo
        SET     CompletionTime = CompletionTime + @CompletionTime,
          FileState = @FileState,
          TimeStamp = @TimeStamp
        WHERE       RequestID       = @RequestID
        AND     DataGranuleID   = @DataGranuleID
        AND     FileID          = @FileID

        SELECT @status = @@error

        IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
                RETURN(@status)
        END
END
go
```

Procedure: InRFIUpdStateStatus

Code

```
////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:InRFIUpdStateStatus
--
-- DESCRIPTION:                 Update the value of InRequestFileInfo.FileState
--                 and InRequestFileInfo.FileStatus
--
-- TABLES ACCESSED:InRequestFileInfo
--
-- RETURNS:              Status (Success = 0)
--
-- INPUTS:       PARAMETER               DOMAIN
-- -------      ---------                   ------
--
--                   RequestID             int,
--                   DataGranuleID          int,
--                   FileID                varchar(255),
--                   FileState              char(15),
--                   FileStatus             smallint
--
-- OUTPUTS:    PARAMETER               DOMAIN
-- --------     ---------                   ------
--
--                   NONE
--
-- --////////////////////////////////////////////////////

CREATE PROCEDURE InRFIUpdStateStatus
        @RequestID                  int,
        @DataGranuleID               int,
        @FileID                    varchar(255),
        @FileState                  char(15),
        @FileStatus                 smallint
AS
BEGIN
        DECLARE @status int

        BEGIN TRANSACTION

        UPDATE          InRequestFileInfo
        SET     FileState = @FileState,
          FileStatus = @FileStatus
        WHERE           RequestID       = @RequestID
        AND     DataGranuleID    = @DataGranuleID
        AND     FileID                = @FileID

        SELECT @status = @@error
```

```
        IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
                RETURN(@status)
        END
END
go
```

Procedure: InRFIUpdStatus

Code

```
-- --////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:InRFIUpdStatus
--
-- DESCRIPTION:                 Update the value of InRequestFileInfo.FileStatus
--
-- TABLES ACCESSED:InRequestFileInfo
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:      PARAMETER               DOMAIN
-- -------     ---------                ------
--
--                  RequestID           int,
--                  DataGranuleID       int,
--                  FileID              varchar(255),
--                  FileStatus          smallint
--
--
-- OUTPUTS:     PARAMETER               DOMAIN
-- --------    ---------                ------
--
--                  NONE
--
--////////////////////////////////////////////////////

CREATE PROCEDURE InRFIUpdStatus
        @RequestID              int,
        @DataGranuleID          int,
        @FileID                 varchar(255),
        @FileStatus             smallint
AS
BEGIN
        DECLARE @status int

        BEGIN TRANSACTION
```

```
        UPDATE          InRequestFileInfo
        SET     FileStatus = @FileStatus
        WHERE           RequestID       = @RequestID
        AND     DataGranuleID    = @DataGranuleID
        AND     FileID                  = @FileID

        SELECT @status = @@error

        IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
                RETURN(@status)
        END
END
go
```

Procedure: InRPDInsert

Code

```
-- *--//////////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      InRPDInsert
--
-- DESCRIPTION:         Insert a record into the InRequestProcessData table.
--
-- TABLES ACCESSED:     InRequestProcessData
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:      PARAMETER               DOMAIN
-- -------      ---------               ------
--
--              RequestID               int,
--              DataGranuleID           int,
--              DataType                datatype,
--              DataDescriptor          varchar(60),
--              DataGranuleVolume       float(48),
--              DataVersion             int,
--              NodeName                varchar(255),
--              TotalFileCount          int
--
--
-- OUTPUTS:     PARAMETER               DOMAIN
-- --------     ---------               ------
--
--              NONE
--
-- --//////////////////////////////////////////////////////////
```

```
CREATE PROCEDURE InRPDInsert
      @RequestID            int,
      @DataGranuleID        int,
      @DataType             datatype,
      @DataDescriptor       varchar(60),
      @DataGranuleVolume    float(48),
      @DataVersion          int,
      @NodeName             varchar(255),
      @TotalFileCount       int
AS
BEGIN
      DECLARE @status int

      BEGIN TRANSACTION
      INSERT InRequestProcessData(
            RequestID,
            DataGranuleID,
            DataType,
            DataDescriptor,
            DataGranuleVolume,
            DataVersion,
            NodeName,
            RetryCount,
            TotalFileCount
      )
      VALUES (
            @RequestID,
            @DataGranuleID,
            @DataType,
            @DataDescriptor,
            @DataGranuleVolume,
            @DataVersion,
            @NodeName,
            0,
            @TotalFileCount
      )
      SELECT @status = @@error

         IF (@status != 0)
         BEGIN
                  ROLLBACK TRANSACTION
                  RETURN (@status)
         END
         ELSE
         BEGIN
         COMMIT TRANSACTION
         RETURN(@status)
         END
END
go
```

Procedure: InRPDRecov

Code

```
-- --////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      InRPDRecov
--
-- DESCRIPTION:  To determine if a granule was completed and if not, to
-- set its RetryCount and/or the FileStatus for the associated files
-- appropriately.
--
-- TABLES ACCESSED:    InRequestProcessData, InRequestFileInfo
--
--
-- RETURNS:            Status (Success = 0)
--
-- INPUTS:     PARAMETER              DOMAIN
-- -------     ---------              ------
--
--              RequestID              int
--              DataGranuleID          int
--              WarmStartFlag          int
--
--
-- OUTPUTS:    PARAMETER              DOMAIN
-- --------    ---------              ------
--
--              ResultingAction        varchar(20)
--
-- PDL:
-- ____
--
-- Select DataGranuleState and RetryCount from InRequestProcessData table
--     for the input granule
-- If granule is in InRequestProcessData table
--     if DataGranuleState == Archived, XferErr, PreprocErr, ArchErr,
--         Terminated, or Cancelled
--         set ResultingAction to "SkipGranule"
--     elseif RetryCount == 0
--         if WarmStartFlag == 1
--             Update RetryCount in InRequestProcessData table to 1
--         end if
--         set ResultingAction to "Continue"
--     else
--         set ResultingAction to "SkipGranuleAgain" to indicate that cannot
--             restart granule a 2nd time
--         Update FileStatus in InRequestFileInfo table for all files for
--             this granule to Internal Error
-- else
--     set status to granule not found
-- Return
```

```
--
-- *--//////////////////////////////////////////////////////

CREATE PROCEDURE InRPDRecov
      @RequestID        int,
      @DataGranuleID    int,
      @WarmStartFlag    int,
      @ResultingAction  varchar(20) output
AS
BEGIN

      DECLARE @status          int
      DECLARE @rows            int
      DECLARE @DataGranuleState varchar(20)
      DECLARE @RetryCount      int
      DECLARE @GranuleCompleted smallint

      SELECT @DataGranuleState = DataGranuleState,
            @RetryCount      = RetryCount,
               @GranuleCompleted = GranuleCompleted
      FROM InRequestProcessData
      WHERE RequestID    = @RequestID AND
           DataGranuleID = @DataGranuleID

      select @rows = @@rowcount
      select @status = @@error

      IF (@rows = 0)
      BEGIN
            select @status = 73100
            RAISERROR 73100
            "Granule not in InRequestProcessData table for RequestID %1! and DataGranuleID %2!",
            @RequestID, @DataGranuleID
            RETURN(@status)
      END

-- Check if DataGranuleState is complete.

      IF (((charindex("Arch", @DataGranuleState) > 0) OR
         (charindex("XferErr", @DataGranuleState) > 0) OR
        (charindex("PreprocErr", @DataGranuleState) > 0) OR
        (charindex("Terminated", @DataGranuleState) > 0) OR
        (charindex("Cancelled", @DataGranuleState) > 0)) AND
        (@GranuleCompleted = 1))
      BEGIN
         SELECT @ResultingAction = "SkipGranule"
      END

-- If RetryCount is 0, update it to 1.

      ELSE IF (@RetryCount = 0)
      BEGIN
         IF (@WarmStartFlag = 1)
```

```
        BEGIN
          BEGIN TRANSACTION

          UPDATE InRequestProcessData
          SET    RetryCount = @RetryCount + 1
          WHERE  RequestID     = @RequestID AND
                 DataGranuleID = @DataGranuleID

          SELECT @status = @@error

          IF (@status != 0)
          BEGIN
            ROLLBACK TRANSACTION
            RETURN (@status)
          END
          ELSE
          BEGIN
            COMMIT TRANSACTION
          END
         END

        SELECT @ResultingAction = "Continue"
      END

-- Have already retried this granule once before.
      ELSE
      BEGIN
        BEGIN TRANSACTION

        UPDATE InRequestFileInfo
        SET    FileStatus = 245  -- Internal Error
        WHERE  RequestID     = @RequestID AND
               DataGranuleID = @DataGranuleID

        SELECT @status = @@error

        IF (@status != 0)
        BEGIN
          ROLLBACK TRANSACTION
          RETURN (@status)
        END
        ELSE
        BEGIN
          COMMIT TRANSACTION
          RETURN(@status)
        END

        SELECT @ResultingAction = "SkipGranuleAgain"
      END

      RETURN(@status)

END
go
```

Procedure: InRPDSetGranComp

Code

```
-- --////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      InRPDSetGranComp
--
-- DESCRIPTION:         Update the value of
--                      InRequestProcessData.GranuleCompleted to 1
--
-- TABLES ACCESSED:     InRequestProcessData
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:      PARAMETER               DOMAIN
-- -------      ---------               ------
--
--              RequestID               int
--              DataGranuleID           int
--
--
-- OUTPUTS:     PARAMETER               DOMAIN
-- --------     ---------               ------
--
--              NONE
--
-- --////////////////////////////////////////////////////
CREATE PROCEDURE InRPDSetGranComp
        @RequestID                      int,
        @DataGranuleID                  int
AS
BEGIN
        DECLARE @status int

        BEGIN TRANSACTION
        UPDATE  InRequestProcessData
        SET     GranuleCompleted = 1
        WHERE   RequestID       = @RequestID
        AND     DataGranuleID   = @DataGranuleID
        SELECT @status = @@error
        IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
                RETURN(@status)
        END
 END
go
```

Procedure: InRPDUpdEndTime

Code

