

305-CD-022-002

EOSDIS Core System Project

Release B SDPS Interoperability Subsystem Design Specification for the ECS Project

March 1996

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

Hughes Information Technology Systems
Upper Marlboro, MD

Release B SDPS Interoperability Subsystem Design Specification for the ECS Project

March 1996

Prepared Under Contract NAS5-60000
CDRL Item #046

APPROVED BY

Rick Kochhar /s/

3/18/96

Rick Kochhar, Release B CCB Chairman
EOSDIS Core System Project

Date

**Hughes Information Technology Systems
Upper Marlboro, Maryland**

This page intentionally left blank.

Preface

This document is one of eighteen comprising the detailed design specifications of the SDPS and CSMS subsystem for Release B of the ECS project. A complete list of the design specification documents is given below. Of particular interest are documents number 305-CD-020, which provides an overview of the subsystems and 305-CD-039, the Data Dictionary, for those reviewing the object models in detail.

The SDPS and CSMS subsystem design specification documents for Release B of the ECS Project include:

305-CD-020	Release B Overview of the SDPS and CSMS Segment System Design Specification
305-CD-021	Release B SDPS Client Subsystem Design Specification
305-CD-022	Release B SDPS Interoperability Subsystem Design Specification
305-CD-023	Release B SDPS Data Management Subsystem Design Specification
305-CD-024	Release B SDPS Data Server Subsystem Design Specification
305-CD-025	Release B SDPS Ingest Subsystem Design Specification
305-CD-026	Release B SDPS Planning Subsystem Design Specification
305-CD-027	Release B SDPS Data Processing Subsystem Design Specification
305-CD-028	Release B CSMS Segment Communications Subsystem Design Specification
305-CD-029	Release B CSMS Segment Systems Management Subsystem Design Specification
305-CD-030	Release B GSFC Distributed Active Archive Center Design Specification
305-CD-031	Release B LaRC Distributed Active Archive Center Design Specification
305-CD-033	Release B EDC Distributed Active Archive Center Design Specification
305-CD-034	Release B ASF Data Center Distributed Active Archive Center Design Specification
305-CD-035	Release B NSIDC Distributed Active Archive Center Design Specification
305-CD-036	Release B JPL Distributed Active Archive Center Design Specification
305-CD-037	Release B ORNL Distributed Active Archive Center Design Specification
305-CD-038	Release B System Monitoring and Coordination Center Design Specification
305-CD-039	Release B Data Dictionary for Subsystem Design Specification

Object models presented in this document have been exported directly from CASE or DBMS 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 drawings on line using the Portable Document Format (PDF) electronic copy available via the ECS Data Handling System (EDHS) at: URL <http://edhs1.gsfc.nasa.gov>.

This document is a formal contract deliverable with an approval code of 2; as such it requires Government review and approval prior to acceptance and use. This document is under ECS contractor configuration control. Once this document is approved, Contractor approved changes are handled in accordance with Class I and Class II change control requirements described in the EOS Configuration Management Plan, and changes to this document shall be made by document change notice (DCN) or by complete revision.

Any questions should be addressed to:

Data Management Office
The ECS Project Office
Hughes Information Technology Systems
1616 McCormick Drive
Upper Marlboro, MD 20774-5372

Abstract

This document presents the design of the Interoperability Subsystem of the Earth Observing System Data and Information System (EOSDIS) Core System (ECS). It defines the Interoperability Subsystem's Release B CSCI interfaces and structures, as well as subsystem design based on Level 4 requirements.

Keywords: SDPS, interoperability, CSCI, HWCI, advertising, advertisement, gateway, WAIS, HTTP, HTML, DBMS

This page intentionally left blank.

Change Information Page

List of Effective Pages			
Page Number	Issue		
Title	Submitted as Final		
iii through x	Submitted as Final		
1-1 and 1-2	Submitted as Final		
2-1 through 2-4	Submitted as Final		
3-1 through 3-6	Submitted as Final		
4-1 through 4-62	Submitted as Final		
AB-1 through AB-6	Submitted as Final		
Document History			
Document Number	Status/Issue	Publication Date	CCR Number
305-CD-022-001	Preliminary	October 1995	95-0735
305-CD-022-002	Submitted as Final	March 1996	96-0224

This page intentionally left blank.

Contents

1. Introduction

1.1	Identification	1-1
1.2	Scope	1-1
1.3	Document Organization	1-1
1.4	Status and Schedule	1-2

2. Related Documentation

2.1	Parent Documents	2-1
2.2	Applicable Documents	2-1
2.3	Information Documents Not Referenced	2-2

3. Interoperability Subsystem Overview

3.1	Subsystem Overview	3-1
3.2	Subsystem Structure	3-1
3.3	Subsystem Design Rationale	3-1
3.4	Subsystem Physical Design	3-5

4. ADSRV - Advertising Service CSCI

4.1	CSCI Overview	4-1
4.2	Advertising Service Context	4-1
4.3	Advertising Service Object Model	4-1
4.3.1	EcPoHandle Class	4-7
4.3.2	EcPoPersistentBase Class	4-10
4.3.3	IoAdAdvertisement Class	4-13
4.3.4	IoAdContact Class	4-19
4.3.5	IoAdContactSearchCommand Class	4-25
4.3.6	IoAdMimeTypeAdv Class	4-30
4.3.7	IoAdProduct Class	4-33
4.3.8	IoAdProductSearchCommand Class	4-37
4.3.9	IoAdProvider Class	4-38
4.3.10	IoAdProviderSearchCommand Class	4-42
4.3.11	IoAdSearchCommand Class	4-43
4.3.12	IoAdService Class	4-48
4.3.13	IoAdServiceSearchCommand Class	4-52

4.3.14	IoAdSignatureServiceAdv Class	4-54
4.3.15	IoAdSignatureServiceSearchCommand Class	4-57
4.3.16	Advertising Service Subscription Model	4-59
4.4	CSCI Structure	4-59
4.4.1	Advertising Application Server CSC (AdvDBMSApplServer)	4-60
4.4.2	Advertising DBMS Server CSC (AdvDBMSServer)	4-60
4.4.3	Persistent Object Framework CSC	4-60
4.4.4	HTML Framework CSC	4-60
4.4.5	HTML Interfaces CSC	4-60
4.4.6	Core Library CSC	4-60
4.4.7	Client Library CSC	4-60
4.4.8	Installer CSC	4-60
4.4.9	Advertising Navigating Server CSC	4-61
4.5	CSCI Management and Operation	4-61

Abbreviations and Acronyms

List of Figures

3.2-1	Interoperability Subsystem Context	3-2
3.4-1	Software to Hardware Mapping Diagram	3-5
4.3-1	EcPoPersist Object Model Diagram	4-2
4.3-2	IoAdAdvHandles Object Model Diagram	4-3
4.3-3	IoAdHandles Object Model Diagram	4-4
4.3-4	IoAdSearchCommand Object Model Diagram	4-5
4.3-5	IoAdServiceHandles Object Model Diagram	4-6

List of Tables

2-1	Interoperability Subsystem Interface	3-3
3.2-2	Interoperability Data Flow Definitions	3-4
4.4-1	ADSRV Components	4-59

1. Introduction

1.1 Identification

This Release B SDPS Interoperability Subsystem Design Specification for the ECS Project, Contract Data Requirement List (CDRL) Item 046, with requirements specified in Data Item Description (DID) 305/DV2, is a required deliverable under the Earth Observing System Data and Information System (EOSDIS) Core System (ECS), Contract NAS5-60000. This publication is part of a series of documents comprising the Science and Communications Development Office design specification for the Communications and System Management Segment (CSMS) and the Science and Data Processing Subsystem (SDPS) for Release B.

1.2 Scope

Release B provides support to EOS AM-1 Mission Operations and Science Operations, and it provides support to ESDIS Ground System Certification Testing for the EOS AM-1 and Landsat 7 missions. Release B provides services for the Landsat 7, COLOR, ADEOS II, SAGE III, DAO, TERS, ERS, RADARSAT, and ALT RADAR missions, and it provides product generation support for COLOR.

The Release B SDPS Interoperability Subsystem Design Specification defines the progress of the Interoperability subsystem design. It defines the Interoperability Subsystem computer software design, as well as subsystem design based on Level 4 requirements.

This subsystem is on the incremental development track. Its design and capabilities are released in and reviewed in the form of Evaluation Packages (EP), and is therefore, not part of the formal Release B Incremental Design Review. The overview material for this subsystem has been included in this document for information purposes only.

This document reflects the February 14, 1996 Technical Baseline, maintained by the ECS Configuration Control Board in accordance with ECS Technical Direction No. 11 dated December 6, 1994.

1.3 Document Organization

The document is organized to describe the Release B SDPS Interoperability Subsystem design as follows:

- Section 1 provides information regarding the identification, scope, organization and status.
- Section 2 provides a listing of related documents.
- Section 3 provides an overview of the Subsystem, focusing on the high-level design concepts. This provides general background information to put interoperability into context.
- Section 4 contains the structure of the Computer Software Configuration Items (CSCI) comprising the Interoperability Subsystem.
- The section Abbreviations and Acronyms contains an alphabetized list of the definitions for abbreviations and acronyms used in this volume.

1.4 Status and Schedule

This submittal of DID 305/DV2 meets the milestone specified in the Contract Data Requirements List (CDRL) of NASA Contract NAS5-60000. This document reflects the changes in the Release B Interoperability Subsystem that will occur from Release A to Release B. The Release A design was documented as part of the Release A Critical Design Review. This submittal is an update to that design.

2. Related Documentation

2.1 Parent Documents

The parent document is the document from which the scope and content of this Interoperability Subsystem Design Specification is derived.

194-207-SE1-001 System Design Specification for the ECS Project

2.2 Applicable Documents

The following documents are referenced within this SDPS Design Specification, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this volume.

209-CD-001-003	Interface Control Document Between EOSDIS Core System (ECS) and the NASA Science Internet
209-CD-002-003	Interface Control Document Between EOSDIS Core System (ECS) and ASTER Ground Data System
209-CD-003-003	Interface Control Document Between EOSDIS Core System (ECS) and EOS-AM Project for AM-1 Spacecraft Analysis Software
209-CD-004-003	Data Format Control Document for the Earth Observing System (EOS) AM-1 Project Data Base
209-CD-005-005	Interface Control Document Between EOSDIS Core System (ECS) and Science Computing Facilities (SCF)
209-CD-006-005	Interface Control Document Between EOSDIS Core System (ECS) and National Oceanic and Atmospheric Administration (NOAA) Affiliated Data Center (ADC)
209-CD-007-003	Interface Control Document Between EOSDIS Core System (ECS) and TRMM Science Data and Information System (TSDIS)
209-CD-008-004	Interface Control Document Between EOSDIS Core System (ECS) and the Goddard Space Flight Center (GSFC) Distributed Active Archive Center (DAAC)
209-CD-009-002	Interface Control Document Between EOSDIS Core System (ECS) and the Marshall Space Flight Center (MSFC) Distributed Active Archive Center (DAAC)
209-CD-011-004	Interface Control Document Between EOSDIS Core System (ECS) and the Version 0 System
305-CD-020-002	Overview of Release B SDPS and CSMS System Design Specification for the ECS Project
308-CD-001-005	Software Development Plan for the ECS Project

313-CD-006-002	Release B CSMS/SDPS Internal Interface Control Document for the ECS Project
423-41-03	Goddard Space Flight Center, EOSDIS Core System (ECS) Contract Data Requirements Document

2.3 Information Documents Not Referenced

The following documents, although not referenced herein and/or not directly applicable, do amplify and clarify the information presented in this document. These documents are not binding on the content of this SDPS Subsystem Design Specifications.

205-CD-002-002	Science User's Guide and Operations Procedure Handbook for the ECS Project. Part 4: Software Developer's Guide to Preparation, Delivery, Integration, and Test with ECS
206-CD-001-002	Version 0 Analysis Report for the ECS Project
209-CD-010-001	Interface Control Document Between EOSDIS Core System (ECS) and the Langley Research Center (LaRC) Distributed Active Archive Center (DAAC)
302-CD-002-001	SDPS/CSMS Release A and FOS Release A and B Facilities Plan for the ECS Project
101-303-DV1-001	Individual Facility Requirements for the ECS Project, Preliminary
194-317-DV1-001	Prototyping and Studies Plan for the ECS Project
318-CD-000-XXX	Prototyping and Studies Progress Report for the ECS Project (monthly)
333-CD-003-002	SDP Toolkit Users Guide for the ECS Project
601-CD-001-004	Maintenance and Operations Management Plan for the ECS Project
604-CD-001-004	Operations Concept for the ECS Project: Part 1-- ECS Overview
604-CD-002-003	Operations Concept for the ECS project: Part 2B -- ECS Release B,
604-CD-003-002	Operations Concept for the ECS Project: Part 2A -- ECS Release A
604-CD-004-001	Operations Concept for the ECS Project: Part 2 -- FOS
101-620-OP2-001	List of Recommended Maintenance Equipment for the ECS Project
194-703-PP1-001	System Design Review (SDR) Presentation Package for the ECS Project
194-813-SI4-002	Planning and Scheduling Prototype Results Report for the ECS Project
194-813-SI4-003	DADS Prototype One FSMS Product Operational Evaluation
194-813-SI4-004	DADS Prototype One STK Wolfcreek 9360 Automated Cartridge System Hardware Characterization Report
813-RD-009-001	DADS Prototype Two Multi-FSMS Product Integration Evaluation
828-RD-001-002	Government Furnished Property for the ECS Project
193-TP-626-001	GCDIS/UserDIS Study ECS Technical Paper, Draft 0.2
193-WP-118-001	Algorithm Integration and Test Issues for the ECS Project

193-WP-611-001	Science-based System Architecture Drivers for the ECS Project, Revision 1.0
193-WP-623-001	ECS Evolutionary Development White Paper
194-TP-266-002	Data Distribution Architecture Logical Object Model (LOM) for the ECS Project, Version 2.01
194-TP-267-001	Data Server Architecture Logical Object Model (LOM) for the ECS Project, Version 2.00
194-TP-313-001	ECS User Characterization Methodology and Results
194-TP-316-002	Data Compression Study for the ECS Project
194-TP-548-001	User Scenario Functional Analysis [for the ECS Project]
194-TP-569-001	PDPS Prototyping at ECS Science and Technology Laboratory, Progress Report #4
194-WP-901-002	EOSDIS Core System Science Information Architecture, White Paper, Working Paper
194-WP-902-002	ECS Science Requirements Summary, White Paper, Working Paper
194-WP-904-002	Multi-Track Development for the ECS Project, White Paper, Working Paper
194-WP-913-003	User Environment Definition for the ECS Project, White Paper, Working Paper
194-WP-914-001	CORBA Object Request Broker Survey for the ECS Project, White Paper, Working Paper
194-WP-918-001	DADS Prototype One FSMS Product Operational Evaluation, White Paper, Draft Report
194-WP-925-001	Science Software Integration and Test, White Paper, Working Paper
222-TP-003-003	Release Plan Content Description for the ECS Project
420-WP-001-001	Maximizing the Use of COTS Software in the SDPS SDS Software Design, White Paper
430-TP-001-001	SDP Toolkit Implementation with Pathfinder SSM/I Precipitation Rate Algorithm, Technical Paper
410-TD-001-002	ECS User Interface Style Guide Technical Data
423-16-01	Goddard Space Flight Center, Data Production Software and Science Computing Facility (SCF) Standards and Guidelines
423-16-02	Goddard Space Flight Center, PGS Toolkit Requirements Specification for the ECS Project
423-41-02	Goddard Space Flight Center, Functional and Performance Requirements Specification for the Earth Observing System Data and Information System (EOSDIS) Core System

540-022

Goddard Space Flight Center, Earth Observing System (EOS)
Communications (Ecom) System Design Specification

560-EDOS-0211.0001

Goddard Space Flight Center, Interface Requirements Document
Between EDOS and the EOS Ground System (EGS)

3. Interoperability Subsystem Overview

3.1 Subsystem Overview

The SDPS Interoperability Subsystem includes one CSCI, the Advertising Service. Users access this subsystem to search for and locate both ECS and non-ECS services, providers, and data. The subsystem receives advertising information from ECS and non-ECS service providers, and provides interfaces for users to access and subscribe to the advertising information.

