Safety and Fitness Electronic Records (SAFER) System

Logical Architecture Document

Working Draft

January 31,1997

Prepared for

Federal Highway Administration

Prepared by:

The Johns Hopkins University
Applied Physics Laboratory

Table of Contents

Section 1.0 Introduction

Section 2.0 Logical Architecture

Section 3.0 Functional Decomposition

List of Figures

Figure **Section 3:Page**

1.0 Introduction

This Logical Architecture Document includes the products developed during the functional analysis of the Safety and Fitness Electronic Records (SAFER) System. This document, along with the companion Operational Concept and Physical Architecture Documents, describe the SAFER Architecture proposed by the Johns Hopkins Applied Physics Laboratory (APL).

1.1 System Overview

SAFER fits within the larger context of existing and planned information systems related to Commercial Vehicle Operations (CVO) and Intelligent Transportation Systems (ITS). SAFER supports the overall CVO Safety Assurance mission to improve the safety of commercial vehicle operations throughout North America.

The SAFER System will provide fixed and mobile commercial vehicle inspection sites, other systems, and users with electronic access to the data residing within existing and planned Federal and State motor carrier safety information systems.

The SAFER System will provide information within seconds on a motor carrier's safety risk rating, roadside inspection history, and accident record, thereby ensuring that enforcement officers working at the roadside will have the most recent information available when deciding which vehicles and drivers should be inspected. The system will also provide insurers and shippers with electronic access to the safety information they need to support their business operations. The system is being designed to support other Intelligent Transportation System (ITS) applications, such as electronic verification, checking safety credentials at the time of vehicle registration, and other commercial vehicle administrative processes.

SAFER development will adhere to the ITS CVO Information Systems Architecture being developed by JHU/APL for FHWA.

The SAFER System will provide users with either a summary of a carrier's safety record ("snapshot" or a more detailed report ("profile"). The system will be both re-active (i.e., responding to specific requests) and pro-active (i.e., allowing users to request that they be informed when the snapshot changes substantially). Users will be able to request information for specific carriers, or for carriers meeting certain selection criteria. Users will specify the desired response time and delivery mechanism.

Initially, information will be available for interstate carriers only. The system will be expanded to provide access to State information systems and intrastate carriers. Expanded system functions will also include allowing users to request industry-wide safety statistics, and provisions for providing larger user sites with change updates rather than complete updates.

To carry out its functions, SAFER will need access to motor carrier identification information for both interstate and intrastate carriers. Over time, SAFER will become the authoritative source for certain forms of motor carrier identification information.

To support the long term operations and maintenance of the SAFER System, users will be charged for SAFER services. The rate structure will be developed in concert with the system design.

1.2 Approach

The Logical Architecture identifies each of the functions required to implement the SAFER System along with the functional relationships and interfaces to the outside world. Neither functional allocation to physical subsystem nor implementation of the subsystems themselves are specified in the logical architecture. These items will be addressed in the Physical Architecture Document.

The format and content of the Logical Architecture Document reflect the structured analysis approach originally developed by Tom DeMarco. Data and control flows are integrated within a single set of diagrams rather than partitioned onto separate data flow and control flow diagrams.

The structured analysis model was generated using Popkin's System Architect Computer Aided Software Engineering (CASE) tool.

1.3 Document Organization

Following the introductory section, Section 2 presents the logical architecture model for the SAFER System by introducing the context diagram which describes the data and control interfaces between SAFER and external users. The logical architecture is further described in Section 3 by presenting each of the underlying flow diagrams at increasing levels of decomposition. Following each flow diagram is a narrative describing the functional relationships depicted graphically on the diagram. A definition of process inputs and outputs is also provided. Section 4 presents the Data Dictionary which defines the data and control elements indicated in the flow diagrams and Section 5 provides a requirements traceability matrix which graphically shows that each of the requirements specified in the SAFER Requirements Document are functionally represented in the logical architecture.

2.0 SAFER Logical Architecture

This section provides a functional decomposition of the SAFER System. System functions are described in concert with their data and control interfaces.

2.1 SAFER Context

The context diagram precisely defines the architectural boundary layer between the SAFER System and the external world. The single process bubble encapsulates the complete set of functional elements needed to support all SAFER services. Elements on the diagram which interface with but are not completely encompassed by the SAFER System are defined as "system externals".

List of Externals

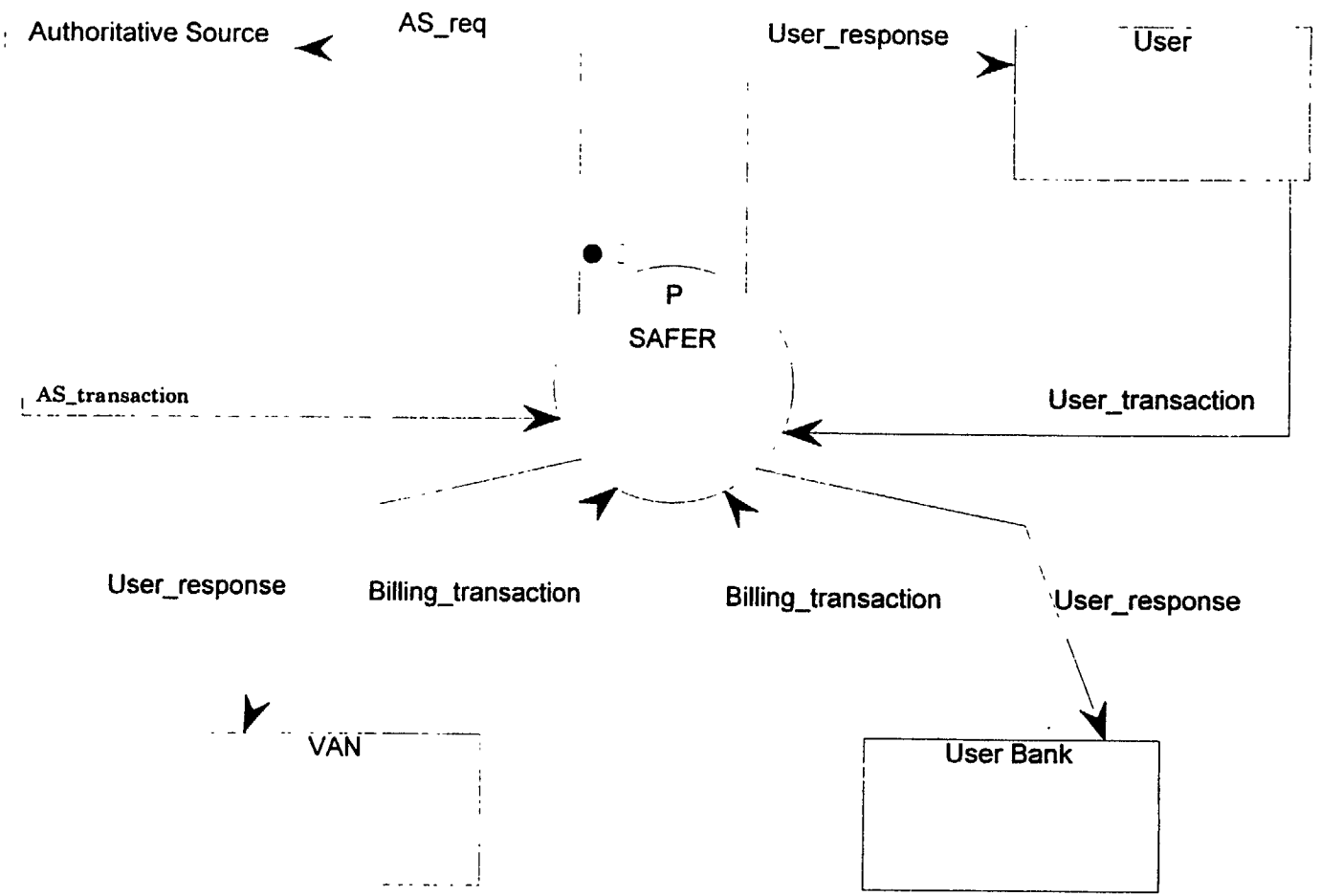
The following list includes all externals represented on the context diagram. Accompanying each external name is a brief description describing the external and its relationship with the SAFER architecture.

Authoritative Source: The single and only source of a particular type of information flowing into the System, e.g., the Motor Carrier Management Information System (MCMIS) is the current authoritative source for interstate carrier census and safety information.

User: Any system, organization, group, or person that interfaces with the SAFER System for the purpose of sending, receiving or exchanging commercial vehicle safety information or other types of information needed to support such activity.

VAN: Value Added Network is a provider of network services that have been augmented with enhanced application-level capabilities thereby adding value to the basic network functions.

User Bank: Any financial institution recognized within the banking community as an organization certified to perform financial transactions on behalf of the User.



SAFER Context, READ ONLY
System Architect
Fri Apr 28, 1995 07:38
Comment -

0.0 **SAFER**

SAFER provides users with electronic access to the carrier safety data residing within existing and planned federal and state motor carrier safety information systems. SAFER provides either a summary of a carrier's safety record (snapshot) or a more detail record (profile). The System is both re-active (i.e., responding to specific requests) and pro-active (i.e., allowing users to request that they be informed when the snapshot changes substantially. Users are able to request information for specific carriers, or for carriers meeting certain selection criteria. Users may specify the desired response time and delivery mechanism.

Input: T **Output: F**

AS-transaction

A transaction from an Authoritative Source may be an acknowledgment, a profile generated in response to a specific user request, SAFER snapshot database refresh, amended data, or an event trigger.

Billing-transaction

Information supplied by the VAN for billing purposes may include: the number of messages sent/received between users and the System, the number of log-in sessions and session length, the amount of data transmitted.

User-transaction

A user transaction includes requests for data (snapshot, profile), account establishment and maintenance, carrier data maintenance, payment for System services, subscriber list definition and modification, and acknowledgments.