```
-- --////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      InRPDUpdEndTime
--
-- DESCRIPTION:         Update the value of
--                      InRequestProcessData.ProcessingEndDateTime
--
-- TABLES ACCESSED:     InRequestProcessData
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:      PARAMETER               DOMAIN
-- -------      ---------               ------
--
--              RequestID               int
--              DataGranuleID           int
--              ProcessingEndDateTime   varchar(30)
--
--
-- OUTPUTS:     PARAMETER               DOMAIN
-- --------     ---------               ------
--
--              NONE
--
-- --////////////////////////////////////////////////////

CREATE PROCEDURE InRPDUpdEndTime
        @RequestID                      int,
        @DataGranuleID                  int,
        @ProcessingEndDateTime          varchar(30)
AS
BEGIN
        DECLARE @status int

        BEGIN TRANSACTION
        UPDATE  InRequestProcessData
        SET     ProcessingEndDateTime = @ProcessingEndDateTime
        WHERE   RequestID       = @RequestID
        AND     DataGranuleID   = @DataGranuleID

        SELECT @status = @@error

        IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
```

```
        COMMIT TRANSACTION
        RETURN(@status)
    END

END
go
```

Procedure: InRPDUpdHandle

Code

```
-- --////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    InRPDUpdHandle
--
-- DESCRIPTION:       Update the value of
--                    InRequestProcessData.GranuleHandle
--
-- TABLES ACCESSED:   InRequestProcessData
--
-- RETURNS:           Status (Success = 0)
--
-- INPUTS:      PARAMETER              DOMAIN
-- -------      ---------              ------
--
--              RequestID              int,
--              DataGranuleID          int,
--              GranuleHandle          char(36)
--
--
-- OUTPUTS:     PARAMETER              DOMAIN
-- --------     ---------              ------
--
--              NONE
--
-- --////////////////////////////////////////////////////

CREATE PROCEDURE InRPDUpdHandle
    @RequestID                  int,
    @DataGranuleID              int,
    @GranuleHandle              char(36)
AS
BEGIN
    DECLARE @status int

    BEGIN TRANSACTION

    UPDATE InRequestProcessData
    SET GranuleHandle  = @GranuleHandle
    WHERE RequestID    = @RequestID
    AND   DataGranuleID = @DataGranuleID

    SELECT @status = @@error
```

```
        IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
                RETURN(@status)
        END

END
go
```

Procedure: InRPDUpdStaging

Code

```
-- --////////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    InRPDUpdStaging
--
-- DESCRIPTION:       Update the value of InRequestProcessData.StagingTagID.
--
-- TABLES ACCESSED:   InRequestProcessData
--
-- RETURNS:           Status (Success = 0)
--
-- INPUTS:      PARAMETER               DOMAIN
-- -------      ---------               ------
--
--             RequestID               int,
--             DataGranuleID           int,
--             StagingTagID            varchar(255)
--
--
-- OUTPUTS:     PARAMETER               DOMAIN
-- --------     ---------               ------
--
--             NONE
--
--////////////////////////////////////////////////////////

CREATE PROCEDURE InRPDUpdStaging
        @RequestID                  int,
        @DataGranuleID              int,
        @StagingTagID               varchar(255)
AS
BEGIN
        DECLARE @status int

        BEGIN TRANSACTION

        UPDATE InRequestProcessData
```

```
        SET StagingTagID = @StagingTagID
        WHERE RequestID = @RequestID
        AND DataGranuleID = @DataGranuleID

        SELECT @status = @@error

        IF (@status != 0)
        BEGIN
            ROLLBACK TRANSACTION
            RETURN (@status)
        END
        ELSE
        BEGIN
            COMMIT TRANSACTION
            RETURN(@status)
        END

END
go
```

Procedure: InRPDUpdStagReq

Code

```
--
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    InRPDUpdStagReq
--
-- DESCRIPTION:       Update the value of InRequestProcessData.StagingTagID.
--
-- TABLES ACCESSED:   InRequestProcessData
--
-- RETURNS:           Status (Success = 0)
--
-- INPUTS:      PARAMETER              DOMAIN
-- -------      ---------              ------
--
--              RequestID              int,
--              StagingTagID           varchar(255)
--
--
-- OUTPUTS:     PARAMETER              DOMAIN
-- --------     ---------              ------
--
--              NONE
--
-- *--//////////////////////////////////////////////////////

CREATE PROCEDURE InRPDUpdStagReq
        @RequestID                  int,
        @StagingTagID               varchar(255)
AS
BEGIN
```

```
    DECLARE @status int

    BEGIN TRANSACTION

    UPDATE InRequestProcessData
    SET StagingTagID = @StagingTagID
    WHERE RequestID = @RequestID

    SELECT @status = @@error

    IF (@status != 0)
    BEGIN
        ROLLBACK TRANSACTION
        RETURN (@status)
    END
    ELSE
    BEGIN
        COMMIT TRANSACTION
        RETURN(@status)
    END

END
go
```

Procedure: InRPDUpdStartTime

Code

```
-- --////////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:     InRPDUpdStartTime
--
-- DESCRIPTION:        Update the value of
--                     InRequestProcessData.ProcessingStartDateTime
--
-- TABLES ACCESSED:    InRequestProcessData
--
-- RETURNS:            Status (Success = 0)
--
-- INPUTS:     PARAMETER              DOMAIN
-- -------     ---------              ------
--
--             RequestID              int
--             DataGranuleID          int
--             ProcessingStartDateTime varchar(30)
--
--
-- OUTPUTS:    PARAMETER              DOMAIN
-- --------    ---------              ------
--
--             NONE
--
--////////////////////////////////////////////////////////
```

```
CREATE PROCEDURE InRPDUpdStartTime
      @RequestID                  int,
      @DataGranuleID              int,
      @ProcessingStartDateTime    varchar(30)
AS
BEGIN
      DECLARE @status int

      BEGIN TRANSACTION
      UPDATE  InRequestProcessData
      SET     ProcessingStartDateTime = @ProcessingStartDateTime
      WHERE   RequestID       = @RequestID
      AND     DataGranuleID   = @DataGranuleID

      SELECT @status = @@error

      IF (@status != 0)
      BEGIN
            ROLLBACK TRANSACTION
            RETURN (@status)
      END
      ELSE
      BEGIN
            COMMIT TRANSACTION
            RETURN(@status)
      END

END
go
```

Procedure: InRPDUpdStateArch

Code

```
-- --////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    InRPDUpdStateArch
--
-- DESCRIPTION:              Update the value of InRequestProcessData.DataGranuleState
--                  and the appropriate time based on the value the state.
--
-- TABLES ACCESSED:   InRequestProcessData
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:        PARAMETER              DOMAIN
-- -------      ---------                    ------
--
--                   RequestID              int,
--                   DataGranuleID          int,
--                   DataGranuleState       varchar(20),
--             StateTime             int
--
--
```

```
-- OUTPUTS:    PARAMETER              DOMAIN
-- --------    ---------              ------
--
--                  NONE
--
-- --////////////////////////////////////////////////////

CREATE PROCEDURE InRPDUpdStateArch
        @RequestID                  int,
        @DataGranuleID                      int,
        @DataGranuleState           varchar(20),
        @StateTime                  int
AS
BEGIN
        DECLARE @status int

        BEGIN TRANSACTION

        UPDATE      InRequestProcessData
        SET         DataGranuleState = @DataGranuleState,
                    TimeToArchive    = TimeToArchive + @StateTime
        WHERE       RequestID      = @RequestID
        AND         DataGranuleID    = @DataGranuleID

        SELECT @status = @@error

      IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
        END

        RETURN(@status)
END
go
```

Procedure: InRPDUpdStateDefault

Code

```
-- --////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    InRPDUpdStateDefault
--
-- DESCRIPTION:            Update the value of InRequestProcessData.DataGranuleState
--              and the appropriate time based on the value the state.
--
-- TABLES ACCESSED:   InRequestProcessData
--
-- RETURNS:        Status (Success = 0)
```

```
--
-- INPUTS:      PARAMETER              DOMAIN
-- -------     ---------              ------
--
--                  RequestID              int,
--                  DataGranuleID          int,
--                  DataGranuleState       varchar(20),
--                  StateTime              int
--
--
-- OUTPUTS:     PARAMETER              DOMAIN
-- --------    ---------              ------
--
--                  NONE
--
-- --////////////////////////////////////////////////////////

CREATE PROCEDURE InRPDUpdStateDefault
        @RequestID                  int,
        @DataGranuleID                      int,
        @DataGranuleState           varchar(20),
        @StateTime                  int
AS
BEGIN
        DECLARE @status int

        BEGIN TRANSACTION

    UPDATE  InRequestProcessData
    SET     DataGranuleState = @DataGranuleState
    WHERE   RequestID       = @RequestID
    AND     DataGranuleID    = @DataGranuleID

    SELECT @status = @@error

    IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
        END

        RETURN(@status)
END
go
```

Procedure: InRPDUpdStatePreproc

Code

```
-- --////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    InRPDUpdStatePreproc
--
-- DESCRIPTION:              Update the value of InRequestProcessData.DataGranuleState
--                  and the appropriate time based on the value the state.
--
-- TABLES ACCESSED:   InRequestProcessData
--
-- RETURNS:          Status (Success = 0)
--
-- INPUTS:        PARAMETER              DOMAIN
-- -------      ---------                    ------
--
--                   RequestID              int,
--                   DataGranuleID          int,
--                   DataGranuleState       varchar(20),
--             StateTime             int
--
--
-- OUTPUTS:    PARAMETER              DOMAIN
-- --------     ---------              ------
--
--                   NONE

--
-- --////////////////////////////////////////////////////

CREATE PROCEDURE InRPDUpdStatePreproc
          @RequestID                  int,
          @DataGranuleID              int,
          @DataGranuleState           varchar(20),
          @StateTime                  int
AS
BEGIN
          DECLARE @status int