3.2 Subsystem Structure

The Interoperability Subsystem for Release B is composed of one CSCI, the Advertising Service (ADSRV) and one HWCI, Advertising Service HWCI (ADSHW). The HWCI is shared with Data Management subsystem and the details can be found in Section 8 of Data Management Subsystem Design Specification (305-CD-023-001).

The Interoperability Subsystem context diagram is presented in Figure 3.2-1. It illustrates the relationships between the Interoperability subsystem and the other SDPS subsystems. The following bullets outline only major flows.

- The subsystem accepts advertisements and search requests from the Client subsystem.
- The subsystem accepts search requests from the Ingest, Planning, and Processing subsystems.
- The subsystem also accepts advertisements and search requests from the Data Server and Data Management subsystems.
- Advertisements, search requests, and installation instructions are also received from non-ECS service providers such as International Partners, Science Computing Facilities, Affiliated Data Centers, etc.
- Lifecycle commands and mode requests are received from the Management Subsystem and events are logged with the Management Subsystem.

The Interoperability Subsystem interfaces are listed in Table 3.2-1. The flow names are defined in Table 3.2-2.

3.3 Subsystem Design Rationale

This Interoperability Subsystem is being developed on the incremental development track. The general design emphasizes a user friendly interface for a scalable, evolvable, and easily maintained subsystem. The subsystem design is driven primarily by the following drivers:

- Provide an integrated view of the data and service network.
- Evolve Interoperability capabilities as technology and research progress.
- Allow for an extensible provider network.
- Provide a framework for the "publish and subscribe paradigm" in order to access data and services.

The following new features are being implemented in Release B:

- Links within the advertising service to definitions within the Data Dictionary Service.
- Support for subscriptions on insertion, deletion, and modification of advertisements.

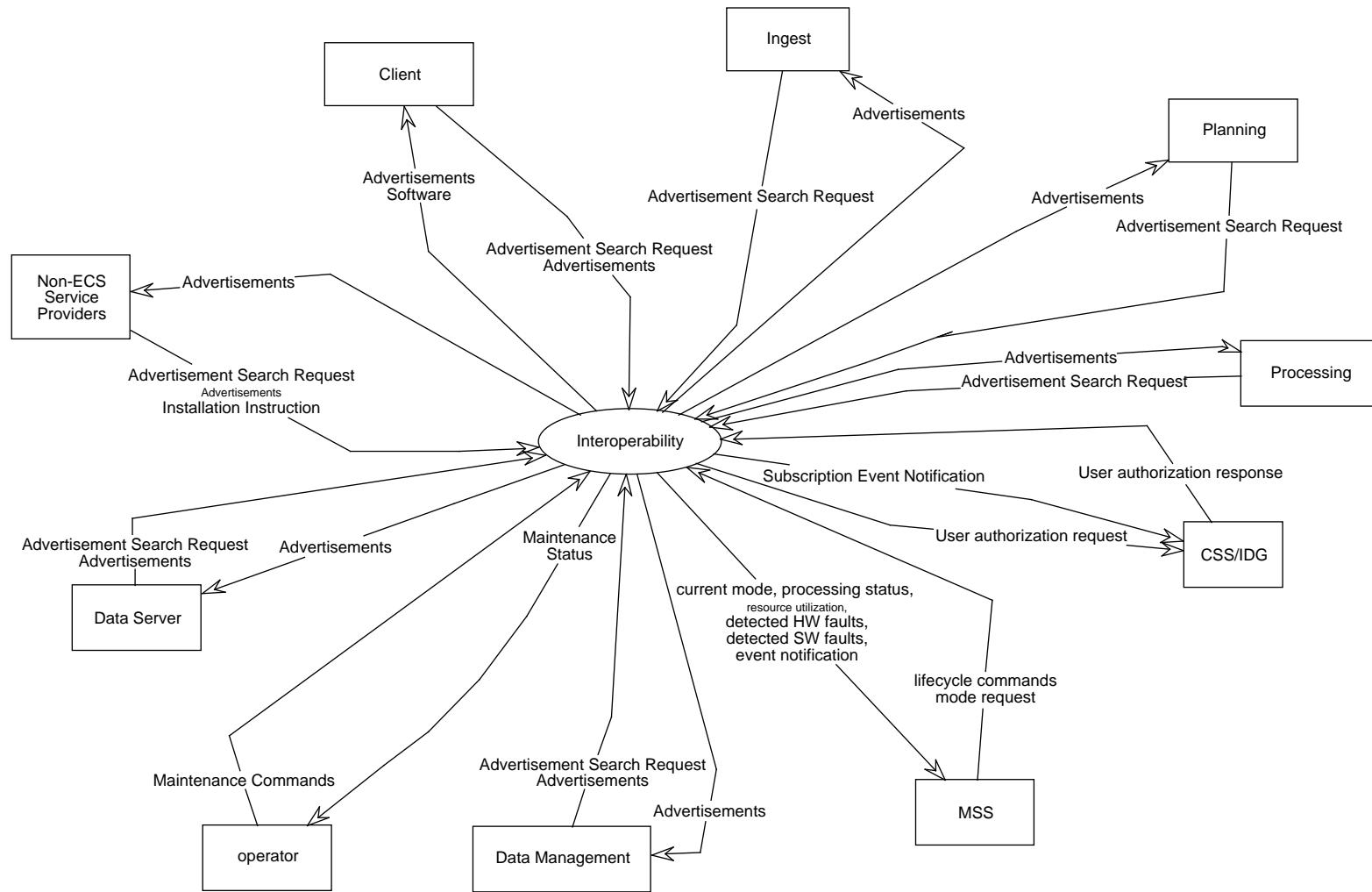


Figure 3.2-1. Interoperability Subsystem Context

Table 3.2-1. Interoperability Subsystem Interface (1 of 2)

Source	Destination	Data Types	Data Volume	Frequency
Client	Interoperability	Advertisements	low	as requested
Client	Interoperability	Advertisement Search Requests	low	as requested
Interoperability	Client	Advertisements	low	in response to requests
Interoperability	Client	Software	low	in response to request
Data Management	Interoperability	Advertisements	low	as required
Data Management	Interoperability	Advertisement Search Request	low	as required
Interoperability	Data Management	Advertisements	low	as requested
Data Server	Interoperability	Advertisements	low	as required
Data Server	Interoperability	Advertisement Search Request	low	as required
Interoperability	Data Server	Advertisements	low	in response to request
Non-ECS Service Providers	Interoperability	Advertisements	low	as required
Non-ECS Service Providers	Interoperability	Advertisement Search Request	low	as required
Non-ECS Service Providers	Interoperability	Installation Instruction	low	as required
Interoperability	Non-ECS Service Providers	Advertisements	low	as requested
Ingest	Interoperability	Advertisement Search Requests	low	as required
Interoperability	Ingest	Advertisements	low	in response to requests
Planning	Interoperability	Advertisement Search Request	low	as required
Interoperability	Planning	Advertisements	low	in response to requests
Processing	Interoperability	Advertisement Search Request	low	as required
Interoperability	Processing	Advertisements	low	in response to requests
MSS	Interoperability	lifecycle commands	low	as required
MSS	Interoperability	mode request	low	as required
Interoperability	MSS	current mode	low	as requested

Table 3.2-1. Interoperability Subsystem Interface (2 of 2)

Source	Destination	Data Types	Data Volume	Frequency
Interoperability	MSS	processing status	low	as required
Interoperability	MSS	resource utilization	low	as required
Interoperability	MSS	detected HW faults	low	as required
Interoperability	MSS	detected SW faults	low	as required
Interoperability	MSS	event notification	low	as required
Interoperability	CSS/IDG	user authentication request	low	as required
Interoperability	CSS/IDG	subscription event notification	low	as required
CSS/IDG	Interoperability	user authentication response	low	as requested
operator	Interoperability	Maintenance Commands	low	as required
Interoperability	operator	Maintenance status	low	in response to request

In the table, where an exact number is unavailable, the data volume is estimated as low (less than 1 MB), medium (between 1 MB and 1 GB), or high (greater than 1 GB) per use defined in the frequency column. The frequency information will be updated as the interfaces are fully defined.

The shadings show changes from Release A to Release B. An additional change from Release A to Release B is the deletion of the interfaces to the Global Change Master Directory (GCMD).

Table 3.2-2 defines the flows specific to Interoperability. These flow names are used in Table 3.2-1 and the context diagram.

Table 3.2-2. Interoperability Data Flow Definitions (1 of 2)

Data Type	Data Type Description
Advertisements	There are three types of advertisements, service advertisements, product advertisements, and provider advertisements. These are submitted by ECS subsystems and non-ECS providers and are returned as the result of a search of the Advertising database as part of the Interoperability subsystem.
Advertisement Search Request	The calling subsystem creates a search request to retrieve an advertisement. In Release B this is expressed in an Earth Science Query Language.
Subscription Event Notification	A notification is sent upon a specific event occurring. It is sent to the subscription server when an event such as advertisement insertion occurs. The subscription server handles the notification to the subscribers. The notification can be either a process to process notification or an electronic mail message. An example of an event is a new advertisement being defined. The notification would include the universal reference (UR) to the advertisement.

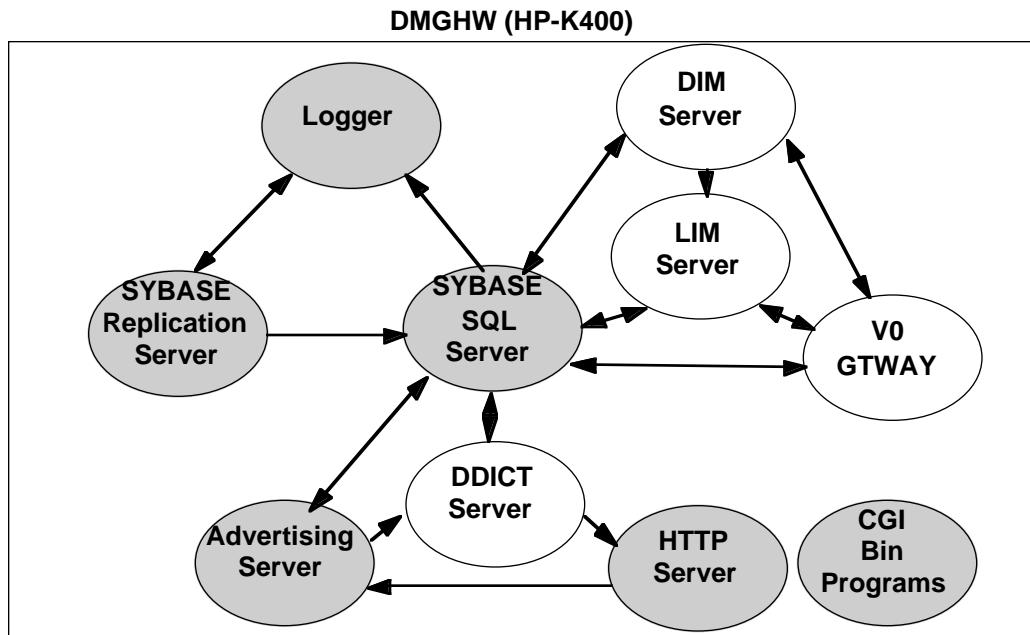
Table 3.2-2. Interoperability Data Flow Definitions (2 of 2)

Data Type	Data Type Description
Software	The software flow is a software component that must get installed on the client in order to run a particular service. For example, if the user is attempting to install a search service, but for some reason does not have the Earth Science Search Tool installed, the advertising service installer program will install this software on the user's workstation.
Installation Instructions	Non-ECS providers must provide a set of commands that should be run in order to install client software for advertised services. For example, suppose the Non-ECS Provider is advertising a subsetting service. The Interoperability service must know what client software needs to be installed and what the commands are to do this. An example is an ftp script.
Maintenance Commands	Maintenance commands are things like backup and recovery. These are performed by the operator on the Advertising Service DBMS and Navigation Server.
Maintenance Status	Maintenance status is the general status messages that occur during the maintenance execution.

3.4 Subsystem Physical Design

The advertising service processes reside on the Data Management Subsystem Hardware.

INTEROPERABILITY SUBSYSTEM COMPONENT INTERACTION DIAGRAM



*Components used by the Interoperability Subsystem.

Figure 3.4-1. Software to Hardware Mapping Diagram

3.5 Key Mechanisms Used

The advertising service uses the following key mechanisms, described in other 305 documents:

- URs
- Server Request Framework
- Subscriptions
- Managed Process Framework
- Event Logging

4 ADSRV - Advertising Service CSCI

4.1 CSCI Overview

The Advertising Service provides the interfaces needed to support Client defined interactive submission, browsing, searching, and retrieving of advertisements. Although there will be a single format for submitting advertisements to the service, advertisements should be accessible via several different interfaces to support database searching, text searching, and hyper linked access and retrieval according to several different viewing styles (e.g., plain ASCII text, interactive form, or HTML document).

A data server or other provider will advertise its data collections and services with the Advertising Service. The advertisement will include a listing of all products (and other Earth Science Data Types) available in the collection and a set of product attributes. The Advertising Service database is replicated across the DAACs for easy access to the information. The COTS DBMS provides the replication functions. Advertisements include directory level metadata, therefore, the attributes reflected in the advertising service include the ECS Core Metadata Directory-Level attributes that apply to collections. The client will send user queries which access only directory level metadata directly to the advertising service (rather than sending it as a distributed query to the various sites which provided the advertising information). A user who wishes to find out what data sets are available on the network can search (i.e., formulate a query) or browse (i.e., navigate through hyperlinked pages of advertisements) the advertising information. Both types of 'directory searching' are available on the user's desktop; the user can choose whichever approach is most convenient in the current work context.

Since the ADSRV CSCI is an incremental track development component, requirements, schedule, scenarios, issues and design are documented in a Software Development File (SDF) for ADSRV. The details provided in this document are to specify the interfaces to and from the ADSRV since these are controlled through the formal configuration management process as are formal development track components.

4.2 Advertising Service Context

ADSRV CSCI is the only CSCI in the Interoperability Subsystem. Therefore the context of the CSCI is identical to the subsystem context which is shown in Figure 3.2-1.

4.3 Advertising Service Object Model

Figure 4.3-1. EcPoPersist Object Model Diagram includes two classes, EcPoHandle and EcPoPersistentBase. These classes make up the persistent object framework. They provide behavior that simplifies the management of storing/retrieving complex inter-related object structures to the database. The Handle/PersistentBase classes provide reference counting, deferred fetching, a uniform interface for database interaction, coordinated-fetch across encapsulated objects, and object caching for performance and structure.

