311-CD-104-005

EOSDIS Core System Project

Release 4
Interoperability Subsystem (IOS) Database Design and Database Schema Specifications for the ECS Project

Final

March 1999

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

Raytheon Systems Company
Upper Marlboro, Maryland

Release 4
Interoperability Subsystem (IOS)
Database Design and Database Schema Specifications for the ECS Project

Final

March 1999

Prepared Under Contract NAS5-60000
CDRL Item #050

RESPONSIBLE ENGINEER

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

SUBMITTED BY

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

Raytheon Systems Company
Upper Marlboro, Maryland

311-CD-104-005

This page intentionally left blank.

Preface

This document describes the database design and database schema specifications for the Interoperability Subsystem (IOS). It is one of eight documents comprising the detailed database design and database schema specifications for the as-delivered ECS subsystems. A complete list of the eight documents follows:

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

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

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

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

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

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

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

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

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

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

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

Abstract

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

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

This page intentionally left blank.

Change Information Page

List of Effective Pages	
Page Number	**Issue**
Title	Submitted as Final
iii through xii	Submitted as Final
1-1 through 1-2	Submitted as Final
2-1 through 2-2	Submitted as Final
3-1 through 3-2	Submitted as Final
4-1 through 4-72	Submitted as Final
5-1 through 5-2	Submitted as Final
6-1 through 6-4	Submitted as Final
7-1 through 7-2	Submitted as Final
AB-1 through AB-2	Submitted as Final

Document History			
Document Number	**Status/Issue**	**Publication Date**	**CCR Number**
311-CD-104-001	Draft	January 1998	97-1755
311-CD-104-002	Draft	May 1998	98-0620
311-CD-104-003	Draft	July 1998	98-0866
311-CD-104-004	Draft	December 1998	98-1267
311-CD-104-005	Submitted as Final	March 1999	99-0166

This page intentionally left blank.

Contents

Preface

Abstract

1. Introduction

1.1 Identification 1-1

1.2 Scope 1-1

1.3 Purpose 1-1

1.4 Audience 1-1

2. Related Documents

2.1 Applicable Documents 2-1

2.2 Information Documents 2-2

3. Database Configurations

3.1 Server Configurations 3-1

3.2 Storage Device Layouts 3-1

4. Database Design

4.1 Design Overview 4-1

4.1.1 Physical Data Model Entity Relationship Diagram 4-1

4.1.2 Database Table Specifications 4-5

4.1.3 Column Specifications 4-21

4.1.4 Column Domains 4-41

4.1.5 Column Default Values 4-41

4.1.6 Referential Integrity Rules 4-41
4.1.7 Views....4-41
4.1.8 Declarative Integrity Constraints 4-41
4.1.9 Triggers 4-46
4.1.10 Stored Procedures....4-54
4.2 Flat File Usage 4-71
4.2.1 File Descriptions 4-71
4.2.2 Field Specifications....4-71
4.2.3 Domain Definitions....4-71

5. Performance and Tuning Factors

5.1 Indexes 5-1
5.2 Segments 5-1
5.3 Named Caches....5-2

6. Database Security

6.1 Approach 6-1
6.2 Initial Users 6-2
6.3 Groups 6-2
6.4 Objects....6-2
6.5 Roles....6-3

7. Scripts

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

Figures

4-1 ERD Key 4-2

4-2 IOS ERD 4-3
6-1 Sybase General Approach to SQL Server Security 6-1

Tables

4-1 Database Tables 4-5
4-2 Trigger Descriptions 4-47
4-3 Summary List of Stored Procedures 4-55
5-1 Segment Indexes 5-1
5-2 Segment Descriptions 5-1
6-1 Permission Key 6-2
6-2 Group Specifications 6-2
7-1 Installation Scripts 7-1
7-2 De-Installation Scripts 7-1
7-3 Backup and Recovery Scripts 7-2
7-4 Miscellaneous Scripts 7-2

Abbreviations and Acronyms

This page intentionally left blank.

1. Introduction

1.1 Identification

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

1.2 Scope

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

1.3 Purpose

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

1.4 Audience

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

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

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

Section 3 specifies the IOS database physical architecture.

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

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

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

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

This page is left intentionally blank.

2. Related Documents

2.1 Applicable Documents

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

305-CD-100	Release 4 Segment Design Specification for the ECS Project
920-TDG-009	DAAC Hardware Database Mapping/GSFC
920-TDN- 009	DAAC Hardware Database Mapping/NSIDC
920-TDE-009	DAAC Hardware Database Mapping/EDC
920-TDL-009	DAAC Hardware Database Mapping/LARC
920-TDS-009	DAAC Hardware Database Mapping/SMC
920-TDG-010	DAAC Database Configuration/GSFC
920-TDN-010	DAAC Database Configuration/NSIDC
920-TDE-010	DAAC Database Configuration/EDC
920-TDL-010	DAAC Database Configuration/LARC
920-TDS-010	DAAC Database Configuration/SMC
920-TDG-011	DAAC Sybase Log Mapping/GSFC
920-TDN-011	DAAC Sybase Log Mapping/NSIDC
920-TDE-011	DAAC Sybase Log Mapping/EDC
920-TDL-011	DAAC Sybase Log Mapping/LARC
920-TDS-011	DAAC Sybase Log Mapping/SMC
922-TDG-013	Disk Partitions/GSFC
922-TDN-013	Disk Partitions/NSIDC
922-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-004	Release 4 Mission Operation Procedures for the ECS Project

These documents are accessible via the EDHS homepage.

3. Database Configurations

3.1 Server Configurations

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

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

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

3.2 Storage Device Layouts

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

922-TDG-013	Disk Partitions/GSFC
922-TDN-013	Disk Partitions/NSIDC
922-TDE-013	Disk Partitions/EDC
922-TDL-013	Disk Partitions/LARC
922-TDS-013	Disk Partitions/SMC

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

4. Database Design

4.1 Design Overview

The IOS database implements a majority of the persistent data requirements for the IOS. Other non-database data requirements, used for system support, are implemented in flat files-see Section 4.2 for descriptions of these flat files. The database is designed to satisfy IOS business rules while maintaining data integrity, consistency, and performance. Database tables are implemented using the Sybase Relational Database Management System (RDBMS) Version 11.0.2.2. All components of the IOS database are described in the following sections; information is presented in sufficient detail to support operational needs.