          BEGIN TRANSACTION

       UPDATE  InRequestProcessData
       SET     DataGranuleState = @DataGranuleState,
               TimeToPreprocess = TimeToPreprocess + @StateTime
       WHERE   RequestID       = @RequestID
       AND     DataGranuleID    = @DataGranuleID

       SELECT @status = @@error

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

```
        RETURN (@status)
    END
    ELSE
    BEGIN
        COMMIT TRANSACTION
    END

    RETURN(@status)
END
go
```

Procedure: InRPDUpdStateXfer

Code

```
-- --////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    InRPDUpdStateXfer
--
-- DESCRIPTION:            Update the value of InRequestProcessData.DataGranuleState
--              and the appropriate time based on the value the state.
--
-- TABLES ACCESSED:   InRequestProcessData
--
-- RETURNS:        Status (Success = 0)
--
-- INPUTS:      PARAMETER           DOMAIN
-- -------     ---------           ------
--
--              RequestID           int,
--              DataGranuleID       int,
--              DataGranuleState    varchar(20),
--              StateTime           int
--
--
-- OUTPUTS:    PARAMETER           DOMAIN
-- --------    ---------           ------
--
--              NONE
--
-- --////////////////////////////////////////////////////

CREATE PROCEDURE InRPDUpdStateXfer
    @RequestID              int,
    @DataGranuleID                  int,
    @DataGranuleState       varchar(20),
    @StateTime              int
AS
BEGIN
    DECLARE @status int
```

```
        BEGIN TRANSACTION

    UPDATE  InRequestProcessData
    SET     DataGranuleState = @DataGranuleState,
            TimeToXfer       = TimeToXfer + @StateTime
    WHERE   RequestID        = @RequestID
    AND     DataGranuleID    = @DataGranuleID

    SELECT @status = @@error

    IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
        END

        RETURN(@status)
END
go
```

Procedure: InRPHColdStart

Code

```
-- --////////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:InRPHColdStart
--
-- DESCRIPTION:                 Deletes all records from the InRequestProcessHeader,
--                 InRequestProcessData, and InRequestFileInfo tables.
--                 Those entries with a valid RequestState (not NULL)
--                 are moved into the InRequestSummaryHeader and
--                 InRequestSummaryData tables.  The CurrentVolume and
--                 CurrentRequests columns in the
--                 InExternalDataProviderInfo table are initialized to 0.
--                 The CurrentTotalVolume and CurrentTotalRequests
--                 columns in the InSystemParameters table are
--                 initialized to 0.
--
-- TABLES ACCESSED:InRequestProcessHeader, InRequestProcessData,
--                 InRequestFileInfo, InRequestSummaryHeader,
--                 InRequestSummaryData, InExternalDataProviderInfo,
--                 InSystemParameters
--
-- RETURNS:                Status (Success = 0)
--
-- INPUTS:         PARAMETER               DOMAIN
-- -------      ---------                  ------
--              NONE
--
```

```
-- OUTPUTS:    PARAMETER               DOMAIN
-- --------    ---------               ------
--             NONE
--
-- PDL:
-- ____
--
-- Delete all rows with NULL RequestState from the InRequestFileInfo,
-- InRequestProcessData, and InRequestProcessHeader tables
-- Copy the request data remaining in the InRequestProcessHeader and
--    InRequestProcessData tables into the InRequestSummaryHeader and
--    InRequestSummaryData tables and delete the request data from the
--    InRequestFileInfo, InRequestProcessData, and InRequestProcessHeader
--    tables
-- Update CurrentVolume and CurrentRequests in InExternalDataProviderInfo table
--    to 0 for all rows
-- Update CurrentTotalVolume and CurrentTotalRequest in the InSystemParameters
--    table to 0
--
-- --////////////////////////////////////////////////////////

CREATE PROCEDURE InRPHColdStart
AS
        DECLARE @status                 int
      DECLARE @TerminatedRequestStateKey tinyint

      /* Delete the Requests with NULL RequestState from the active */
      /* process tables */
      /* Note that the order of the deletes is significant.  */

      BEGIN TRANSACTION

      /* Delete from  InRequestFileInfo, InRequestProcessData,    */
      /* and InRequestProcessHeader. */
      /* This function is done within the stored procedure InRPHWarmStart. */

      EXEC InRPHWarmStart

      SELECT @status = @@error

      IF (@status != 0)
      BEGIN
            ROLLBACK TRANSACTION
            RETURN(@status)
      END
      ELSE
            COMMIT TRANSACTION

      -- Copy the request data remaining in the InRequestProcessHeader and
      -- InRequestProcessData tables into the InRequestSummaryHeader and
      -- InRequestSummaryData tables and delete the request data from the
      -- InRequestFileInfo, InRequestProcessData, and InRequestProcessHeader
      -- tables.
```

```
CREATE TABLE #tmpRPHInfo (
       RequestID              int         NULL,
       RequestStateKey        tinyint     NULL,
       TotalSuccessfulGranules int        NULL,
       TimeToXfer             int         NULL,
       TimeToPreprocess       int         NULL,
       TimeToArchive          int         NULL)

SELECT @status = @@error

IF (@status != 0)
   RETURN(@status)

CREATE TABLE #tmpTotSuccess (
       RequestID              int     NULL,
       TotalSuccessfulGranules int     NULL)

SELECT @status = @@error

IF (@status != 0)
   RETURN(@status)

-- Count the granules that were archived into a temporary table
INSERT #tmpTotSuccess
SELECT RequestID, TotalSuccessfulGranules = COUNT(RequestID)
FROM InRequestProcessData
WHERE DataGranuleState = "Archived"
GROUP BY RequestID

SELECT @status = @@error

IF (@status != 0)
   RETURN(@status)

-- Compute the total times for all requests in another temporary
-- table.
INSERT #tmpRPHInfo
SELECT RequestID, NULL, 0, SUM(TimeToXfer), SUM(TimeToPreprocess),
       SUM(TimeToArchive)
FROM InRequestProcessData
GROUP BY RequestID

SELECT @status = @@error

IF (@status != 0)
   RETURN(@status)

-- Store the RequestState from the InRequestProcessHeader table into
-- the temporary table.
UPDATE #tmpRPHInfo
SET RequestStateKey = InRequestProcessHeader.RequestStateKey
FROM InRequestProcessHeader
WHERE InRequestProcessHeader.RequestID = #tmpRPHInfo.RequestID
```

```
	SELECT @status = @@error

	IF (@status != 0)
	   RETURN(@status)

	-- Change any RequestState which is not a complete state to Terminated.
	SELECT @TerminatedRequestStateKey = RequestStateKey
	FROM InValRequestState
	WHERE RequestState = "Terminated"

	SELECT @status = @@error

	IF (@status != 0)
	   RETURN(@status)

	UPDATE #tmpRPHInfo
	SET RequestStateKey = @TerminatedRequestStateKey
	FROM #tmpRPHInfo, InValRequestState
	WHERE InValRequestState.RequestState NOT in ("Successful", "Cancelled", "Failed",
"Partial_Failure")
	AND #tmpRPHInfo.RequestStateKey = InValRequestState.RequestStateKey

	SELECT @status = @@error

	IF (@status != 0)
	   RETURN(@status)

	-- Store the total successful granule counts from the first temporary
	-- table into the 2nd temporary table.
	UPDATE #tmpRPHInfo
	SET TotalSuccessfulGranules = #tmpTotSuccess.TotalSuccessfulGranules
	FROM #tmpRPHInfo, #tmpTotSuccess
	WHERE #tmpRPHInfo.RequestID = #tmpTotSuccess.RequestID

	SELECT @status = @@error

	IF (@status != 0)
	   RETURN(@status)

	/* First copy the Summary Header Information */

	BEGIN TRANSACTION

	INSERT INTO InRequestSummaryHeader
	SELECT  InRequestProcessHeader.RequestID,
	      DANFileName,
	      ExternalDataProvider,
	      IngestType,
	      Mission,
	      ProcessingStartDateTime,
	      ProcessingEndDateTime,
	      RequestPriority,
	      #tmpRPHInfo.RequestStateKey,
	      TimeToXfer,
```

```
        TimeToPreprocess,
        TimeToArchive,
        TotalDataVolume,
        TotalFileCount,
        TotalGranuleCount,
        TotalSuccessfulGranules
FROM    InRequestProcessHeader, #tmpRPHInfo
WHERE   InRequestProcessHeader.RequestID = #tmpRPHInfo.RequestID

SELECT @status = @@error

IF (@status != 0)
        BEGIN
          ROLLBACK TRANSACTION
          RETURN(@status)
        END

/* Next Copy the Summary Data information */

INSERT  InRequestSummaryData
SELECT  RequestID,
        DataGranuleID,
        DataType,
        DataGranuleVolume,
        DataGranuleState,
        NodeName,
        ProcessingEndDateTime,
        ProcessingStartDateTime,
        RetryCount,
        TotalFileCount,
        TimeToArchive,
        TimeToPreprocess,
        TimeToXfer
FROM    InRequestProcessData
SELECT @status = @@error
IF (@status != 0)
        BEGIN
          ROLLBACK TRANSACTION
          RETURN(@status)
        END
ELSE
        COMMIT TRANSACTION

/* Delete the Requests from the active process tables */
/* Note that the order of the deletes is significant.  */

BEGIN TRANSACTION

DELETE  InRequestFileInfo

SELECT @status = @@error

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

```
        ROLLBACK TRANSACTION
        RETURN(@status)
      END

DELETE  InRequestProcessData

SELECT @status = @@error

IF (@status != 0)
      BEGIN
        ROLLBACK TRANSACTION
        RETURN(@status)
      END

DELETE  InRequestProcessHeader

SELECT @status = @@error

IF (@status != 0)
      BEGIN
        ROLLBACK TRANSACTION
        RETURN(@status)
      END
ELSE
      COMMIT TRANSACTION

-- Set CurrentVolume and CurrentRequests to 0 for all external data
-- providers in the InExternalDataProviderInfo table.  Also set
-- CurrentTotalVolume and CurrentTotalRequests to 0 in the
-- InSystemParameters table.