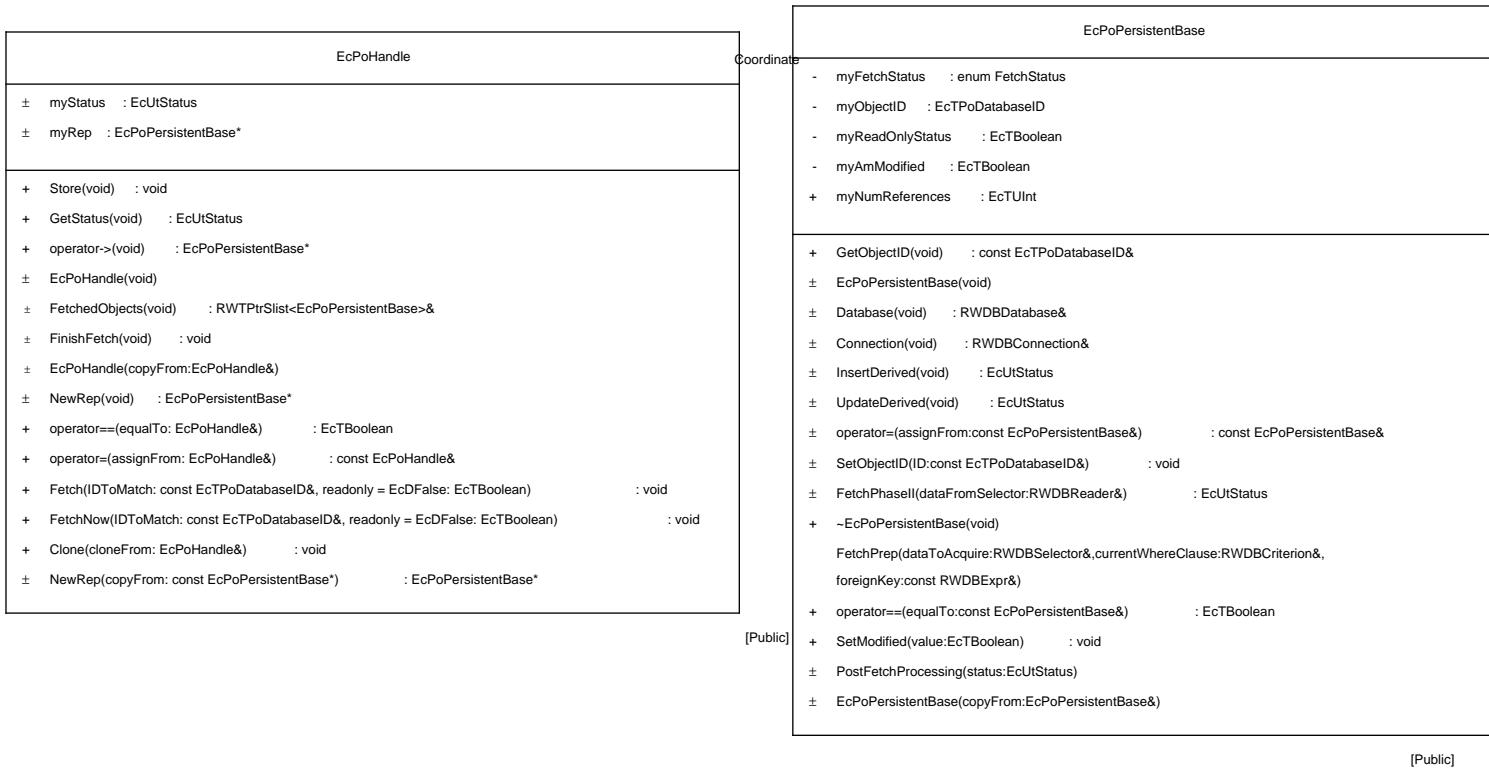


Figure 4.3-1. EcPoPersist Object Model Diagram

Figure 4.3.2. IoAdAdvHandles Object Model Diagram provides objects representing service advertisements (i.e. class IoAdService), provider advertisements (i.e. class IoAdProvider), and product advertisements (i.e. class IoAdAdvertisement). Each of these advertisement classes is derived from the generalized class (class IoAdAdvertisement).

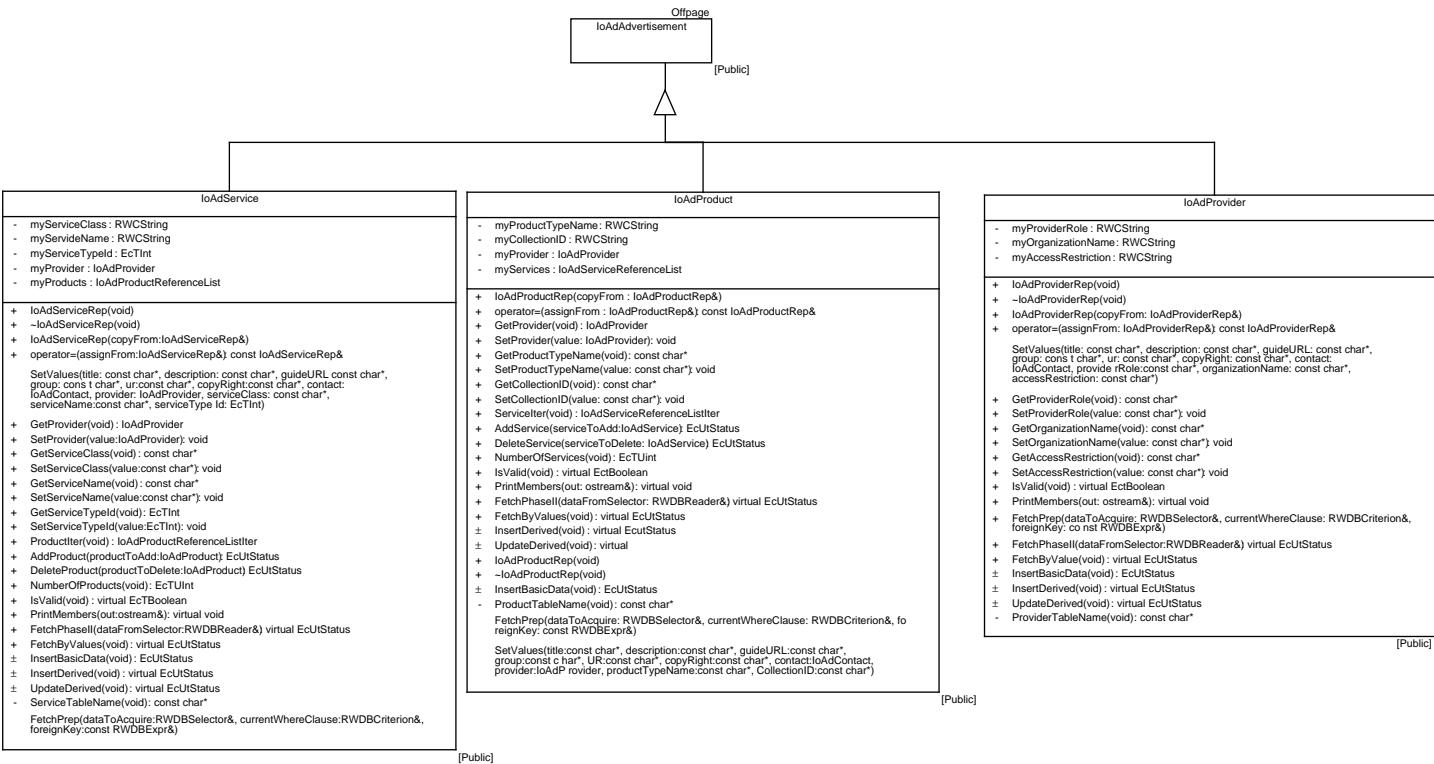


Figure 4.3-3. IoAdHandles Object Model Diagram shows EcPoHandle class as an abstract base class to allow reference counting, deferred fetching, and caching of persistent objects. It is the generalized class for IoAdContact and IoAdAdvertisement.

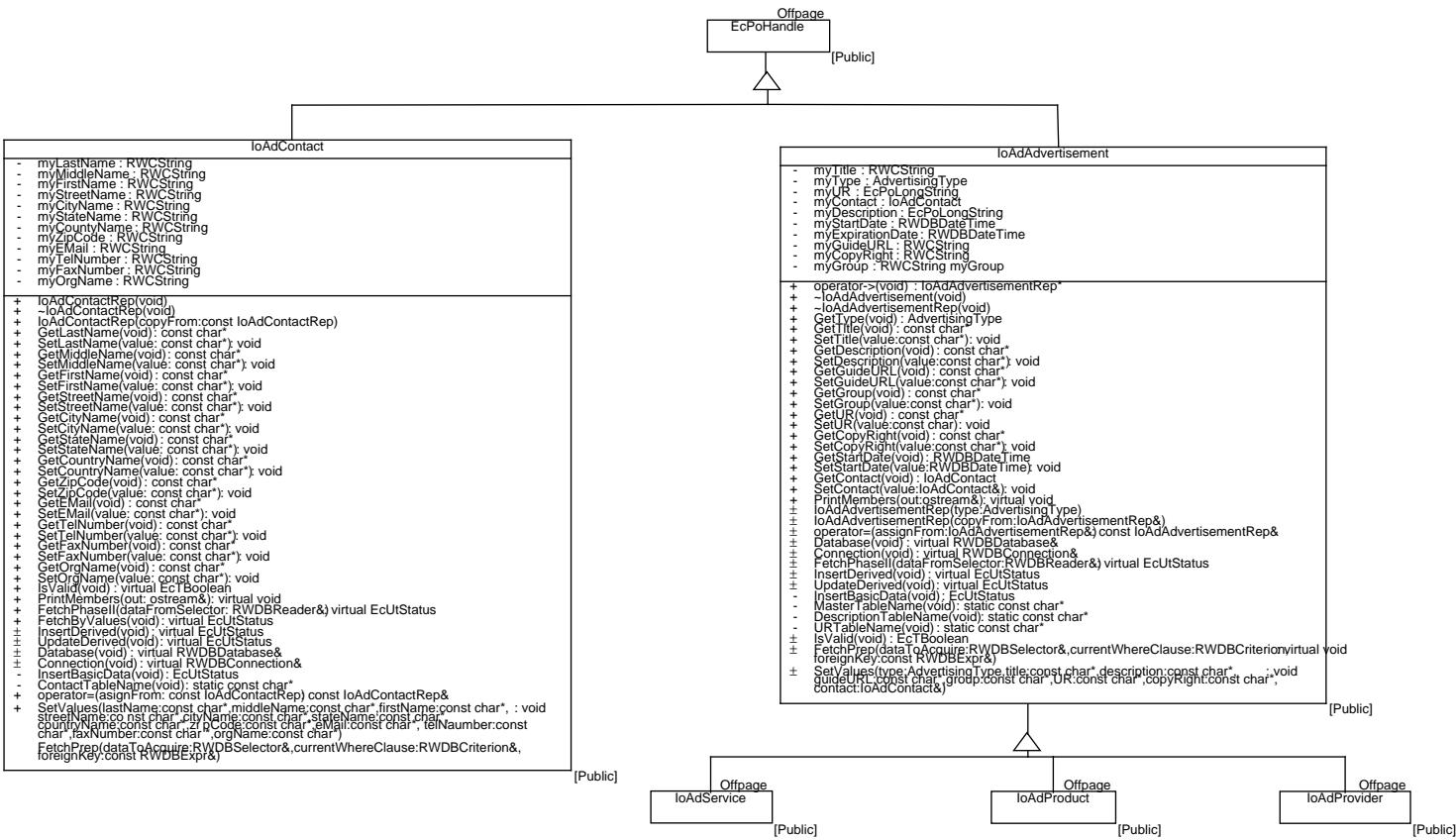


Figure 4.3-3. IoAdHandles Object Model Diagram

Figure 4.3-4. IoAdSearchCommand Object Model Diagram shows classes to support searching capabilities for various types of advertisements (Product, Service, Provider, and Signature Service). It also provides searching for advertisement contact.