4.1.1 Physical Data Model Entity Relationship Diagram

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

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

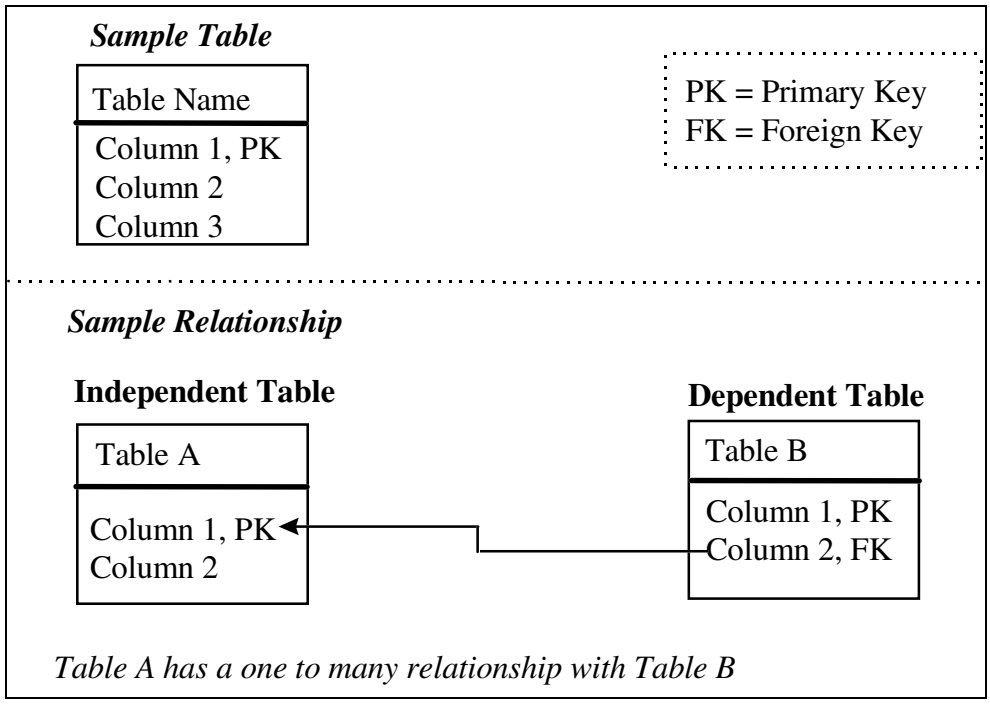

Figure 4-1. ERD Key

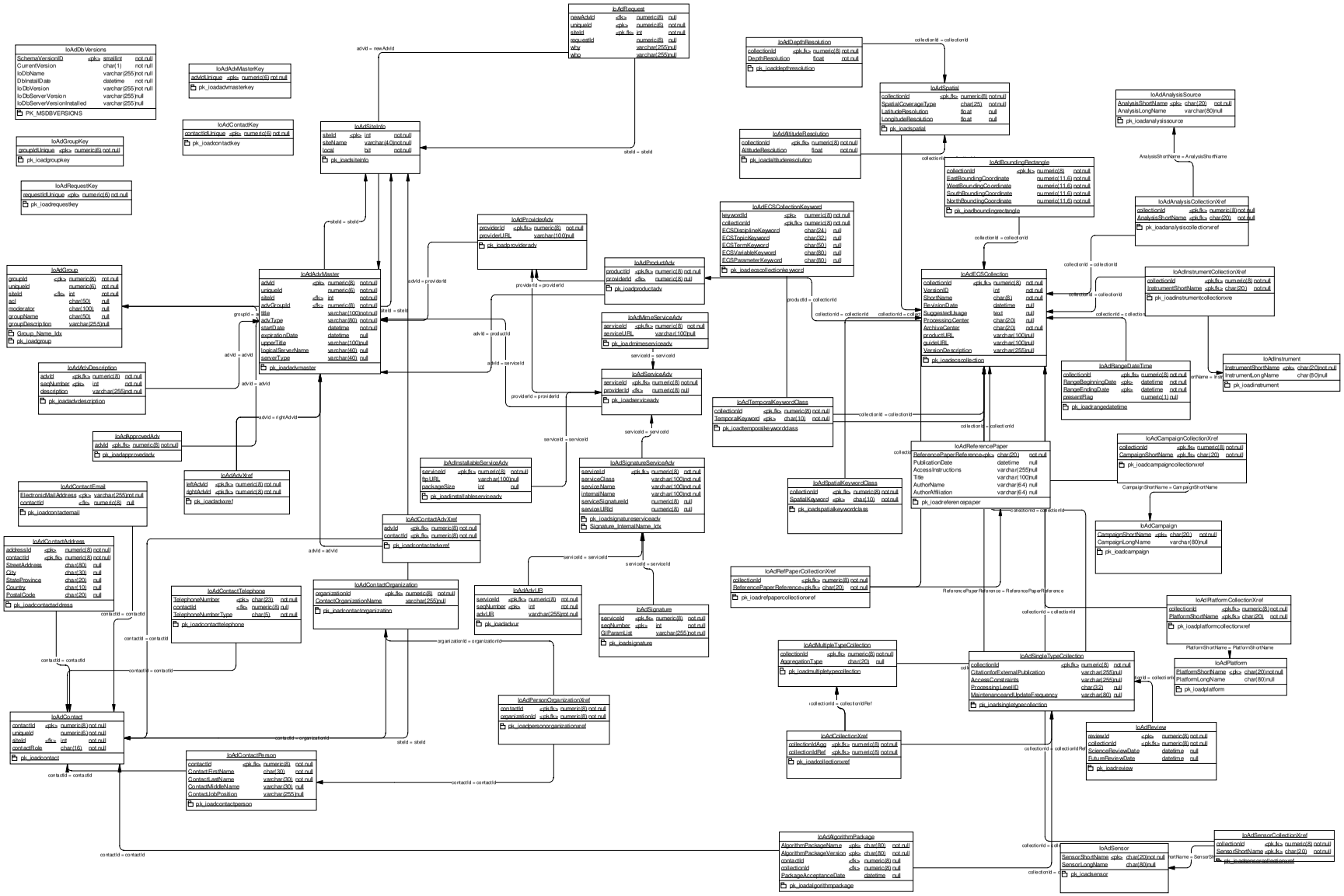

Figure 4-2. IOS ERD

4.1.2 Database Table Specifications

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

Table 4-1. Database Tables

Database Table Name	Database Table Name
IoAdAdvMaster	IoAdInstallableServiceAdv
IoAdAdvMasterKey	IoAdInstrument
IoAdAdvUR	IoAdInstrumentCollectionXref
IoAdAdvXref	IoAdMimeServiceAdv
IoAdAlgorithmPackage	IoAdMultipleTypeCollection
IoAdAltitudeResolution	IoAdPersonOrganizationXref
IoAdAnalysisCollectionXref	IoAdPlatform
IoAdAnalysisSource	IoAdPlatformCollectionXref
IoAdApprovedAdv	IoAdProductAdv
IoAdBoundingRectangle	IoAdProviderAdv
IoAdCampaign	IoAdRangeDateTime
IoAdCampaignCollectionXref	IoAdReferencePaper
IoAdCollectionXref	IoAdRefPaperCollectionXref
IoAdContact	IoAdRequest
IoAdContactAddress	IoAdRequestKey
IoAdContactAdvXref	IoAdReview
IoAdContactEmail	IoAdSensor
IoAdContactKey	IoAdSensorCollectionXref
IoAdContactOrganization	IoAdServiceAdv
IoAdContactPerson	IoAdSignature
IoAdContactTelephone	IoAdSignatureServiceAdv
IoAdDepthResolution	IoAdSingleTypeCollection
IoAdECSCollection	IoAdSiteInfo
IoAdECSCollectionKeyword	IoAdSpatial
IoAdGroup	IoAdSpatialKeywordClass
IoAdGroupKey	IoAdTemporalKeywordClass

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

Table: IoAdAdvMaster

Description: This table stores the general information of approved advertisements.

Column List:

Name	Type	P	M
AdvGroupId	numeric(8)	No	Yes
AdvId	numeric(8)	Yes	Yes
AdvType	varchar(80)	No	Yes
ExpirationDate	datetime	No	No
LogicalServerName	varchar(40)	No	No
ServerType	varchar(40)	No	No
SiteId	int	No	Yes
StartDate	datetime	No	Yes
Title	varchar(100)	No	Yes
UniqueId	numeric(6)	No	Yes
UpperTitle	varchar(100)	No	No

Table: IoAdAdvMasterKey

Description: This is a look up table to maintain the highest value of advId.

Column List:

Name	Type	P	M
advIdUnique	numeric(6)	Yes	Yes

Table: IoAdAdvUR

Description: This table stores the address of the service advertisement universal reference.

Column List:

Name	Type	P	M
advUR	varchar(255)	No	Yes
seqNumber	int	Yes	Yes
serviceId	numeric(8)	Yes	Yes

Table: IoAdAdvXref

Description: This table maintains the references between product advertisement and service advertisement.

Column List

Name	Type	P	M
leftAdvId	numeric(8)	Yes	Yes
rightAdvId	numeric(8)	Yes	Yes

Table: IoAdAlgorithmPackage

Description: This table stores the algorithm package information associated with the single type collections.

Column List:

Name	Type	P	M
AlgorithmPackageVersion	char(80)	Yes	Yes
AlgorithmPakageName	char(80)	Yes	Yes
collectionId	numeric(8)	No	No
contactId	numeric(8)	No	No
PackageAcceptanceDate	datetime	No	No

Table: IoAdAltitudeResolution

Description: This table stores the altitude resolution associated with the ECS collections.

Column List:

Name	Type	P	M
AltitudeResolution	float	No	Yes
collectionId	numeric(8)	Yes	Yes

Table:IoAdAnalysisCollectionXref

Description: This table maintains the cross-reference between analysis sources and the ECS collections.

Column List:

Name	Type	P	M
AnalysisShortName	char(20)	Yes	Yes
collectionId	numeric(8)	Yes	Yes

Table: IoAdAnalysisSource

Description: This table defines analysis sources germane to ECS.

Column List:

Name	Type	P	M
AnalysisLongName	varchar(80)	No	No
AnalysisShortName	char(20)	Yes	Yes

Table: IoAdApprovedAdv

Description: This table identifies approved advertisements.

Column List:

Name	Type	P	M
advId	numeric(8)	Yes	Yes

Table: IoAdBoundingRectangle

Description: This table stores the spatial area coverage for ECS collection.

Column List:

Name	Type	P	M
CollectionId	numeric(8)	Yes	Yes
EastBoundingCoordinate	numeric(11,6)	No	Yes
NorthBoundingCoordinate	numeric(11,6)	No	Yes
SouthBoundingCoordinate	numeric(11,6)	No	Yes
WestBoundingCoordinate	numeric(11,6)	No	Yes

Table: IoAdCampaign

Description: This table stores the scientific endeavors to which the collection is associated.

Column List:

Name	Type	P	M
CampaignLongName	varchar(80)	No	No
CampaignShortName	char(20)	Yes	Yes

Table:IoAdCampaignCollectionXref

Description: This table maintains the cross-reference between campaign and ECS collection.

Column List:

Name	Type	P	M
CampaignShortName	char(20)	Yes	Yes
CollectionId	numeric(8)	Yes	Yes

Table: IoAdCollectionXref

Description: This table contains the cross-reference between single type collection and multiple type collection.

Column List:

Name	Type	P	M
collectionIdAgg	numeric(8)	Yes	Yes
collectionIdRef	numeric(8)	Yes	Yes

Table: IoAdContact

Description: This table stores the contact information of the advertisement.

Column List:

Name	Type	P	M
contactId	numeric(8)	Yes	Yes
contactRole	char(16)	No	Yes
siteId	int	No	Yes
uniqueId	numeric(6)	No	Yes

Table: IoAdContactAddress

Description: This table stores the address of points of contact for the advertisement.

Column List:

Name	Type	P	M
addressId	numeric(8)	Yes	Yes
City	char(30)	No	No
contactId	numeric(8)	Yes	Yes
Country	char(10)	No	No
PostalCode	char(20)	No	No
StateProvince	char(20)	No	No
StreetAddress	char(80)	No	No

Table: IoAdContactAdvXref

Description: This table contains the cross-reference between contact and advertisement.

Column List:

Name	Type	P	M
advId	numeric(8)	Yes	Yes
contactId	numeric(8)	Yes	Yes

Table: IoAdContactEmail

Description: This table stores the Email address of the points of contact for the advertisement.

Column List:

Name	Type	P	M
contactId	numeric(8)	No	No
ElectronicMailAddress	varchar(255)	Yes	Yes

Table: IoAdContactKey

Description: This is a look up table to maintain the highest value of the contactId.

Column List:

Name	Type	P	M
contactIdUnique	numeric(6)	Yes	Yes

Table:IoAdContactOrganization

Description: This table stores the name of the organizations.

Column List:

Name	Type	P	M
ContactOrganizationName	varchar(255)	No	No
organizationId	numeric(8)	Yes	Yes

Table: IoAdContactPerson

Description: This table stores the name of the points of contact for the advertisement.

Column List:

Name	Type	P	M
ContactFirstName	char(30)	No	Yes
contactId	numeric(8)	Yes	Yes
ContactJobPosition	varchar(255)	No	No
ContactLastName	varchar(30)	No	Yes
ContactMiddleName	varchar(30)	No	No

Table: IoAdContactTelephone

Description: This table stores the phone information of the points of contact for the advertisement.

Column List:

Name	Type	P	M
contactId	numeric(8)	No	No
TelephoneNumber	char(23)	Yes	Yes
TelephoneNumberType	char(5)	No	Yes

Table: IoAdDepthResolution

Description: This table stores the depth resolution associated with ECS collections.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
DepthResolution	float	No	Yes

Table: IoAdECSCollection

Description: This table stores the ECS collection associated to the product advertisement.

Column List:

Name	Type	P	M
ArchiveCenter	char(20)	No	Yes
collectionId	numeric(8)	Yes	Yes
guideURL	varchar(100)	No	No
ProcessingCenter	char(20)	No	No
productURL	varchar(100)	No	No
RevisionDate	datetime	No	No
ShortName	char(8)	No	Yes
SuggestedUsage	text	No	No
VersionID	int	No	Yes
VersionDescription	varchar(255)	No	No

Table:IoAdECSCollectionKeyword

Description: This table stores all types of keywords used in searching ECS collections.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
ECSDisciplineKeyword	char(24)	No	No
ECSParameterKeyword	char(80)	No	No
ECSTermKeyword	char(50)	No	No
ECSTopicKeyword	char(32)	No	No
ECSVariableKeyword	char(80)	No	No
keywordId	numeric(8)	Yes	Yes

Table: IoAdGroup

Description: This table stores the moderation group for the advertisement.

Column List:

Name	Type	P	M
acl	char(50)	No	No
groupDescription	varchar(255)	No	No
groupId	numeric(8)	Yes	Yes
groupName	char(50)	No	No
moderator	char(50)	No	No
siteId	int	No	Yes
uniqueId	numeric(6)	No	Yes

Table: IoAdGroupKey

Description: This is a look up table to maintain the highest value of the groupId currently being used.

Column List:

Name	Type	P	M
groupIdUnique	numeric(6)	Yes	Yes

Table:IoAdInstallableServiceAdv

Description: This table stores the software information which is installed to the client site upon request.

Column List:

Name	Type	P	M
ftpURL	varchar(100)	No	No
packageSize	int	No	No
serviceId	numeric(8)	Yes	Yes

Table: IoAdInstrument

Description: This table stores the instrument information associated with ECS collection.

Column List:

Name	Type	P	M
InstrumentLongName	char(80)	No	No
InstrumentShortName	char(20)	Yes	Yes

Table:IoAdInstrumentCollectionXref

Description: This table maintains the cross-reference between ECS collection and Instrument.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
InstrumentShortName	char(20)	Yes	Yes

Table: IoAdMimeServiceAdv

Description: This table stores the MIME (Multipurpose Internet Mail Extension) service advertisement.

Column List:

Name	Type	P	M
serviceId	numeric(8)	Yes	Yes
serviceURL	varchar(100)	No	No

Table:IoAdMultipleTypeCollection

Description: This table stores the multiple type ECS collections.

Column List:

Name	Type	P	M
AggregationType	char(20)	No	No
collectionId	numeric(8)	Yes	Yes

Table:IoAdPersonOrganizationXref

Description: This table maintains the cross-reference between Contact Person and Contact Organization.