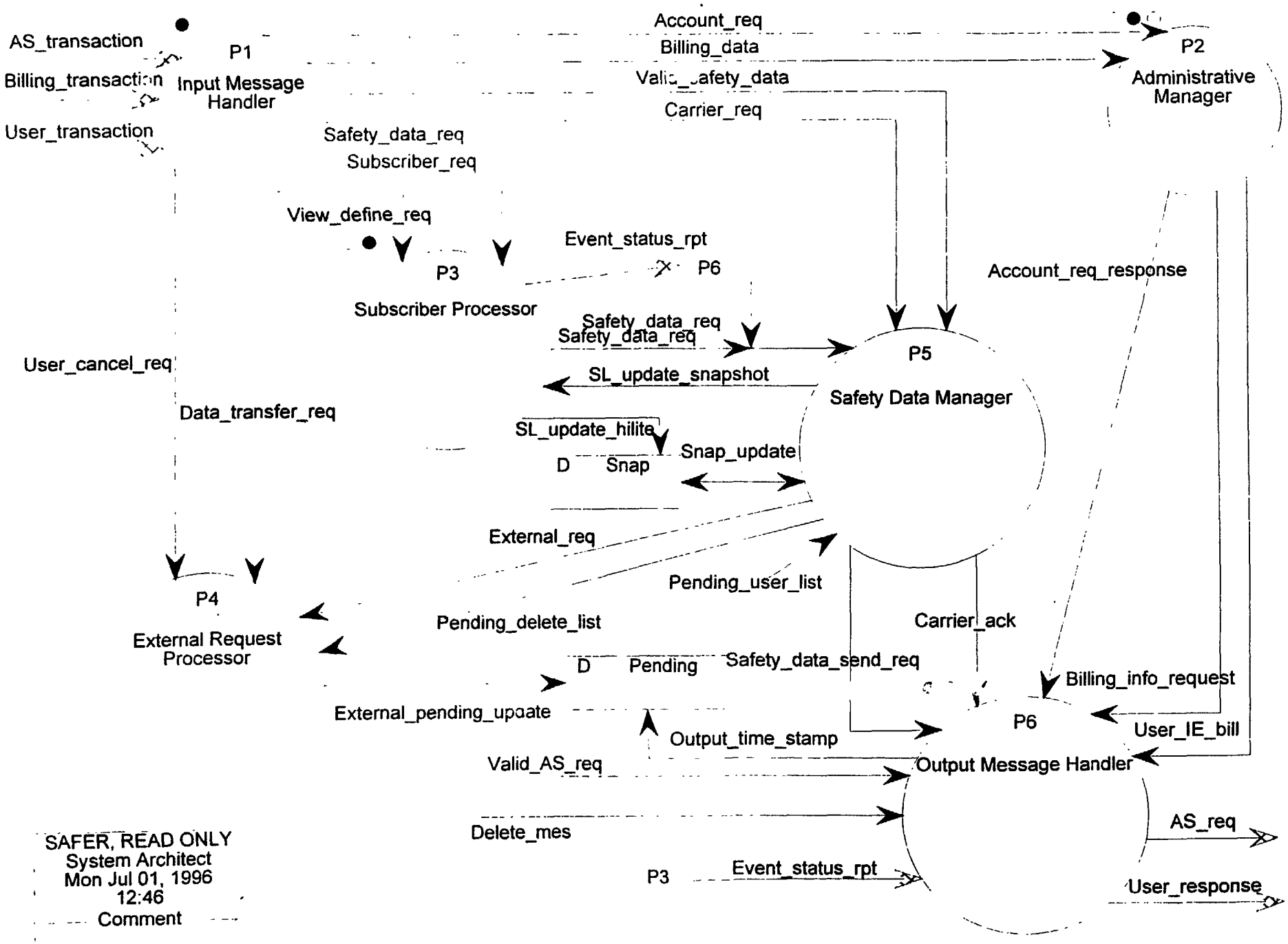
Input: F **Output: T**

AS-req

A request for data from an Authoritative Source. This may also route updated carrier information to the appropriate source and send interchange and functional acknowledgments to the Authoritative Source.

User-response

Response to user request for data (snapshot, profile), account establishment and maintenance, carrier data maintenance, payment for System services, subscriber definition and modification, or forwarded amended data. Also included are interchange and functional acknowledgments.


AS_transaction
Billing_transaction
User_transaction
P1
Input Message Handler
Account_req
Billing_data
Valid_safety_data
Carrier_req
P2
Administrative Manager
Safety_data_req
Subscriber_req
View_define_req
Event_status_rpt
P3
P6
Subscriber Processor
Account_req_response
Safety_data_req
Safety_data_req
P5
User_cancel_req
SL_update_snapshot
Safety Data Manager
Data_transfer_req
SL_update_hilite
D
Snap
Snap_update
External_req
Pending_user_list
P4
External Request Processor
Pending_delete_list
Carrier_ack
D
Pending
Safety_data_send_req
Billing_info_request
External_pending_update
P6
User_IE_bill
Output_time_stamp
Valid_AS_req
Output Message Handler
Delete_mes
AS_req
P3
Event_status_rpt
User_response
SAFER, READ ONLY
System Architect
Mon Jul 01, 1996
12:46
Comment

1.0 Input Message Handler

The Input Message Handler accepts incoming transactions, translates the transactions into internal format, determines if the sender of the transaction has the proper authorization level, performs application-level edit checks, and sends a valid transaction to the appropriate processor.

Input: T **Output: F**

AS-transaction

A transaction from an Authoritative Source may be an acknowledgment, a profile generated in response to a specific user request, SAFER snapshot database refresh, amended data, or an event trigger.

Billing-transaction

Information supplied by the VAN for billing purposes may include: the number of messages sent/received between users and the System, the number of log-in sessions and session length, the amount of data transmitted.

User-transaction

A user transaction includes requests for data (snapshot, profile), account establishment and maintenance, carrier data maintenance, payment for System services, subscriber list definition and modification, and acknowledgments.

Input: F **Output: T**

Account-req

A request to create, update or delete a user account or an organization account

Billing-data

Billing information received from the VAN for an organization and its users for a given billing period OR a notification of payment by an organization for a given billing period OR the result of a bank account check

Carrier-req

Request to add, update or delete carrier information.

Data-transfer-req

Request for data to be passed through SAFER to an authoritative source.

Safety-data-req

A request for snapshot or profile data.

Subscriber-req

Request from a user to create/update a subscriber list. Passed from P1 to P3. Contains the user's method of data delivery, area of interest, and events to be monitored.

User-cancel-req

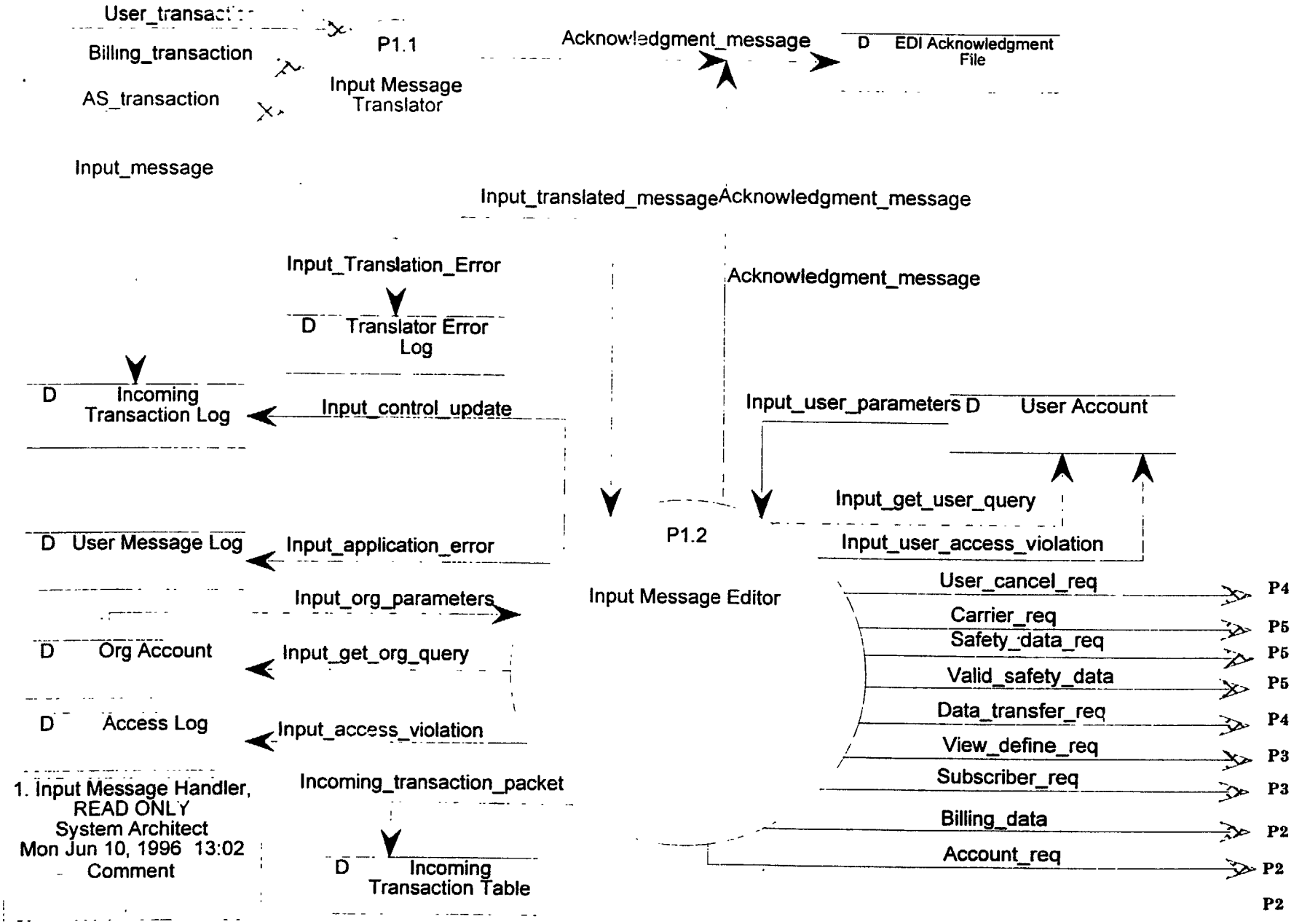
Data needed to cancel a low priority request.

Valid-safety-data

Incoming safety data(damended, snapshot or profile) that has been translated, checked for access authorization and has undergone application-level edit checks.

View-define-req

Request to update the system view table. Passed from P1 to P3. This request is available to only certain system-level SAFER users.


User_transact
Billing_transaction
AS_transaction
P1.1
Input Message Translator
Acknowledgment_message
D EDI Acknowledgment File
Input_message
Input_translated_message
Acknowledgment_message
Input_Translation_Error
D Translator Error Log
Acknowledgment_message
D Incoming Transaction Log
Input_control_update
Input_user_parameters
D User Account
Input_get_user_query
P1.2
Input_user_access_violation
D User Message Log
Input_application_error
Input_org_parameters
Input Message Editor
User_cancel_req
P4
Carrier_req
P5
Safety_data_req
P5
Valid_safety_data
P5
Data_transfer_req
P4
View_define_req
P3
Subscriber_req
P3
Billing_data
P2
Account_req
P2
P2
P2
D Org Account
Input_get_org_query
D Access Log
Input_access_violation
1. Input Message Handler,
READ ONLY
System Architect
Mon Jun 10, 1996 13:02
Comment
Incoming_transaction_packet
D Incoming Transaction Table

1.1 Input Message Translator

The Input Message Translator receives a transmission (User-transaction, Billing-transaction, AS-transaction), determines the validity of the interchange, functional group, and transaction set control segments, performs compliance checking, and generates functional acknowledgments giving the results of the compliance checking.