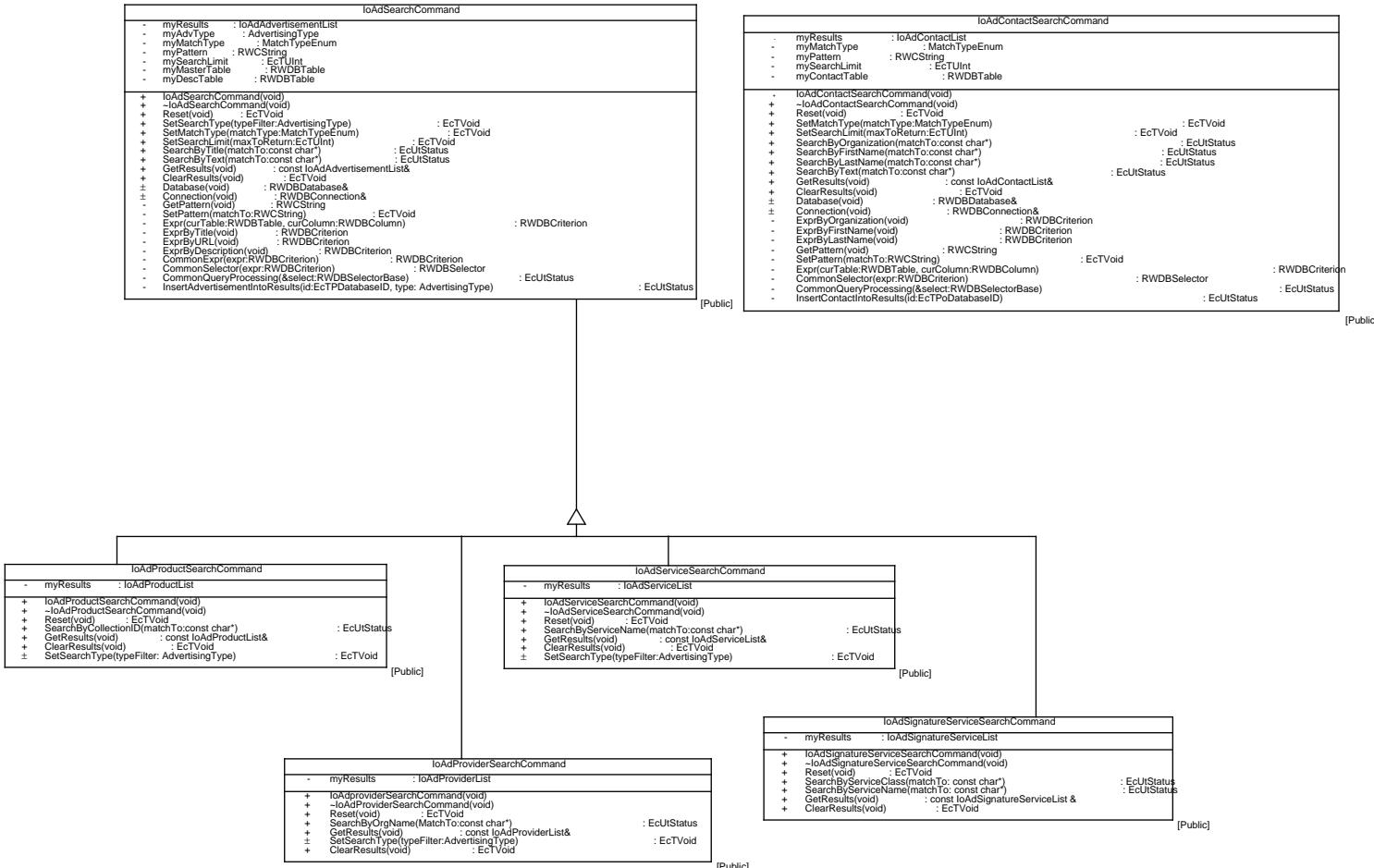
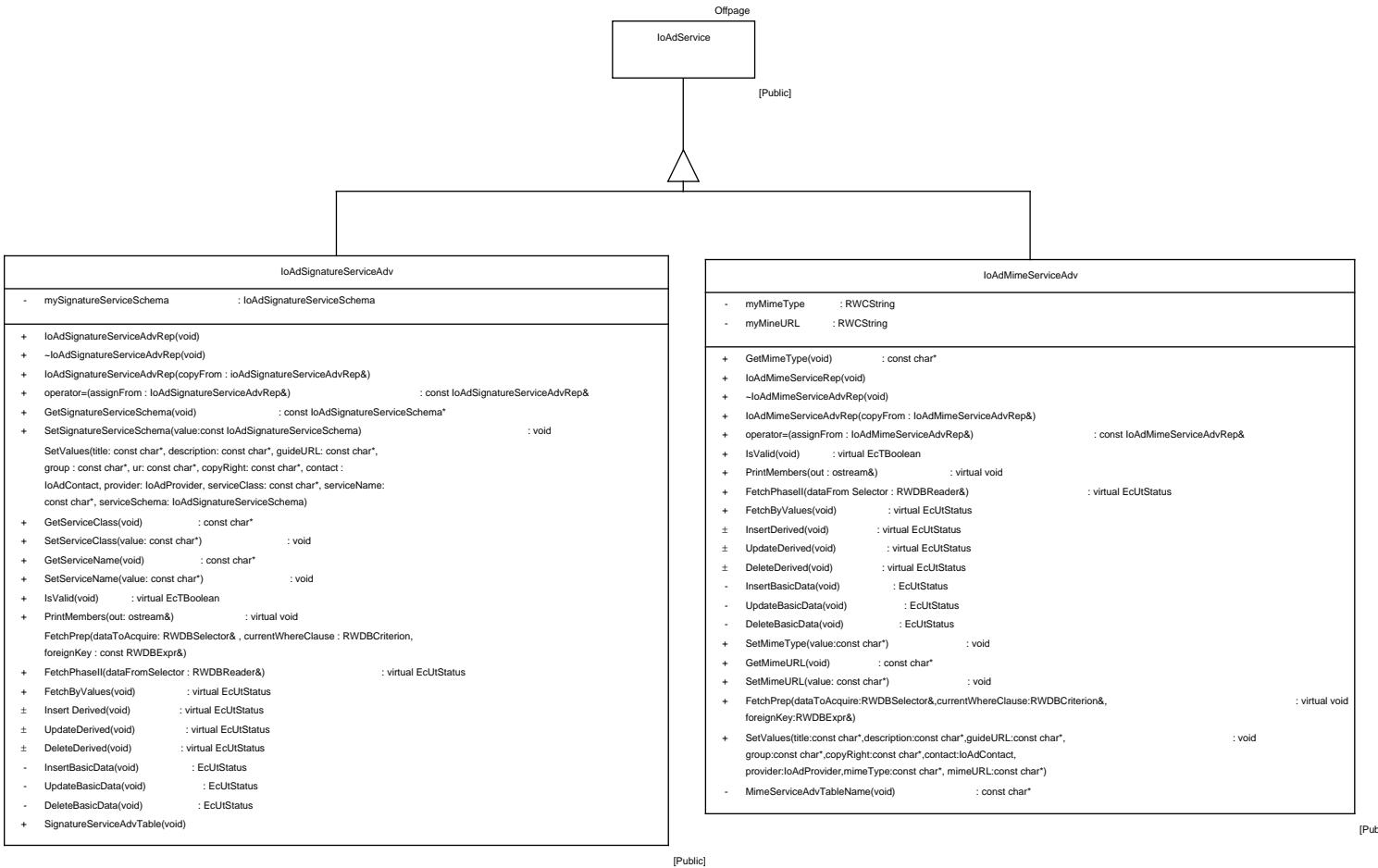


Figure 4.3-4. **IoAdSearchCommand Object Model Diagram**

Figure 4.3-5. *IoAdServiceHandles* Object Model Diagram declares *IoAdService* class as a generalized class for *IoAdSignatureServiceAdv* and *IoAdMimeServiceAdv* classes.



4.3.1 EcPoHandle Class

Parent Class: Not Applicable

Public: Yes

Distributed Object: No

Purpose and Description:

Public View: This class works in concert with EcPoPersistentBase to form the abstract base classes for a persistent object framework. This includes:
o Deferred fetching
o Simplified data storing
o Reference counting
o Caching for performance and structure preservation. All objects inheriting from this class separate the request of a fetch from the actual database interaction. Fetches are deferred until data or services relating to the fetched object are accessed. This allows performance efficient access. In addition, this class supports a simplified model of data storage. The user of the class does not have to track whether the current object was already in the database or is a new object. The system will insert or update as appropriate. Derived classes of this class are not the actual objects requested by the user, but only handles to the objects. Thus when a user fetches a complex object out of the database, the user need not be concerned with memory management. All objects are reference counted. When references are no longer exist to the object, the object are deleted. This class maintains a cache of stored and fetched objects. If multiple clients fetch the same logical object from memory, they will be sharing it. This enables complex objects that share sub-components to be stored/fetched and maintain their structure.

Protected View: Concrete and abstract subclasses must define the following operations:
o An operator -> that returns the appropriate representation type. Concrete subclasses must define the following operations:
o A NewRep() and NewRep(copyFrom) operation that returns new representation
o A FetchedObjects() operation that returns access to the concrete cache.

Private View: None

Attributes:

myRep - This is the current concrete representation.

Data Type: EcPoPersistentBase*

Privilege: Protected

Default Value:

myStatus - This is the current status.

Data Type: EcUtStatus

Privilege: Protected

Default Value:

Operations:

Clone - This operation makes this handle use a new copy (clone) of an existing representation. Since copy does representation sharing, this is the only way to get a logical duplicate of the object.

Arguments: cloneFrom: EcPoHandle&

Return Type: void

Privilege: Public

EcPoHandle - Default constructor. It creates this object with a new representation.

Arguments: void

Return Type: Void

Privilege: Protected

EcPoHandle - Copy constructor. It creates this object by sharing the representation of another handle.

Arguments: copyFrom:EcPoHandle&

Return Type: Void

Privilege: Protected

Fetch - This operation binds this object to a database object specified by IDToMatch and "logically" load the data from that object into this object.

Arguments: IDToMatch: const EcTPoDatabaseID&, readonly = EcDFalse: EcTBoolean

Return Type: void

Privilege: Public

FetchNow - This operation is the same as matching Fetch, but WILL NOT defer. It gets the data for this object, based on the argument ID.

Arguments: IDToMatch: const EcTPoDatabaseID&, readonly = EcDFalse: EcTBoolean

Return Type: void

Privilege: Public

FetchedObjects - This operation returns the container of concrete cached objects.

Arguments: void

Return Type: RWTPtrSlist<EcPoPersistentBase>&

Privilege: Protected

FinishFetch - This operation forces the completion of any pending fetches. Derived handle class referencing representation data should always call this first.

Arguments: void

Return Type: void

Privilege: Protected

GetStatus - This operation returns the status for the object's data. It checks if the object's data is valid or not. If the status is OK, it forces a database fetch.

Arguments: void

Return Type: EcUtStatus

Privilege: Public

NewRep - This operation returns a new concrete representation.

Arguments: void

Return Type: EcPoPersistentBase*

Privilege: Protected

NewRep - This operation returns a new concrete representation based on an existing one.

Arguments: copyFrom: const EcPoPersistentBase*

Return Type: EcPoPersistentBase*

Privilege: Protected

Store - This operation stores the current data to the database. If this object has never been in the database, does an insert, otherwise, does an update. The status object will contain our validity information.

Arguments: void

Return Type: void

Privilege: Public

operator-> - If the client is looking for a PersistentBase representation, this operation returns it.

Arguments: void

Return Type: EcPoPersistentBase*

Privilege: Public

operator= - This operation resets this object handle to share another handle's representation.

Arguments: assignFrom: EcPoHandle&

Return Type: const EcPoHandle&

Privilege: Public

operator== - This operation compares the representations of this and another handle.

Arguments: equalTo: EcPoHandle&

Return Type: EcTBoolean

Privilege: Public

Associations:

The EcPoHandle class has associations with the following classes:

Class: EcPoPersistentBase Coordinate

4.3.2 EcPoPersistentBase Class

Parent Class: Not Applicable

Public: Yes

Distributed Object: No

Purpose and Description:

Public View: Not much of one. This class supports the concept of a persistent object having a unique ID. Protected View: This class provides the framework used by EcPoHandle to implement its deferred fetching capabilities. The methods for each subclass are specified here, but called by our Handle. Private View: None.

Attributes:

myAmModified - This attribute contains a Boolean value. If TRUE, the object's data is changed and that it is set to modify so that the database will UPDATE and not INSERT.

Data Type: EcTBoolean

Privilege: Private

Default Value:

myFetchStatus - This is the various states of fetching (FetchNotRequested, FetchRequested, FetchDone).

Data Type: enum FetchStatus

Privilege: Private

Default Value:

myNumReferences - This is the counter of the number of handles accesses us.

Data Type: EcTUInt

Privilege: Public

Default Value:

myObjectID - This attribute contains the current object ID.

Data Type: EcTPoDatabaseID

Privilege: Private

Default Value:

myReadOnlyStatus - This attribute contains the read only status which has a value of either TRUE or FALSE.

Data Type: EcTBoolean

Privilege: Private

Default Value:

Operations:

Connection - This operation returns a connection for the current object.

Arguments: void

Return Type: RWDBConnection&

Privilege: Protected

Database - This operation returns the database for the current object.

Arguments: void

Return Type: RWDBDatabase&

Privilege: Protected

EcPoPersistentBase - Default constructor. Set reference count to 1 handle.

Arguments: void

Return Type: Void

Privilege: Protected

EcPoPersistentBase - Copy constructor.

Arguments: copyFrom:EcPoPersistentBase&

Return Type: Void

Privilege: Protected

FetchPhaseII - This operation processes the data in 'dataFromSelector' into the current object. The data in 'dataFromSelector' comes from executing the selector prepared in 'FetchPrep'.

Arguments: dataFromSelector:RWDBReader&

Return Type: EcUtStatus

Privilege: Protected

FetchPrep - This method shall process the selector, "dataToAcquire", to retrieve all relevant data for this object that can be returned in a single database fetch. This method shall add any constraints to the "currentWhereClause". This method shall add a constraint that selects only the data that are identified by its primary key equalling "foreignKey".

Arguments: dataToAcquire:RWDBSelector&,currentWhereClause:RWDBCriterion&,foreignKey:const RWDBExpr&

GetObjectID - This operation returns the unique object ID (primary key) for the current object. A valid ID determines whether we are currently in the database or not.

Arguments: void

Return Type: const EcTPoDatabaseID&

Privilege: Public

InsertDerived - This operation will insert the current object into the database and set the current object's ID with a new unique value.

Arguments: void

Return Type: EcUtStatus

Privilege: Protected

PostFetchProcessing - This operation sets our internal state data after a fetch. Any derived class that does its own kind of fetching should also call it with the status from the Fetch.

Arguments: status:EcUtStatus

Return Type: Void

Privilege: Protected

SetModified - This operation determines whether we are modified.

Arguments: value:EcTBoolean

Return Type: void

Privilege: Public

SetObjectID - This routine will set our object's ID. The ID determines whether the object is in the database or not.

Arguments: ID:const EcTPoDatabaseID&

Return Type: void

Privilege: Protected

UpdateDerived - This method will update an existing object in the database that matches our current ID with the data in this object.

Arguments: void

Return Type: EcUtStatus

Privilege: Protected

operator= - This operation enables assignment. It is similar to copy.

Arguments: assignFrom:const EcPoPersistentBase&

Return Type: const EcPoPersistentBase&

Privilege: Protected

operator== - This operation returns a True value if IDs are the same.

Arguments: equalTo:const EcPoPersistentBase&

Return Type: EcTBoolean

Privilege: Public

~EcPoPersistentBase - Default destructor.

Arguments: void

Return Type: Void

Privilege: Public

Associations:

The EcPoPersistentBase class has associations with the following classes:

Class: EcPoHandle Coordinate

4.3.3 IoAdAdvertisement Class

Parent Class: EcPoHandle

Public: Yes

Distributed Object: No

Purpose and Description:

Public View: This is an abstract handle class for all advertisements. It provides the operator-> operation to access members of the base class IoAdAdvertisementRep.

Protected View: We do not delete data in our dtor, our derived concrete class shall. We do not declare "NewRep()", our derived concrete class shall. We do not declare a "FetchedObjects" cache, our derived concrete class shall. Private View: None.

Attributes:

myContact - Who/What is responsible for the advertised entity.

Data Type: IoAdContact

Privilege: Private

Default Value:

myCopyRight - Any relevant copyright that applies to the entity being advertised.

Data Type: RWCString

Privilege: Private

Default Value:

myDescription - Long textual description of the advertised entity.

Data Type: EcPoLongString

Privilege: Private

Default Value:

myExpirationDate - When am I no longer valid.

Data Type: RWDBDateTime

Privilege: Private

Default Value:

myGroup - Logical group I am part of for administration.

Data Type: RWCString myGroup

Privilege: Private

Default Value:

myGuideURL - The Web URL for a guide to my entity.

Data Type: RWCString

Privilege: Private

Default Value:

myStartDate - When I am valid for advertisement.

Data Type: RWDBDateTime

Privilege: Private

Default Value:

myTitle - Stores the unique description for me.

Data Type: RWCString

Privilege: Private

Default Value:

myType - What derived type are we part of.

Data Type: AdvertisingType

Privilege: Private

Default Value:

myUR - A universal reference to access my advertised entity.

Data Type: EcPoLongString

Privilege: Private

Default Value:

Operations:

Connection - This operation returns a valid connection.

Arguments: void

Return Type: virtual RWDBConnection&

Privilege: Protected

Database - This operation returns a valid database with data in it.

Arguments: void

Return Type: virtual RWDBDatabase&

Privilege: Protected

DescriptionTableName - This operation contain names to encapsulate database table.

Arguments: void

Return Type: static const char*

Privilege: Private

FetchPhaseII - This operation gets all the data from FetchPrep.

Arguments: dataFromSelector:RWDBReader&

Return Type: virtual EcUtStatus

Privilege: Protected

FetchPrep - This operation prepares the selector to acquire all of classes data.

Arguments: dataToAcquire:RWDBSelector&,currentWhereClause:RWDBCriterion,
foreignKey:const RWDBExpr&

Return Type: virtual void

Privilege: Protected

GetContact - This operation gets who/what is responsible for the advertised entity.

Arguments: void

Return Type: IoAdContact

Privilege: Public

GetCopyRight - This operation gets any relevant copyrights that apply to the entity being advertised.

Arguments: void

Return Type: const char*

Privilege: Public

GetDescription - This operation gets a long textual description of the advertised entity.

Arguments: void

Return Type: const char*

Privilege: Public

GetGroup - This operation gets the logical group I am part of for administration.

Arguments: void

Return Type: const char*

Privilege: Public

GetGuideURL - This operation gets the Web URL for a guide to my entity.

Arguments: void

Return Type: const char*

Privilege: Public

GetStartDate - This operation gets the start date when the ad is valid.