Column List:

Name	Type	P	M
ContactId	numeric(8)	Yes	Yes
OrganizationId	numeric(8)	Yes	Yes

Table: IoAdPlatform

Description: This table stores the platforms associated with the acquisition of the collection.

Column List:

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

Table:IoAdPlatformCollectionXref

Description: This table maintains the cross-reference between Platform and ECS collection.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
PlatformShortName	char(20)	Yes	Yes

Table: IoAdProductAdv

Description: This table stores the product advertisement.

Column List:

Name	Type	P	M
productId	numeric(8)	Yes	Yes
providerId	numeric(8)	No	No

Table:IoAdProviderAdv

Description: This table stores the provider advertisement.

Column List:

Name	Type	P	M
providerId	numeric(8)	Yes	Yes
providerURL	varchar(100)	No	No

Table:IoAdRangeDateTime

Description: This table stores the start/end time of a collection.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
presentFlag	numeric(1)	No	No
RangeBeginningDate	datetime	Yes	Yes
RangeEndingDate	datetime	Yes	Yes

Table: IoAdReferencePaper

Description: This table stores the reference document of a collection.

Column List:

Name	Type	P	M
AccessInstructions	varchar(255)	No	No
AuthorAffiliation	varchar(64)	No	No
AuthorName	varchar(64)	No	No
PublicationDate	datetime	No	No
ReferencePaperReference	char(20)	Yes	Yes
Title	varchar(100)	No	No

Table:IoAdRefPaperCollectionXref

Description: This table maintains the cross-reference between Reference Paper and ECS collection.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
ReferencePaperReference	char(20)	Yes	Yes

Table: IoAdRequest

Description: This table stores the information that external providers make a request to change the current approved advertisement.

Column List:

Name	Type	P	M
newAdvId	numeric(8)	Yes	Yes
oldAdvId	numeric(8)	No	No
oldAdvModTime	datetime	No	No
requestId	numeric(8)	No	No
requestTime	datetime	No	No
siteId	int	No	Yes
type	int	No	Yes
uniqueId	numeric(6)	No	Yes
who	varchar(255)	No	No
why	varchar(255)	No	No

Table: IoAdRequestKey

Description: This is a look up table to maintain the highest value of the new AdvId.

Column List:

Name	Type	P	M
requestIdUnique	numeric(6)	Yes	Yes

Table: IoAdReview

Description: This table stores the dates for the QA peer review for a single type collection.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
FutureReviewDate	datetime	No	No
reviewId	numeric(8)	Yes	Yes
ScienceReviewDate	datetime	No	No

Table: IoAdSensor

Description: This table stores the sensory subcomponents of an instrument.

Column List:

Name	Type	P	M
SensorLongName	char(80)	No	No
SensorShortName	char(20)	Yes	Yes

Table:IoAdSensorCollectionXref

Description: This table maintains the cross-reference between Sensors and ECS collection.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
SensorShortName	char(20)	Yes	Yes

Table: IoAdServiceAdv

Description: This table stores the service advertisement.

Column List:

Name	Type	P	M
providerId	numeric(8)	No	No
serviceId	numeric(8)	Yes	Yes

Table: IoAdSignature

Description: This table stores the signature service associated with the signature service advertisement.

Column List:

Name	Type	P	M
GlParamList	varchar(255)	No	Yes
seqNumber	int	Yes	Yes
serviceId	numeric(8)	Yes	Yes

Table:IoAdSignatureServiceAdv

Description: This table stores the signature service advertisement. Signature service is the C++ operation to perform the service.

Column List:

Name	Type	P	M
internalName	varchar(100)	No	Yes
serviceClass	varchar(100)	No	Yes
serviceId	numeric(8)	Yes	Yes
serviceName	varchar(100)	No	Yes
serviceSignatureId	numeric(8)	No	No
serviceURId	numeric(8)	No	No

Table:IoAdSingleTypeCollection

Description: This table stores the information specific to a single type collection.

Column List:

Name	Type	P	M
AccessConstraints	varchar(255)	No	No
CitationforExternalPublication	varchar(255)	No	No
collectionId	numeric(8)	Yes	Yes
MaintenanceandUpdateFrequency	varchar(80)	No	No
ProcessingLevelID	char(32)	No	No

Table: IoAdSiteInfo

Description: This table stores the DAAC which an advertisement belongs to.

Column List:

Name	Type	P	M
local	bit	No	Yes
siteId	int	Yes	Yes
siteName	varchar(40)	No	Yes

Table: IoAdSpatial

Description: This table stores the spatial extent of ECS collection.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
LatitudeResolution	float	No	No
LongitudeResolution	float	No	No
SpatialCoverageType	char(25)	No	Yes

Table: IoAdSpatialKeywordClass

Description: This table stores the keyword of spatial region for ECS collection.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
SpatialKeyword	char(10)	Yes	Yes

Table: IoAdTemporalKeywordClass

Description: This table stores the keyword of temporal characteristics for ECS collection.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
TemporalKeyword	char(10)	Yes	Yes

4.1.3 Column Specifications

Brief definitions of each of the columns[1] within the IOS database and their applicable valid values, or references to other documents containing the valid values, are contained herein.

Column: AccessConstraints

Description: Restrictions and legal prerequisites for accessing the collection. These include any access constraints applied to assure the protection of privacy or intellectual property, and any special restrictions or limitations on obtaining the collection. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: AccessInstructions

Description: Instructions describing how to obtain electronic access to a stand-alone document. May simply be an anonymous ftp site address, or a World Wide Web homepage URL.

Column: acl

Description: The name of the access control list used by the moderator to create/update/delete approved advertisement for the given group.

Column: addressId

Description: The unique ID which identifies the address of the contact.

Column: advGroupId

Description: The IDs which uniquely identify the group that advertisement belong to.

---

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

Column: advId

Description: An ID which uniquely identifies an advertisement.

Note: AdvId is generated from siteId and uniqueId. Refer to stored procedure "IoAdgetAvdMasterKey".

Column: advIdUnique

Description: The last value of advId used.

Note: This key is used by the store procedure "IoAdAdvMasterKey" to generate the next advId in sequence.

Column: advType

Description: The type of advertisement.

Column: advUR

Description: The address of the service advertisement UR.

Column: AggregationType

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

Valid Values: See 420-TP-015

Column: AlgorithmPackageVersion

Description: This attribute specifies the version of the full package being delivered. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: AlgorithmPackageName

Description: This attribute is the name given to the complete delivered package submitted for algorithm integration and test. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: AltitudeResolution

Description: The minimum distance possible between two adjacent altitude values, expressed in Altitude Distance Units of measure. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015.

Column: AnalysisLongName

Description: The expanded or long name of the analysis source identified using AnalysisShortName. AnalysisLongName is intended to categorize collections by the processes which collected (e.g., census survey) or produced them (e.g., NMC 16-level Nested Grid Model). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: AnalysisShortName

Description: AnalysisShortname is the unique identifier of the collection or anlysis process(s) which best characterize the ECS Collection or Granule. ECS Collections or Granules may be characterized by both a collection and an analysis data set which include data collected using the NWS ASOS network (PlatformType = Network, PlatformShortName = ASOS) which was processed using an NMC analysis model (e.g., AnalysisType = Model, AnalysisShortName = RAFS, AnalysisDescription = Regional Area Forecast System, AnalysisTechinque = Regional Optimal Interpolation.) This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ArchiveCenter

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

Valid Values: See 420-TP-015

Column: AuthorAffiliation

Description: The name of an agency or center with which the author of the document works for or is affiliated with.

Column: AuthorName

Description: The name of the author of the reference paper.

Column: CampaignLongName

Description: The expanded name of the campaign/experiment (e.g., Global Climate Observing System). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: CampaignShortName

Description: The unique identifier by which a campaign/project/experiment is known. The campaign/project is the scientific endeavor associated with the acquisition of the collection. Collections may be associated with multiple campaigns. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: CitationforExternalPublication

Description: The recommended reference to be used when referring to this collection in publications. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: City

Description: The city of the person or organization of the point of contact. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: collectionId

Description: The unique ID which identifies an ECS collection. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: collectionIdAgg

Description: The aggregate type of collections.

Column: collectionIdRef

Description: The collection which is included in the collectionIdAgg.

Column: ContactFirstName

Description: First name of the individual to which the contact role (producer, archiver, distributor, or data originator) applies. People are points of contact, rather than organization, in cases where the association of the person to the data set is more significant than the association of the organization to the data set. They may also be included if both a single person and organization are provided as points of contact. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: contactId

Description: The unique ID which identifies the contact information.

Note: contacId is generated from uniqueId and siteId. Refer to the store procedure "IoAdGetContactKey". This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: contactIdUnique

Description: The highest value of the ContactId that is currently used.

Column: ContactJobPosition

Description: The title of the individual (i.e., Team Leader, Principal Investigator). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ContactLastName

Description: Last name of the individual with the contact role (producer, archiver, distributor, or data originator) applies. People are points of contact, rather than organization, in cases where the association of the person to the data set is more significant than the association of the organization to the data set. They may also be included if both a single person and organization are provided as points of contact. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ContactMiddleName

Description: The middle name of the point of contact. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ContactOrganizationName

Description: This is the name of the organization. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: contactRole

Description: Classification of individuals who are associated with a given collection. This is an SDSRV Subsystem column "ContactRole."

Valid Values: See 420-TP-015

Column: Country

Description: The country of the address of the point of contact. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: DepthResolution

Description: The minimum distance possible between two adjacent depth values, expressed in depth distance units of measure - mandatory. This is an SDSRV Subsystem column.

Valid Values: >0.0, See 420-TP-015

Column: EastBoundingCoordinate

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

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

Column: ECSDisciplineKeyword

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

Valid Values: See 420-TP-015

Column: ECSParameterKeyword

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

Valid Values: See 420-TP-015

Column: ECSTermKeyword

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

Valid Values: See 420-TP-015

Column: ECSTopicKeyword

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

Valid Values: See 420-TP-015

Column: ECSVariableKeyword

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

Valid Values: See 420-TP-015

Column:ElectronicMailAddress

Description: The address of the electronic mailbox of the organization or individual. The address, following NASA Global Change Master Directory format, should be of the form 'network name>network address'. Examples of network names are NSN, SPAN, telemail, ARPANET, and Internet. Examples of network address are NSSDCA::NG.MIKEMARTIN/NASA, MMARTIN@JPL.MILVAX, or miken@eos.hitc.com. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: expirationDate

Description: Date that the advertisement will be expired.

Column: ftpURL

Description: The address of FTP site from where the software can be downloaded.

Column: FutureReviewDate

Description: Date of next planned AQ peer review. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: GlParamList

Description: The compressed list contains metadata information.

Column: groupDescription

Description: The brief description of the group.

Column: groupId

Description: The unique IDs which identify the group.

Note: GroupId is generated from uniqueId and SiteId. Refer to the stored procedure "IoAdGetGroupKey".

Column: groupIdUnique

Description: The highest value of the groupId.

Column: groupName

Description: This attribute is used to tie a “kind” of metadata to physical processing in the DBWrappers. For example: “QAGranule” relates the DsMdQaGranules table for a function (Physical Delete) and sequence of SQL code (EXEC DsDbProcDeleteQaGranule) to process the data. This is an SDSRV Subsystem column “GroupName.”