BEGIN TRANSACTION

UPDATE  InExternalDataProviderInfo
SET     CurrentVolume = 0,
      CurrentRequests = 0

SELECT @status = @@error

IF (@status != 0)
BEGIN
      ROLLBACK TRANSACTION
      RETURN(@status)
END

UPDATE InSystemParameters
SET CurrentTotalVolume = 0,
   CurrentTotalRequests = 0

SELECT @status = @@error

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

```
        RETURN(@status)
    END
    ELSE
        COMMIT TRANSACTION

    RETURN(@status)
go
```

Procedure: InRPHDelCompReq

Code

```
-- --////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:     InRPHDelCompReq
--
-- DESCRIPTION:        When processing has been completed on an ingest
--     request for more than the MonitorTimeForCompletedRequests in the
--     InSystemParameters table:
--     (1) data in the RequestProcessHeader table is copied into the
--     InRequestSummaryHeader table, (2) data in the InRequestProcessData
--     table is copied into the InRequestSummaryData table, (3) the
--     file information for the ingest request is deleted from the
--     InRequestFileInfo table, (4) the granule information for the ingest
--     request is deleted from the InRequestProcessData table, and (5)
--     the record for the ingest request is deleted from the
--     InRequestProcessHeader table.
--
-- TABLES ACCESSED:    InRequestProcessHeader, InRequestProcessData,
--                     InRequestFileInfo, InRequestSummaryHeader,
--                     InRequestSummaryData, InSystemParameters
--
-- RETURNS:            Status (Success = 0)
--
-- INPUTS:     PARAMETER              DOMAIN
-- -------     ---------              ------
--
--             CurrentTime   varchar(30)
--
-- OUTPUTS:    PARAMETER              DOMAIN
-- --------    ---------              ------
--
--             NONE
--
-- --////////////////////////////////////////////////////


CREATE PROCEDURE InRPHDelCompReq
    @CurrentTime            varchar(30)
AS
BEGIN

    DECLARE @status                 int
    DECLARE @MonitorTime            int
```

```
DECLARE @CompleteTime            datetime

CREATE TABLE #tmpRPHComp (
      RequestID              int     NULL,
      TotalSuccessfulGranules int     NULL,
      TimeToXfer             int     NULL,
      TimeToPreprocess       int     NULL,
      TimeToArchive          int     NULL)

SELECT @status = @@error

IF (@status != 0)
BEGIN
      SELECT @status = 73200
      RAISERROR 73200 "Temporary table #tmpRPHCComp can not be created."
      RETURN (@status)
END

CREATE TABLE #tmpTotComp (
      RequestID              int     NULL,
      TotalSuccessfulGranules int     NULL)

SELECT @status = @@error

IF (@status != 0)
BEGIN
      SELECT @status = 73201
      RAISERROR 73201 "Temporary table #tmpTotComp can not be created."
      RETURN (@status)
END

-- Retrieve MonitorTimeForCompletedRequest
SELECT @MonitorTime = MonitorTimeForCompletedRequest * (-1)
FROM InSystemParameters

SELECT @CompleteTime = dateadd(mi, @MonitorTime, @CurrentTime)

-- Count the completed requests' granules that were archived into a
-- temporary table
INSERT #tmpTotComp
SELECT InRequestProcessData.RequestID,
      TotalSuccessfulGranules = COUNT(InRequestProcessData.RequestID)
FROM InRequestProcessData, InRequestProcessHeader, InValRequestState
WHERE DataGranuleState = "Archived"
AND
RequestState IN ("Cancelled", "Partial_Failure", "Failed", "Successful")
AND InRequestProcessHeader.RequestStateKey = InValRequestState.RequestStateKey
AND (InRequestProcessHeader.ProcessingEndDateTime <= @CompleteTime)
AND InRequestProcessHeader.RequestID = InRequestProcessData.RequestID

GROUP BY InRequestProcessData.RequestID
HAVING  DataGranuleState = "Archived"
AND
RequestState IN ("Cancelled", "Partial_Failure", "Failed", "Successful")
```

```
AND InRequestProcessHeader.RequestStateKey = InValRequestState.RequestStateKey
AND (InRequestProcessHeader.ProcessingEndDateTime <= @CompleteTime)
AND InRequestProcessHeader.RequestID = InRequestProcessData.RequestID

SELECT @status = @@error

IF (@status != 0)
BEGIN
      RAISERROR 73202 "Insert into temporary table #tmpTotComp failed."
      RETURN (@status)
END

-- Compute the total times for all completed requests in another
-- temporary table.
 INSERT #tmpRPHComp
 SELECT InRequestProcessData.RequestID, 0, SUM(TimeToXfer),
      SUM(TimeToPreprocess), SUM(TimeToArchive)
 FROM InRequestProcessData, InRequestProcessHeader, InValRequestState
 WHERE
 RequestState IN ("Cancelled", "Partial_Failure", "Failed", "Successful")
 AND InRequestProcessHeader.RequestStateKey = InValRequestState.RequestStateKey
 AND InRequestProcessHeader.ProcessingEndDateTime <= @CompleteTime
 AND InRequestProcessHeader.RequestID = InRequestProcessData.RequestID

 GROUP BY InRequestProcessData.RequestID
 HAVING
 RequestState IN ("Cancelled", "Partial_Failure", "Failed", "Successful")
 AND InRequestProcessHeader.RequestStateKey = InValRequestState.RequestStateKey
 AND InRequestProcessHeader.ProcessingEndDateTime <= @CompleteTime
 AND InRequestProcessHeader.RequestID = InRequestProcessData.RequestID

SELECT @status = @@error

IF (@status != 0)
BEGIN
      SELECT @status = 73203
      RAISERROR 73203 "Insert into temporary table #tmpRPHComp failed."
      RETURN (@status)
END

-- Store the total successful granule counts from the first temporary
-- table into the 2nd temporary table.
 UPDATE #tmpRPHComp
 SET TotalSuccessfulGranules = #tmpTotComp.TotalSuccessfulGranules
 FROM #tmpRPHComp, #tmpTotComp
 WHERE #tmpRPHComp.RequestID = #tmpTotComp.RequestID

SELECT @status = @@error

IF (@status != 0)
BEGIN
      SELECT @status = 73204
      RAISERROR 73204 "Update of temporary table #tmpRPHComp failed."
      RETURN (@status)
```

```
END

/* First copy the Summary Header Information */

BEGIN TRANSACTION

INSERT INTO InRequestSummaryHeader
SELECT  InRequestProcessHeader.RequestID,
        DANFileName,
        ExternalDataProvider,
        IngestType,
        Mission,
        ProcessingStartDateTime,
        ProcessingEndDateTime,
        RequestPriority,
        RequestStateKey,
        TimeToXfer,
        TimeToPreprocess,
        TimeToArchive,
        TotalDataVolume,
        TotalFileCount,
        TotalGranuleCount,
        TotalSuccessfulGranules
FROM    InRequestProcessHeader, #tmpRPHComp
WHERE   InRequestProcessHeader.RequestID = #tmpRPHComp.RequestID

SELECT @status = @@error

IF (@status != 0)
        BEGIN
          ROLLBACK TRANSACTION
          RETURN(@status)
        END

/* Next Copy the Summary Data information */

INSERT  InRequestSummaryData
SELECT  InRequestProcessData.RequestID,
        DataGranuleID,
        DataType,
        DataGranuleVolume,
        DataGranuleState,
        NodeName,
        ProcessingEndDateTime,
        ProcessingStartDateTime,
        RetryCount,
        TotalFileCount,
        InRequestProcessData.TimeToArchive,
        InRequestProcessData.TimeToPreprocess,
        InRequestProcessData.TimeToXfer
FROM    InRequestProcessData, #tmpRPHComp
WHERE   InRequestProcessData.RequestID = #tmpRPHComp.RequestID

SELECT @status = @@error
```

```
IF (@status != 0)
BEGIN
        ROLLBACK TRANSACTION
        RETURN(@status)
END
ELSE
BEGIN
      COMMIT TRANSACTION
END

/* Delete the RequestID from the active process tables */
/* Note that the order of the deletes is significant.  */

BEGIN TRANSACTION

DELETE  InRequestFileInfo
FROM InRequestFileInfo, #tmpRPHComp
WHERE   InRequestFileInfo.RequestID = #tmpRPHComp.RequestID

SELECT @status = @@error

IF (@status != 0)
BEGIN
        ROLLBACK TRANSACTION
        RETURN(@status)
END

DELETE  InRequestProcessData
FROM InRequestProcessData, #tmpRPHComp
WHERE   InRequestProcessData.RequestID = #tmpRPHComp.RequestID

SELECT @status = @@error

IF (@status != 0)
BEGIN
        ROLLBACK TRANSACTION
        RETURN(@status)
END

declare @RequestID int
select @RequestID = min(RequestID)
      from #tmpRPHComp

while @RequestID != null
begin
      DELETE  InRequestProcessHeader
      WHERE   InRequestProcessHeader.RequestID = @RequestID

      SELECT @status = @@error

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

```
            RETURN(@status)
        END

        select @RequestID = min(RequestID)
            from #tmpRPHComp
            where RequestID > @RequestID
    end