Arguments: void

Return Type: RWDBDateTime

Privilege: Public

GetTitle - This operation gets a unique description of the title .

Arguments: void

Return Type: const char*

Privilege: Public

GetType - This operation gets the derived type that the advertisement is part of.

Arguments: void

Return Type: AdvertisingType

Privilege: Public

GetUR - This operation gets the universal reference to access the advertised entity.

Arguments: void

Return Type: const char*

Privilege: Public

InsertBasicData - This operation stores all attributes in database.

Arguments: void

Return Type: EcUtStatus

Privilege: Private

InsertDerived - This operation inserts our data if it is logically valid.

Arguments: void

Return Type: virtual EcUtStatus

Privilege: Protected

IoAdAdvertisementRep - Default constructor.

Arguments: type:AdvertisingType

Return Type: Void

Privilege: Protected

IoAdAdvertisementRep - Copy constructor.

Arguments: copyFrom:IoAdAdvertisementRep&

Return Type: Void

Privilege: Protected

IsValid - This operation checks the contents of the object for consistency and data rules:

o Contact is valid o Title, Description, and Group are not NULL

Arguments: void

Return Type: EcTBoolean

Privilege: Protected

MasterTableName - This operation contain names to encapsulate database table.

Arguments: void

Return Type: static const char*

Privilege: Private

PrintMembers - This operation writes contents to stream.

Arguments: out:ostream&

Return Type: virtual void

Privilege: Public

SetContact - This operation sets who/what is responsible for the advertised entity.

Arguments: value:IoAdContact&

Return Type: void

Privilege: Public

SetCopyRight - This operation sets any relevant copyrights that apply to the entity being advertised.

Arguments: value:const char*

Return Type: void

Privilege: Public

SetDescription - This operation sets the long textual description of the advertised entity.

Arguments: value:const char*

Return Type: void

Privilege: Public

SetGroup - This operation sets the logical group I am part of for administration.

Arguments: value:const char*

Return Type: void

Privilege: Public

SetGuideURL - This operation sets the Web URL for a guide to my entity.

Arguments: value:const char*

Return Type: void

Privilege: Public

SetStartDate - This operation sets the start date when the ad is valid.

Arguments: value:RWDBDate

Return Type: void

Privilege: Public

SetTitle - This operation sets a unique description of the title.

Arguments: value:const char*

Return Type: void

Privilege: Public

SetUR - This operation sets the universal reference to access the advertised entity.

Arguments: value:const char

Return Type: void

Privilege: Public

SetValues - This operation sets values for the parameters passed in the argument lists.

Arguments: type:AdvertisingType,title:const char*,description:const char*,
guideURL:const char*,group:const char*,UR:const char*,copyRight:const char*,
contact:IoAdContact&

Return Type: void

Privilege: Protected

URTableName - This operation contains names to encapsulate database table.

Arguments: void

Return Type: static const char*

Privilege: Private

UpdateDerived - This operation updates this object in the database for its current object ID.

Arguments: void

Return Type: virtual EcUtStatus

Privilege: Protected

operator-> - This operation enables the -> operation to access members of the base class IoAdAdvertisementRep.

Arguments: void

Return Type: IoAdAdvertisementRep*

Privilege: Public

operator= - This operator enables the assignment of two objects.

Arguments: assignFrom:IoAdAdvertisementRep&

Return Type: const IoAdAdvertisementRep&

Privilege: Protected

~IoAdAdvertisement - Default destructor.

Arguments: void

Return Type: Void

Privilege: Public

~IoAdAdvertisementRep - Default destructor.

Arguments: void

Return Type: Void

Privilege: Public

Associations:

The IoAdAdvertisement class has associations with the following classes:

None

4.3.4 IoAdContact Class

Parent Class: EcPoHandle

Public: Yes

Distributed Object: No

Purpose and Description:

Public View This class represents a person or organization responsible for an advertisement. It basically holds data. It supports the operations of the persistent object framework. Data and Services should not be accessed directly, but through IoAdContact.

IoAdContact performs handle services. Because fetching is deferred, any access can generate a database error. Therefore, one should check the handles status when appropriate.

Protected View: None. Private View: None.

Attributes:

myCityName - Stores the city name of the address of the contact person.

Data Type:RWCString

Privilege:Private

Default Value:

myCountyName - Stores the country name of the address of the contact person.

Data Type:RWCString

Privilege:Private

Default Value:

myEMail - Stores the e-mail address of the contact person.

Data Type:RWCString

Privilege:Private

Default Value:

myFaxNumber - Stores the fax phone number of the contact person.

Data Type:RWCString

Privilege:Private

Default Value:

myFirstName - Stores the first name of the contact person.

Data Type:RWCString

Privilege:Private

Default Value:

myLastName - Stores the last name of the contact person.

Data Type:RWCString

Privilege:Private

Default Value:

myMiddleName - Stores the middle name of the contact person.

Data Type:RWCString

Privilege:Private

Default Value:

myOrgName - Stores the name of the organization of the contact person.

Data Type:RWCString

Privilege:Private

Default Value:

myStateName - Stores the state of the address of the contact person

Data Type:RWCString

Privilege:Private

Default Value:

myStreetName - Stores the street name of the address of the contact person.

Data Type:RWCString

Privilege:Private

Default Value:

myTelNumber - Stores the telephone number of the contact person.

Data Type:RWCString

Privilege:Private

Default Value:

myZipCode - Stores the zip code of the address of the contact person.

Data Type:RWCString

Privilege:Private

Default Value:

Operations:

Connection - This operation returns the connection for our class.

Arguments:void

Return Type:virtual RWDBConnection&

Privilege:Protected

ContactTableName - This operation returns contact database table name.

Arguments:void

Return Type:static const char*

Privilege:Private

Database - This operation returns the database for our class.

Arguments:void

Return Type:virtual RWDBDatabase&

Privilege:Protected

FetchByValues - This routine selects an object from the database based on its state. For contact, the unique application state is FirstName, LastName, and Org.

Arguments:void

Return Type:virtual EcUtStatus

Privilege:Public

FetchPhaseII - This operation gets all the data from FetchPrep.

Arguments:dataFromSelector: RWDBReader&

Return Type:virtual EcUtStatus

Privilege:Public

FetchPrep - This operation prepares the selector to acquire all of classes data.

Arguments:dataToAcquire:RWDBSelector&,currentWhereClause:RWDBCriterion&,

foreignKey:const RWDBExpr&

GetCityName - This operation returns the city name of the address of the contact person.

Arguments:void

Return Type:const char*

Privilege:Public

GetCountryName - This operation returns the country name of the address of the contact person.

Arguments:void

Return Type:const char*

Privilege:Public

GetEMail - This operation returns the e-mail address of the contact person.

Arguments:void

Return Type:const char*

Privilege:Public

GetFaxNumber - This operation returns the fax phone number of the contact person.

Arguments:void

Return Type:const char*

Privilege:Public

GetFirstName - This operation returns the first name of the contact person.

Arguments:void

Return Type:const char*

Privilege:Public

GetLastName - This operation returns the last name of the contact person.

Arguments:void

Return Type:const char*

Privilege:Public

GetMiddleName - This operation returns the middle name of the contact person.

Arguments:void

Return Type:const char*

Privilege:Public

GetOrgName - This operation returns the name of the organization of the contact person.

Arguments:void

Return Type:const char*

Privilege:Public

GetStateName - This operation returns the state name of the address of the contact person.

Arguments:void

Return Type:const char*

Privilege:Public

GetStreetName - This operation returns the street name of the address of the contact person.

Arguments:void

Return Type:const char*

Privilege:Public

GetTelNumber - This operation returns the phone number of the contact person.

Arguments:void

Return Type:const char*

Privilege:Public

GetZipCode - This operation returns the zip code of the address of the contact person.

Arguments:void

Return Type:const char*

Privilege:Public

InsertBasicData - This operation stores all attributes in the database.

Arguments:void

Return Type:EcUtStatus

Privilege:Private

InsertDerived - This operation inserts our data if it is logically valid.

Arguments:void

Return Type:virtual EcUtStatus

Privilege:Protected

IoAdContactRep - Default constructor.

Arguments:void

Return Type:Void

Privilege:Public

IoAdContactRep - Copy constructor.

Arguments:copyFrom:const IoAdContactRep

Return Type:Void

Privilege:Public

IsValid - This operation determines whether the data meets the requirements.

Arguments:void

Return Type:virtual EcTBoolean

Privilege:Public

PrintMembers - This operation prints contents to stream.

Arguments:out: ostream&

Return Type:virtual void

Privilege:Public

SetCityName - This operation sets the city name of the address of the contact person.

Arguments:value: const char*

Return Type:void

Privilege:Public

SetCountryName - This operation sets the country name of the address of the contact person.

Arguments:value: const char*

Return Type:void

Privilege:Public

SetEMail - This operation sets the e-mail address of the contact person.

Arguments:value: const char*

Return Type:void

Privilege:Public

SetFaxNumber - This operation sets the fax phone number of the contact person.

Arguments:value: const char*

Return Type:void

Privilege:Public

SetFirstName - This operation sets the first name of the contact person.

Arguments:value: const char*

Return Type:void

Privilege:Public

SetLastName - This operation sets the last name of the contact person.

Arguments:value: const char*

Return Type:void

Privilege:Public

SetMiddleName - This operation sets the middle name of the contact person.

Arguments:value: const char*

Return Type:void

Privilege:Public

SetOrgName - This operation sets the name of the organization of the contact person.

Arguments:value: const char*

Return Type:void

Privilege:Public

SetStateName - This operation sets the state name of the address of the contact person.

Arguments:value: const char*

Return Type:void

Privilege:Public

SetStreetName - This operation sets the street name of the address of the contact person.

Arguments:value: const char*

Return Type:void

Privilege:Public

SetTelNumber - This operation sets the phone number of the contact person.

Arguments:value: const char*

Return Type:void

Privilege:Public

SetValues - This operation sets values for the parameters passed in the argument lists.
Arguments:lastName:const char*,middleName:const char*,firstName:const char*,
streetName:co nst char*,cityName:const char*,stateName:const char*,
countryName:const char*,zi pCode:const char*,eMail:const char*, telNaumber:const
char*,faxNumber:const char *,orgName:const char*
Return Type:void
Privilege:Public

SetZipCode - This operation sets the zip code of the address of the contact person.
Arguments:value: const char*
Return Type:void
Privilege:Public

UpdateDerived - This operation updates this object in the database for its current object ID.
Arguments:void
Return Type:virtual EcUtStatus
Privilege:Protected

operator= - This operation enables the assignment of two objects.
Arguments:asignFrom: const IoAdContactRep
Return Type:const IoAdContactRep&
Privilege:Public

~IoAdContactRep - Default destructor.
Arguments:void
Return Type:Void
Privilege:Public

Associations:

The IoAdContact class has associations with the following classes:
None

4.3.5 IoAdContactSearchCommand Class

Parent Class: Not Applicable

Public: Yes

Distributed Object: No

Purpose and Description:

Public View: This class provides interfaces for applications to search the set of all derived classes of advertisement contacts by specifying criterion. The persistent data will be stored

into a result list for additional searches or access. Users should set up options of how to search (filtering, patterns, how many results to return) and then call the search interfaces. While concrete, derived objects are placed in the results list, we return a list of advertisement contact objects. If one wants to access members of the advertisement contact, use the associated Contact-List class to see the Advertisement Contacts. Protected View: None. Private View: None.

Attributes:

myContactTable - This is an object to monitor the physical database table.

Data Type: RWDBTable

Privilege: Private

Default Value:

myMatchType - This is the match type (Prefix/Contain/Exact) that the pattern will be compared.

Data Type: MatchTypeEnum

Privilege: Private

Default Value:

myPattern - This is the matched pattern to be searched.

Data Type: RWCString

Privilege: Private

Default Value:

myResults - This attribute contains results of found advertisement contacts.

Data Type: IoAdContactList

Privilege: Private

Default Value:

mySearchLimit - This is the search limit for the match type (Prefix/Contain/Exact) in which the pattern will be compared.

Data Type: EcTUInt

Privilege: Private

Default Value:

Operations:

ClearResults - This operation clears all found advertisement contacts.

Arguments: void

Return Type: EcTVoid

Privilege: Public

CommonQueryProcessing - This operation processes the search of persistent object list for pattern matched.

Arguments: &select:RWDBSelectorBase

Return Type: EcUtStatus

Privilege: Private

CommonSelector - This operation constructs the search attribute for persistent objects.

Arguments: expr:RWDBCriterion

Return Type: RWDBSelector

Privilege: Private

Connection - This operation returns the default database connection.

Arguments: void

Return Type: RWDBConnection&

Privilege: Protected

Database - This operation connects to the database server.

Arguments: void

Return Type: RWDBDatabase&

Privilege: Protected

Expr - This operation formulates the search criterion according to the matched pattern.

Arguments: curTable:RWDBTable, curColumn:RWDBColumn

Return Type: RWDBCriterion

Privilege: Private

ExprByFirstName - This operation formulates the search criterion for First Name Search.

Arguments: void

Return Type: RWDBCriterion

Privilege: Private

ExprByLastName - This operation formulates the search criterion for Last Name Search.

Arguments: void

Return Type: RWDBCriterion

Privilege: Private

ExprByOrganization - This operation formulates the search criterion for Organization Name search.

Arguments: void

Return Type: RWDBCriterion

Privilege: Private

GetPattern - This operation retrieves the matched pattern to be searched.

Arguments: void

Return Type: RWCString

Privilege: Private

GetResults - This operation returns the found advertisement contacts.

Arguments: void

Return Type: const IoAdContactList&

Privilege: Public

InsertContactIntoResults - This operation creates an advertisement with the characteristic specified.

Arguments: id:EcTPoDatabaseID

Return Type: EcUtStatus

Privilege: Private

IoAdContactSearchCommand - Default constructor.

Arguments: void

Return Type: Void

Privilege: Public

Reset - This operation resets all the search criterion to default. Match types defaulted to prefix, and default search limit returns all.

Arguments: void

Return Type: EcTVoid

Privilege: Public

SearchByFirstName - This operation searches for any Advertisement Contact that contains a substring of matchTo in its First Name and appends found contact to the current results set.

Arguments: matchTo:const char*

Return Type: EcUtStatus

Privilege: Public

SearchByLastName - This operation searches for any Advertisement Contact that contains a substring of matchTo in its Last Name and appends found contact to the current results set.

Arguments: matchTo:const char*

Return Type: EcUtStatus

Privilege: Public

SearchByOrganization - This operation searches for any Advertisement Contact that contains a substring of matchTo in its Organization and append found contact to the current results set.

Arguments: matchTo:const char*

Return Type: EcUtStatus

Privilege: Public

SearchByText - This operation searches for any Contact that contains a substring of matchTo in its Organization, Last/First Name and append found Advertisement Contact to the current results set.

Arguments: matchTo:const char*

Return Type: EcUtStatus

Privilege: Public

SetMatchType - This operation sets the match type (Prefix/Contain/Exact) that the pattern will be compared.

Arguments: matchType:MatchTypeEnum

Return Type: EcTVoid

Privilege: Public

SetPattern - This operation stores the matched pattern to be searched.

Arguments: matchTo:RWCString

Return Type: EcTVoid

Privilege: Private

SetSearchLimit - This operation enables the specification of the maximum number of results to return for a given search. If not specified, all that match will be returned.

Arguments: maxToReturn:EcTUInt

Return Type: EcTVoid

Privilege: Public

~IoAdContactSearchCommand - Default destructor.