Valid Values: See 420-TP-015

Column: guideURL

Description: The address of UR where the guide information for the collection is stored.

Note: The data origin of this attribute has not been determined. Potential source is Document Data Server.

Column: InstrumentLongName

Description: The expanded name of the primary sensory instrument (e.g., Advanced Spaceborne Thermal Emission and Reflective Radiometer). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: InstrumentShortName

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

Valid Values: See 420-TP-015

Column: internalName

Description: The type of service for an instrument.

Column: keywordId

Description: The unique IDs which identify the keyword.

Column: LatitudeResolution

Description: The minimum difference between two adjacent latitude values expressed in Geographic Coordinate Units of measure. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: leftAdvId

Description: The advId which represents a product advertisement.

Column: local

Description: The Boolean value to identify the local DAAC.

Column: logicalServerName

Description: The logical server name is the cross-reference name to the physical server where the advertisement data is located. It is a static identification to the physical server which may vary.

Valid Values: "sdsrv@gsfc", "sdsrv@LaRc", "sdsrv@edc", "sbsrv@nsidc", "sbsrv@gsfc", "sbsrv@LaRc", "sbsrv@edc", "dim@nsidc", "dim@gsfc", "dim@LaRc", "dim@edc", "dim@nsidc"

Column: LongitudeResolution

Description: The minimum difference between two adjacent longitude values expressed in Geographic Coordinate Units of Measure. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: MaintenanceandUpdateFrequency

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

Valid Values: See 420-TP-015

Column: moderator

Description: The name of the moderator assigned to the group.

Column: newAdvId

Description: The unique ID which identifies the new approved advertisement specified in additions or modifications.

Valid Values: See AdvId valid values.

Column: NorthBoundingCoordinate

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

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

Column: oldAdvId

Description: The old approved advertisement specified in deletion or modification.

Valid Values: See AdvId valid values.

Column: oldAdvModTime

Description: The modification time of the old approved advertisement.

Column: organizationId

Description: The unique ID which identifies the contact organization. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: PackageAcceptanceDate

Description: The date that the particular version of the package is accepted.

Column: packageSize

Description: The size of the software in kilobytes.

Column: PlatformLongName

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

Valid Values: See 420-TP-015

Column: PlatformShortName

Description: The unique identifier of a specific platform (e.g., AM1, TRMM, PM). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: PostalCode

Description: The zip or other postal code of the address of the point of contact. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: presentFlag

Description: If this is 1, then it means the RangeEndingDate is present in the table, IoAdRangeDateTime. If this is 0, then the effective RangeEndingDate is the current date.

Column: ProcessingCenter

Description: Center where collection was or is being processed (i.e., name of DAAC or SCF). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ProcessingLevelID

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

Valid Values: See 420-TP-015

Column: productId

Description: The unique ID which identifies the product advertisement.

Note: The productId is equivalent to AdvId in the IoAdAdvMaster table.

Valid Values: See AdvId valid values.

Column: productURL

Description: The address of UR where providers store the product information.

Note: This field is optional. Providers can enter the UR location along with product advertisement submission.

Column: providerId

Description: The unique ID which identifies a provider advertisement.

Note: The providerId is equivalent to advId in the IoAdvMaster table.

Column: providerURL

Description: The address of UR which stores the provider advertisement.

Column: PublicationDate

Description: The date of formal or informal publication of the reference paper.

Column: RangeBeginningDate

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

Valid Values: See 420-TP-015

Column: RangeEndingDate

Description: The last year (and optionally month, or month and day) of the temporal coverage period being described (GSFC AVHRR). This date represents the end date of the latest granule contained in the product. This is an SDSRV Subsystem column.

Note: MM/DD/YY format is product specific for sage_atmos_dyn, sage_atmos_comp, erbe_erp. MMDDYYYY format is product specific for LARC_FIRE, LARC_GTE

Valid Values: See 420-TP-015

Column: ReferencePaperReference

Description: Contains the unique ID of the Reference Paper is issued by publisher, such as 'NOS NSG', or 'JPL Publication 91-29'.

Column: requestId

Description: The unique ID assigned to each DAAC.

Column: requestIdUnique

Description: The highest value of the new AdvId which is used currently.

Column: requestTime

Description: This is the time of the request.

Column: reviewId

Description: The unique ID which identifies the review for the given collection.

Column: RevisionDate

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

Valid Values: See 420-TP-015

Column: rightAdvId

Description: The advId which represents a service advertisement.

Column: ScienceReviewDate

Description: Date of last QA peer review.

Column: SensorLongName

Description: The expanded name of the sensor. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: SensorShortName

Description: A sensor is defined sensory sub-component of an instrument (e.g., InstrumentShortName = ASTER, NumberofSensors = 3, SensorShortName = SWIR, SensorShortName = TR, SensorShortName = VNIR). This is an SDSRV Subsystem column.

Note: If the instrument has only one sensor or the sensor is an instrument (e.g., AVHRR), the information should go to both Sensor table and Instrument table.

Valid Values: See 420-TP-015

Column: seqNumber

Description: Sequence number of line in which the address of service UR is stored.

Column: serverType

Description: This data represents the name of the server that contains the advertising data from a particular source.

Valid Values: "SDSRV", "SBSRV", "DIM"

Column: serviceClass

Description: The name of the C++ class the operation belongs to. This is an SDSRV Subsystem column (ServiceClass).

Valid Values: See 420-TP-015

Column: serviceId

Description: The unique ID which identifies the Mime service advertisement. This is an SDSRV Subsystem column (ServiceId).

Note: The Mime service is the Internet (WWW) service.

Valid Values: See 420-TP-015

Column: serviceName

Description: The name of the C++ an operation. This is an SDSRV Subsystem column (ServiceName).

Valid Values: See 420-TP-015

Column: serviceSignatureId

Description: The ID which identifies the signature service.

Column: serviceURId

Description: The ID which identifies the address of service UR.

Column: serviceURL

Description: The address of UR for the Mime service.

Note: MIME service is the Internet (WWW) service.

Column: ShortName

Description: This name will identify the short name associated with the collection or granule. This includes the ECS Technical Baseline product names (e.g., CERO2, MOD12). This is the official reference name used in identifying the contents of the data collection. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: siteId

Description: The numeric ID assigned to the DAAC.

Note: The value of the siteId varies depending on the home DAAC. The value of the siteId for the home DAAC is always 1.

Column: siteName

Description: The name of the DAAC.

Column: SouthBoundingCoordinate

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

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

Column: SpatialCoverageType

Description: This attribute denotes whether the locality/coverage requires horizontal, vertical, or both in the spatial domain and coordinate system definitions. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: SpatialKeyword

Description: This attribute specifies a word or phrase which serves to summarize the spatial regions covered by the collection. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: startDate

Description: Date that the advertisement become valid. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: StateProvince

Description: The state or province of the address of the point of contact. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: StreetAddress

Description: The street of the address of the point of contact. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: SuggestedUsage

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

Number of organization or individual who is point of contact. The general format of the number includes country, area, and STD codes, as required for the full telephone number. Multi-extensions should be single entries rather than part of a single entry text.

Valid Values: See 420-TP-015

Column: TelephoneNumber

Description: The telephone number of the organization or individual identified as the point of contact. The general format of the number includes country, area and STD codes, as required for the full telephone number. Multi-extensions should be single entries rather than part of a single entry text. This is an SDSRV Subsystem column

Valid Values: See 420-TP-015

Column: TelephoneNumberType

Description: The type of telephone number being provided in this instance of the phone number, in order to reach the organization or individual who serves as a point of contact. Voice number is used to speak to the organization or individual, the TDD/TTY number which hearing-impaired can converse with organization or individual, or the fa(x)csimile of the organization or individual. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: TemporalKeyword

Description: This attribute specifies a word or phrase which serves to summarize the temporal characteristics referenced in the collection. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: Title

Description: The full title of the reference paper. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: title

Description: The full title of the advertisement.

Column: type

Description: The type of request.

Column: uniqueId

Description: The IDs which uniquely identify an advertisement within the local site.

Column: upperTitle

Description: The name of the advertisement in upper case characters.

Column: VersionDescription

Description: Version description of the data collection.

Column: VersionID

Description: Version identifier of the data collection. This is an SDSRV Subsystem column.

Note: The default is "1"

Valid Values: See 420-TP-015

Column: WestBoundingCoordinate

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

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

Column: who

Description: The name of the person or organization who made the request.

Column: why

Description: The explanation of why the change is being requested.

4.1.4 Column Domains

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

4.1.5 Column Default Values

A default value is used to supply a value for a column when one is not defined at row insert time. Values default to null values where no other specification is identified for the column in Section 4.1.3; reference 311-CD-107-002 for those columns marked as replicated columns from the SDSRV Subsystem.

4.1.6 Referential Integrity Rules

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

4.1.7 Views

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

4.1.8 Declarative Integrity Constraints

Sybase version 11.0.2.2 allows the enforcement of referential integrity via the use of declarative integrity constraints. Integrity constraints allow the SQL server to enforce primary and foreign key integrity checks automatically without requiring programming. Sybase 11 is ANSI-92 compliant, therefore, its constraints support “restrict-only” operations. This means that a row can not be deleted or the key values modified if there are rows in other database tables having a foreign key dependency on that row. Cascade delete and update operations cannot be performed if a declarative integrity constraint has been used. The following information describes the declarative integrity constraints contained in the IOS database.

Dependencies on Table: loAdAdvMaster

Reference by List

Referenced by	Primary Key	Foreign Key
loAdAdvXref	advId	RightAdvId
loAdApprovedAdv	advId	AdvId
loAdContactAdvXref	advId	AdvId
loAdProductAdv	advId	ProductId
loAdProviderAdv	advId	ProviderId
loAdRequest	advId	NewAdvId
loAdAdvXref	advId	LeftAdvId
loAdServiceAdv	advId	ServiceId

Dependencies on Table: loAdAnalysisSource

Reference by List

Referenced by	Primary Key	Foreign Key
loAdAnalysisCollectionXref	AnalysisShortName	AnalysisShortName

Dependencies on Table: loAdCampaign

Reference by List

Referenced by	Primary Key	Foreign Key
loAdCampaignCollectionXref	CampaignShortName	CampaignShortName

Dependencies on Table: loAdContact

Reference by List

Referenced by	Primary Key	Foreign Key
loAdContactAddress	contactId	contactId
loAdContactEmail	contactId	contactId
loAdContactOrganization	contactId	organizationId
loAdContactPerson	contactId	contactId
loAdContactTelephone	contactId	contactId
loAdAlgorithmPackage	contactId	contactId
loAdContactAdvXref	contactId	contactId

Dependencies on Table: loAdContactOrganization

Reference by List

Referenced by	Primary Key	Foreign Key
loAdPersonOrganizationXref	organizationId	organizationId

Dependencies on Table: loAdContactPerson

Reference by List

Referenced by	Primary Key	Foreign Key
loAdPersonOrganizationXref	contactId	contactId

Dependencies on Table: loAdECSCollection

Reference by List

Referenced by	Primary Key	Foreign Key
loAdAnalysisCollectionXref	collectionId	collectionId
loAdBoundingRectangle	collectionId	collectionId
loAdCampaignCollectionXref	collectionId	collectionId
loAdECSCollectionKeyword	collectionId	collectionId
loAdMultipleTypeCollection	collectionId	collectionId
loAdRangeDateTime	collectionId	collectionId
loAdRefPaperCollectionXref	collectionId	collectionId
loAdSingleTypeCollection	collectionId	collectionId
loAdSpatial	collectionId	collectionId
loAdSpatialKeywordClass	collectionId	collectionId
loAdTemporalKeywordClass	collectionId	collectionId
loAdInstrumentCollectionXref	collectionId	collectionId
loAdPlatformCollectionXref	collectionId	collectionId
loAdSensorCollectionXref	collectionId	collectionId