This process will use a commercial translation product to analyze each transmission's compliance with the specified standard. At a minimum, the translator should convert incoming (and outgoing) messages according to the specified standard (EDI, EDIFACT, industry standard). It must also be able to check : minimum/maximum field size, presence of required fields, optional and conditional requirements met, proper sequence of data elements within data segment and segments within a transaction set, and correct data types. It should also be able to generate acknowledgments and log incoming and outgoing messages. It should verify that the number of functional groups specified in the Interchange Control Trailer were received, and the Interchange Control Number is the same in both the Interchange Control Header and Trailer. The control number must be the same in the Functional Group Header and Trailer, and the number of transactions received must match the number specified in the Functional Group Trailer. The Transaction set control number must be the same in both the Transaction Set Header and Trailer. The number of segments received for the transaction must match the number specified in the Transaction Set Trailer. Valid transactions are mapped to internal application structures.

A Functional Acknowledgment will be generated, as part of the validation process, for each interchange, if requested, and each functional group, and via Acknowledgment-message, sent to the EDI Acknowledgment Log for later processing. All valid incoming transactions are logged to the Incoming Transaction Log, along with a System generated date and time (Greenwich Mean Time), through Input-message. Valid translated messages, Input-translated-message, are sent to the Input Message Editor for any application-level checks. If errors are found with a transaction, it is assumed an error message will be written to a translator generated error file.

Input: T **Output: F**

AS-transaction

A transaction from an Authoritative Source may be an acknowledgment, a profile generated in response to a specific user request, SAFER snapshot database refresh, amended data, or an event trigger.

Billing-transaction

Information supplied by the VAN for billing purposes may include: the number of messages sent/received between users and the System, the number of log-in sessions and session length, the amount of data transmitted.

User-transaction

A user transaction includes requests for data (snapshot, profile), account establishment and maintenance, carrier data maintenance, payment for System services, subscriber list definition and modification, and acknowledgments.

Input: F Output: T

Acknowledgment-message

This may report the success of processing a received interchange header and trailer. It may also be "used to report the results of the syntactical analysis of the functional groups of transaction sets; they report the extent to which the syntax complies with the standards for the transaction sets and functional groups. They do not report on the semantic meaning of the transaction sets (for example, on the ability of the receiver to comply with the request of the sender)." [x12.20] This also includes incoming acknowledgments which may indicate errors found in SAFER transmitted transactions.

Input-Translation-Error

An input message identified as having interchange, functional group or transaction errors.

Input-message

Key information from the incoming transaction, along with the System generated date and time, and identifiers, are logged.

Input-translated-message

An input transaction successfully translated into internal format. Security and application-level edit checks may be required before the transaction is ready to be used by the application.

1.2 Input Message Editor

The Input Message Editor performs application-level edit checks on the translated message, Input-translated-message. Some of the following functions may be performed by a commercial translator.

This process helps to maintain integrity of SAFER data by limiting System access to authorized users. A data store with valid users, User Account, is maintained and contains the user IDS, the transactions sets he is authorized to use, his access privileges, the transaction standard he has agreed to use, and his account balance. These fields are maintained through transactions to establish and update user accounts. The set of transactions is established by SAFER based on the type of user. A test database and automatic test procedures will be provided by SAFER for new users. When a new user has demonstrated the ability to send and receive transactions correctly, the user will be considered validated, and is then authorized to send and receive his set of transactions.

The User Account data store is searched for the user ID supplied through Inputget-user-query, and supplies the user ID, privileges, and account balance through Input-user-parameters. If the user ID is not found in the data store, and the request is not to establish a new account, an access violation has occurred. This is logged in the Access Log, for tracking, through Input-access-violation. If the user ID is found, but the transaction is not authorized for the user, or the standard is not the one agreed to, the transaction is not authorized. An access violation count is incremented and the transaction is sent by Input-application-error to the User Message Log for later processing by the Output Message Handler. An updated access violation count is sent to the User Account by Input-user-access-violation. A check is made on the access violation count, and if equals or exceeds the System limit, the transaction is not authorized, and is sent to the User Message Log through Input-application-error. If the user's account is overdrawn, a message is written to the User Message Log through Input-application-error.

If no errors have occurred, application-level editing is performed. This may include semantic checks (reasonableness, null values, unusual or large monetary amounts, duplicate or missing data, invalid data combinations, out of range data), and any special editing required to map the input elements into internal format required by the application that could not be done by the Input Message Translator. This may include converting time fields in the incoming transactions to Greenwich Mean Time.

For transactions with errors, an error message, Input-application-error, is sent to the User Message Log for later processing by the Output Message Handler. Otherwise, message priority is determined based on transaction type and request options, internal control numbers are assigned to each transaction and the Incoming Transaction Log is updated with these numbers, through Input-control-update. Key information for each incoming transaction is stored in the Incoming Transaction Table via "Input-transaction-packet", for message reconciliation. Messages without errors (Data-transfer-request, AS-ack, Valid-data, Event-define-req, Subscriber-req, Safety-data-req, Carrier-req, Billing-data, and Account-data, User-cancel-req) are sent to the appropriate request processor. Acknowledgments received by SAFER are sent to the EDI Acknowledgment Log via "Acknowledgment-message."

Input: F **Output: F**

View-define-req
Request to update the system view table. Passed from P1 to P3. This request is available to only certain system-level SAFER users.

Input: T **Output: F**

Input-erg-parameters
Identifying and account information for an organization.

Input-translated-message
An input transaction successfully translated into internal format. Security and application-level edit checks may be required before the transaction is ready to be used by the application.

Input-user-parameters
Authorization levels for sender and other identifying information.

Input: F **Output: T**

Account-req
A request to create, update or delete a user account or an organization account

Acknowledgment-message
This may report the success of processing a received interchange header and trailer. It may also be "used to report the results of the syntactical analysis of the functional groups of transaction sets; they report the extent to which the syntax complies with the standards for the transaction sets and functional groups. They do not report on the semantic meaning of the transaction sets (for example, on the ability of the receiver to comply with the request of the sender)." **[x12.20]** This also includes incoming acknowledgments which may indicate errors found in SAFER transmitted transactions.

Billing-data
Billing information received from the VAN for an organization and its users for a given billing period OR a notification of payment by an organization for a given billing period OR the result of a bank account check

Carrier-req
Request to add, update or delete carrier information.

Data-transfer-req
Request for data to be passed through SAFER to an authoritative source.

Incoming-transaction-packet
Provides data needed for transaction reconciliation.

Input-access-violation
Contains the data needed to log an unauthorized System access attempt.

Input-application-error
An input message identified as having application-level errors, as being unauthorized for the user, or a message rejected due to insufficient funds.

Input-control-update
Contains the internal control numbers generated by SAFER for the transaction.

Input-get-erg-query
Contains the fields necessary to retrieve organizational information for input processing.

Input-get-user-query
Contains the fields necessary to retrieve user information for input processing using the sender id(s) supplied in the transmission.

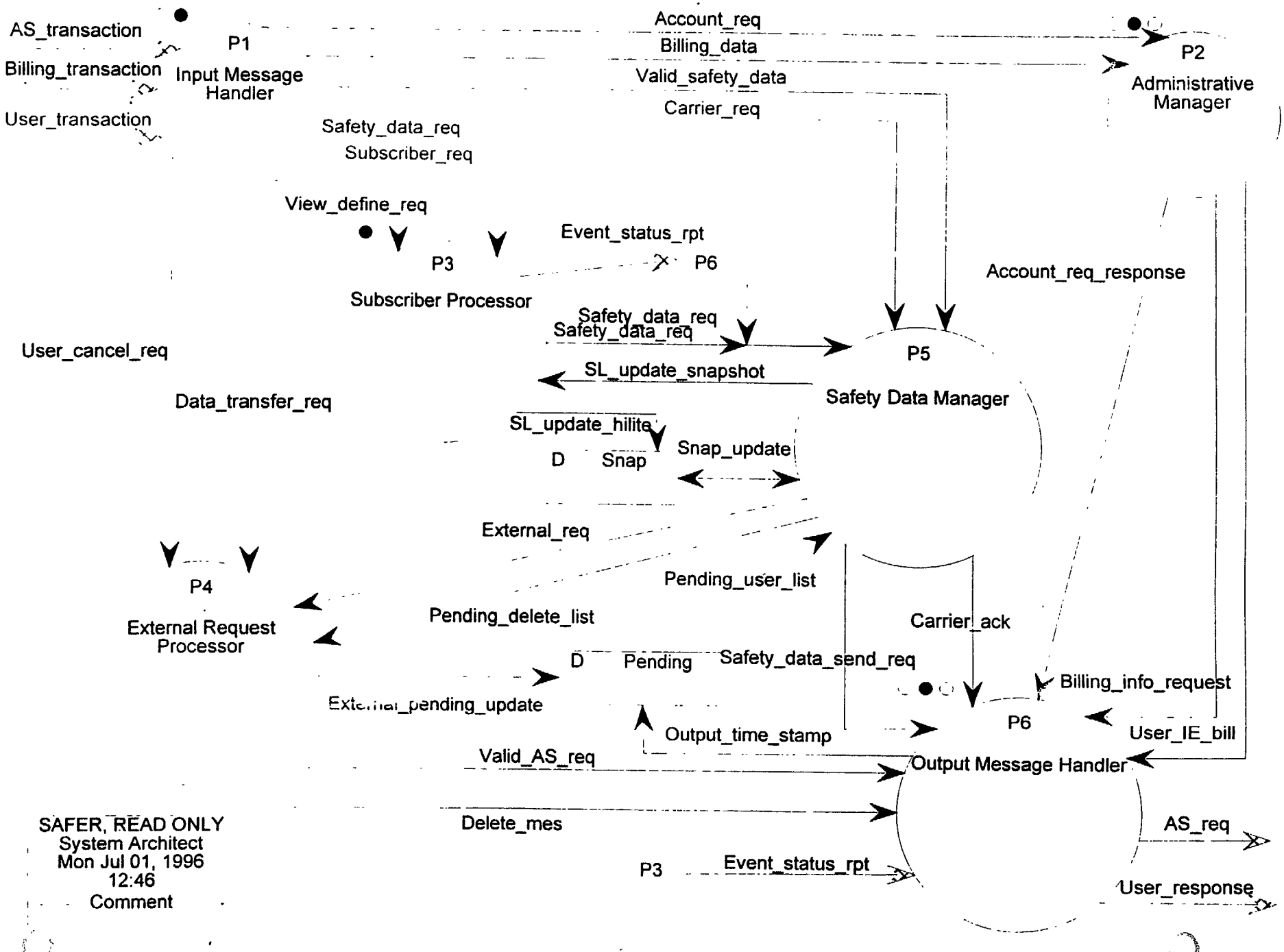
Input-user-access-violation
Contains the data necessary to update the user access violation count.