Arguments: void

Return Type: Void

Privilege: Public

Associations:

The IoAdContactSearchCommand class has associations with the following classes:

None

4.3.6 IoAdMimeServiceAdv Class

Parent Class: IoAdService

Public: Yes

Distributed Object: No

Purpose and Description:

Public View: This entity class supports operations to allow the definition, storage, and retrieval of an advertisement of a mime service. Member data and operations should not be accessed directly, but through IoAdMimeServiceAdv. IoAdMimeServiceAdv performs handle services. Because fetching is deferred, any access can generate a database error. Therefore, one should check the handle status when appropriate. Unless specified, the string values can be empty. Protected View: We inherit our Database() and Connection() from IoAdAdvertisement. Private View: None.

Attributes:

myMimeType - Stores the name of the Mime type of the the service.

Data Type: RWCString

Privilege: Private

Default Value:

myMineURL - Stores the URL of the Mime type of the service to be accessed.

Data Type: RWCString

Privilege: Private

Default Value:

Operations:

DeleteBasicData - This operation deletes all attributes in database.

Arguments: void

Return Type: EcUtStatus

Privilege: Private

DeleteDerived - This operation deletes this object in the database for its current object ID.

Arguments: void

Return Type: virtual EcUtStatus

Privilege: Protected

FetchByValues - This routine selects an object from database based on its state. For contact, the unique application state is MimeTypeID, MimeURL and ServiceID.

Arguments: void

Return Type: virtual EcUtStatus

Privilege: Public

FetchPhaseII - This operation gets all the data from FetchPrep.

Arguments: dataFrom Selector : RWDBReader&

Return Type: virtual EcUtStatus

Privilege: Public

FetchPrep - This operation prepares the selector to acquire all of classes data.

Arguments: dataToAcquire:RWDBSelector&,currentWhereClause:RWDBCriterion&,
foreignKey:RWDBExpr&

Return Type: virtual void

Privilege: Public

GetMimeType - This operation returns the name of the Mime type of the service.

Arguments: void

Return Type: const char*

Privilege: Public

GetMimeTypeURL - This operation returns the URL of the Mime type of the service.

Arguments: void

Return Type: const char*

Privilege: Public

InsertBasicData - This operation stores all attributes in database.

Arguments: void

Return Type: EcUtStatus

Privilege: Private

InsertDerived - This operation inserts our data if it is logically valid. This routine is part of the persistent framework.

Arguments: void

Return Type: virtual EcUtStatus

Privilege: Protected

IoAdMimeServiceAdvRep - Copy constructor.

Arguments: copyFrom : IoAdMimeServiceAdvRep&

Return Type: Void

Privilege: Public

IoAdMimeServiceRep - Default constructor.

Arguments: void

Return Type: Void

Privilege: Public

IsValid - This operation checks the contents of the object for consistency and data rules:
o Provider is valid o Ad Base is valid oMimeType and MimeURL are not NULL.

Arguments: void

Return Type: virtual EcTBoolean

Privilege: Public

MimeServiceAdvTableName - This operation gets the Mime Service Advertisement database Table Name.

Arguments: void

Return Type: const char*

Privilege: Private

PrintMembers - This operation writes contents to stream.

Arguments: out : ostream&

Return Type: virtual void

Privilege: Public

SetMimeType - This operation is used to set a new Mime type name for the service.

Arguments: value:const char*

Return Type: void

Privilege: Public

SetMimeURL - This operation is used to set a new URL of the Mime type of the service.

Arguments: value: const char*

Return Type: void

Privilege: Public

SetValues - This operation sets values for the parameters passed in the argument lists.

Arguments: title:const char*,description:const char*,guideURL:const char*,
group:const char*,copyRight:const char*,contact:IoAdContact,
provider:IoAdProvider,mimeType:const char*, mimeURL:const char*

Return Type: void

Privilege: Public

UpdateBasicData - This operation stores all attributes in database.

Arguments: void

Return Type: EcUtStatus

Privilege: Private

UpdateDerived - This operation updates this object in the database for its current object ID.

Arguments: void

Return Type: virtual EcUtStatus

Privilege: Protected

operator= - This operation enables the assignment of two objects.

Arguments: assignFrom : IoAdMimeServiceAdvRep&

Return Type: const IoAdMimeServiceAdvRep&

Privilege: Public

~IoAdMimeServiceAdvRep - Default destructor.

Arguments: void

Return Type: Void

Privilege: Public

Associations:

The IoAdMimeServiceAdv class has associations with the following classes:

None

4.3.7 IoAdProduct Class

Parent Class: IoAdAdvertisement

Public: Yes

Distributed Object: No

Purpose and Description:

Public View: This entity class supports operations to allow the definition, storage and retrieval of a advertisement of a data product. Typically, this product is an ECS collection. It can also be other kinds of collections or other general data. Member data and functions should not be accessed directly, but through IoAdProduct. IoAdProduct performs handle services. Because fetching is deferred, any access can generate a database error. Therefore, one should check the handles status when appropriate. Unless specified, the string values can be empty. Protected View: We inherit our Database() and Connection() from IoAdAdvertisement. Private View: None.

Attributes:

myCollectionID - Stores the ID of the collection.

Data Type: RWCString

Privilege: Private

Default Value:

myProductName - Stores the product name.

Data Type: RWCString

Privilege: Private

Default Value:

myProvider - Who is providing this product.

Data Type: IoAdProvider

Privilege: Private

Default Value:

myServices - What services can be applied to this data.

Data Type: IoAdServiceReferenceList

Privilege: Private

Default Value:

Operations:

AddService - This operation defines a new service to apply to this product

Arguments: serviceToAdd:IoAdService

Return Type: EcUtStatus

Privilege: Public

DeleteService - This operation removes an existing service that applied to this product.

Arguments: serviceToDelete: IoAdService

Return Type: EcUtStatus

Privilege: Public

FetchByValues - This routine selects an object from the database based on its state. For contact, the unique application is ProductName and CollectionID

Arguments: void

Return Type: virtual EcUtStatus

Privilege: Public

FetchPhaseII - This operation gets all the data from FetchPrep. This routine is part of the persistent framework.

Arguments: dataFromSelector: RWDBReader&

Return Type: virtual EcUtStatus

Privilege: Public

FetchPrep

Arguments: dataToAcquire: RWDBSelector&, currentWhereClause:
RWDBCriterion&, fo reignKey: const RWDBExpr&

GetCollectionID - This operation gets the ID of the collection.

Arguments: void

Return Type: const char*

Privilege: Public

GetProductName - This operation returns the product type name.

Arguments: void

Return Type: const char*

Privilege: Public

GetProvider - This operation gets the provider who is providing the product.

Arguments: void

Return Type: IoAdProvider

Privilege: Public

InsertBasicData - This operation stores all attributes in database.

Arguments: void

Return Type: EcUtStatus

Privilege: Protected

InsertDerived - This operation inserts our data if it is logically valid. This routine is part of the persistent framework.

Arguments: void

Return Type: virtual EcutStatus

Privilege: Protected

IoAdProductRep - Copy constructor.

Arguments: copyFrom : IoAdProductRep&

Return Type: Void

Privilege: Public

IoAdProductRep - Default constructor.

Arguments: void

Return Type: Void

Privilege: Public

IsValid - This operation checks the contents of the object for consistency and data rules:

o Provider is valid o Ad Base is valid o Collection ID is not NULL

Arguments: void

Return Type: virtual EctBoolean

Privilege: Public

NumberOfServices - This operation defines how many services apply to this product.

Arguments: void

Return Type: EcTUint

Privilege: Public

PrintMembers - This operation writes contents to stream.

Arguments: out: ostream&
Return Type: virtual void
Privilege: Public

ProductName - This is a private function to encapsulate table name.

Arguments: void
Return Type: const char*
Privilege: Private

ServiceIter - This operation provides a RWTPSlist-like iterator for all defined services

Arguments: void
Return Type: IoAdServiceReferenceListIter
Privilege: Public

SetCollectionID - This operation sets the ID of the collection.

Arguments: value: const char*
Return Type: void
Privilege: Public

SetProductTypeName - This operation is used to set a new product type name.

Arguments: value: const char*
Return Type: void
Privilege: Public

SetProvider - This operation sets the myProvider attribute.

Arguments: value: IoAdProvider
Return Type: void
Privilege: Public

SetValues

Arguments: title:const char*, description:const char*, guideURL:const char*, group:const char*, UR:const char*, copyRight:const char*, contact:IoAdContact, provider:IoAdProvider, productName:const char*, CollectionID:const char*

UpdateDerived - This operation updates this object in the database for its current object ID. This routine is part of the persistent framework.

Arguments: void
Return Type: virtual
Privilege: Protected

operator= - This operator enables the assignment of two objects.

Arguments: assignFrom : IoAdProductRep&
Return Type: const IoAdProductRep&
Privilege: Public

~IoAdProductRep - Default destructor.

Arguments: void

Return Type: Void

Privilege: Public

Associations:

The IoAdProduct class has associations with the following classes:

None

4.3.8 IoAdProductSearchCommand Class

Parent Class: IoAdSearchCommand

Public: Yes

Distributed Object: No

Purpose and Description:

Public View: This class provides interfaces for applications to search the set of product advertisements by specifying options and criterion. The persistent data will be stored into a results list for additional searches or access. Users should set up options of how to search (filtering, patterns, how many results to return) and then call the search interfaces.

Protected View: None. Private View: None.

Attributes:

myResults - Contains the matched Product Advertisement to user.

Data Type: IoAdProductList

Privilege: Private

Default Value:

Operations:

ClearResults - This operation clears the matched object linked list.

Arguments: void

Return Type: EcTVoid

Privilege: Public

GetResults - This operation returns the matched Product Advertisement to user.

Arguments: void

Return Type: const IoAdProductList&

Privilege: Public

IoAdProductSearchCommand - Default constructor.

Arguments: void

Return Type: Void

Privilege: Public

Reset - This operation resets all the search criterion to default. Default Adv type searches Product Advertisement type.

Arguments: void

Return Type: EcTVoid

Privilege: Public

SearchByCollectionID - This operation searches for Product Advertisement that contains a substring of matchTo in its Collection ID and appends found Product advertisements to the current results set.

Arguments: matchTo:const char*

Return Type: EcUtStatus

Privilege: Public

SetSearchType - This operation sets the type of search.

Arguments: typeFilter: AdvertisingType

Return Type: EcTVoid

Privilege: Protected

~IoAdProductSearchCommand - Default destructor.

Arguments: void

Return Type: Void

Privilege: Public

Associations:

The IoAdProductSearchCommand class has associations with the following classes:

None

4.3.9 IoAdProvider Class

Parent Class:IoAdAdvertisement

Public:Yes

Distributed Object:No

Purpose and Description:

Public View: This entity class supports operations to allow the definition, storage and retrieval of a advertisement of a data/service provider. Typically, this provider is a DAAC. It can also be any other organization that provides data or services. Member data and

functions should not be accessed directly, but through IoAdProvider. IoAdProvider performs handle services. Because fetching is deferred, any access can generate a database error. Therefore, one should check the handles status when appropriate. Unless specified, the string values can be empty. Protected View: We inherit our Database() and Connection() from IoAdAdvertisement. Private View: None.

Attributes:

myAccessRestriction - Stores a text description of any access restrictions imposed by the provider.

Data Type: RWCString

Privilege: Private

Default Value:

myOrganizationName - Stores the organization name of the provider.

Data Type: RWCString

Privilege: Private

Default Value:

myProviderRole - Stores the name of the role performed by the provider of the advertisement.

Data Type: RWCString

Privilege: Private

Default Value:

Operations:

FetchByValue - This routine selects an object from the database based on its state. For contact, the unique application state is ProviderRole and OrganizationName.

Arguments: void

Return Type: virtual EcUtStatus

Privilege: Public

FetchPhaseII - This operation gets all the data from FetchPrep. This routine is part of the persistent framework.

Arguments: dataFromSelector:RWDBReader&

Return Type: virtual EcUtStatus

Privilege: Public

FetchPrep

Arguments: dataToAcquire: RWDBSelector&, currentWhereClause: RWDBCriterion&, foreignKey: const RWDBExpr&

Return Type: virtual void

Privilege: Public

GetAccessRestriction - This operation returns a text description of access restrictions imposed by the provider.

Arguments: void

Return Type: const char*

Privilege: Public

GetOrganizationName - This operation returns the organization name of the provider.

Arguments: void

Return Type: const char*

Privilege: Public

GetProviderRole - This operation returns the name of the role of the provider.

Arguments: void

Return Type: const char*

Privilege: Public

InsertBasicData - This operation stores all attributes in database.

Arguments: void

Return Type: EcUtStatus

Privilege: Protected

InsertDerived - This operation inserts our data if it is logically valid. This routine is part of the persistent framework.

Arguments: void

Return Type: virtual EcUtStatus

Privilege: Protected

IoAdProviderRep - Default constructor.

Arguments: void

Return Type: Void

Privilege: Public

IoAdProviderRep - Copy constructor.

Arguments: copyFrom: IoAdProviderRep&

Return Type: Void

Privilege: Public

IsValid - This operation checks the contents of the object for consistency and data rules:

o Ad Base is valid o Organization name and Provider role is not NULL

Arguments: void

Return Type: virtual EctBoolean

Privilege: Public

PrintMembers - This operation writes contents to stream.

Arguments: out: ostream&

Return Type: virtual void

Privilege: Public

ProviderTableName - This is a private function to encapsulate table name.

Arguments: void

Return Type: const char*

Privilege: Private

SetAccessRestriction - This operation set a new text description of access restrictions imposed by the provider.

Arguments: value: const char*

Return Type: void

Privilege: Public

SetOrganizationName - This operation sets the organization name of the provider.

Arguments: value: const char*

Return Type: void

Privilege: Public

SetProviderRole - This operation sets the role name of the provider.

Arguments: value: const char*

Return Type: void

Privilege: Public

SetValues

Arguments: title: const char*, description: const char*, guideURL: const char*, group: const char*, ur: const char*, copyRight: const char*, contact: IoAdContact, providerRole: const char*, organizationName: const char*, accessRestriction: const char*

UpdateDerived - This operation updates this object in the database for its current object ID. This routine is part of the persistent framework.

Arguments: void

Return Type: virtual EcUtStatus

Privilege: Protected

operator= - This operator enables the assignment of two objects.

Arguments: assignFrom: IoAdProviderRep&

Return Type: const IoAdProviderRep&

Privilege: Public

~IoAdProviderRep - Default destructor.

Arguments: void

Return Type: Void

Privilege: Public

Associations:

The IoAdProvider class has associations with the following classes:

None

4.3.10 IoAdProviderSearchCommand Class

Parent Class: IoAdSearchCommand

Public: Yes

Distributed Object: No

Purpose and Description:

Public View: This class provides interfaces for applications to search the set of product advertisements by specifying options and criterion. The persistent data will be stored into a results list for additional searches or access. Users should set up options of how to search (filtering, patterns, how many results to return) and then call the search interfaces.

Protected View: None. Private View: None.

Attributes:

myResults - This is the result that contains the matched Provider Advertisement.

Data Type: IoAdProviderList

Privilege: Private

Default Value:

Operations:

ClearResults - This operation clears the matched object linked list.

Arguments: void

Return Type: EcTVoid

Privilege: Public

GetResults - This operation returns the matched Provider Advertisement to user.