Dependencies on Table: loAdGroup

Reference by List

Referenced by	Primary Key	Foreign Key
loAdAdvMaster	groupId	advGroupId

Dependencies on Table: loAdInstrument

Reference by List

Referenced by	Primary Key	Foreign Key
loAdInstrumentCollectionXref	InstrumentShortName	InstrumentShortName

Dependencies on Table: loAdMultipleTypeCollection

Reference by List

Referenced by	Primary Key	Foreign Key
loAdCollectionXref	collectionId	collectionIdRef
loAdCollectionXref	collectionId	collectionIdAgg

Dependencies on Table: loAdPlatform

Reference by List

Referenced by	Primary Key	Foreign Key
loAdPlatformCollectionXref	PlatformShortName	PlatformShortName

Dependencies on Table: loAdProductAdv

Reference by List

Referenced by	Primary Key	Foreign Key
loAdECSCollection	productId	collectionId

Dependencies on Table: loAdProviderAdv

Reference by List

Referenced by	Primary Key	Foreign Key
loAdServiceAdv	providerId	providerId
loAdProductAdv	providerId	providerId

Dependencies on Table: loAdReferencePaper

Reference by List

Referenced by	Primary Key	Foreign Key
loAdRefPaperCollectionXref	ReferencePaperReference	ReferencePaperReference

Dependencies on Table: loAdSensor

Reference by List

Referenced by	Primary Key	Foreign Key
loAdSensorCollectionXref	SensorShortName	SensorShortName

Dependencies on Table: loAdServiceAdv

Reference by List

Referenced by	Primary Key	Foreign Key
loAdInstallableServiceAdv	serviceId	serviceId
loAdMimeServiceAdv	serviceId	serviceId
loAdSignatureServiceAdv	serviceId	serviceId

Dependencies on Table: loAdSignatureServiceAdv

Reference by List

Referenced by	Primary Key	Foreign Key
loAdAdvUR	serviceId	serviceId
loAdSignature	serviceId	serviceId

Dependencies on Table: loAdSingleTypeCollection

Reference by List

Referenced by	Primary Key	Foreign Key
loAdAlgorithmPackage	collectionId	collectionId
loAdReview	collectionId	collectionId
loAdCollectionXref	collectionId	collectionIdRef

Dependencies on Table: IoAdSiteInfo

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdContact	siteId	siteId
IoAdGroup	siteId	SiteId
IoAdAdvMaster	siteId	SiteId
IoAdRequest	siteId	SiteId

Dependencies on Table: IoAdSpatial

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdAltitudeResolution	collectionId	collectionId
IoAdDepthResolution	collectionId	collectionId

4.1.9 Triggers

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

Triggers are implemented at the database table level. Table 4-2 below provides an alphabetically ordered set of triggers used to assure data integrity for the IOS database. This table includes the trigger name, the database table it is related to, and a brief description of the trigger. Table 4-2 is followed immediately by trigger source code listings. Each trigger listing is found in the same order as in Table 4-2.

Table 4-2. Trigger Descriptions

Trigger Name	Related Database Table	Description
loAdApprovedAdvDuplicateCheck	loAdApprovedAdv	Insert and update trigger. Create an error return if the approved ad description conflicts with logically unique information in an existing ad.
loAdGenerateAddressId	loAdContactAddress	Trigger that generates Address Id which is maximum Address Id + 1.
loAdGenerateContactId	loAdContact	Insert trigger. Trigger that generates Contact Id which is concatenation of Site Id + Contact Id
loAdGenerateGroupId	loAdGroup	Trigger that generates Group Id which is Site Id + Group Id.
loAdGenerateKeywordId	loAdECSCollectionKeyword	Trigger that generates Keyword Id which is maximum Keyword Id + 1.
loAdGenerateReviewId	loAdReview	Trigger that generates Review Id which is maximum Review Id + 1.
loAdRequestDuplicateCheck	loAdRequest	Trigger that generates Request Id which is Site Id + Request Id.- Raise an error if this request's ad description conflicts with logically unique information in an existing approved ad.
loAdSetMaster	loAdAdvMaster	Insert trigger. Sets the Advertisement Id which is Site Id + Unique Id and sets the UpperTitle column to uppercase of Title column in the database table
loAdUpperTitle	loAdAdvMaster	Update trigger. Sets the UpperTitle column to uppercase.

Trigger: loAdApprovedAdvDuplicateCheck

Trigger Code

```
--////////////////////////////////////////////////////////
-- Name : loAdApprovedAdvDuplicateCheck
--
-- Description:
--
-- Raise an error if this approved ad description
-- conflicts with logically
-- unique information in an existing ad.
--
-- Implementation Notes:
--////////////////////////////////////

create trigger loAdApprovedAdvDuplicateCheck on loAdApprovedAdv
for insert, update as

   declare @id int
   declare @approved bit

   select @id = advId
   from inserted

   exec loAdAdvDuplicateCheck @old=@id, @new=@id, @approved=1
return
go
```

Trigger: loAdGenerateAddressId

Trigger Code

```
--////////////////////////////////////////////////////////
-- Name: loAdGenerateAddressId
--
-- Description:
--
-- Trigger that generates Address Id which is maximum Address Id + 1
--
-- Implementation Notes:
--////////////////////////////////////////////////////////

create trigger loAdGenerateAddressId on loAdContactAddress
for insert as
   begin

         declare @maxId          AdvIdType

        select @maxId = max(addressId)
                  from loAdContactAddress
           if @maxId = NULL
             select   @maxId = 0
```

```
            update loAdContactAddress
                set addressId = @maxId + 1
                where addressId = 0
        end
return
go
```

Trigger: loAdGenerateContactId

Trigger Code

```
--//////////////////////////////////////////////////////////
-- Name: loAdGenerateContactId
--
-- Description:
--
-- Trigger that generates Contact Id which is Site Id + Contact Id
--
-- Implementation Notes:
--//////////////////////////////////////////////////////////

create trigger loAdGenerateContactId on loAdContact
for insert as
   begin
        declare @siteId         SiteIdType,
                @uniqueId       AdvIdType,
                @contactId      AdvIdType
      declare @error            int
        begin
           select @contactId = 0, @siteId = 0, @uniqueId = 0
           exec loAdGetContactKey @siteId = @siteId output,
                                  @uniqueId = @uniqueId output,
                                  @contactId = @contactId output,
                                  @error = @error output
           if @error != 0
                return
           update loAdContact
                set uniqueId = @uniqueId,
                   contactId = @contactId,
                   siteId = @siteId
                where uniqueId = 0
        end
   end
return
go
```

Trigger: loAdGenerateGroupId

Trigger Code

```
--//////////////////////////////////////////////////////
-- Name: loAdGenerateGroupId
--
-- Description:
--
-- Trigger that generates Group Id which is Site Id + Group Id
--
-- Implementation Notes:
--//////////////////////////////////////////////////////

create trigger loAdGenerateGroupId on loAdGroup
for insert as
   begin
         declare @siteId          SiteIdType,
                 @uniqueId        AdvIdType,
                 @groupId         AdvIdType
      declare @error              int
         begin
            select @groupId = 0, @siteId = 0, @uniqueId = 0
            exec loAdGetGroupKey @siteId = @siteId output,
                                 @uniqueId = @uniqueId output,
                                 @groupId = @groupId output,
                                 @error = @error output
            if @error != 0
                  return
            update loAdGroup
                  set uniqueId = @uniqueId,
                     groupId = @groupId,
                     siteId = @siteId
                  where uniqueId = 0
         end
   end
return
go
```

Trigger: loAdGenerateKeywordId

Trigger Code

```
--//////////////////////////////////////////////////////
-- Name: loAdGenerateKeywordId
--
-- Description:
--
-- Trigger that generates Keyword Id which is maximum Keyword Id + 1
--
-- Implementation Notes:
--//////////////////////////////////////////////////////

create trigger loAdGenerateKeywordId on loAdECSCollectionKeyword
for insert as
   begin
```

```
        declare @maxId          AdvIdType

      select @maxId = max(keywordId)
                  from loAdECSCollectionKeyword
          if @maxId = NULL
            select   @maxId = 0

           update loAdECSCollectionKeyword
                 set keywordId = @maxId + 1
                 where keywordId = 0
        end
return
go
```

Trigger: loAdGenerateReviewId

Trigger Code

```
--////////////////////////////////////////////////////////
-- Name: loAdGenerateReviewId
--
-- Description:
--
-- Trigger that generates Review Id which is maximum Review Id + 1
--
-- Implementation Notes:
--////////////////////////////////////////////////////////

create trigger loAdGenerateReviewId on loAdReview
for insert as
   begin

        declare @maxId          AdvIdType

      select @maxId = max(reviewId)
                  from loAdReview
          if @maxId = NULL
            select   @maxId = 0

           update loAdReview
                 set reviewId = @maxId + 1
                 where reviewId = 0
        end
return
go
```

Trigger: loAdRequestDuplicateCheck

Trigger Code

```
--//////////////////////////////////////////////////////
--
-- Name : IoAdRequestDuplicateCheckANDGenerateRequestId
--
-- Description:
--
-- Trigger that generates Request Id which is Site Id + Request Id
-- Raise an error if this request's ad
-- description conflicts with logically
-- unique information in an existing approved ad.
--
-- Implementation Notes:
--//////////////////////////////////////////////////////

create trigger IoAdRequestDuplicateCheck on IoAdRequest
for insert, update as
   begin
         declare @siteId          SiteIdType,
                 @uniqueId        AdvIdType,
                 @requestId       AdvIdType
      declare @error              int
-- only do that for insert, delete later ...
      if ( select count(*) from deleted ) = 0
         begin
            select @requestId = 0, @siteId = 0, @uniqueId = 0
            exec IoAdGetRequestKey @siteId = @siteId output,
                                   @uniqueId = @uniqueId output,
                                   @requestId = @requestId output,
                                   @error = @error output
            if @error != 0
                  return
            update IoAdRequest
                  set uniqueId = @uniqueId,
                     requestId = @requestId,
                     siteId = @siteId
                  where uniqueId = 0
         end
   end

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

   declare @kind int, @old int, @new int

   -- Processing is based on the kind of request
   --

   select @kind = type
   from inserted

   -- We need to call AdvDuplicateCheck with two IDs, the ID of
   -- the ad being inserted (@new) and possibly the ID of the ad it
   -- is replacing (@old). If it is not replacing anything, old should
   -- equal new.
```

```
    if (@kind = 1)                                  -- Addition
    begin
            select @new = newAdvId
            from inserted

    declare @approved bit
            exec loAdAdvDuplicateCheck @old=@new, @new=@new, @approved=0
    end
    else if (@kind = 2)                     -- Deletion
    begin
            return
    end
    else if (@kind = 3)                     -- Modification
    begin
            select @new = newAdvId
            from inserted

            select @old = oldAdvId
            from inserted

            exec loAdAdvDuplicateCheck @old=@old, @new=@new, @approved=0
    end
    else
    begin
            raiserror 30010 "Unknown Request Type in loAdRequestDuplicateCheck"
            return
    end

return
go
```