Safety-data-req
A request for snapshot or profile data.

Subscriber-req
Request from **a user** to create/update a subscriber list. Passed from P1 to P3. Contains the user's method of data delivery, area of interest, and events to be monitored.

User-cancel-req
Data needed to cancel a low priority request.

Valid-safety-data
Incoming safety data(damended, snapshot or profile) that has been translated, checked for access authorization and has undergone application-level edit checks.


AS_transaction
Billing_transaction
User_transaction
P1
Input Message Handler
Account_req
Billing_data
Valid_safety_data
Carrier_req
P2
Administrative Manager
Safety_data_req
Subscriber_req
View_define_req
Event_status_rpt
P3
P6
Subscriber Processor
Account_req_response
Safety_data_req
Safety_data_req
User_cancel_req
P5
SL_update_snapshot
Safety Data Manager
Data_transfer_req
SL_update_hilite
D
Snap
Snap_update
External_req
P4
Pending_user_list
External Request Processor
Pending_delete_list
Carrier_ack
D
Pending
Safety_data_send_req
Billing_info_request
External_pending_update
Output_time_stamp
P6
User_IE_bill
Valid_AS_req
Output Message Handler
Delete_mes
AS_req
P3
Event_status_rpt
User_response
SAFER, READ ONLY
System Architect
Mon Jul 01, 1996
12:46
Comment

2.0 Administrative Manager

The Administrative Manager (AM) is responsible for managing: Accounts - It must facilitate and control the creation, deletion and update of account information. Accounts are established and maintained at the organization and user levels. An organization can be associated with one or more users; a user can be associated with one or more organizations;
Billing operations - It must be able to request monthly billing information from the VAN, process that information based on the resource expenditure rates of each organizational user and bill each accordingly. These functions will be provided by a Commercial-Off-The-Shelf (COTS) financial/billing product.
System Resources - It must provide system resource management most likely in the form of a COTS data archive and retrieval product. The product must be capable of performing online backup and file management functions without disrupting data exchange services.

Input: T **Output: F**

Account-req

A request to create, update or delete a user account or an organization account

Billing-data

Billing information received from the VAN for an organization and its users for a given billing period OR a notification of payment by an organization for a given billing period OR the result of a bank account check

Input: F **Output: T**

Account-req-response

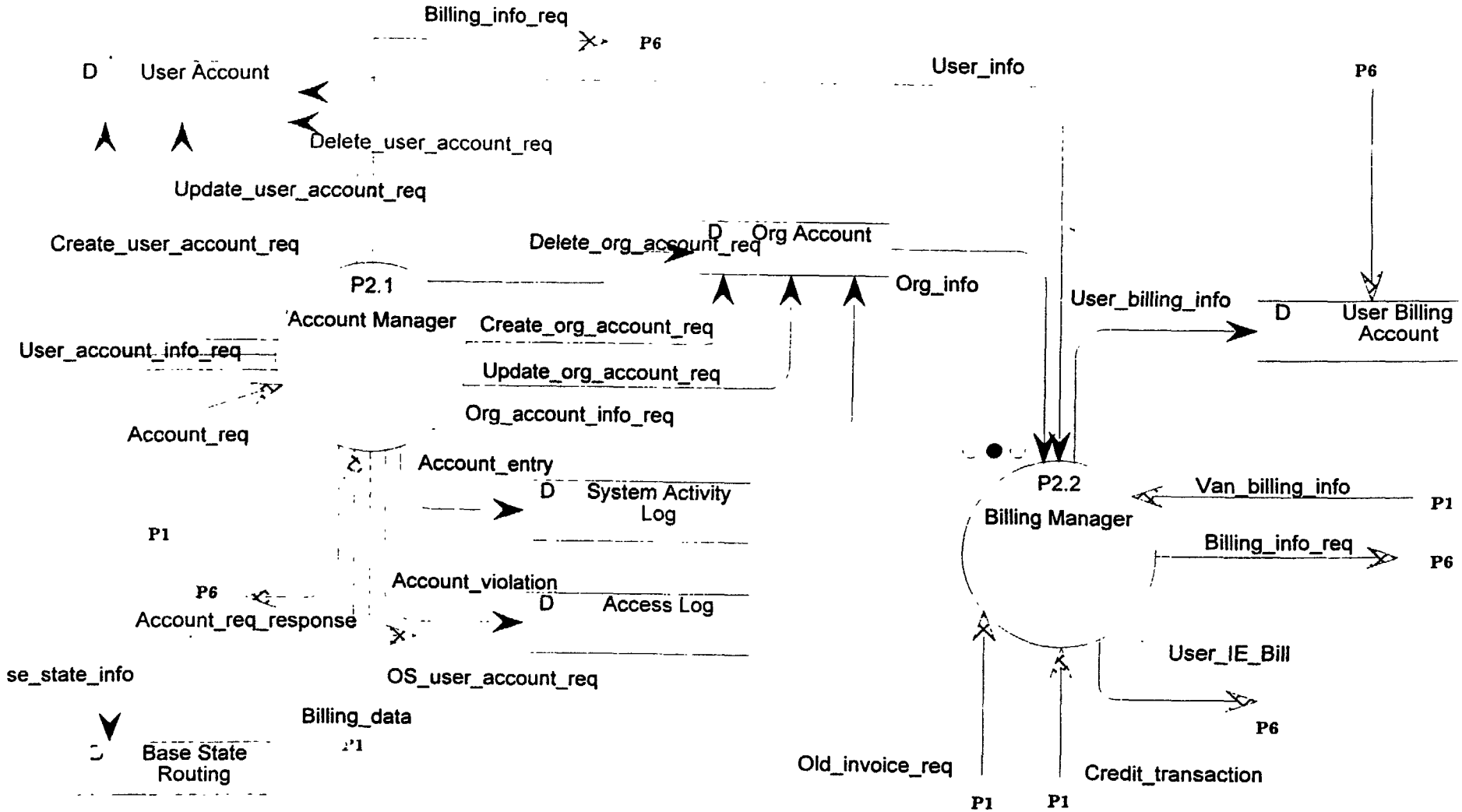
A response sent back to the user indicating the status of his request

Billing-info-request

A request made to the VAN at some specified billing interval for billing information for each organization with an account on the system.

User-IE-bill

The bill sent to each organization each billing period.


Billing_info_req
P6
D User Account
User_info
P6
Delete_user_account_req
Update_user_account_req
Create_user_account_req
Delete_org_account_req
D Org Account
P2.1
Account Manager
Org_info
User_billing_info
D User Billing Account
Create_org_account_req
User_account_info_req
Update_org_account_req
Org_account_info_req
Account_req
Account_entry
D System Activity Log
P2.2
Billing Manager
Van_billing_info
P1
P1
Billing_info_req
P6
P6
Account_violation
D Access Log
Account_req_response
User_IE_Bill
P6
se_state_info
OS_user_account_req
Billing_data
P1
D Base State Routing
Old_invoice_req
Credit_transaction
P1
P1
2. Administrative Manager, READ ONLY
System Architect
Tue Sep 03, 1996

2.1 Account Manager

The Account Manager is responsible for processing and executing account create, update, delete and summary requests. An account is created and maintained in the System on behalf of a user organization. Each organization must provide the System with the names and user information of one or more valid users. Base states must open SAFER org accounts in order to provide data to SAFER.

Create Org Account Request

Accept a Create Org Account request. Determine if an account had already been created for the requesting org. If yes, send an error message to the requester indicating that the account had already been created; Else, check to ensure that the requester's user-id is in the user list of the Create Account request and has the Org-account-create_priv or the Super-user privilege. If not, send an error message to the requester indicating that only a user with the Org-account-create-priv or Super-user privileges can create an account; Else, issue an Org-bank-account-check-req (structure in Billing-info-req) to determine if the organization's specified bank has sufficient funds to cover the dollar value of the SAFER account to be created as specified in Account-$-value. If the response from the bank, Org-bank-check-response (structure in Billing-data) validates that sufficient funds are available, update the Base State Routing data store if the requesting organization is a Base State, add the Org Account with the Account-id and the organization information, add the users, specified in the user list, to the User Account with the Account-id as a key field and send a successful account create message to the requester; Else, issue a Failed-bank-check-msg to the user. Log any action taken to the System Activity Log.

Delete Org Account Request

Accept a Delete Org Account request. Determine if the account, identified by Account-id, exists in Org Account. If not, send an error message to the requester, indicating that the requested account does not exist; Else, check to ensure that the user-id of the requester is associated with the Account-id in User Account table and has the Org-account-delete-priv or the Super-user privileges; If not, log the delete attempt in access violation log and send a message the requester indicating error; Else, deactivate all user accounts associated with the organization and send a message to the requester indicating that the organization account is deactivated and will be permenantly removed in certain days. After certain days, the org account and all its user accounts are deleted from SAFER database. Log any action taken to the System Activity Log.

Update Org Account Request

Accept an Update Org Account request. Determine if the account, identified by Account-id, exists in Org Account. If not, send an error message to the requester indicating that the requested account does not exist; Else, check to ensure that the user-id of the requester is associated with Account-id in the User Account table and has the Org-account-update-priv or the Super-user privilege. If not, log the update attempt in access violation log, and send an error message to the requester indicating that only a user with the Org-account-update-priv or Super-user privileges can update an account; Else, verify that the requested fields are updatable. If not, send an invalid field update request

message to the requester. Else, update the Org Account and the user accounts if requested, and send a successful account update message to the requester. Log any action taken to the System Activity Log.

Org Account Info Request

Accept an Org Account Info request. Determine if the account, identified by Account-id, exists in Org Account. If not send an error message to the requester indicating that the requested account does not exist; Else check to ensure that the user-id is in the User Account for the organization and has the privilege to request org account information. If not, log the attempt in access violation log, and send an error message to the requester indicating error. Otherwise retrieve Org info, bank info and users info and send the result to the requester via Account-req. Log any action taken into System Activity Log.