Arguments: void

Return Type: const IoAdProviderList&

Privilege: Public

IoAdproviderSearchCommand - Default constructor.

Arguments: void

Return Type: Void

Privilege: Public

Reset - This operation resets all the search criterion to default. Default Adv type searches Provider Advertisement type.

Arguments: void

Return Type: EcTVoid

Privilege: Public

SearchByOrgName - This operation searches for Provider Advertisement that contains a substring of matchTo in its organization Name and appends found Product Advertisements to the current results set.

Arguments: MatchTo:const char*

Return Type: EcUtStatus

Privilege: Public

SetSearchType - This operation hides from users our base's SetSearchType method.

Arguments: typeFilter:AdvertisingType

Return Type: EcTVoid

Privilege: Protected

~IoAdProviderSearchCommand - Default destructor.

Arguments: void

Return Type: Void

Privilege: Public

Associations:

The IoAdProviderSearchCommand class has associations with the following classes:

None

4.3.11 IoAdSearchCommand Class

Parent Class: Not Applicable

Public: Yes

Distributed Object: No

Purpose and Description:

Public View: This class provides interfaces for applications to search the set of all derived classes of advertisements by specifying criterion. The persistent data will be stored into a results list for additional searches or access. Users should set up options of how to

search (filtering, patterns, how many results to return) and then call the search interfaces. While concrete, derived objects are placed in the results list, we return a list of abstract ads objects. If one wants to access members of the derived type of ads, use the associated derived-ad-List class to filter the ad-list. For example, to see Product Ads, use the ProductList class. Protected View: None. Private View: None.

Attributes:

myAdvType - What kind of search type (product, service, provider).

Data Type: AdvertisingType

Privilege: Private

Default Value:

myDescTable - The name of the description table of the database.

Data Type: RWDBTable

Privilege: Private

Default Value:

myMasterTable - The name of the master table.

Data Type: RWDBTable

Privilege: Private

Default Value:

myMatchType - Contains types of pattern (Prefix/Contain/Exact) matching allowable on string searches.

Data Type: MatchTypeEnum

Privilege: Private

Default Value:

myPattern - Contains the matched pattern to be searched.

Data Type: RWCString

Privilege: Private

Default Value:

myResults - Contains accumulated set of results for this search.

Data Type: IoAdAdvertisementList

Privilege: Private

Default Value:

mySearchLimit - This is the match type (Prefix/Contain/Exact) that the pattern will be compared, or else all the matched will be found.

Data Type: EcTUInt

Privilege: Private

Default Value:

Operations:

ClearResults - This operation clears the matched object linked list.

Arguments: void

Return Type: EcTVoid

Privilege: Public

CommonExpr - This operation formulates the search criterion for Adv Type search.

Arguments: expr:RWDBCriterion

Return Type: RWDBCriterion

Privilege: Private

CommonQueryProcessing - This operation processes the search of persistent object list for pattern matched.

Arguments: &select:RWDBSelectorBase

Return Type: EcUtStatus

Privilege: Private

CommonSelector - This operation constructs the search attribute for persistent objects.

Arguments: expr:RWDBCriterion

Return Type: RWDBSelector

Privilege: Private

Connection - This operation returns the default database connection.

Arguments: void

Return Type: RWDBConnection&

Privilege: Protected

Database - This operation connects to the database server.

Arguments: void

Return Type: RWDBDatabase&

Privilege: Protected

Expr - This operation formulates the search criterion according to the matched pattern.

Arguments: curTable:RWDBTable, curColumn:RWDBCOLUMN

Return Type: RWDBCriterion

Privilege: Private

ExprByDescription - This operation formulates the search criterion for Description search.

Arguments: void

Return Type: RWDBCriterion

Privilege: Private

ExprByTitle - This operation formulates the search criterion for Title search.

Arguments: void

Return Type: RWDBCriterion

Privilege: Private

ExprByUrl - This operation formulates the search criterion for URL search.

Arguments: void

Return Type: RWDBCriterion

Privilege: Private

GetPattern - This operation retrieves the matched pattern to be searched.

Arguments: void

Return Type: RWCString

Privilege: Private

GetResults - This operation returns the matched Advertisement object to user.

Arguments: void

Return Type: const IoAdAdvertisementList&

Privilege: Public

InsertAdvertisementIntoResults - This operation creates an advertisement with the characteristic specified.

Arguments: id:EcTPDatabaseID, type: AdvertisingType

Return Type: EcUtStatus

Privilege: Private

IoAdSearchCommand - Default constructor.

Arguments: void

Return Type: Void

Privilege: Public

Reset - This operation resets all the search criterion to default. Default Adv type searches all product/provider/service type, match Type is defaulted to prefix, and default search limit returns all.

Arguments: void

Return Type: EcTVoid

Privilege: Public

SearchByText - This operation searches for any Advertisement that contains a substring of matchTo in its Title, URL, or Description and appends found advertisements to the current results set.

Arguments: matchTo:const char*

Return Type: EcUtStatus

Privilege: Public

SearchByTitle - This operation searches for any Advertisement that contains a substring of matchTo in its Title and appends found advertisements to the current results set.

Arguments: matchTo:const char*

Return Type: EcUtStatus

Privilege: Public

SetMatchType - This operation sets the match type (Prefix/Contain/Exact) that the pattern will be compared.

Arguments: matchType:MatchTypeEnum

Return Type: EcTVoid

Privilege: Public

SetPattern - This operation stores the matched pattern to be searched.

Arguments: matchTo:RWCString

Return Type: EcTVoid

Privilege: Private

SetSearchLimit - This operation sets the match type (Prefix/Contain/Exact) that the pattern will be compared or else all the matched will be found.

Arguments: maxToReturn:EcTUInt

Return Type: EcTVoid

Privilege: Public

SetSearchType - This operation sets all the Adv Type desired, if not specified, all Adv Type will be returned.

Arguments: typeFilter:AdvertisingType

Return Type: EcTVoid

Privilege: Public

~IoAdSearchCommand - Default destructor.

Arguments: void

Return Type: Void

Privilege: Public

Associations:

The IoAdSearchCommand class has associations with the following classes:

None

4.3.12 IoAdService Class

Parent Class: IoAdAdvertisement

Public: Yes

Distributed Object: No

Purpose and Description:

Public View: This entity class supports operations to allow the definition, storage and retrieval of a advertisement of a service. Typically, this service processes ECS data products. It can also be other kinds of automated services or non automated services (e.g. help desk). Member data and operations should not be accessed directly, but through IoAdService. IoAdService performs handle services. Because fetching is deferred, any access can generate a database error. Therefore, one should check the handles status when appropriate. Unless specified, the string values can be empty. Protected View: We inherit our Database() and Connection() from IoAdAdvertisement. Private View: None.

Attributes:

myProducts - What products can we apply to.

Data Type: IoAdProductReferenceList

Privilege: Private

Default Value:

myProvider - Who is providing this product.

Data Type: IoAdProvider

Privilege: Private

Default Value:

myServiceClass - Stores the name of the service class for the advertised service.

Data Type: RWCString

Privilege: Private

Default Value:

myServiceTypeId - Stores the signature of the service.

Data Type: EcTInt

Privilege: Private

Default Value:

myServiceName - Stores the name of the advertised service.

Data Type: RWCString

Privilege: Private

Default Value:

Operations:

AddProduct - This operation defines a new product applies to this service. This in turn also defines this service is applicable to the given product.

Arguments: productToAdd:IoAdProduct

Return Type: EcUtStatus

Privilege: Public

DeleteProduct - This operation removes a defined product from this service. This in turn also removes this service from the product.

Arguments: productToDelete:IoAdProduct

Return Type: EcUtStatus

Privilege: Public

FetchByValues - This routine selects an object from the database based on its state. For contact, the unique applicable state is ServiceClassID, ServiceName, and ServiceID

Arguments: void

Return Type: virtual EcUtStatus

Privilege: Public

FetchPhaseII - This operation gets all the data from FetchPrep.

Arguments: dataFromSelector:RWDBReader&

Return Type: virtual EcUtStatus

Privilege: Public

FetchPrep - This operation prepares the selector to acquire all of classes data.

Arguments: dataToAcquire:RWDBSelector&,

currentWhereClause:RWDBCriterion&, foreignKey:const RWDBExpr&

GetProvider - This operation gets the provider who is providing the product.

Arguments: void

Return Type: IoAdProvider

Privilege: Public

GetServiceClass - This operation returns the current service class name of the advertised service.

Arguments: void

Return Type: const char*

Privilege: Public

GetServiceName - This operation returns the name of the advertised service.

Arguments: void

Return Type: const char*

Privilege: Public

GetServiceTypeId - This operation defines the signature of the service.

Arguments: void

Return Type: EcTInt

Privilege: Public

InsertBasicData - This operation stores all attributes in database.

Arguments: void

Return Type: EcUtStatus

Privilege: Protected

InsertDerived - This operation inserts our data if it is logically valid. This routine is part of the persistent framework.

Arguments: void

Return Type: virtual EcUtStatus

Privilege: Protected

IoAdServiceRep - This constructor sets our advertising type. It initializes our list of products to know we are the source object.

Arguments: void

Return Type: Void

Privilege: Public

IoAdServiceRep - Copy constructor.

Arguments: copyFrom:IoAdServiceRep&

Return Type: Void

Privilege: Public

IsValid - This operation checks the contents of the object for consistency and data rules.

o Provider is valid o Ad Base is valid o Service class and service name are not NULL

Arguments: void

Return Type: virtual EcTBoolean

Privilege: Public

NumberOfProducts - This operation defines the number of products that service applies to.

Arguments: void

Return Type: EcTUInt

Privilege: Public

PrintMembers - This operation writes contents to stream.

Arguments: out:ostream&

Return Type: virtual void

Privilege: Public

ProductIter - This operation provides a RWTPSlist-like iterator for all applicable products.

Arguments: void

Return Type: IoAdProductReferenceListIter

Privilege: Public

ServiceTableName - This is a private function to encapsulate database table.

Arguments: void

Return Type: const char*

Privilege: Private

SetProvider - This operation sets the myProvider attribute.

Arguments: value:IoAdProvider

Return Type: void

Privilege: Public

SetServiceClass - This operation is used to set a new service class name for the advertised service.

Arguments: value:const char*

Return Type: void

Privilege: Public

SetServiceName - This operation is used to set the name of the advertised service.

Arguments: value:const char*

Return Type: void

Privilege: Public

SetServiceTypeId - This operation sets the signature of the service.

Arguments: value:EcTInt

Return Type: void

Privilege: Public

SetValues

Arguments: title: const char*, description: const char*, guideURL const char*, group: const char*, ur:const char*, copyRight:const char*, contact: IoAdContact, provider: IoAdProvider, serviceClass: const char*, serviceName:const char*, serviceType Id: EcTInt

UpdateDerived - This operation updates this object in the database for its current object ID. This routine is part of the persistent framework.

Arguments: void

Return Type: virtual EcUtStatus

Privilege: Protected

operator= - This operator enables the assignment of two objects.

Arguments: assignFrom:IoAdServiceRep&

Return Type: const IoAdServiceRep&

Privilege: Public

~IoAdServiceRep - Default destructor.

Arguments: void

Return Type: Void

Privilege: Public

Associations:

The IoAdService class has associations with the following classes:

None

4.3.13 IoAdServiceSearchCommand Class

Parent Class: IoAdSearchCommand

Public: Yes

Distributed Object: No

Purpose and Description:

Public View: This class provides interfaces for applications to search the set of service advertisements by specifying options and criterion. The persistent data will be stored into a results list for additional searches or access. Users should set up options of how to search (filtering, patterns, how many results to return) and then call the search interfaces.
Protected View: None. Private View: None.

Attributes:

myResults - This is the result that contains the matched Service Advertisement.

Data Type: IoAdServiceList

Privilege: Private

Default Value:

Operations:

ClearResults - This operation clears the matched object linked list.

Arguments: void

Return Type: EcTVoid

Privilege: Public

GetResults - This operation returns the matched Service Advertisement to user.

Arguments: void

Return Type: const IoAdServiceList&

Privilege: Public

IoAdServiceSearchCommand - Default constructor.

Arguments: void

Return Type: Void

Privilege: Public

Reset - This operation resets all the search criterion to default. Default Adv type searches Service Advertisement type.

Arguments: void

Return Type: EcTVoid

Privilege: Public

SearchByServiceName - This operation searches for Service Advertisement that contains a substring of matchTo in its ServiceName and appends found Service advertisements to the current results set.

Arguments: matchTo:const char*

Return Type: EcUtStatus

Privilege: Public

SetSearchType - This operation sets the type of search .

Arguments: typeFilter:AdvertisingType

Return Type: EcTVoid

Privilege: Protected

~IoAdServiceSearchCommand - Default destructor.

Arguments: void

Return Type: Void

Privilege: Public

Associations:

The IoAdServiceSearchCommand class has associations with the following classes:

None

4.3.14 IoAdSignatureServiceAdv Class

Parent Class: IoAdService

Public: Yes

Distributed Object: No

Purpose and Description:

Public View: This entity class supports operations to allow the definition, storage and retrieval of an advertisement of a signature service. Member data and operations should not be accessed directly, but through IoAdSignatureServiceAdv. IoAdSignatureServiceAdv performs handle services. Because fetching is deferred, any access can generate a database error. Therefore, one should check the handles status when appropriate. Unless specified, the string values can be empty. Protected View: We inherit our Database() and Connection() from IoAdAdvertisement. Private View: None

Attributes:

mySignatureServiceSchema - Which Signature Service Schema is associated with.

Data Type: IoAdSignatureServiceSchema

Privilege: Private

Default Value:

Operations:

DeleteBasicData - This operation deletes all attributes in database.

Arguments: void

Return Type: EcUtStatus

Privilege: Private

DeleteDerived - This operation deletes this object in the database for its current object ID.

This routine is part of the persistent framework.

Arguments: void

Return Type: virtual EcUtStatus

Privilege: Protected

FetchByValues - This routine selects an object from the database based on its state. For contact, the unique application state is ServiceClassID, ServiceName, and ServiceID.

Arguments: void

Return Type: virtual EcUtStatus

Privilege: Public

FetchPhaseII - This operation gets all the data from FetchPrep.

Arguments: dataFromSelector : RWDBReader&

Return Type: virtual EcUtStatus

Privilege: Public

FetchPrep

Arguments: dataToAcquire: RWDBSelector& , currentWhereClause : RWDBCriterion, foreignKey : const RWDBExpr&

GetServiceClass - This operation gets the service class type supplied by the subsystem.

Arguments: void

Return Type: const char*

Privilege: Public

GetServiceName - This operator gets the service operator for particular service class type such as Ingest/Ceres02 or Delete/Ceres02.

Arguments: void

Return Type: const char*

Privilege: Public

GetSignatureServiceSchema - This operation gets the signature service schema .

Arguments: void

Return Type: const IoAdSignatureServiceSchema*

Privilege: Public

Insert Derived - This operation inserts our data if it is logically valid. This routine is part of the persistent framework.

Arguments: void

Return Type: virtual EcUtStatus

Privilege: Protected

InsertBasicData - This operation stores all attributes in database.

Arguments: void

Return Type: EcUtStatus

Privilege: Private

IoAdSignatureServiceAdvRep - This constructor is responsible for initializing the IoAdSignatureServiceAdvRep class.

Arguments: void

Return Type: Void

Privilege: Public

IoAdSignatureServiceAdvRep - Copy constructor.