Trigger: loAdSetMaster

Trigger Code

```
--////////////////////////////////////////////////////////
-- Name: loAdSetMaster
--
-- Description:
--
-- Trigger for loAdSetMaster insert. It sets
-- Advertisement Id which is Site Id + Unique Id
-- It also sets the UpperTitle column to the uppercase
-- of Title column in the loAdAdvMaster table
--
-- Implementation Notes:
--////////////////////////////////////////////////////////

create trigger loAdSetMaster on loAdAdvMaster
for insert as
begin
            declare @siteId             SiteIdType,
```

```
                @uniqueId       AdvIdType,
                @advId          AdvIdType
        declare @error          int

                select @advId = 0, @siteId = 0, @uniqueId = 0
                exec loAdGetAdvMasterKey @siteId = @siteId output,
                                         @uniqueId = @uniqueId output,
                                         @advId = @advId output,
                                         @error = @error output
                if @error != 0
                        return
                update loAdAdvMaster
                        set uniqueId = @uniqueId,
                           advId = @advId,
                           siteId = @siteId
                        where uniqueId = 0

                update loAdAdvMaster
                        set upperTitle = upper(title)
                        where advId = @advId
end
return
go
Trigger: loAdUpperTitle
Trigger Code
--////////////////////////////////////////////////////
-- Name: loAdUpperTitle
--
-- Description:
--
-- Trigger that sets the UpperTitle column to the uppercase
-- of Title column in the loAdAdvMaster table
--
-- Implementation Notes:
--////////////////////////////////////////////////////

create trigger loAdUpperTitle on loAdAdvMaster
for update as
        begin
           update loAdAdvMaster
                set m.upperTitle = upper(m.title)
                from loAdAdvMaster m, inserted i
                where m.advId = i.advId
        end
return
go
```

4.1.10 Stored Procedures

In addition to triggers, the Sybase RDBMS supports the implementation of stored procedures for the purpose of enforcing business rules that ensure data integrity within the database. Stored procedures are parsed and compiled SQL code that reside in the database and may be called by

name by an application, trigger or another stored procedure. Table 4-3 contains a summary listing of the stored procedures in the IOS database including the procedure name and a brief definition of the purpose of the procedure. Each stored procedure within the IOS database is listed following Table 4-3.

Table 4-3. Summary List of Stored Procedures

Procedure Name	Description
IoAdAdvDuplicateCheck	Check if the new ad (@new)'s data duplicates any existing approved ad- except for an older version of it (@old), which it of course duplicates.-- Raise an error if this there is a duplication.
IoAdCreateCompositeId	Generate Composite Key from siteId and uniqueId.
IoAdGetAdvMasterKey	Generate key AdvId from siteId and uniqueId.—Raise an error if update and select from IoAdAdvMasterKey- table not successful
IoAdGetContactKey	Generate key ContactId from siteId and uniqueId.-- Raise an error if update and select from IoAdContactKey- table not successful
IoAdGetGroupKey	Generate key GroupId from siteId and uniqueId. Raise an error if update and select from IoAdGroupKey- table not successful
IoAdGetRequestKey	Generate key RequestId from siteId and uniqueId. Raise an error if update and select from IoAdRequestKey- table not successful
IoAdProductDupChkForApproved	Procedure that detects the duplication of key fields in this entity.
IoAdProductDupChkForRequest	Procedure that detects the duplication of key fields in this entity.
IoAdProviderDuplicateCheck	Procedure that detects the duplication of key fields in this entity.
IoAdServiceDuplicateCheck	Procedure that detects the duplication of key fields in this entity.
IoAdSigServiceDuplicateCheck	Procedure that detects the duplication of key fields in this entity.

Procedure: IoAdAdvDuplicateCheck

Code

```
--////////////////////////////////////////////////////////
-- Name : IoAdAdvDuplicateCheck
--
-- Description:
--
-- Check if the new ad (@new)'s data duplicates any existing approved ad
-- except for an older version of it (@old), which it of course duplicates.
```

```
-- Raise an error if this there is a duplication.
--
-- Implementation Notes:
--//////////////////////////////////////////////////////

create proc loAdAdvDuplicateCheck
(@old int = null,
 @new int = null,
 @approvedCheck bit = 0)
as

    -- Validate arguments
    --

    if (@old is null OR @new is null )
    begin
            raiserror 30010 "IDs not supplied to procedure loAdAdvDuplicateCheck"
            return
    end

    -- Determine ad's type
    --

    declare @typeStr varchar(80)
    select @typeStr = advType
    from
            loAdAdvMaster tm
    where
            tm.advId = @new

    -- Call right routine based on ad's type
    --

    if (@typeStr = 'ad.provider')
            exec loAdProviderDuplicateCheck @old=@old, @new=@new
    else if (@typeStr = 'ad.product' and @approvedCheck = 1)
            exec loAdProductDupChkForApproved @old=@old, @new=@new
    else if (@typeStr = 'ad.product' and @approvedCheck = 0)
            exec loAdProductDupChkForRequest @old=@old, @new=@new
    else if (@typeStr = 'ad.service.sig')
            exec loAdSigServiceDuplicateCheck @old=@old, @new=@new
    else if (@typeStr like 'ad.service%')
            exec loAdServiceDuplicateCheck @old=@old, @new=@new
    else
    begin
            raiserror 30010 "Unknown Type of Ad in procedure loAdAdvDuplicateCheck"
            return
    end
return
go
```

Procedure: loAdCreateCompositeId

Code

```
--create schema authorization dbo


--////////////////////////////////////////////////////
-- Name : loAdCreateCompositeId
--
-- Description:
--
-- Generate Composite Key from siteId and uniqueId.
--
-- Implementation Notes:
--////////////////////////////////////////////////////

create proc loAdCreateCompositeId
 (@id1     SiteIdType    output,
  @id2     UniqueIdType,
  @id3     AdvIdType       output,
  @error   int                 output)
as
    declare @siteStr                varchar(2),
          @uniqueStr        varchar(6),
              @idStr                varchar(8)

    select @id1 = siteId
                from loAdSiteInfo
                where local = 1

    select @error = @@error
    if @error != 0
          return
    select @siteStr = right
          ("00" + ltrim(convert(varchar(2), @id1)), 2)

    select @idStr = right
          ("000000" + ltrim(convert(varchar(6), @id2)), 6)

    select @id3 = convert(numeric(8,0), @siteStr + @idStr)

return
go
```

Procedure: loAdGetAdvMasterKey

Code

```
--////////////////////////////////////////////////////
-- Name : loAdGetAdvMasterKey
--
-- Description:
```

```
--
-- Generate key AdvId from siteId and uniqueId
-- Raise an error if update and select from loAdAdvMasterKey
-- table not successful
--
-- Implementation Notes:
--//////////////////////////////////////////////////////

create proc loAdGetAdvMasterKey
 (@siteId          SiteIdType output,
  @uniqueId        AdvIdType output,
  @advId           AdvIdType output,
  @error           int          output)
as
    begin transaction

--      Check the MasterAdv Key table

           if (select count(*) from loAdAdvMasterKey ) != 1
           begin
             declare @maxId         UniqueIdType
             delete from loAdAdvMasterKey
         select @maxId = max(uniqueId)
                     from loAdAdvMaster g, loAdSiteInfo s
                     where s.local = 1 and s.siteId = g.siteId
             if @maxId = NULL
               select   @maxId = 0
             insert into loAdAdvMasterKey (advIdUnique) values (@maxId)
           end

--      Generate key AdvId

             begin
         update loAdAdvMasterKey
                      set advIdUnique = advIdUnique + 1
             select @error = @@error
             if @error != 0
               begin
                 raiserror 30515
                 "loAdAdvMasterKey not supplied to procedure loAdGetAdvMasterKey"
                     rollback transaction
                     return
               end
         select @uniqueId = advIdUnique
               from loAdAdvMasterKey
             select @error = @@error
             if @error != 0
               begin
                 raiserror 30515
                 "loAdAdvMasterKey not supplied to procedure loAdGetAdvMasterKey"
                     rollback transaction
                     return
               end
             select @advId = 0, @siteId = 0
```

```
            exec loAdCreateCompositeId @id1 = @siteId output,
                                       @id2 = @uniqueId,
                                       @id3 = @advId output,
                                       @error = @error output
            if @error != 0
              begin
                raiserror 30515
                "loAdSiteInfo not supplied to procedure loAdGetAdvMasterKey"
                    rollback transaction
                    return
              end
          end
    commit transaction
return
go
```

Procedure: loAdGetContactKey

Code

```
--//////////////////////////////////////////////////////////
-- Name : loAdGetContactKey
--
-- Description:
--
-- Generate key ContactId from siteId and uniqueId
-- Raise an error if update and select from loAdContactKey
-- table not successful
--
-- Implementation Notes:
--//////////////////////////////////////////////////////////

create proc loAdGetContactKey
 (@siteId        SiteIdType output,
  @uniqueId      AdvIdType output,
  @contactId     AdvIdType output,
  @error         int       output)
as
    begin transaction

--      Check the Contact Key table

          if (select count(*) from loAdContactKey ) != 1
          begin
            declare @maxId        UniqueIdType
            delete from loAdContactKey

            select @maxId = max(uniqueId)
                   from loAdContact g, loAdSiteInfo s
                   where s.local = 1
                   and s.siteId = g.siteId
```

```
            if @maxId = NULL
              select   @maxId = 0
            insert into IoAdContactKey (contactIdUnique) values (@maxId)
          end

--      Generate key ContactId

          begin
        update IoAdContactKey
                    set contactIdUnique = contactIdUnique + 1
            select @error = @@error
            if @error != 0
              begin
                raiserror 30515
                "IoAdContactKey not supplied to procedure IoAdGetContactKey"
                    rollback transaction
                    return
              end
        select @uniqueId = contactIdUnique
             from IoAdContactKey
            select @error = @@error
            if @error != 0
              begin
                raiserror 30515
                "IoAdContactKey not supplied to procedure IoAdGetContactKey"
                    rollback transaction
                    return
              end
            select @contactId = 0, @siteId = 0
            exec IoAdCreateCompositeId @id1 = @siteId output,
                                       @id2 = @uniqueId,
                                       @id3 = @contactId output,
                                  @error = @error output
            if @error != 0
              begin
                raiserror 30515
                "IoAdSiteInfo not supplied to procedure IoAdGetContactKey"
                    rollback transaction
                    return
              end
          end
    commit transaction
return
go
```