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

Procedure: InRPHInsert

Code

```
-- --//////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:InRPHInsert
--
-- DESCRIPTION:                 Insert a record into the InRequestProcessHeader table.
--
-- TABLES ACCESSED:InRequestProcessHeader
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:      PARAMETER               DOMAIN
-- -------     ---------                ------
--
--                  RequestID           int,
--                  RequestPriority     varchar(10),
--                  SequenceID          int,
--                  IngestType          varchar(40),
--                  ExternalDataProvider    varchar(20),
--                  ExpirationDateTime      varchar(30),
--                  TotalDataVolume     float(48),
--                  TotalGranuleCount       int,
--                  Mission             varchar(60),
--                  TotalFileCount      int,
--                  ProcessingStartDateTime varchar(30),
--                  DANFileName         varchar(255),
--          TransferFlag        smallint
--          SpecProc            smallint
--
--
-- OUTPUTS:    PARAMETER               DOMAIN
-- --------    ---------               ------
--
--          NONE
--
-- *--//////////////////////////////////////////////////////

CREATE PROCEDURE InRPHInsert
```

```
        @RequestID                  int,
        @RequestPriority            varchar(10),
        @SequenceID                 int,
        @IngestType                 varchar(40),
        @ExternalDataProvider       varchar(20),
        @ExpirationDateTime         varchar(30),
        @TotalDataVolume            float(48),
        @TotalGranuleCount          int,
        @Mission                    varchar(60),
        @TotalFileCount             int,
        @ProcessingStartDateTime    varchar(30),
        @DANFileName                varchar(255),
    @TransferFlag               smallint,
    @SpecProc                           smallint
AS
BEGIN
        DECLARE @status int

        BEGIN TRANSACTION
        INSERT InRequestProcessHeader (
                RequestID,
                SequenceID,
                ExpirationDateTime,
                ExternalDataProvider,
                IngestType,
                Mission,
                ProcessingStartDateTime,
                RequestPriority,
                DANFileName,
                TotalDataVolume,
                TotalFileCount,
                TotalGranuleCount,
          TransferFlag,
          SpecProc )
        VALUES          (
                @RequestID,
                @SequenceID,
                @ExpirationDateTime,
                @ExternalDataProvider,
                @IngestType,
                @Mission,
                @ProcessingStartDateTime,
                @RequestPriority,
                @DANFileName,
                @TotalDataVolume,
                @TotalFileCount,
                @TotalGranuleCount,
          @TransferFlag,
          @SpecProc )

        SELECT @status = @@error

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

```
            ROLLBACK TRANSACTION
            RETURN (@status)
        END
        ELSE
        BEGIN
                    COMMIT TRANSACTION
                    RETURN(@status)
            END
END
go
```

Procedure: lnRPHUpdDDNDest

Code

```
--//////////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      lnRPHUpdDDNDest
--
-- DESCRIPTION:         Update the value of CDSName, UUID, and
--                      DDNDestination in the lnRequestProcessHeader table
--                      for a given RequestID.
--
-- TABLES ACCESSED:     lnRequestProcessHeader
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:      PARAMETER               DOMAIN
-- -------      ---------               ------
--
--              RequestID               int,
--              CDSName                 varchar(255),
--              UUID                    char(36),
--              DDNDestination          int

-- OUTPUTS:     PARAMETER               DOMAIN
-- --------     ---------               ------
--
--              NONE
--
-- --//////////////////////////////////////////////////////////

CREATE PROCEDURE lnRPHUpdDDNDest
        @RequestID                      int,
        @CDSName                        varchar(255),
        @UUID                           char(36),
        @DDNDestination                 int
AS
BEGIN
        DECLARE @status int

        BEGIN TRANSACTION

        UPDATE lnRequestProcessHeader
```

```
        SET     CDSName         = @CDSName,
                UUID            = @UUID,
                DDNDestination  = @DDNDestination
        WHERE   RequestID       = @RequestID

        SELECT @status = @@error

        IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN(@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
                RETURN(@status)
        END

END
go
```

Procedure: InRPHUpdEndTime

Code

```
--//////////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      InRPHUpdEndTIme
--
-- DESCRIPTION:         Update the value of
--                      InRequestProcessHeader.ProcessingEndDateTime
--
-- TABLES ACCESSED:     InRequestProcessHeader
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:      PARAMETER               DOMAIN
-- -------      ---------               ------
--
--              RequestID               int
--              ProcessingEndDateTime   varchar(30)
--
--
-- OUTPUTS:     PARAMETER               DOMAIN
-- --------     ---------               ------
--
--              NONE
--
--//////////////////////////////////////////////////////////

CREATE PROCEDURE InRPHUpdEndTime
        @RequestID                      int,
        @ProcessingEndDateTime          varchar(30)
AS
```

```
BEGIN
      DECLARE @status int

      BEGIN TRANSACTION

      UPDATE  InRequestProcessHeader
      SET     ProcessingEndDateTime = @ProcessingEndDateTime
      WHERE   RequestID       = @RequestID

      SELECT @status = @@error

          IF (@status != 0)
          BEGIN
                    ROLLBACK TRANSACTION
                    RETURN (@status)
          END
          ELSE
          BEGIN
                    COMMIT TRANSACTION
                    RETURN(@status)
          END
END
go
```

Procedure: InRPHUpdExpired

Code

```
-- --////////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:      InRPHUpdExpired
--
-- DESCRIPTION:         Update the value of
--                      InRequestProcessHeader.ExpiredFlag to 1 for the
--                    input RequestID.
--
-- TABLES ACCESSED:     InRequestProcessHeader
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:      PARAMETER               DOMAIN
-- -------      ---------               ------
--
--              RequestID               int
--
--
-- OUTPUTS:     PARAMETER               DOMAIN
-- --------     ---------               ------
--
--              NONE
--
-- --////////////////////////////////////////////////////////
```

```
CREATE PROCEDURE InRPHUpdExpired
      @RequestID                  int
AS
      DECLARE @status int

      BEGIN TRANSACTION

      UPDATE InRequestProcessHeader
      SET    ExpiredFlag     = 1
      WHERE  RequestID       = @RequestID

      SELECT @status = @@error

      COMMIT TRANSACTION
      RETURN(@status)
go
```

Procedure: InRPHUpdFailReq

Code

```
-- --////////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:     InRPHUpdFailReq
--
-- DESCRIPTION:        Update the value of InRequestFileInfo.FileStatus
--                     for all files for a given RequestID.  Update the
--                     value of InRequestProcessHeader.RequestState for the
--                     RequestID.
--
-- TABLES ACCESSED:    InRequestFileInfo, InRequestProcessHeader
--
-- RETURNS:            Status (Success = 0)
--
-- INPUTS:     PARAMETER              DOMAIN
-- -------     ---------              ------
--
--             RequestID              int,
--             FileStatus             smallint
--
-- OUTPUTS:    PARAMETER              DOMAIN
-- --------    ---------              ------
--
--             NONE
--
-- --////////////////////////////////////////////////////////

CREATE PROCEDURE InRPHUpdFailReq
      @RequestID                  int,
      @FileStatus                 smallint
AS
BEGIN
      DECLARE @status             int
```

```
    DECLARE @FailedRequestStateKey tinyint

    BEGIN TRANSACTION

    UPDATE InRequestFileInfo
    SET     FileStatus      = @FileStatus
    WHERE   RequestID       = @RequestID

    SELECT @status = @@error

    IF (@status != 0)
    BEGIN  -- Not rolling back transaction since the Request failed
          -- and it is desired that as many of the state/statuses get
          -- set as possible.
           RETURN(@status)
    END

    COMMIT TRANSACTION

    SELECT @FailedRequestStateKey  = RequestStateKey FROM InValRequestState
    WHERE RequestState = "Failed"

    SELECT @status = @@error

    IF (@status != 0)
       RETURN(@status)

    BEGIN TRANSACTION

    UPDATE InRequestProcessHeader
    SET     RequestStateKey  = @FailedRequestStateKey
    WHERE   RequestID       = @RequestID

    IF (@status != 0)
    BEGIN -- Not rolling back the transaction since the Request failed
          -- and it is desired that as many of the states get set as
          -- possible.
RETURN(@status)
    END

    COMMIT TRANSACTION
    RETURN(@status)
END
Go
```

Procedure: InRPHUpdPercentCompleteArch

Code

```
--////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:InRPHUpdPercentCompleteArch
--
-- DESCRIPTION:                 Increment the value of
```

```
--                  InRequestProcessHeader.ArchPercentComplete by
--                  the input PercentComplete parameter
--
-- TABLES ACCESSED:InRequestProcessHeader
--
-- RETURNS:            Status (Success = 0)
--
-- INPUTS:      PARAMETER           DOMAIN
-- -------     ---------                   ------
--
--                  RequestID               int,
--                  PercentComplete             int
--
--
-- OUTPUTS:    PARAMETER           DOMAIN
-- --------    ---------                   ------
--
--                  NONE
--
-- --////////////////////////////////////////////////////

CREATE PROCEDURE InRPHUpdPercentCompleteArch
        @RequestID              int,
        @PercentComplete        int
AS
BEGIN
        DECLARE @status int

        BEGIN TRANSACTION
        UPDATE      InRequestProcessHeader
        SET     ArchPercentComplete = @PercentComplete + ArchPercentComplete
        WHERE       RequestID       = @RequestID

        SELECT @status = @@error

        IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
                RETURN(@status)
        END
END
go
```

Procedure: InRPHUpdPercentCompletePre

Code

```
-- --//////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:InRPHUpdPercentCompletePre
--
-- DESCRIPTION:                 Increment the value of
--                 InRequestProcessHeader.PreprocPercentComplete by
--                 the input PercentComplete parameter
--
-- TABLES ACCESSED:InRequestProcessHeader
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:      PARAMETER               DOMAIN
-- -------     ---------                   ------
--
--                 RequestID               int,
--                 PercentComplete             int
--
-- OUTPUTS:    PARAMETER               DOMAIN
-- --------    ---------                   ------
----                NONE
---- *--//////////////////////////////////////////////////////

CREATE PROCEDURE InRPHUpdPercentCompletePre
        @RequestID                  int,
        @PercentComplete            int
AS
BEGIN
        DECLARE @status int

        BEGIN TRANSACTION
        UPDATE      InRequestProcessHeader
        SET     PreprocPercentComplete = @PercentComplete + PreprocPercentComplete
        WHERE       RequestID       = @RequestID

        SELECT @status = @@error

        IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
                RETURN(@status)
        END
END
go
```

Procedure: InRPHUpdPercentCompleteXfer

Code