Create User Account Request

Accept a Create User Account request. Determine if requesting organization identified by Account-id exists in SAFER. If not, send an error message to the requester indicating that the supplied org account does not exist: Else, check to ensure that the user-id of the requester exists for the Account-id in User Account table, and has the User-account-create-priv or the Super-user privilege. If not, log the attempt in access violation log, and send an error message to the requester indicating that only a user with the User-account-create-priv or Super-user privileges can create an account; Else, Determine if new user account(s) exist. For existent one(s), send an error message to the requester indicating that those user(s) exist; while for non-existent user(s), add the user(s) to the User Account, based on Account-id and User-id(s) and send a successful user add message to the requester. Log any action taken to the System Activity Log.

Delete User Account Request

Accept a Delete User Account request. Determine if the account, identified by Account-id, exists in Org Account. If not, send an error message to the requester indicating that the requested org accoun does not exist; Else, determine if the User-id(s) to be deleted is associated with the supplied Account id in Org Account. If not send an error message to the requester; Else, check to ensure that the user-id of the requester is in the User Account and has the User-account-delete-priv or the Super-user privilege. If not, log the attempt in access violation log, and send an error message to the requester indicating that only a user with the User-account-delete-priv or Super-user privileges can delete an user account; Else check to ensure the requester is not deleting itself, if so, send a message to the requester indicating a user can not delete itself. For the user(s) ready to be deleted, deactivate the user account(s) and send a message to the requester indicating those user accounts are deactivated and will be permanently removed in certain days. After certain days, delete those user records from User Account. Log any action taken to the System Activity Log.

Update User Account Request

Accept an Update User Account request. Determine if the account, identified by Account-id, exists in Org Account. If not, send an error message to the requester, indicating that the requested account does not exist; Else, check to ensure that the user-id of the requester is in the User Account, is

associated with Account-id and has the User-account-update priv or the Super-user privilege. If not, log the attempt in access violation log, and send an error message to the requester indicating that only a user with the User-account-update-priv or Super-user privileges can update user accounts; Else, verify that the requested fields are updatable. If not, send an invalid field update request message to the user. Else, update the User Account(s) associated with Account-id and User-id with the supplied fields and send a successful user update message to the requester. Log any action taken to the System Activity Log.

User Account Info request

Accept a User Account Info Request.Determine if the account, identified by Account-id, exists in Org Account. If not, send a message back to the requester indicating error. Else check to ensure if the requester exists for the org and is requesting info on users for the org. If not, send an error message back. If the requester is requesting other users, check to ensure he has the privilege to do so. If not, log the attempt in access violation log, and send an error message back indicating error. Otherwise, retrieve user info and send the result back to the requester. Log any action taken to System Activity Log.

Input: T **Output: F**

Account-req
A request to create, update or delete a user account or an organization account

Billing-data
Billing information received from the VAN for an organization and its users for a given billing period OR a notification of payment by an organization for a given billing period OR the result of a bank account check

Input: F **Output: T**

Account-entry
A tracking entry regarding an account transaction made to the Activity Log

Account-reqresponse
A response sent back to the user indicating the status of his request

Account-violation
Track entry regarding SAFER users' illegal actions on SAFER accounts, for example, trying to update other users' info without sufficient privileges.

Base-state-info
The name and network address of each base state

Billing-info-req

A request made to the VAN for billing information for each organization with an account on the system

Create-erg-account-req
A request to create an organization account

Create-user-account-req
A request to add(create) a user to the system

Delete-erg-account-req
A request to delete an organization account

Delete-user-account-req
A request to delete a user from the system

OS-user-account-req
A request of the operating system to create, update or delete a user or user information to/from the system

Org-account-info-req
A request for SAFER account summary for the organization.

Update-erg-account-req
A request to update the information associated with an organization account

Update-user-account-req
A request to update the information associated with a system user

User-account-info-req
A request for User account summary.

2.2 Billing Manager

The Billing Manager (BM) must be able to request billing information from the VAN for all SAFER System users at some periodic rate. It must be able to process billing information received from the VAN and update each user billing account, in the SAFER Database, accordingly and send each user a user information exchange bill, User-IE-bill, at some periodic rate for services used during that period.

At some predefined time period, e.g. once a month, issue a billing information request for each organization account. On receipt of the monthly billing data, store the organization billing information in the Org Billing Account and the user billing information in the User Billing Account. Debit the Account-$-value in the Org Account by the monthly billing charge for each organization. Generate a bill for each organization, User-IE-bill, as an organization cost summary and a user cost summary for each organization. On receipt of an organization payment notification, Org-payment-notification, credit the Account-$-value in the Org Account for the organization whose payment was received.

Input: T **Output: F**

Credit-transaction
Receipt of payment. Initiates an update of the safer account.

Input: T **Output: F**

Old-invoice-req
A request for an invoice from a previous billing cycle.

Org-info
Identification information about an organization

User-info
Information about a user account

Van-billing-info
This is data received from the VAN which is used to assess and allocate charges for an organization using the VAN for communications. It includes the VAN's charges for service for
each organization.

Input: F **Output: T**

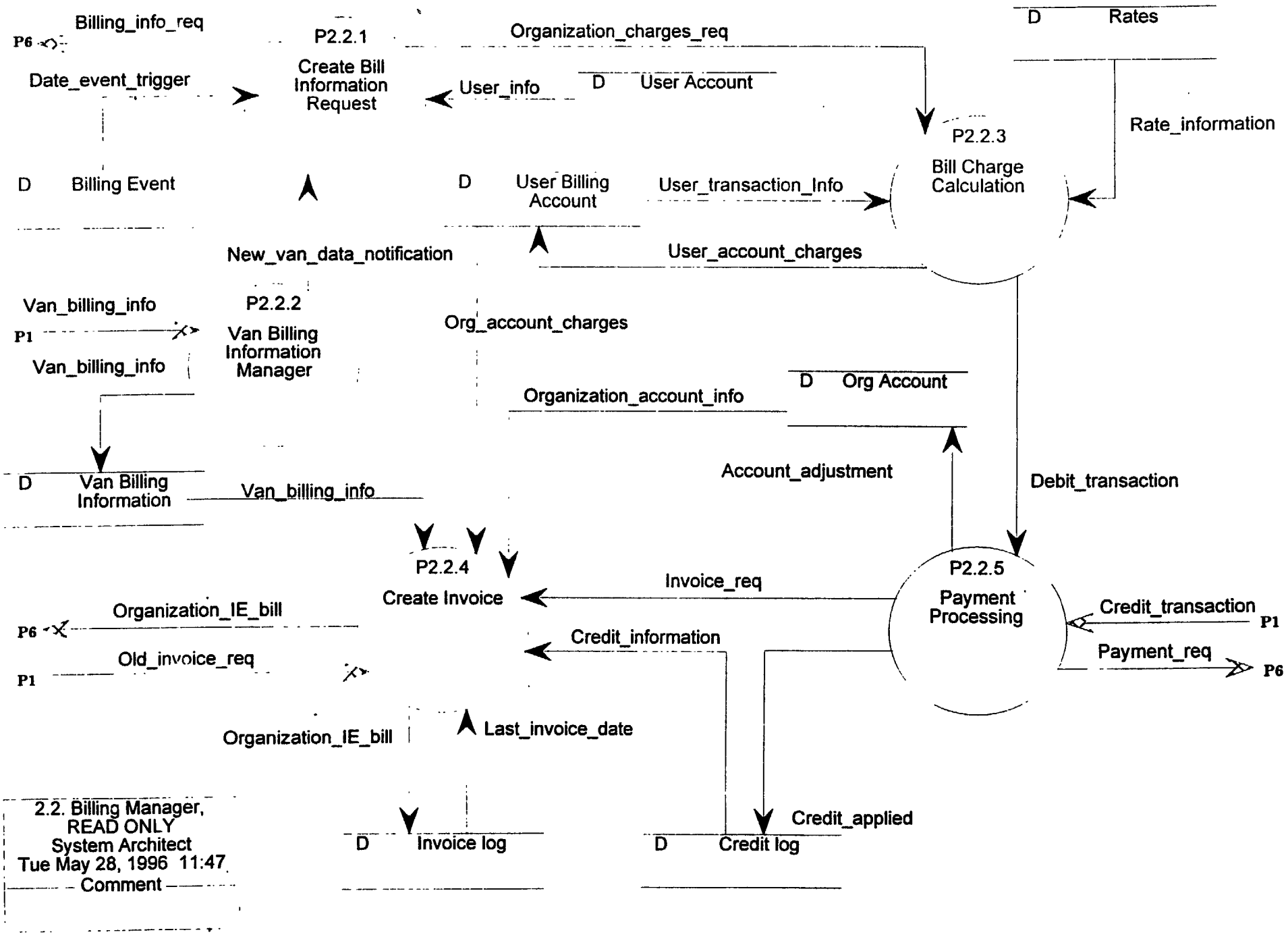
Billing-inforeq
A request made to the VAN for billing information for each organization with an account on the system

User-IE-Bill

The bill sent to each organization each billing period

User-billing-info

Information received from the VAN regarding the transaction charges associated with each user of an organization for a given billing period


Billing_info_req
P6
P2.2.1
Create Bill Information Request
Organization_charges_req
D Rates
Date_event_trigger
User_info
D User Account
Rate_information
P2.2.3
Bill Charge Calculation
D Billing Event
D User Billing Account
User_transaction_Info
New_van_data_notification
User_account_charges
Van_billing_info
P1
P2.2.2
Van Billing Information Manager
Org_account_charges
Van_billing_info
D Org Account
Organization_account_info
D Van Billing Information
Van_billing_info
Account_adjustment
Debit_transaction
P2.2.4
Create Invoice
Invoice_req
P2.2.5
Payment Processing
Credit_transaction
P1
Organization_IE_bill
P6
Credit_information
Payment_req
P6
Old_invoice_req
P1
Organization_IE_bill
Last_invoice_date
2.2. Billing Manager,
READ ONLY
System Architect
Tue May 28, 1996 11:47
Comment
D Invoice log
Credit_applied
D Credit log

2.2.1 Create Bill Information Request

This process is triggered by a Date-Event-Trigger (to be determined) received from the EVENT data store. Given the trigger the process creates a message requesting transaction and billing information from the Value Added Network (VAN) for all organizations using the VAN to interact with SAFER. A Billing-info-req is sent to the VAN.

This process also creates a request to the Billing Service Level Charge Data Process (2.22) for organizations which use communications access methods other than the VAN such **as a l-800** dial up, INTERNET etc. An organization-charges-req is sent.

Input: T **Output:** F

Date-event-trigger

A determined date sent to initiate creation of a request in P2.2.1 to gather and send billing information to an organization.

New-van-data-notification

Notification that new van billing information has arrived.

User-info

Information about a user account

Input: F **Output: T**

Billing-info-req

A request made to the VAN for billing information for each organization with an account on the system

Organization-charges-req

A request made to process P2.2.3 for billing information for each organization with an account on the SAFER system.