Procedure: IoAdGetGroupKey

Code

```
--////////////////////////////////////////////////////////
-- Name : IoAdGetGroupKey
--
```

```
-- Description:
--
-- Generate key GroupId from siteId and uniqueId
-- Raise an error if update and select from loAdGroupKey
-- table not successful
--
-- Implementation Notes:
--//////////////////////////////////////////////////////

create proc loAdGetGroupKey
 (@siteId          SiteIdType output,
  @uniqueId        AdvIdType output,
  @groupId         AdvIdType output,
  @error           int          output)
as
    begin transaction

--      Check the Group Key table

           if (select count(*) from loAdGroupKey ) != 1
           begin
             declare @maxId         UniqueIdType
             delete from loAdGroupKey
             select @maxId = max(uniqueId)
                       from loAdGroup g, loAdSiteInfo s
                       where s.local = 1 and s.siteId = g.siteId
             if @maxId = NULL
               select   @maxId = 0
             insert into loAdGroupKey (groupIdUnique) values (@maxId)
           end

--      Generate key GroupId

             begin
          update loAdGroupKey
                       set groupIdUnique = groupIdUnique + 1
             select @error = @@error
             if @error != 0
               begin
                 raiserror 30515
                 "loAdGroupKey not supplied to procedure loAdGetGroupKey"
                      rollback transaction
                      return
               end
          select @uniqueId = groupIdUnique
               from loAdGroupKey
             select @error = @@error
             if @error != 0
               begin
                 raiserror 30515
                 "loAdGroupKey not supplied to procedure loAdGetGroupKey"
                      rollback transaction
                      return
               end
```

```
            select @groupId = 0, @siteId = 0
            exec loAdCreateCompositeId @id1 = @siteId output,
                                       @id2 = @uniqueId,
                                       @id3 = @groupId output,
                                    @error = @error output
            if @error != 0
              begin
                raiserror 30515
                "loAdSiteInfo not supplied to procedure loAdGetGroupKey"
                    rollback transaction
                    return
              end
           end
    commit transaction
return
go
```

Procedure: loAdGetRequestKey

Code

```
--//////////////////////////////////////////////////////
-- Name : loAdGetRequestKey
--
-- Description:
--
-- Generate key RequestId from siteId and uniqueId
-- Raise an error if update and select from loAdRequestKey
-- table not successful
--
-- Implementation Notes:
--//////////////////////////////////////////////////////

create proc loAdGetRequestKey
 (@siteId          SiteIdType output,
  @uniqueId        AdvIdType output,
  @requestId       AdvIdType output,
  @error           int          output)
as
    begin transaction

--      Check the Request Key table

            if (select count(*) from loAdRequestKey ) != 1
            begin
              declare @maxId         UniqueIdType
              delete from loAdRequestKey
           select @maxId = max(uniqueId)
                       from loAdRequest g, loAdSiteInfo s
                       where s.local = 1 and s.siteId = g.siteId
              if @maxId = NULL
                select   @maxId = 0
```

```
            insert into IoAdRequestKey (requestIdUnique) values (@maxId)
          end

--      Generate key RequestId

          begin
        update IoAdRequestKey
                    set requestIdUnique = requestIdUnique + 1
            select @error = @@error
            if @error != 0
              begin
                raiserror 30515
                "IoAdRequestKey not supplied to procedure IoAdGetRequestKey"
                    rollback transaction
                    return
              end
        select @uniqueId = requestIdUnique
              from IoAdRequestKey
            select @error = @@error
            if @error != 0
              begin
                raiserror 30515
                "IoAdRequestKey not supplied to procedure IoAdGetRequestKey"
                    rollback transaction
                    return
              end
            select @requestId = 0, @siteId = 0
            exec IoAdCreateCompositeId @id1 = @siteId output,
                                       @id2 = @uniqueId,
                                       @id3 = @requestId output,
                                       @error = @error output
            if @error != 0
              begin
                raiserror 30515
                "IoAdSiteInfo not supplied to procedure IoAdGetRequestKey"
                    rollback transaction
                    return
              end
          end
    commit transaction
return
go
```

Procedure: IoAdProductDupChkForApproved

Code

```
--////////////////////////////////////////////////////////
-- Name: IoAdProductDupChkForApproved
--
-- Description:
--
```

```
-- Procedure that detects the duplication of key fields in
-- this entity.
--
-- This entity is uniquely determined by the following composite key
--          VersionID in IoAdECSCollection
--          ShortName in IoAdECSCollection
--          o Group Id
--          o Provider Id
--          o Title
--
-- Raise an error if this there is a duplication.
--
-- Implementation Notes:
--////////////////////////////////////////////////////////

create proc IoAdProductDupChkForApproved
(@old int = null,
 @new int = null)
as
--
-- Check VersionID, ShortName in IoAdECSCollection
--

          declare @VersionID       int,
                              @ShortName     char(8)

          select @VersionID = VersionID, @ShortName = ShortName
          from IoAdECSCollection
          where collectionId = @new

          if @@rowcount = 0
          begin
                    raiserror 31117 "There is no such record in Collection"
          return
          end


   declare @group_id int, @title varchar(100), @provider int

   -- First, get the group, title that we inserted into the master
   --

   select @group_id = advGroupId, @title = title
   from IoAdAdvMaster tm
   where
          tm.advId = @new

   -- Then, get the title that we inserted into the master
   --

   select @title = title
             from IoAdAdvMaster tm
             where
                       tm.advId = @new
```

```
    -- Then, get the provider that we inserted into the product table
    --

    select @provider = providerId
    from loAdProductAdv tp
    where
            tp.productId = @new



    -- Now, compare primary keys
    --

    if (select count(*)
    from loAdApprovedAdv ta, loAdAdvMaster tm, loAdProductAdv tp,
                    loAdECSCollection tc
            where
            tm.advId = tp.productId AND
            tm.advId = tc.collectionId AND
            tm.advId = ta.advId AND
            tm.advId != @old AND
            tm.advId != @new AND
            tp.providerId = @provider AND
            tc.VersionID = @VersionID AND
            tc.ShortName = @ShortName AND
            tm.advGroupId = @group_id AND
            tm.title = @title) > 0
    begin
            raiserror 31117 "Attempting to insert data with same
                    group, provider, version, ShortName and title as exisiting entry"
            return
    end
return
go
```

Procedure: loAdProductDupChkForRequest

Code

```
--////////////////////////////////////////////////////////
-- Name: loAdProductDupChkForRequest
--
-- Description:
--
-- Procedure that detects the duplication of key fields in
-- this entity.
--
-- This entity is uniquely determined by the following composite key
--          o Group Id
--          o Provider Id
--          o Title
```

```
--
-- Raise an error if this there is a duplication.
--
-- Implementation Notes:
--////////////////////////////////////////////////////////

create proc loAdProductDupChkForRequest
(@old int = null,
 @new int = null)
as

    declare @group_id int, @title varchar(100), @provider int

    -- First, get the group, title that we inserted into the master
    --

    select @group_id = advGroupId, @title = title
    from loAdAdvMaster tm
    where
            tm.advId = @new

    -- Then, get the title that we inserted into the master
    --

    select @title = title
              from loAdAdvMaster tm
              where
                        tm.advId = @new

    -- Then, get the provider that we inserted into the product table
    --

    select @provider = providerId
    from loAdProductAdv tp
    where
            tp.productId = @new



    -- Now, compare primary keys
    --

    if (select count(*)
    from loAdApprovedAdv ta, loAdAdvMaster tm, loAdProductAdv tp
            where
            tm.advId = tp.productId AND
            tm.advId = ta.advId AND
            tm.advId != @old AND
            tm.advId != @new AND
            tp.providerId = @provider AND
            tm.advGroupId = @group_id AND
            tm.title = @title) > 0
    begin
            raiserror 31117 "Attempting to insert data with same
```

```
                group, provider, version, ShortName and title as exisiting entry"
        return
   end
return
go
```

Procedure: loAdProviderDuplicateCheck

Code

```
--//////////////////////////////////////////////////////
-- Name: loAdProviderDuplicateCheck
--
-- Description:
--
-- Procedure that detects the duplication of key fields in
-- this entity.
--
-- This entity is uniquely determined by the following composite key
--        o Group Id
--        o Title
--
-- Raise an error if this there is a duplication.
--
-- Implementation Notes:
--//////////////////////////////////////////////////////

create proc loAdProviderDuplicateCheck
(@old int = null,
 @new int = null)
as
   declare @group_id int, @title varchar(100)

   -- First, get the group, title that we inserted into the master
   --

   select @group_id = advGroupId, @title = title
   from loAdAdvMaster tm
   where
        tm.advId = @new


   -- Now, compare primary keys, making sure to ignore old and new
   --

   if (select count(*)
   from loAdApprovedAdv ta, loAdAdvMaster tm, loAdProviderAdv tp
   where
        ta.advId = tm.advId AND
        tm.advId = tp.providerId AND
        tm.advId != @old AND
        tm.advId != @new AND
```

```
        tm.advGroupId = @group_id AND
        tm.title = @title) > 0
    begin
        raiserror 31117 "Attempting to insert data with same
                group, and title as exisiting entry"
        return
    end
return
go
```

Procedure: loAdServiceDuplicateCheck

Code

```
--//////////////////////////////////////////////////////
-- Name: loAdServiceDuplicateCheck
--
-- Description:
--
-- Procedure that detects the duplication of key fields in
-- this entity.
--
-- This entity is uniquely determined by the following composite key
--      o Group Id
--      o Provider Id
--      o title
--
-- Raise an error if this there is a duplication.
--
-- Implementation Notes:
--//////////////////////////////////////////////////////

create proc loAdServiceDuplicateCheck
(@old int = null,
 @new int = null)
as
    declare @group_id int, @title varchar(100), @provider int

    -- First, get the group, title that we inserted into the master
    --

    select @group_id = advGroupId, @title = title
    from loAdAdvMaster tm
    where
        tm.advId = @new


    -- Then, get the provider that we inserted into the product table
    --

    select @provider = providerId
    from loAdServiceAdv ts
```

```
    where
            ts.serviceId = @new



    -- Now, compare primary keys
    --

    if (select count(*)
    from IoAdApprovedAdv ta, IoAdAdvMaster tm, IoAdServiceAdv ts
    where
            ta.advId = tm.advId AND
            tm.advId = ts.serviceId AND
            tm.advId != @old AND
            tm.advId != @new AND
            tm.advGroupId = @group_id AND
            ts.providerId = @provider AND
            tm.title = @title) > 0
    begin
            raiserror 31117 "Attempting to insert data with same
                    group, provider, and title as exisiting entry"
            return
    end

return
go
```

Procedure: IoAdSigServiceDuplicateCheck

Code

```
--//////////////////////////////////////////////////////////
-- Name: IoAdSigServiceDuplicateCheck
--
-- Description:
--
-- Procedure that detects the duplication of key fields in
-- this entity.
--
-- This entity is uniquely determined by the following composite key
--          o Group Id
--          o Provider Id
--          o Internal Name
--
-- Raise an error if this there is a duplication.
--
-- Implementation Notes:
--//////////////////////////////////////////////////////////

create proc IoAdSigServiceDuplicateCheck
(@old int = null,
 @new int = null)
```

```
as
   declare @group_id int, @internalName varchar(100), @provider int

   -- First, get the group that we inserted into the master
   --

   select @group_id = advGroupId
   from IoAdAdvMaster tm
   where
         tm.advId = @new



   -- Then, get the internal Name that we inserted into the master
   --

   select @internalName = internalName
   from IoAdSignatureServiceAdv tss
   where
         tss.serviceId = @new



   -- Then, get the provider that we inserted into the product table
   --

   select @provider = providerId
   from IoAdServiceAdv ts
   where
         ts.serviceId = @new



   -- Now, compare primary keys
   --

   if (select count(*)
   from IoAdApprovedAdv ta, IoAdAdvMaster tm, IoAdServiceAdv ts,
          IoAdSignatureServiceAdv tss
   where
         ta.advId = tm.advId AND
         tm.advId = ts.serviceId AND
         tm.advId = tss.serviceId AND
         tm.advId != @old AND
         tm.advId != @new AND
         tm.advGroupId = @group_id AND
         ts.providerId = @provider AND
         tss.internalName = @internalName) > 0
   begin
         raiserror 31117 "Attempting to insert data with same
                  group, provider, and internalName as exisiting entry"
         return
   end
```

```
   -- Now we make sure that the service criteria also are met
   --

   exec IoAdServiceDuplicateCheck @old=@old, @new=@new

return
go
```

4.2 Flat File Usage

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

4.2.1 File Descriptions

No files used.