```
-- --////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:InRPHUpdPercentCompleteXfer
--
-- DESCRIPTION:                 Increment the value of
--                 InRequestProcessHeader.XferPercentComplete by
--                 the input PercentComplete parameter
--
-- TABLES ACCESSED:InRequestProcessHeader
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:      PARAMETER               DOMAIN
-- -------     ---------                   ------
--
--                 RequestID               int,
--                 PercentComplete             int
--
--
-- OUTPUTS:    PARAMETER               DOMAIN
-- --------    ---------                   ------
--                 NONE
-- --////////////////////////////////////////////////////
CREATE PROCEDURE InRPHUpdPercentCompleteXfer
        @RequestID                  int,
        @PercentComplete            int
AS
BEGIN
        DECLARE @status int
        BEGIN TRANSACTION
        UPDATE      InRequestProcessHeader
        SET     XferPercentComplete = @PercentComplete + XferPercentComplete
        WHERE       RequestID       = @RequestID

        SELECT @status = @@error

        IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
                RETURN(@status)
        END
END
go
```

Procedure: InRPHUpdPriority

Code

```
-- --////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:InRPHUpdPriority
--
-- DESCRIPTION:                 Update the value of
--                 InRequestProcessHeader.RequestPriority
--
-- TABLES ACCESSED:InRequestProcessHeader
--
-- RETURNS:              Status (Success = 0)
--
-- INPUTS:      PARAMETER            DOMAIN
-- -------     ---------                  ------
--
--                  RequestID               int,
--                  RequestPriority         varchar(10)
--
-- OUTPUTS:    PARAMETER            DOMAIN
-- --------    ---------                  ------
--
--                  NONE

--
-- ////////////////////////////////////////////////////

CREATE PROCEDURE InRPHUpdPriority
        @RequestID                  int,
        @RequestPriority            varchar(10)
AS
BEGIN
        DECLARE @status int

        BEGIN TRANSACTION
        UPDATE      InRequestProcessHeader
        SET     RequestPriority = @RequestPriority
        WHERE       RequestID      = @RequestID

        SELECT @status = @@error
        IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
                RETURN(@status)
        END
END
Go
```

Procedure: InRPHUpdPriorityDataProv

Code

```
-- --////////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:InRPHUpdPriority
--
-- DESCRIPTION:                 Update the value of
--                 InRequestProcessHeader.RequestPriority for the
--                 ExternalDataProvider
--
-- TABLES ACCESSED:InRequestProcessHeader
--
-- RETURNS:              Status (Success = 0)
--
-- INPUTS:       PARAMETER             DOMAIN
-- -------     ---------                   ------
--
--                 ExternalDataProvider    varchar(20),
--                 RequestPriority             varchar(10)
--
--
-- OUTPUTS:    PARAMETER             DOMAIN
-- --------     ---------                   ------
--
--                 NONE
--
-- --////////////////////////////////////////////////////////

CREATE PROCEDURE InRPHUpdPriorityDataProv
        @ExternalDataProvider         varchar(20),
        @RequestPriority                 varchar(10)
AS
BEGIN
        DECLARE @status int

        BEGIN TRANSACTION
        UPDATE        InRequestProcessHeader
        SET     RequestPriority = @RequestPriority
        WHERE         ExternalDataProvider = @ExternalDataProvider

        SELECT @status = @@error

      IF (@status != 0)
       BEGIN
            ROLLBACK TRANSACTION
            RETURN (@status)
      END
      ELSE
```

```
    BEGIN
        COMMIT TRANSACTION
        RETURN(@status)
    END

END
go
```

Procedure: InRPHUpdState

Code

```
-- --////////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:InRPHUpdState
--
-- DESCRIPTION:                 Update the value of InRequestProcessHeader.RequestState
--
-- TABLES ACCESSED:InRequestProcessHeader
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:      PARAMETER           DOMAIN
-- -------     ---------               ------
--
--                  RequestID           int,
--                  RequestState        varchar(15)
--
--
-- OUTPUTS:     PARAMETER           DOMAIN
-- --------    ---------               ------
--
--                  NONE
--
-- --////////////////////////////////////////////////////////

CREATE PROCEDURE InRPHUpdState
        @RequestID                  int,
        @RequestState               varchar(15)
AS
BEGIN
        DECLARE @status int

        BEGIN TRANSACTION
        UPDATE      InRequestProcessHeader
        SET     RequestStateKey = InValRequestState.RequestStateKey
    FROM InRequestProcessHeader, InValRequestState
        WHERE       RequestID   = @RequestID
    AND     InValRequestState.RequestState = @RequestState

        SELECT @status = @@error

    IF (@status != 0)
```

```
        BEGIN
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
                COMMIT TRANSACTION
                RETURN(@status)
        END
END
go
```

Procedure: InRPHUpdStateChng

Code

```
-- --////////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME: InRPHUpdStateChng
--
-- DESCRIPTION:  To update the Request State for an ingest request.
--
-- TABLES ACCESSED:     InRequestProcessHeader
--
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:      PARAMETER               DOMAIN
-- -------      ---------               ------
--
--              RequestID               int
--              StateControlCommand     varchar(15)
--
--
-- OUTPUTS:     PARAMETER               DOMAIN
-- --------     ---------               ------
--
--              PreviousState           varchar(15)
--
-- PDL:
-- ____
--
-- lock access to record in InRequestProcessHeader table
-- select RequestState into PreviousState
-- if (RequestID not found)
--     Set Status = RequestID not found
--     Return
-- end if
-- if ((StateControlCommand != Mark Active) and
--    (PreviousState == Resuming)) or
--   ((StateControlCommand != Mark Cancelled) and
--    (PreviousState == Cancelling)) or
--   ((StateControlCommand != Mark Suspended) and
--    (PreviousState == Suspending))
--
```

```
--     Set Status = previous control command is not complete
--     return PreviousState
--     Return
--
-- elseif (StateControlCommand == Suspend) (Rel B)
--
--     if (PreviousState == Active)
--
--         Update State to Suspending
--
--     else
--
--         Set Status = invalid PreviousState
--         return PreviousState
--         Return
--
-- elseif (StateControlCommand == Mark Active) (Rel B)
--
--     if (PreviousState == Resuming)
--
--         Update State to Active
--
--     else
--
--         Set Status = invalid PreviousState
--         return PreviousState
--         Return
--
--
-- elseif (StateControlCommand == Mark Suspended) (Rel B)
--
--     if (PreviousState == Suspending)
--
--         Update State to Suspended
--
--     else
--
--         Set Status = invalid PreviousState
--         return PreviousState
--         Return
--
--
-- elseif (StateControlCommand == Mark Cancelled)
--
--     if (PreviousState == Cancelling)
--
--         Update State to Cancelled
--
--     else
--
--         Set Status = invalid PreviousState
--         return PreviousState
--         Return
--
```

```
--
-- elseif (StateControlCommand == Cancel)
--
--     if (PreviousState == Suspended)
--
--         Update State to Cancelled
--
--     elseif (PreviousState == Active)
--
--         Update State to Cancelling
--
--     else
--
--         Set Status = invalid PreviousState
--         return PreviousState
--         Return
--
--
-- elseif (StateControlCommand == Resume) (Rel B)
--
--     if (PreviousState == Suspended)
--
--         Update State to Resuming
--
--     else
--
--         Set Status = invalid PreviousState
--         return PreviousState
--         Return
--
-- else
--
--     Set Status = invalid StateControlCommand
--     return PreviousState
--     Return
--
-- Return
--
-- --////////////////////////////////////////////////////////

CREATE PROCEDURE InRPHUpdStateChng
        @RequestID          int,
        @StateControlCommand varchar(15),
        @PreviousState      varchar(15) output
AS
BEGIN

        DECLARE @status int
        DECLARE @rows   int

        BEGIN TRANSACTION
        SELECT @PreviousState = RequestState
        FROM   InRequestProcessHeader, InValRequestState
        HOLDLOCK
```

```
WHERE  RequestID = @RequestID
AND    InValRequestState.RequestStateKey = InRequestProcessHeader.RequestStateKey

SELECT @rows = @@rowcount

IF (@rows = 0)
BEGIN
      select @status = 73205
      RAISERROR 73205 "RequestID %1! not in InRequestProcessHeader table", @RequestID
      ROLLBACK TRANSACTION
      RETURN(@status)
END

IF (((charindex("MarkActive", @StateControlCommand) <= 0) AND
    (charindex("Resuming", @PreviousState) > 0))
  OR
   ((charindex("MarkCancelled", @StateControlCommand) <= 0) AND
    (charindex("Cancelling", @PreviousState) > 0))
  OR
   ((charindex("MarkSuspended", @StateControlCommand) <= 0) AND
   (charindex("Suspending", @PreviousState) > 0)))
BEGIN
      select @status = 73206
      RAISERROR 73206 "Previous control command is not complete"
      ROLLBACK TRANSACTION
      RETURN(@status)
END

IF (charindex("Suspend", @StateControlCommand) > 0)
BEGIN
      IF (charindex("Active", @PreviousState) > 0)
      BEGIN
            UPDATE InRequestProcessHeader
            SET RequestStateKey = InValRequestState.RequestStateKey
            FROM InRequestProcessHeader, InValRequestState
            WHERE RequestID = @RequestID
            AND   RequestState = "Suspending"

            select @status = @@error

            COMMIT TRANSACTION
            RETURN(@status)
      END

      select @status = 73207
      RAISERROR 73207 "Invalid PreviousState"
      ROLLBACK TRANSACTION
      RETURN(@status)
END

IF (charindex("MarkActive", @StateControlCommand) > 0)
BEGIN
      IF (charindex("Resuming", @PreviousState) > 0)
      BEGIN
```

```
            UPDATE InRequestProcessHeader
               SET RequestStateKey = InValRequestState.RequestStateKey
            FROM InRequestProcessHeader, InValRequestState
            WHERE RequestID = @RequestID
            AND   RequestState = "Active"

            select @status = @@error

            COMMIT TRANSACTION
            RETURN(@status)
      END

      select @status = 73208
      RAISERROR 73208 "Invalid PreviousState"
      ROLLBACK TRANSACTION
      RETURN(@status)
END