2.2.3 Bill Charge Calculation

This process performs the calculations for applying charges to an organization or user for service provided during the billing cycle. This process is triggered by Organization-charges-req from 2.2. I. This process also receives rate information from the rates data store and applies them to the User-transaction-info it retrieves from the User Billing Account data store. The charges are written to the User Billing account data store, and the debit transaction is sent to 2.2.5.

Input: T **Output: F**

Organization-charges-req

A request made to process P2.2.3 for billing information for each organization with an account on the SAFER system.

Rate-information

Information concerning charge per product. products currently include snapshots. profiles, and reports.

User-transaction-Info

Information describing transactions that are billable. (products)

Input: F **Output: T**

Debit-transaction

Charges to be applied to the safer account.

User-account-charges

Records including information describing a charge for a completed transaction of a product.

2.2.4 Create Invoice

This process creates and sends a billing invoice to an organization for services provided during a billing cycle. In order to process the invoice, .Organization-charges are received from process P.2.2.3, Billing Charge Calculation. This process summarizes the charges applied for service and formats the invoice. This process also uses Organization-account-info received from the Organization Account data store to send the invoice to the billing address.

This process sends the invoice, Organization-IE-Bill to the billing address. It also sends the Organization-IE-Bill to process P.2.2.5, Payment Processing. This is used to compare the invoice total to received payments.

Input: T **Output: F**

Credit-information

Transactions that resulted in a positive adjustment of an organizations account balance.

Invoice-req

Request for an invoice to be produced and sent to an organization.

Last-invoice-date

The date of the last invoice issued to an organization.

Old-invoice-req

A request for an invoice from a previous billing cycle.

Org-account-charges

Transactions that resulted in a negative adjustment of an organizations account balance.

Organization-account-info

This data provides information regarding the organization. It includes the level of service contracted with the organization, its name and billing address.

Van-billing-info

This is data received from the VAN which is used to assess and allocate charges for an organization using the VAN for communications. It includes the VAN's charges for service for
each organization.

Input: F **Output: T**

Organization-IE-bill

The bill sent to an organization.

2.2.5 Payment Processing

This process receives payment from an organization for a specified billing cycle and posts the payment. The process receives a Payment-Transaction from P1 indicating payment receipt and amount from an organization. This process compares the payment received to the Organization-IE-Bill data sent in the invoice to the organization. When the comparison is complete, Post-payment data is sent to the Organization Billing Account data store indicating the amount received and any credit or debit pertaining to the organization's billing account.

Input: T **Output: F**

Credit-transaction

Receipt of payment. Initiates an update of the safer account.

Input: T **Output: F**

Debit-transaction

Charges to be applied to the safer account.

Input: F **Output: T**

Account-adjustment

A positive or negative adjustment to an organizations account.

Input: F **Output: T**

Credit-applied

Records including information describing a credit to an organizations account.

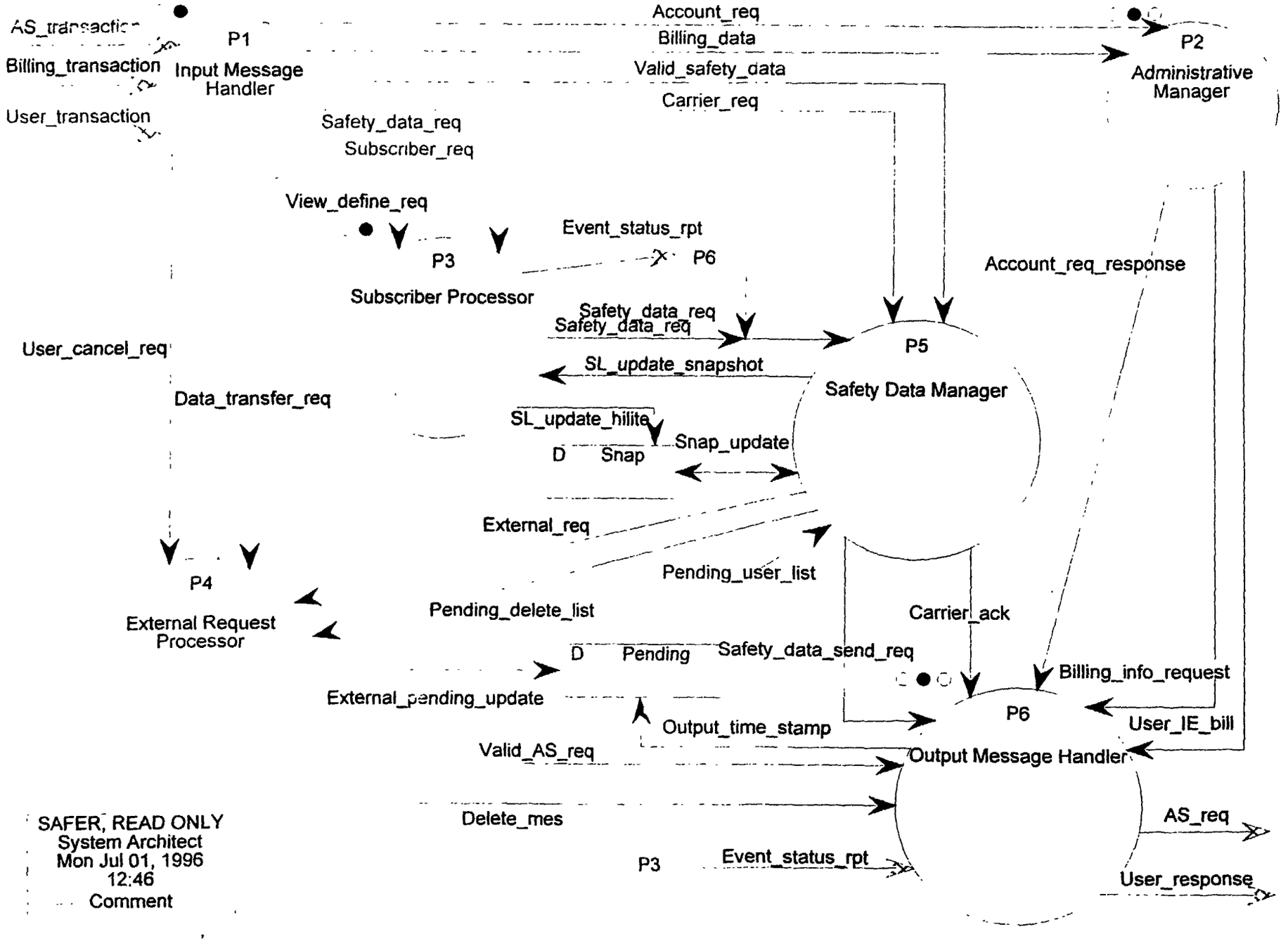
Input: F **Output: T**

Invoice-req

Request for an invoice to be produced and sent to an organization.

Payment-req

A request for payment. This includes electronic funds transfer, or credit card transfer.


AS_transaction
Billing_transaction
User_transaction
P1
Input Message Handler
Account_req
Billing_data
Valid_safety_data
Carrier_req
P2
Administrative Manager
Safety_data_req
Subscriber_req
View_define_req
P3
Subscriber Processor
Event_status_rpt
P6
Account_req_response
Safety_data_req
Safety_data_req
User_cancel_req
P5
Safety Data Manager
SL_update_snapshot
Data_transfer_req
SL_update_hilite
D
Snap
Snap_update
External_req
Pending_user_list
P4
External Request Processor
Pending_delete_list
Carrier_ack
D
Pending
Safety_data_send_req
Billing_info_request
External_pending_update
Output_time_stamp
P6
User_IE_bill
Valid_AS_req
Output Message Handler
Delete_mes
AS_req
P3
Event_status_rpt
User_response
SAFER, READ ONLY
System Architect
Mon Jul 01, 1996
12:46
Comment

3.0 Subscriber Processor

The Subscriber Processor (P3) creates and maintains Subscribtions, examines incoming updated snapshots and distributes appropriate copies. It is also responsible for receiving and processing user requests to update, or add to the view table.

User subscriptions specify a carrier-of-interest group (either by carrier ID's or other groupings, e.g., states), an output type (profile or a view that describes which fields in the snapshot the user is interested in), and a set of events to monitor. They are received via a Subscriber-req. Events are monitored via two types of change conditions or triggers. These include: simple value change of a field, (simple change) and a field change such that a field moves from one prescribed interval of valu ranges to another (range change).

When a new snapshot is received by the safety data manager, the Subscriber processor (P3.1) is provided information via the SL-update-snapshot. The old and new data are examined to determine whether a currently-monitored event has occurred. If that is the case, the new snapshot is processed further. Subscriptions are examined to determine if: a user is monitoring the event which occurred to the carrier represented in the new data is in the user's area of interest. If these conditions are true, SAFER transmits a either a snapshot or a profile back to the user via a Safety-data-req, or places an entry in a queue that will be processed at a later time.

Input: T **Output: F**

SL-update-snapshot

This is the dataflow which P3 receives from P5. It includes a new snapshot, and include an old snapshot.

Subscriber-req

Request from a user to create/update a subscriber list. Passed from Pl to P3. Contains the user's method of data delivery, area of interest, and events to be monitored.

View-define-req

Request to update the system view table. Passed from Pl to P3. This request is available to only certain system-level SAFER users.