4.2.2 Field Specifications

Not applicable.

4.2.3 Domain Definitions

Domain definitions specify the nature and valid content of fields within a file (e.g., specific values for a limited set of data, ranges of numeric data, units of measure for applicable data). This information is generally used by software to edit incoming data for validity prior to writing or changing data within the file. Use of domain values in updating (adding and changing) records within files preserves the integrity of the data within the file. This section is not applicable to IOS.

This page intentionally left blank.

5. Performance and Tuning Factors

5.1 Indexes

An index provides a means of locating a row in a database table based on the value of a specific column(s), without having to scan all data in the table. If used appropriately, indexes can significantly decrease the time it takes to retrieve data. Sybase allows the definition of two types of indexes, *clustered* and *non-clustered*.

In a *clustered* index, the rows in a database table are physically stored in sequence--determined by the index. Clustered indexes are particularly useful, when the data is frequently retrieved in sequential order. Only one clustered index may be defined per database table.

Non-clustered indexes differ from their clustered counterpart, in that data is not physically stored in sorted order—newly added rows are stored at the end of the database table.

A list of all the indexes defined for database tables in the IOS database is given in Table 5-1 along with a description of each index.

Table 5-1. Indexes

Index Name	Related Database Table	Description
Group_Name_Idx	IoAdGroup	Clustered Index.
Signature_InternalName_Idx	IoAdSignatureServiceAdv	Clustered Index

5.2 Segments

Sybase supports the declaration of segments. A segment is a named pointer to a storage device(s). Segments are used to physically allocate a database object to a particular storage device. Segments defined for the IOS and all other subsystem databases are described in Table 5-2.

Table 5-2. Segment Descriptions

Segment Name	Description
default	Tables, indexes, and objects
logsegment	SYSLOGS, Transaction Logs
systemsegment	System tables and indexes

5.3 Named Caches

A cache is a block of memory that is used by Sybase to retain and manage data pages that are currently being processed. A *named cache* is a block of memory that is named and used by the RDBMS to store these frequently accessed data pages. Assigning a database table to named cache causes accessed pages to be loaded into memory and retained. The named cache does not need to be allocated to accommodate the entire database table since the RDBMS manages the cache according to use. Named caches greatly increase performance by eliminating the time associated for disk input and output (I/O). There are no named caches that are currently defined for the IOS database.

6. Database Security

6.1 Approach

The database security discussed within this section is bounded to security implementation within the Sybase RDBMS. A Sybase general approach to security is adopted as illustrated in Figure 6-1.

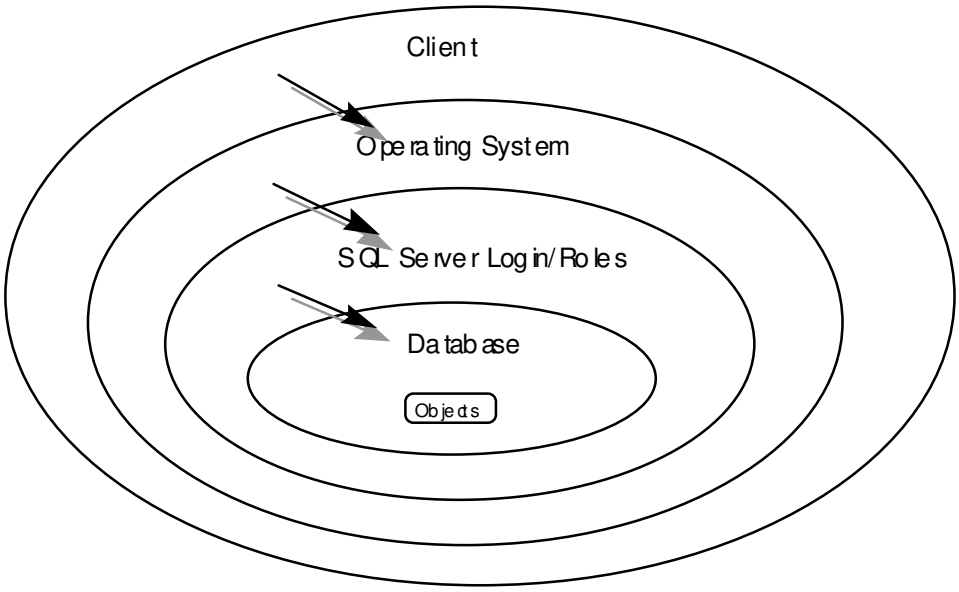

Figure 6-1. Sybase General Approach to SQL Server Security[2]

The client (user) requires a SQL Server login to access the RDBMS. The System Administrator may grant or revoke login by various roles (e.g., sa, sso, oper). Roles are defined in Section 6.5. The login is assigned to a user with certain related permissions for gaining access to particular objects (e.g., database tables, views, commands) within the database. Descriptions of ECS user and system administrator permissions follows.

---

[2] Reference Sybase Student Guide: *Advanced SQL Server Administration*.

6.2 Initial Users

After initial installation users will have been assigned permissions for access to the IOS database. Users include scientists and other interested persons, operations and administrative personnel, and software. The level of access is limited to that associated with their assigned group, objects, and/or role. A definition of each if these groups, objects, and roles is given in the following Sections.

6.3 Groups

Groups are a means of logically associating users with similar data access needs. Once a group has been defined, object and command permissions can be granted to that group. A user who is member of a group inherits all of the permissions granted to that group. No groups have been initially defined in the IOS database for the "IoAdAdvService" default database. Groups are set within the IOS installation scripts for each object/user access. Permissions are identified in Table 6-1. A specification of the groups is contained in Table 6-2.

Table 6-1. Permission Key

Permission	Description
A	All
S	Select
I	Insert
U	Update
D	Delete
E	Execute[3]

Table 6-2. Group Specifications

Group	SYBASE LOGIN	Permissions Granted					
		A	S	I	U	D	E
No groups	EcIoAdServer	Y					

6.4 Objects

Permissions must be assigned for the user to gain access to the objects within the database. Examples of objects associated with the database include database tables, views, stored procedures etc. All select, update, delete, insert and execute permissions are granted to the user, EcIoAdServer.

3

6.5 Roles

Roles were introduced in Sybase 10 to allow a structured means for granting users the permissions needed to perform standard database administration activities and also provide a means for easily identifying such users. There are six pre-defined roles that may be assigned to a user. A definition of each of these roles follows as well as a description of the types of activities that may be performed by each role.

System Administrator (*sa_role*): This role is used to grant a specific user permissions needed to perform standard system administrator duties including:

- installing SQL server and specific SQL server modules
- managing the allocation of physical storage
- tuning configuration parameters
- creating databases

Site Security Officer (*sso_role*): This role is used to grant a specific user the permissions needed to maintain SQL server security including:

- adding server logins
- administrating passwords
- managing the audit system
- granting users all roles except the *sa_role*

Operator (*oper_role*): This role is used to grant a specific user the permissions needed to perform standard functions for the database including:

- dumping transactions and databases
- loading transactions and databases

Navigator (*navigator_role*): This role is used to grant a specific user the permissions needed to manage the navigation server.

Replication (*replication_role*): This role is used to grant a specific user the permissions needed to manage the replication server.

Sybase Technical Support (*sybase_ts_role*): This role is used to grant a specific user the permissions needed to execute *database consistency checker (dbcc)*, a Sybase supplied utility supporting commands that are normally outside of the realm of routine system administrator activities.

This page intentionally left blank

7. Scripts

The scripts identified in this section may be found in the directory named /ecs/formal/IOS/AdvService/src/Database.

7.1 Installation Scripts

Scripts used to support installation of the IOS database are described in Table 7-1.

Table 7-1. Installation Scripts

Script File	Description
create_ios_schema.sql	Creates the schema (tables)
create_ios_trgproc.sql	Establishes the triggers and associates them with the tables in the IOS database.
drop_ios_schema.sql	Drops the old schema if it exists to prepare for the create. drop_ios_trgproc.sql (drops the triggers in the stored procedure.
EcIoAdDbBuild	Initializes the schema, loads the triggers, initializes the database with data, Initially populates the database, adds server names as user for login, grants permissions to user.
sample_data.sql	Contains the insert statements to populate the database.

7.2 De-Installation Scripts

Scripts used to support de-installation of the IOS database are described in Table 7-2.

Table 7-2. De-Installation Scripts

Script File	Description
drop_ios_schema.sql	Drops the old schema if it exists to prepare for the create. drop_ios_trgproc.sql (drops the triggers in the stored procedure.
drop_ios_trgproc.sql	Removes the triggers from the database/table.

7.3 Backup and Recovery Scripts

Scripts used to perform backup and recovery of the IOS database are described in Table 7-3.

Table 7-3. Backup and Recovery Scripts

Script File	Description
bcp.copyin.csh	Copies datafiles into database tables
bcp.copyout.csh	Copies database contents into datafiles.

7.4 Miscellaneous Scripts

Miscellaneous scripts that are applicable to the IOS database are described in Table 7-4.

Table 7-4. Miscellaneous Scripts

Script File	Description
fix_constraints.sql	Adds constraints to tables
grant_iosuser_permission.sql	Grants permissions to users (including software) for access to the IOS database.

Abbreviations and Acronyms

ACL	Access Control List
ANSI	American National Standards Institute
CASE	Computer-Aided Software Engineering
CCB	Change Control Board (Raytheon Convention)
	Configuration Control Board (NASA Convention)
CCR	configuration change request
CD	contractual delivery 214-001
CDRL	Contract Data Requirements List
CSCI	computer software configuration item
CSMS	Communications and Systems Management Segment (ECS)
DAAC	Distributed Active Archive Center
DBMS	database management system
DCN	document change notice
DDICT	Data Dictionary CSCI
DDIST	Data Distribution Services CSCI
DID	data item description
DM	Data Management CSCI
ECS	EOSDIS Core System
EDC	EROS Data Center
EDHS	ECS Data Handling System
EOS	Earth Observing System
EOSDIS	Earth Observing System Data and Information System
ERD	entity-relationship diagram
GSFC	Goddard Space Flight Center
ID	identification
INS	Ingest Subsystem
IOS	Interoperability Subsystem

I/O	Input and Output
LaRC	Langley Research Center (DAAC)
MIME	multipurpose internet mail extension
MSS	Management Support Subsystem
NAS	National Academy of Science
NASA	National Aeronautics and Space Administration
NSIDC	National Snow and Ice Data Center (DAAC)
OO	Object Oriented
PDF	portable document format
PDPS	Planning and Data Processing Subsystem
RDBMS	Relational Database Management System
SDPS	Science Data Processing Segment (ECS)
SDSRV	Science Data Server CSCI
SMC	System Management Center (ECS)
SQL	structured query language
STD	Software Test Document
STMGT	Storage Management CSCI
SUBSRV	Subscription Server
TBS	to be supplied
TD	technical document
TP	technical paper
UR	Universal Reference
URL	Universal Resource Locator
VATC	Verification and Acceptance Testing Center
WWW	World Wide Web