IF (charindex("MarkSuspended", @StateControlCommand) > 0)
BEGIN
      IF (charindex("Suspending", @PreviousState) > 0)
      BEGIN
            UPDATE InRequestProcessHeader
            SET RequestStateKey = InValRequestState.RequestStateKey
            FROM InRequestProcessHeader, InValRequestState
            WHERE RequestID = @RequestID
            AND   RequestState = "Suspended"

            select @status = @@error

            COMMIT TRANSACTION
            RETURN(@status)
      END

      select @status = 73209
      RAISERROR 73209 "Invalid PreviousState"
      ROLLBACK TRANSACTION
      RETURN(@status)
END

IF (charindex("MarkCancelled", @StateControlCommand) > 0)
BEGIN
      IF (charindex("Cancelling", @PreviousState) > 0)
      BEGIN
            UPDATE InRequestProcessHeader
            SET RequestStateKey = InValRequestState.RequestStateKey
            FROM InRequestProcessHeader, InValRequestState
            WHERE RequestID = @RequestID
            AND   RequestState = "Cancelled"

            select @status = @@error

            COMMIT TRANSACTION
```

```
            RETURN(@status)
        END

        select @status = 73210
        RAISERROR 73210 "Invalid PreviousState"
        ROLLBACK TRANSACTION
        RETURN(@status)
    END

    IF (charindex("Cancel", @StateControlCommand) > 0)
    BEGIN
        IF (charindex("Suspended", @PreviousState) > 0)
        BEGIN
            UPDATE InRequestProcessHeader
            SET RequestStateKey = InValRequestState.RequestStateKey
            FROM InRequestProcessHeader, InValRequestState
            WHERE RequestID = @RequestID
            AND   RequestState = "Cancelled''

            select @status = @@error

            COMMIT TRANSACTION
            RETURN(@status)
        END

        IF (charindex("Active", @PreviousState) > 0)
        BEGIN
            UPDATE InRequestProcessHeader
            SET RequestStateKey = InValRequestState.RequestStateKey
            FROM InRequestProcessHeader, InValRequestState
            WHERE RequestID = @RequestID
            AND   RequestState = "Cancelling''

            select @status = @@error

            COMMIT TRANSACTION
            RETURN(@status)
        END

        select @status = 73211
        RAISERROR 73211 "Invalid PreviousState"
        ROLLBACK TRANSACTION
        RETURN(@status)
    END

    IF (charindex("Resume", @StateControlCommand) > 0)
    BEGIN
        IF (charindex("Suspended", @PreviousState) > 0)
        BEGIN
            UPDATE InRequestProcessHeader
            SET RequestStateKey = InValRequestState.RequestStateKey
            FROM InRequestProcessHeader, InValRequestState
            WHERE RequestID = @RequestID
            AND   RequestState = "Resuming"
```

```
            select @status = @@error

            COMMIT TRANSACTION
            RETURN(@status)
        END

        select @status = 73212
        RAISERROR 73212 "Invalid PreviousState"
        ROLLBACK TRANSACTION
        RETURN(@status)
    END

    select @status = 73213
    RAISERROR 73213 "Invalid StateControlCommand %1!", @StateControlCommand
    ROLLBACK TRANSACTION
    RETURN(@status)

END
go
```

Procedure: InRPHWarmStart

Code

```
-- --//////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:InRPHWarmStart
--
-- DESCRIPTION:                 Deletes all records from the InRequestProcessHeader,
--              InRequestProcessData, and InRequestFileInfo tables
--              which have an invalid RequestState (NULL).
--
-- TABLES ACCESSED:InRequestProcessHeader, InRequestProcessData,
--              InRequestFileInfo
--
-- RETURNS:             Status (Success = 0)
--
-- INPUTS:      PARAMETER               DOMAIN
-- -------     ---------                ------
--
--          NONE
--
--
-- OUTPUTS:   PARAMETER                 DOMAIN
-- --------    ---------                ------
--
--          NONE
--
-- PDL:
-- ____
--
-- Delete all rows with NULL RequestState from the InRequestFileInfo,
-- InRequestProcessData, and InRequestProcessHeader tables
--
```

```
-- --//////////////////////////////////////////////////////

CREATE PROCEDURE InRPHWarmStart
AS
BEGIN

    DECLARE @status int

    /* Delete the Requests with NULL RequestState from the active */
    /* process tables */
    /* Note that the order of the deletes is significant.  */

    BEGIN TRANSACTION

    DELETE  InRequestFileInfo
      FROM  InRequestFileInfo, InRequestProcessHeader
    WHERE   RequestStateKey IS NULL
      AND   InRequestFileInfo.RequestID = InRequestProcessHeader.RequestID

    SELECT @status = @@error

    IF (@status != 0)
    BEGIN
        ROLLBACK TRANSACTION
        RETURN(@status)
    END

    DELETE  InRequestProcessData
      FROM  InRequestProcessData, InRequestProcessHeader
    WHERE   RequestStateKey IS NULL
      AND   InRequestProcessData.RequestID = InRequestProcessHeader.RequestID

    SELECT @status = @@error

    IF (@status != 0)
    BEGIN
        ROLLBACK TRANSACTION
        RETURN(@status)
    END

    DELETE  InRequestProcessHeader
      FROM  InRequestProcessHeader
    WHERE   RequestStateKey IS NULL

    SELECT @status = @@error

    IF (@status != 0)
    BEGIN
        ROLLBACK TRANSACTION
        RETURN(@status)
    END
    ELSE
      BEGIN
        COMMIT TRANSACTION
```

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

Procedure: InSPUpdAll

Code

```
-- --////////////////////////////////////////////////////////
-- BEGIN PROLOG
--
-- PROCEDURE NAME:    InSPUpdAll
--
-- DESCRIPTION:       Update the values of CommunicationRetryCount,
--                    CommunicationRetryInterval,
--                    MonitorTimeForCompletedRequest, ScreenUpdateInterval,
--                    MaximumTotalRequests, and MaximumTotalVolume in
--                    the InSystemParameters table.
--
-- TABLES ACCESSED:   InSystemParameters
--
-- RETURNS:           Status (Success = 0)
--
-- INPUTS:      PARAMETER                   DOMAIN
-- -------      ---------                   ------
--
--              CommunicationRetryCount       int,
--              CommunicationRetryInterval    int,
--              MonitorTime                   int,
--              ScreenUpdateInterval          int,
--              MaximumTotalRequests          int,
--              MaximumTotalVolume            float(48)
--
--
-- OUTPUTS:     PARAMETER                   DOMAIN
-- --------     ---------                   ------
--
--              NONE
--
-- --////////////////////////////////////////////////////////

CREATE PROCEDURE InSPUpdAll
        @CommunicationRetryCount       int,
        @CommunicationRetryInterval    int,
        @MonitorTime                   int,
        @ScreenUpdateInterval          int,
        @MaximumTotalRequests          int,
        @MaximumTotalVolume            float(48)
AS
BEGIN
        DECLARE @status int

        BEGIN TRANSACTION
            UPDATE          InSystemParameters
        SET CommunicationRetryCount       = @CommunicationRetryCount,
```

```
        CommunicationRetryInterval     = @CommunicationRetryInterval,
            MonitorTimeForCompletedRequest = @MonitorTime,
        ScreenUpdateInterval          = @ScreenUpdateInterval,
        MaximumTotalRequests           = @MaximumTotalRequests,
        MaximumTotalVolume             = @MaximumTotalVolume

        SELECT @status = @@error

        IF (@status != 0)
        BEGIN
                ROLLBACK TRANSACTION
                RETURN (@status)
        END
        ELSE
        BEGIN
        COMMIT TRANSACTION
        RETURN(@status)
        END