Input: F **Output: T**

Event-status-rpt

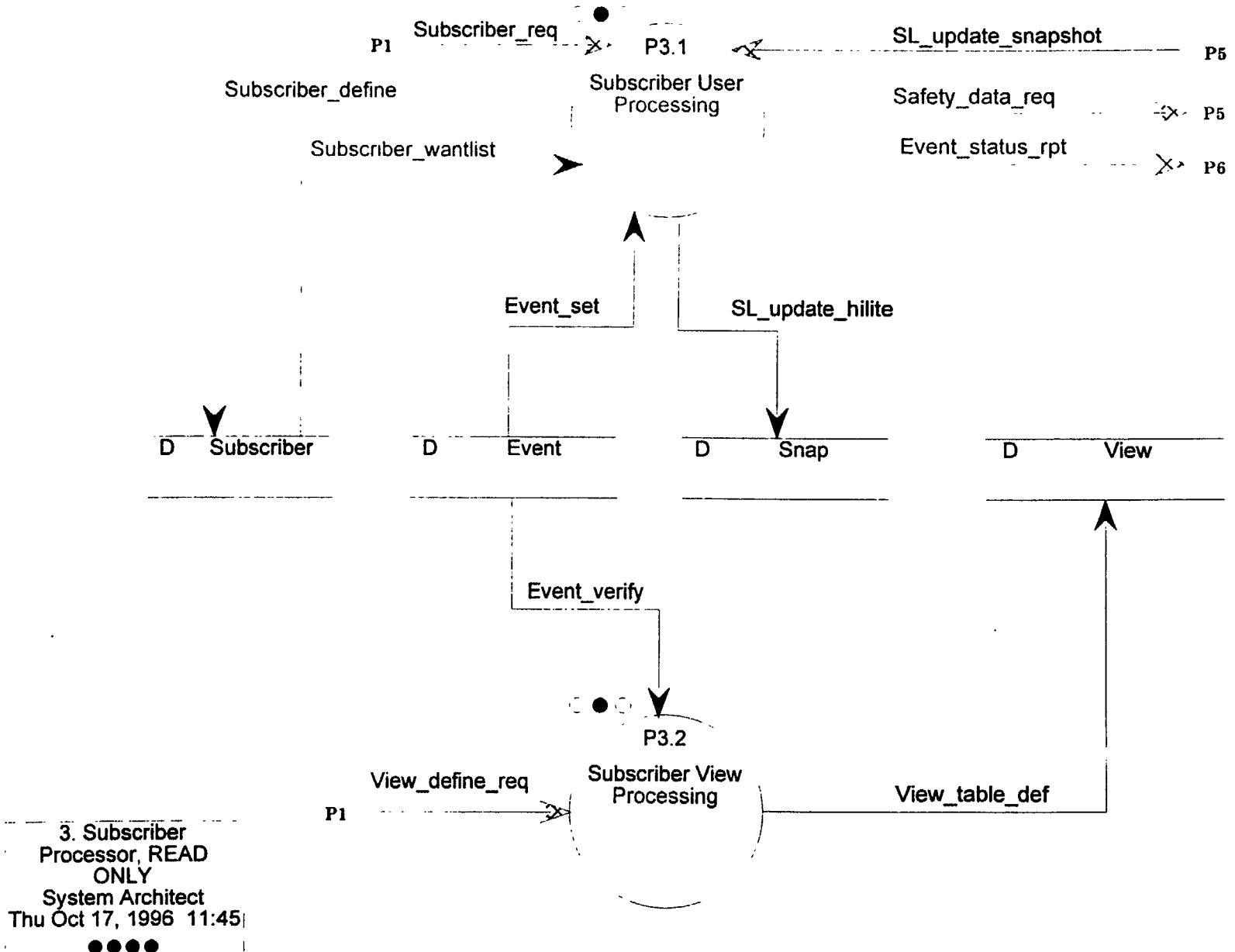
This is a data report sent to the user on request. It either reflects a user's current subscriber status, and/or system event monitoring status.

SL-update-hilite

This dataflow contains the updated hilite flags computed by P3 after comparing old and new snap versions.

Safety-data-req

A request for snapshot or profile data.


P1
Subscriber_req
P3.1
SL_update_snapshot
P5
Subscriber_define
Subscriber User Processing
Safety_data_req
P5
Subscriber_wantlist
Event_status_rpt
P6
Event_set
SL_update_hilite
D Subscriber
D Event
D Snap
D View
Event_verify
P3.2
View_define_req
Subscriber View Processing
View_table_def
P1
3. Subscriber Processor, READ ONLY
System Architect
Thu Oct 17, 1996 11:45

3.1 Subscriber User Processing

Subscriber User Processing creates and maintains User Subscriptions, and examines incoming updated snapshots and distributes appropriate copies to users.

The Subscrption requests (Subscriber-req), are first validated (using Event-set as a baseline). The output type must be either a profile, or one of the valid snap views. A coded list of each user's events of interest (event mask) is created and checked. For example, if the user specifies his area of interest via value matching, those entries are checked by type against a valid system list. A user may request a view of his current subscriber status (retrieved via Subscriber-wantlist) to be sent to him via Event-status-rpt.

Entries are placed into the Subscriber data store keyed by external user ID via the Subscriber-define dataflow. Updates may occur as a result of user instructions (Subscriber-req), or system-generated updates to event tables.

If SL-baseline-data is set to YES then this process will construct a standard safety data request using the subscriber list carrier of interest definition. This will result in the user being sent a baseline set of data.

Incoming updated snapshots are identified via SL-update-snapshot. The specific events currently being monitored (and compared) are retrieved from the event data store via Event-set. The old and new data are examined to determine which specific event triggers occurred for each snapshot. Event hilite flags are written to the Snap database to indicate what has changed.

After determining which of the monitored events changed in the new snapshot, those changes are compared to the event masks stored in the Subscriber list data store for each user (via Subscriber-wantlist). The carriers of interest for those users for which an event match occurred are then checked against the new snapshot to determine if a second level (carrier) match occurred. The carrier of interest matching involves either comparing the incoming Carrier ID to a user-supplied list, or value matching other incoming carrier fields (state, type of operation) to user-specified choices. Precise matches required by a user are noted as each new snapshot is examined. When the supply of new data is exhausted, a Safety-data- req is constructed for each appropriate user and sent to P5, or an entry is inserted into the sendlist-queue.

Input: T **Output: F**

Event-set

This is the data flow which P 3.1 uses to validate a subscriber list request and to also determine which events have been triggered for newly-arrived snapshots.

SL-update-snapshot

This is the dataflow which P3 receives from P5. It includes a new snapshot, and may include an old snapshot.

Subscriber-req

Request from a user to create/update a subscriber list. Passed from P1 to P3. Contains the user's method of data delivery, area of interest, and events to be monitored.

Subscriber-wantlist

This dataflow tells P 3.1 the area of interest a user has expressed in a subscriber list. Newly-arrived snapshots are compared to this area of interest for possible output transmission.

Input: F **Output: T**

Event-status-rpt

This is a data report sent to the user on request. It either reflects a user's current subscriber status, and/or system event monitoring status.

SL-update-hilite

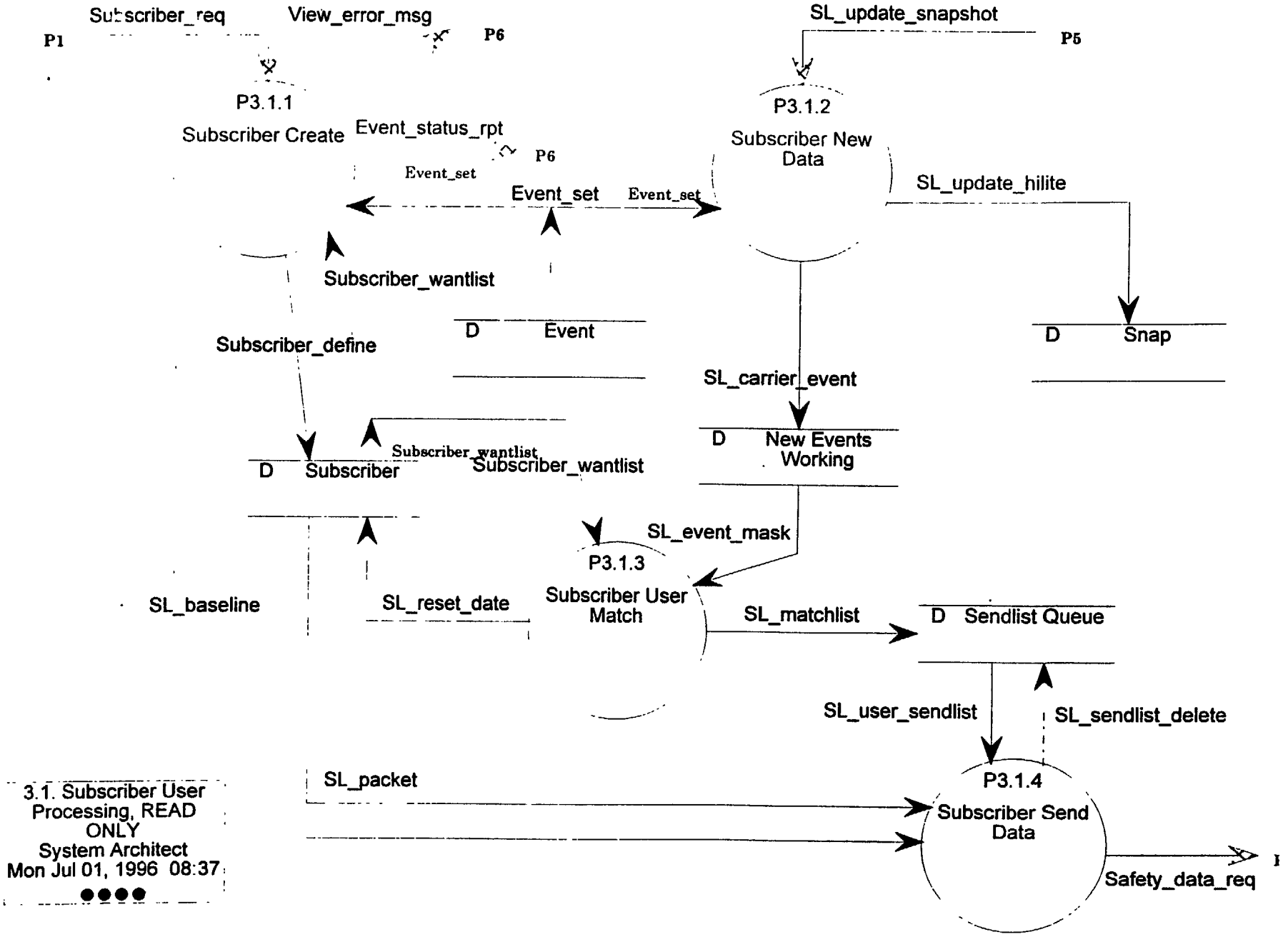
This dataflow contains the updated hilite flags computed by P3 after comparing old and new snap versions.

Safety-data-req

A request for snapshot or profile data.

Subscriber-define

This is a dataflow P 3.1.1 writes to the data store. It is the mechanism to document a user's subscriber list and may be updated subsequently by the user or when an event define request has been received. Some fairly static fields are the request header, etc. The basic elements which may change from time to time are the user's event monitoring (mask), and creation date.


Subscriber_req
P1
View_error_msg
P6
SL_update_snapshot
P5
P3.1.1
Subscriber Create
Event_status_rpt
P6
P3.1.2
Subscriber New Data
Event_set
Event_set
Event_set
SL_update_hilite
Subscriber_wantlist
Subscriber_define
D Event
D Snap
SL_carrier_event
Subscriber_wantlist
D Subscriber
Subscriber_wantlist
D New Events Working
SL_event_mask
P3.1.3
SL_baseline
SL_reset_date
Subscriber User Match
SL_matchlist
D Sendlist Queue
SL_user_sendlist
SL_sendlist_delete
SL_packet
P3.1.4
Subscriber Send Data
Safety_data_req
3.1. Subscriber User Processing, READ ONLY
System Architect
Mon Jul 01, 1996 08:37

3.1.1 Subscriber Create

This process enters the user's subscription into the system. A Subscriber-req is received from Pl. A current event set is retrieved from the data store (via Event-set) and is used to validate the request. If the Event-status-flag is set, a current definition of the user's subscription status is retrieved via Subscriber-wantlist. The user's status and the current events being monitored are sent back to the user via Event-status-rpt.