Arguments: copyFrom : ioAdSignatureServiceAdvRep&

Return Type: Void

Privilege: Public

IsValid - This operation checks the contents of the object for consistency and data rules:
o Provider is valid o Ad Base is valid o Service class and service name are not NULL
Arguments: void
Return Type: virtual EcTBoolean
Privilege: Public

PrintMembers - This operation writes contents to stream.
Arguments: out: ostream&
Return Type: virtual void
Privilege: Public

SetServiceClass - This operation sets the service class type supplied by the subsystem.
Arguments: value: const char*
Return Type: void
Privilege: Public

SetServiceName - This operation sets the service operator for particular service class type such as Ingest/Ceres02 or Delete/Ceres02.
Arguments: value: const char*
Return Type: void
Privilege: Public

SetSignatureServiceSchema - This operation sets the signature service schema.
Arguments: value:const IoAdSignatureServiceSchema
Return Type: void
Privilege: Public

SetValues
Arguments: title: const char*, description: const char*, guideURL: const char*, group : const char*, ur: const char*, copyRight: const char*, contact : IoAdContact, provider: IoAdProvider, serviceClass: const char*, serviceName: const char*, serviceSchema: IoAdSignatureServiceSchema

SignatureServiceAdvTable - This operation stores the signature service table name stored in the database.
Arguments: void
Return Type: Void
Privilege: Public

UpdateBasicData - This operation stores all attributes in database.
Arguments: void
Return Type: EcUtStatus
Privilege: Private

UpdateDerived - This operation updates this object in the database for its current object ID. This routine is part of the persistent framework.

Arguments: void

Return Type: virtual EcUtStatus

Privilege: Protected

operator= - This operation enables the assignment of two objects.

Arguments: assignFrom : IoAdSignatureServiceAdvRep&

Return Type: const IoAdSignatureServiceAdvRep&

Privilege: Public

~IoAdSignatureServiceAdvRep - Default destructor.

Arguments: void

Return Type: Void

Privilege: Public

Associations:

The IoAdSignatureServiceAdv class has associations with the following classes:

None

4.3.15 IoAdSignatureServiceSearchCommand Class

Parent Class: IoAdSearchCommand

Public: Yes

Distributed Object: No

Purpose and Description:

Public View: This class provides interfaces for applications to search the set of Signature type service advertisements by specifying options and criterion. The persistent data will be stored into a results list for additional searches or access. Users should set up options of how to search (filtering, patterns, how many results to return) and then call the search interfaces. Protected View: None. Private View: None.

Attributes:

myResults - This is the found Signature type Service lists.

Data Type: IoAdSignatureServiceList

Privilege: Private

Default Value:

Operations:

ClearResults - This operation clears the matched object linked list.

Arguments: void

Return Type: EcTVoid

Privilege: Public

GetResults - This operation returns the matched Service Advertisement to user.

Arguments: void

Return Type: const IoAdSignatureServiceList &

Privilege: Public

IoAdSignatureServiceSearchCommand - Default constructor.

Arguments: void

Return Type: Void

Privilege: Public

Reset - This operation resets all the search options to default.
o Match type is "Prefix"
o No search limit on rows to match.

Arguments: void

Return Type: EcTVoid

Privilege: Public

SearchByServiceClass - This operation searches for Service Advertisement that contains a substring of matchTo in its ServiceClass and append found Service advertisements to the current results set.

Arguments: matchTo: const char*

Return Type: EcUtStatus

Privilege: Public

SearchByServiceName - This operation searches for Service Advertisement that contains a substring of matchTo in its ServiceName and append found Service advertisements to the current results set.

Arguments: matchTo: const char*

Return Type: EcUtStatus

Privilege: Public

-IoAdSignatureServiceSearchCommand - Default destructor.

Arguments: void

Return Type: Void

Privilege: Public

Associations:

The IoAdSignatureServiceSearchCommand class has associations with the following classes:

None

4.3.16 Advertising Service Subscription Model

The Release A Science Data Server subscription model presented at CDR is a generic subscription service that can be reused across ECS components. A few minor adjustments must be made to the model so that it can support the specification of the service provider upon submittal of the subscription. The service provider will define which ECS component should receive the action as specified in the subscription. The Release A SDSRV subscription model will be documented as a generic ECS component at the Release B Critical Design Review with the necessary modifications. A description of the subscription model can be found in the Release B Data Server Subsystem Design for the ECS Project.

4.4 CSCI Structure

The following table provides a summary of the components which make up this CSCI, to the extent they are currently known. Since this CSCI is on an incremental development, the table presents a current estimate of the CSCI components, but is possible to change as the CSCI evolves.

Table 4.4-1. ADSRV Components

Name	Description	Type DEV - Developmental OTS - Of-the-shelf
Advertising Application Server	Application server that processes the Advertising Client's requests (e.g., attribute search, create advertisement, and etc.).	DEV
Advertising DBMS Server	An off-the-shelf relational DBMS to store the advertisements.	OTS
Persistent Object Framework	Extensible framework for the database objects to inherit from.	DEV
HTML Framework	Framework which generates dynamic HTML pages.	DEV
HTML Interfaces	The actual interfaces that are generated using the HTML Framework	DEV
Core Library	The library used to perform functions used both in the Application Server and the CGI programs spawned by the Navigation Server	DEV
Client Library	The library used by ECS applications to access the Application Server	DEV
Installer	Client program that installs services on the desktop upon user request.	DEV
AdvNavigatingServer	Off the shelf http server to provides hyperlink access to advertisements.	OTS

The following subsection provides a functional descriptions of the ADSRV components.

4.4.1 Advertising Application Server CSC (AdvDBMSAppIServer)

The Advertising Application Server is a developed software component that processes the request from the interface objects described in Section 4.3. The interface objects communicate with the server using DCE protocols. The server also uses a relational DBMS server (Sybase) for searching and persistent storage of the advertisements. The Sybase server is currently shared with the Data Management Subsystem components.

4.4.2 Advertising DBMS Server CSC (AdvDBMSServer)

The Advertising DBMS Server CSC is a physical database server which stores and manages advertisement information. It provides the attribute search capability for users to locate interested advertisements.

4.4.3 Persistent Object Framework CSC

The Persistent Object Framework CSC is a group of classes that provides basic database operations to fetch and store information in a database. It uses the RogueWave DbTools.h++ library underneath the classes to access the database. It is used by the other libraries such as the Core library as base classes for other classes that need access to the advertising database.

4.4.4 HTML Framework CSC

The HTML Framework CSC is a group of classes that provide basic HTML template capabilities. HTML template files are filled in with the appropriate variables passed into the framework.

4.4.5 HTML Interfaces CSC

The HTML Interfaces CSC is the actual client interface to the Web browser built using the HTML Framework. This is the CSC that uses the HTML Framework to build the actual HTML files that are viewed by the users using a Web browser.

4.4.6 Core Library CSC

The Core Library CSC is the library that actually accesses the database using the Persistent Object Framework CSC. The Core Library CSC is used by the CGI programs initiated by the Web server as well as by the back end of the Advertising Application Server CSC.

4.4.7 Client Library CSC

The Client Library CSC is the library supplied to ECS and non-ECS applications in order to provide secure communications to the Advertising Application Server. This library uses DCE RPCs underneath object classes to communicate from some application to the Advertising Application Server. It provides basically the same functionality as the Core Library, but uses DCE as the communications layer.

4.4.8 Installer CSC

The Installer CSC is a program that resides on the client workstation to install services onto the ECS desktop. Users who only want to browse advertisements, but do not want to install or invoke

them, will not require the Installer program. The Installer will need to be resident on the client before any services can be installed or invoked from the Advertising Service.

4.4.9 Advertising Navigating Server CSC

The main purpose of Advertising Navigating Server is to allow Internet users to access ECS Advertising Service and be able to navigate through the advertisements. This CSC is a commercial-off-the-shelf (COTS) product which provides HTML/HTTP access to the Advertising DBMS Application Server. The leading candidate is the Netscape Commerce Server which is being used in Release A for both the Advertising Service and the Document Data Server. The actual procurement for Release B will be delayed however to allow for new alternatives to be presented, given the fluidity of the current Web technology market.

4.5 CSCI Management and Operation

The Advertising Application Server will be a mode managed application residing on the Data Management Subsystem hardware. When advertisements are submitted, an “moderator” will review the advertisement. If the advertisement meets quality standards and is for a service deemed useful for ECS users, the operator will approve the advertisement. The approval of the advertisement makes it visible to users searching the Advertising Service.

Other operator maintenance will include the following functions:

- Database maintenance on the Advertising Service database. This would include functions such as backup and recovery, maintenance of replication server environment.
- Maintenance on the Navigation server including configuration and performance tuning.

This page intentionally left blank.

Abbreviations and Acronyms

ACMHW	Access Control and Management HWCI
ADC	Affiliated Data Center
ADS	Archive data sets
ADSHW	Advertising Service HWCI
ADSRV	Advertising Service CSCI
AITHW	Algorithm Integration & Test HWCI
AITTL	Algorithm Integration and Test Tools (CSCI)
AM	Ante meridian
ANSI	American National Standards Institute
APC	Access/Process Coordinators
API	Application Programming Interface
APID	Application Process Identifier
AQAHW	Algorithm QA HWCI
ASAP	As soon as possible
ASCII	American Standard Code for Information Interchange
ASF	Alaska SAR Facility (DAAC)
ATM	Asynchronous Transfer Mode
CD ROM	Compact disk read only memory
CDRL	Contract Data Requirements List
CERES	Clouds and Earth's Radiant Energy System
CI	Configuration Item
CIESIN	Consortium for International Earth Science Information Network
CLS	Client Subsystem
COTS	Commercial off-the-shelf
CPU	Central processing unit
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CCSDS	Consultative Committee for Space Data Systems
CM	Configuration Management
CSDT	Computer Science Data Types
CSMS	Communications and Systems Management Segment
CSS	Communication Subsystem (CSMS)

DAA	DAN Acknowledge
DAAC	Distributed Active Archive Center
DADS	Data Archive and Distribution System
DAN	Data Availability Notice
DAO	Data Assimilation Office
DAR	Data Acquisition Request
DAS	Data Availability Schedule
DBA	Database administrator
DBMS	Database Management System
DDA	Data Delivery Acknowledgement
DDICT	Data Dictionary CSCI
DDIST	Data Distribution CSCI
DDN	Data Delivery Notice
DDSRV	Document Data Server CSCI
DESKT	Desktop CI
DEV	Developed code
DID	Data Item Description
DIM	Distributed Information Manager
DIMGR	Distributed Information Management CSCI
DIPHW	Distribution & Ingest Peripheral Management HWCI
DMGHW	Data Management HWCI
DMS	Data Management System
DMS	Data Management Subsystem
DP	Data Processing
DPR	December Progress Review
DPREP	Science Data Pre-Processing CSCI
DPS	Data Processing Subsystem
DR	Data Repository
DRPHW	Data Repository HWCI
DS	Data Server
DSM	Distribution Storage Management
DSS	Data Server Subsystem
DT	Data Type
ECS	EOSDIS Core System
EDC	EROS Data Center (DAAC)
EDOS	EOS Data and Operations System

EOC	Earth Observation Center (Japan)
EOS	Earth Observing System
EOSDIS	Earth Observing System Data and Information System
EP	Evaluation Package
EP	Early Prototype
ESDIS	Earth Science Data and Information System
ESDT	Earth Science Data Types
F&PRS	Functional and Performance Requirements Specification
FC	Fiber Channel
FDDI	Fiber distributed data interface
FDF	Flight Dynamics Facility
FOS	Flight Operations Segment
FSMS	File and Storage Management System
Ftp	File transfer protocol
GB	Gigabyte
GDAO	GSFC Data Assimilation Office
GFLOPS	Giga (billions) Floating Point Operations per Second
GOES	Geostationary Operational Environmental Satellite
GRIB	Gridded Binary
GSFC	Goddard Space Flight Center
GTWAY	Version 0 Interoperability Gateway CSCI
GUI	Graphic user interface
HDF	Hierarchical Data Format
HiPPI	High Performance Parallel Interface
HMI	Human machine interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transport Protocol
HWCI	Hardware Configuration Item
I&T	Integration and Test
I/O	Input/Output
ICD	Interface Control Document
ICLHW	Ingest Client HWCI
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronics Engineers
IERS	International Earth Rotation Service
IMS	Information Management Subsystem

IP	International Partner
IR-1	Interim Release 1
IRD	Interface Requirements Document
IS	Ingest Subsystem
ISS	Internetworking Subsystem (CSMS)
JPL	Jet Propulsion Laboratories
LaRC	Langley Research Center
LIM	Local Information Manager
LIMGR	Local Information Management CSCI
LIS	Lightning Imaging Sensor
L0	Level 0
MB	Megabyte
Mbps	Megabits per second
MBps	Megabytes per second
MD	Maryland
MFLOP	Millions of Floating Point Operations per Second
MOC	Mission Operations Center
MODIS	Moderate-Resolution Imaging Spectrometer
MPP	Massively Parallel Processor
MRF	Medium Range Forecast
MSFC	Marshall Space Flight Center
MSS	Management Subsystem (CSMS)
MTBF	Mean time between failures
MTTR	Mean time to restore
NESDIS	National Environmental Satellite Data and Information Service
NMC	National Meteorological Center
NOAA	National Oceanic and Atmospheric Administration
NSIDC	National Snow and Ice Data Center (DAAC)
O/A	Orbit/Altitude
ODC	Other Data Center
ODL	Object Description Language
ORNL	Oak Ridge National Laboratory (DAAC)
OSM	Open Storage Manager
OTS	Off-the-shelf
PAM	Permanent Archive Manager
PCI	Peripherwral Component Interface

PDPS	Planning and Data Processing System
PDR	Preliminary Design Review
PDS	Production Data Set
PDS	Production Data Specialist
PGE	Product Generation Executive
PGS	Product Generation System
PLNHW	Planning HWCI
POSIX	Portable Operating System for UNIX
PRONG	Processing CSCI
Q	Quarter
Q/A	Quality Assurance
QA	Quality Assurance
QAC	Quality and Accounting Capsule
RAID	Redundant Array of Inexpensive Disks
RAM	Random Access Memory
REL	Release
RID	Review Item Discrepancy
RMA	Reliability, Maintainability, Availability
RTF	Rich Text Format
S/C	Spacecraft
SAA	Satellite Active Archives (NOAA)
SCF	Science Computing Facility
SCSI II	Small Computer System Interface
SDF	Software Development File
SDP	Science Data Processing
SDPF	Sensor Data Processing Facility (GSFC)
SDPS	Science Data Processing Segment
SDPS/W	Science Data Processing Software
SDPTK	SDP Toolkit CSCI
SDSRV	Science Data Server CSCI
SFDU	Standard Format Data Unit
SMC	System Management Center
SMP	Symmetric Multi-Processor
SPRHW	Science Processing HWCI
STMGT	Storage Management CSCI
TBD	To be determined

TBR	To be resolved
TDRSS	Tracking and Data Relay Satellite System
TONS	TDRSS Onboard Navigation System
TRMM	Tropical Rainfall Measuring Mission
TSDIS	TRMM Science Data and Information System
UR	Universal Reference
USNO	United States Naval Observatory
V0	Version 0
VC	Virtual Channel
VCDU-ID	Virtual Channel ID
WAIS	Wide Area Information Servers
WAN	Wide Area Network
WKBCH	Workbench CI
WKSHC	Working Storage HWCI
W/S	Workstation
WORM	Write Once Read Many
WS	Working Storage
WWW	World Wide Web