END
go
```

3.2 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. There are no files utilized in INGEST.

3.2.1 Files Definitions

Not Applicable

3.2.2 Attributes

Not Applicable

3.2.3 Attribute Domains

Not Applicable

4. Performance and Tuning Factors

4.1 Indexes

An index provides a means of locating a row in a table based on the value of specific 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 table 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. A list of all the indexes defined against tables in the INGEST database is given here along with a description of each index.

Index List

Table Code	Index Code	P	F	U	C
InDataTypeTemplate	pk_indatatypetemplate	Yes	No	Yes	Yes
InESDTMap	pk_inesdtmap	Yes	Yes	Yes	Yes
InExternalDataProviderInfo	pk_inexternaldataproviderinfo	Yes	No	Yes	Yes
InFileTypeTemplate	InFTTPrimary	No	No	Yes	No
InGranuleServerInfo	pk_ingranuleserverinfo	Yes	Yes	Yes	Yes
InMediaCheckin	pk_inmediacheckin	Yes	No	Yes	Yes
InMediaType	pk_inmediatype	Yes	No	Yes	Yes
InNextAvailableID	pk_innextavailableid	Yes	No	Yes	Yes
InRequestFileInfo	pk_inrequestfileinfo InRFIGranuleID	Yes No	No No	Yes No	Yes No
InRequestProcessData	pk_inrequestprocessdata	Yes	No	Yes	Yes
InRequestProcessHeader	pk_inrequestprocessheader InRPHDataProvider	Yes No	No Yes	Yes No	Yes No
InRequestSummaryData	pk_inrequestsummarydata InRSDPrimary	Yes No	No No	Yes No	Yes No

Table Code	Index Code	P	F	U	C
InRequestSummaryHeader	pk_inrequestsummaryheader InRSHDataProvider InRSHProcessingStartDateTime	Yes No No	No No No	Yes No No	Yes No No
InSourceMCF					
InSystemParameters					
InValDataGranuleState	pk_invaldatagranulestate	Yes	No	Yes	Yes
InValGranuleServerUR	pk_invalgranuleserverur	Yes	No	Yes	Yes
InValIngestType	pk_invalingesttype	Yes	No	Yes	Yes
InValNotifyType	pk_invalnotifytype	Yes	No	Yes	Yes
InValParameterClass	pk_invalparameterclass	Yes	No	Yes	Yes
InValRequestState	pk_invalrequeststate	Yes	No	Yes	Yes

4.2 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. INGEST is defined to use the Sybase default, logsegment, and system segments.

4.3 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 INGEST databases are not defined.

5. Database Security

5.1 Approach

The database security discussed within this section is bounded to security implementation within the Sybase SQL Server RDBMS. A Sybase general approach to security is adopted as illustrated in Figure 5-1.

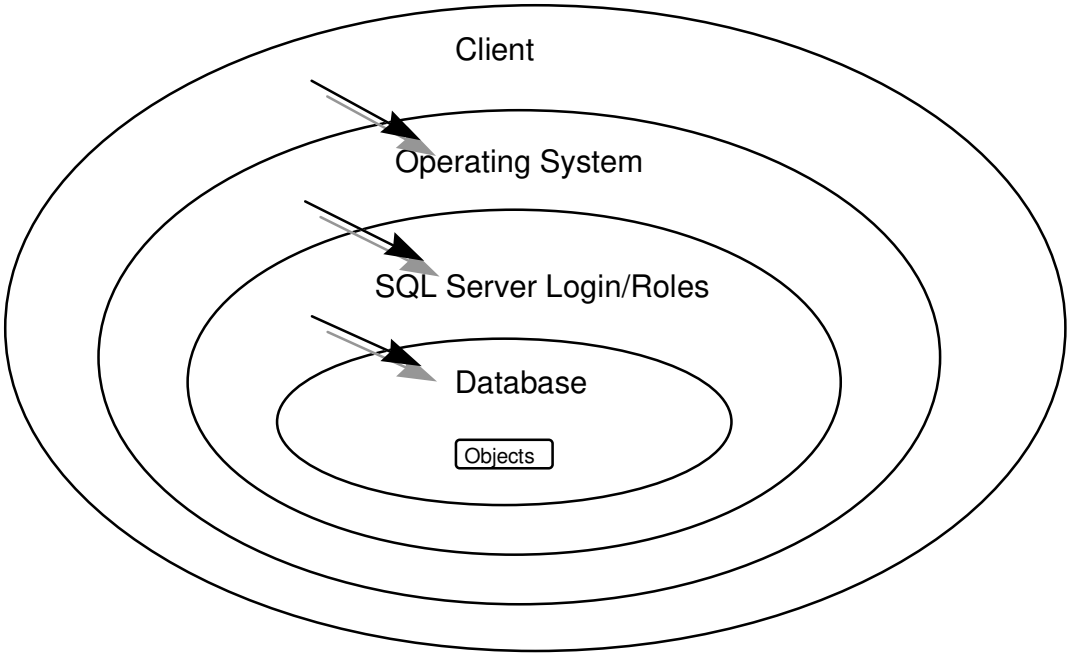

Figure 5-1. Sybase Approach to SQL Server Security

The client/user requires a SQL Server login to access the RDBMS. The System Administrator may grant or revoke various roles to any logins, i.e., *sa, sso, oper*. These roles are identified in Section 5.4. The login needs to be assigned to a user in the database and permissions must be assigned for the user to gain access to the objects within the database. Examples of objects associated with the database include tables, views, commands.

5.2 Initial Users

Upon initial installation the following users will have access to INGEST 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 the documents referenced below.

Login Name	Default Database	Group
EcInReqMgr	Ingest	software
EcInAuto	Ingest	software
EcInGUI	Ingest	software
EcInGran	Ingest	software
EcPolling	Ingest	software
EcInInter	Ingest	software

5.3 Groups

Groups are a means of logically associating users with similar data access needs. Once a group has been defined, object and command permissions can be granted to that group. A user who is member of a group inherits all of the permissions granted to that group. Two groups have been defined in the INGEST database upon initial installation. A definition of each of these groups is contained herein. A user who is a member of the software group has the permissions of the public group also.

Group Name	Database	Permission
software	Ingest	see Object Permissions
public	Ingest	see Object Permissions

5.4 Roles

Roles were introduced in Sybase 10 to allow a structured means for granting users the permissions needed to perform standard database administration activities and also provide a means for easily identifying such users. There are six predefined 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:

a. installing SQL server and specific SQL server modules

b. managing the allocation of physical storage

c. tuning configuration parameters

d. 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:

a. adding server logins

b. administrating passwords

c. managing the audit system

d. 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;

a. dumping transactions and databases

b. 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.

5.5 Object Permissions

Users are assigned to groups which provides a means of logically associating users with similar data access needs. The object and command permissions are granted to the group. A user who is member of the software group inherits all of the permissions granted to that group. A user is automatically a member of the public group and inherits the permissions associated with the public group. In INGEST, the public group grants to the user select privilege on system tables.

Software Group Table ListTable Permissions

Name	Delete	Insert	Select	Update
InDataTypeTemplate	X	X	X	X
InESDTMap	X	X	X	X
InExternalDataProviderInfo	X	X	X	X
InFileTypeTemplate	X	X	X	X
InGranuleServerInfo	X	X	X	X
InMediaCheckin	X	X	X	X
InMediaType	X	X	X	X
InNextAvailableID	X	X	X	X
InRequestFileInfo	X	X	X	X
InRequestProcessData	X	X	X	X
InRequestProcessHeader	X	X	X	X
InRequestSummaryData	X	X	X	X
InRequestSummaryHeader	X	X	X	X
InSourceMCF	X	X	X	X
InSystemParameters	X	X	X	X
InValDataGranuleState	X	X	X	X
InValGranuleServerUR	X	X	X	X
InValIngestType	X	X	X	X
InValNotifyType	X	X	X	X
InValParameterClass	X	X	X	X
InValRequestState	X	X	X	X

6. Scripts

6.1 Installation Scripts

Any scripts used to support installation of the INGEST database are described herein. These files are found in the CLEARCASE directory ecs/formal/INGEST/common/src/sybase.

Script File	Usage
EcInDbBuild.sql	Calls four other scripts to Install/populate the ingest database: EcInDbDropAll.sql, EcInDbUsers.sql,EcInDbTables.sql, and EcInDbPermissions.sql
EcInDbUsers.sql	Adds users to Ingest database
EcInDbTables.sql	Creates Ingest database tables
EcInDbPermissions.sql	Grants permissions tables and objects as needed by IOS applications.

6.2 De-Installation Scripts

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

Script File	Usage
EcInDbDropAll.sql	De-installs database

6.3 Backup/Recovery Scripts

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

There are no scripts defined.

6.4 Miscellaneous Scripts

Miscellaneous scripts applicable to the INGEST database are described herein. Ingest scripts are listed above.

This page intentionally left blank.

Appendix A Entity Relationship Diagram

This page intentionally left blank.

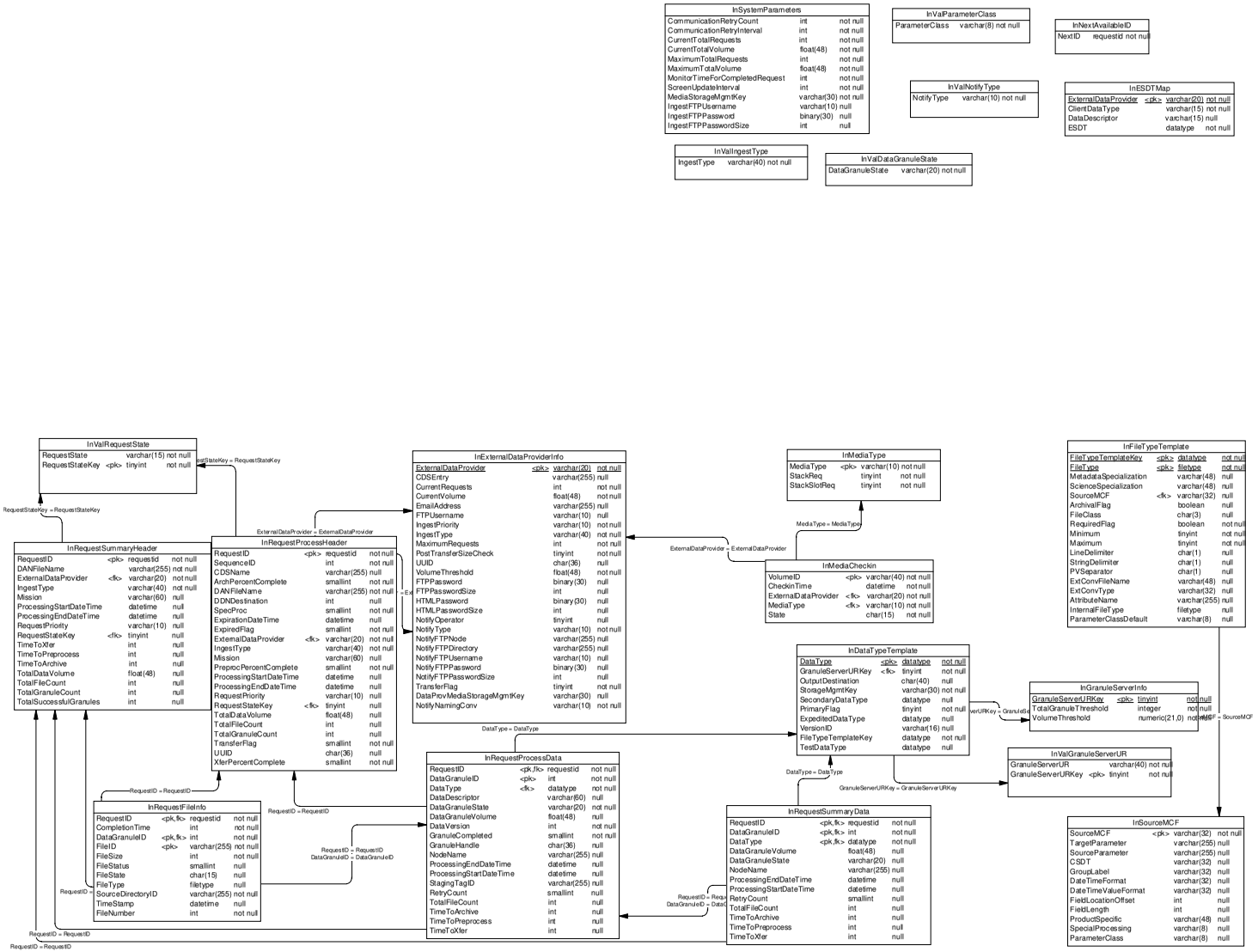

Figure A-1. Ingest ERD

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
SUBSRV	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.