If this request is not a status request, a validity check is performed on incoming fields. For example, event names are compared to the list of currently-monitored events. The event names will indicate whether simple change or range monitoring is desired. An event mask (which will allow efficient matching to new data) for each user is built and dated. A check is made to compare the snapshot view requested and the events to be monitored. If the view does not include the field/event monitored, a message is sent back to the user, and he may update the request if he wishes to. If the request is for periodic processing, the user's SL-process-date is computed. That date, along with the entire subscription definition is written to the data store via Subscriber-define. The unprocessed 'where' clause is stored for each list. The view type desired is indicated and saved in SL-output-type via Subscriber-define.

If SL-baseline-data is set to YES then this process sends SL-baseline to P3.1.4 , This will be used to construct a standard safety data request using the subscriber list carrier of interest definition for a baseline set of data.

Input: F **Output: F**

View-error-msg

Message sent back to user from P3.2 . Identifies error in a view define request.

Input: T **Output: F**

Event-set

This is the data flow which P 3.1 uses to validate a subscriber list request and to also determine which events have been triggered for newly-arrived snapshots.

Subscriber-req

Request from a user to create/update a subscriber list. Passed from Pl to P3. Contains the user's method of data delivery, area of interest, and events to be monitored.

Subscriber-wantlist

This dataflow tells P 3.1 the area of interest a user has expressed in a subscriber list. Newly-arrived snapshots are compared to this area of interest for possible output transmission.

Input: F **Output: T**

Event-status-rpt

This is a data report sent to the user on request. It either reflects a user's current subscriber status, and/or system event monitoring status.

SL-baseline

Data flow from P3.1.1 to P3.1.4 indicating that a baseline set of subscriber user's carriers of interest should be sent upon initialization.

Subscriber-define

This is a dataflow P 3.1.1 writes to the data store. It is the mechanism to document a user's subscriber list and may be updated subsequently by the user or when an event define request has been received. Some fairly static fields are the request header, etc. The basic elements which may change from time to time are the user's event monitoring (mask), and creation date.

3.1.2 Subscriber New Data

This process performs the initial processing required when P5 receives new snapshots from the Authoritative Source. First, this process receives SL-update-snapshot from P5, identifying a new snapshot arriva, and the previous version of that snapshot. A current event set is retrieved from the data store (via Event-set) to use in determining relevant events.

If no old version is present, this case is considered an add operation. To ensure that all users receive the added information all event mask fields are set to 'yes' and entries are written to the working data store.

If both old and new snapshots are present, it is considered an update operation. Using the event set previously retrieved, the process compares old to new, setting the event mask and hilite flags when appropriate event triggers are detected. When the basic flags are set, this process also determines if this event qualifies for the 'any change to a field in my snapshot view' criteria (another event). This information (via SL-carrier-event) is written to the New Events Working data store. The hilite flags are written to the Snap data store via SL-update-hilite.

Input: T **Output: F**

Event-set

This is the data flow which P 3.1 uses to validate a subscriber list request and to also determine which events have been triggered for newly-arrived snapshots.

SL-update-snapshot

This is the dataflow which P3 receives from P5. It includes a new snapshot, and may include an old snapshot.

Input: F **Output: T**

SL-carrier-event

This data flow is written by P 3.1.2 to temporary data store New Events Working. It identifies the newly-arrived snapshot and its event characteristics (mask).

SL-update-hilite

This dataflow contains the updated hilite flags computed by P3 after comparing old and new snap versions.

3.1.3 Subscriber User Match

This process looks for matches between new snapshots and the carriers and events of interest for each user. It retrieves the subscriber-wantlist from the datastore. If the SL-process-date has expired, it is recomputed and written via SL-reset-date to the data store. For each snapshot, it compares the incoming carrier event mask (SL-event-mask) to the event mask of each user. Common events between the two indicate that the snapshot is a potential match and must be examined further.

If the **user's process** type field **indicates 'send** when available, SL-process-date is set to 'today', otherwise left as is. If matchtype = Carrier ID, the process must compare the Carrier ID in the updated snapshot to those in the users list. If a match is found, an entry is written for that user and carrier ID via SL-matchlist to the data store. If matchtype = Field value, the value test must be performed on the updated carrier characteristics. If a match occurs, an entry for that user/carrier is written via SL-matchlist. When processing is completed, the 'New events working' data store is deleted. Note that the 'where' clause is always evaluated, in order to ensure that any carrier changes are reflected in the retrieval.

Input: T Output: **F**

SL-event-mask

This data flow is written by P 3.1.2 to temporary data store for subsequent use by P 3.1.3. It identifies the newly-arrived snapshot and its event characteristics (mask).

Subscriber-wantlist

This dataflow tells P 3.1 the area of interest a user has expressed in a subscriber list. Newly-arrived snapshots are compared to this area of interest for possible output transmission.

Input: F **Output: T**

SL-matchlist

Dataflow written to Sendlist Queue data store. Contains the specific carrier(s) required to be sent to each specific user (due to subscriber list activity).

SL-reset-date

Dataflow written to Subscriber data store. Contains the user id and the SL-process-date which has been recomputed by P3.1.3.

3.1.4 Subscriber Send Data

This process compiles and formulates the standard safety-data-requests sent to the safety data manager P5.

For each user in the Sendlist Queue data store, the SL-user-sendlist is retrieved. If SL-process-date is before or equal to 'today', it will be processed immediately. Certain fields (packet header. etc) are generated or extracted via an SLgacket from the subscriber data store. A Safety-data-req with 1 to 'n' Carrier IDs for each user and sent to P5. After sending the request, it must delete the corresponding entry in the Sendlist Queue.

If SL-baseline is received from P3.1.1, this process constructs a standard safety data request using the subscriber list carrier of interest definition and sends it to P5. The view type is passed in SDR-type via Safety-data-req. For some users, certain fields in certain view might get blanked because of privacy considerations.
The type of output is controlled by SDR-type (Snap,views, or profile). If profile, it will be sent by the current standard method within SAFER. In the first stages of operation, profiles may be sent as printer files (as they are currently constructed). Later, it is expected that they would be transmitted via stardard ED1 methods.

Input: T **Output: F**

SL-baseline
Data flow from P3.1.1 to P3.1.4 indicating that a baseline set of subscriber user's carriers of interest should be sent upon initialization.

SL-packet
This is a user information and a packet header used by P 3.1.4 to uniquely identify and specify a system-generated subscriber-based safety data request.

SL-user-sendlist
Dataflow read from Sendlist Queue data store by P 3.2.4. It contains for each user, the specific carrier(s) required to be sent (now), due to subscriber list activity.

Input: F **Output: T**

SL-sendlist-delete
Dataflow sent to Sendlist Queue data store by P 3.2.4. Since the request has now been issued, the subscriber must be deleted from the Queue.

Safety-data-req
A request for snapshot or profile data.

3.2 Subscriber View Processing

Subscriber View Processing (3.2) creates and maintains the SAFER View Tables. This process examines/validates incoming view table updates.

Requests to update the view table are received via the View-define-req. These requests must be validated and can only be generated by SAFER users with super-user privileges.

Incoming view definition requests are examined to establish specific events and their associated triggers. The proposed events must appear in a system list of valid event names (retrieved via Event verify).

After validation, a view table definition, containing all event names is written to the data store via view-table-def. All validation activities are logged.

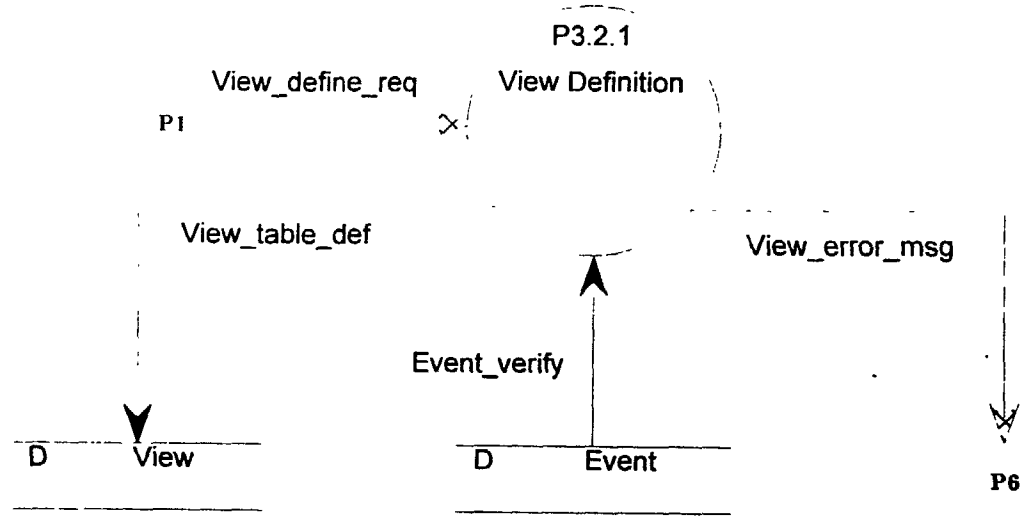
Input: T **Output: F**

Event-verify
This is the data flow extracted from data store to verify an incoming event-define-req.

View-define-req
Request to update the system view table. Passed from P1 to P3. This request is available to only certain system-level SAFER users.

Input: F **Output: T**

View-table-def
This is a dataflow P 3.2.1 writes to theView data store. It is the mechanism to update the system view table when a view define request has been received.



3.2. Subscriber View Processing, READ ONLY
System Architect
Mon Jul 01, 1996 08:59

3.2.1 View Definition

View Definition (P3.2.1) contains the processes which are required to create and maintain the SAFER view table. This is the table which indicates which snapshot fields are being monitored for changes and possible transmission to users in conjunction with a Subscriber list.

The entries in the view table consist of a view name and a list of' event names.

When a view definition request is received (via View-define-req), the following actions are taken: the request must be verified in a detailed manner to ensure that the particular events and the associate parameters are legal and practical; the official SAFER view table must be updated (done via View-table-def).

Input: F **Output: F**

View-define-req

Request to update the system view table. Passed from Pl to P3. This request is available to only certain system-level SAFER users.

Event-verify

This is the data flow extracted from data store to verify an incoming event-define-req.

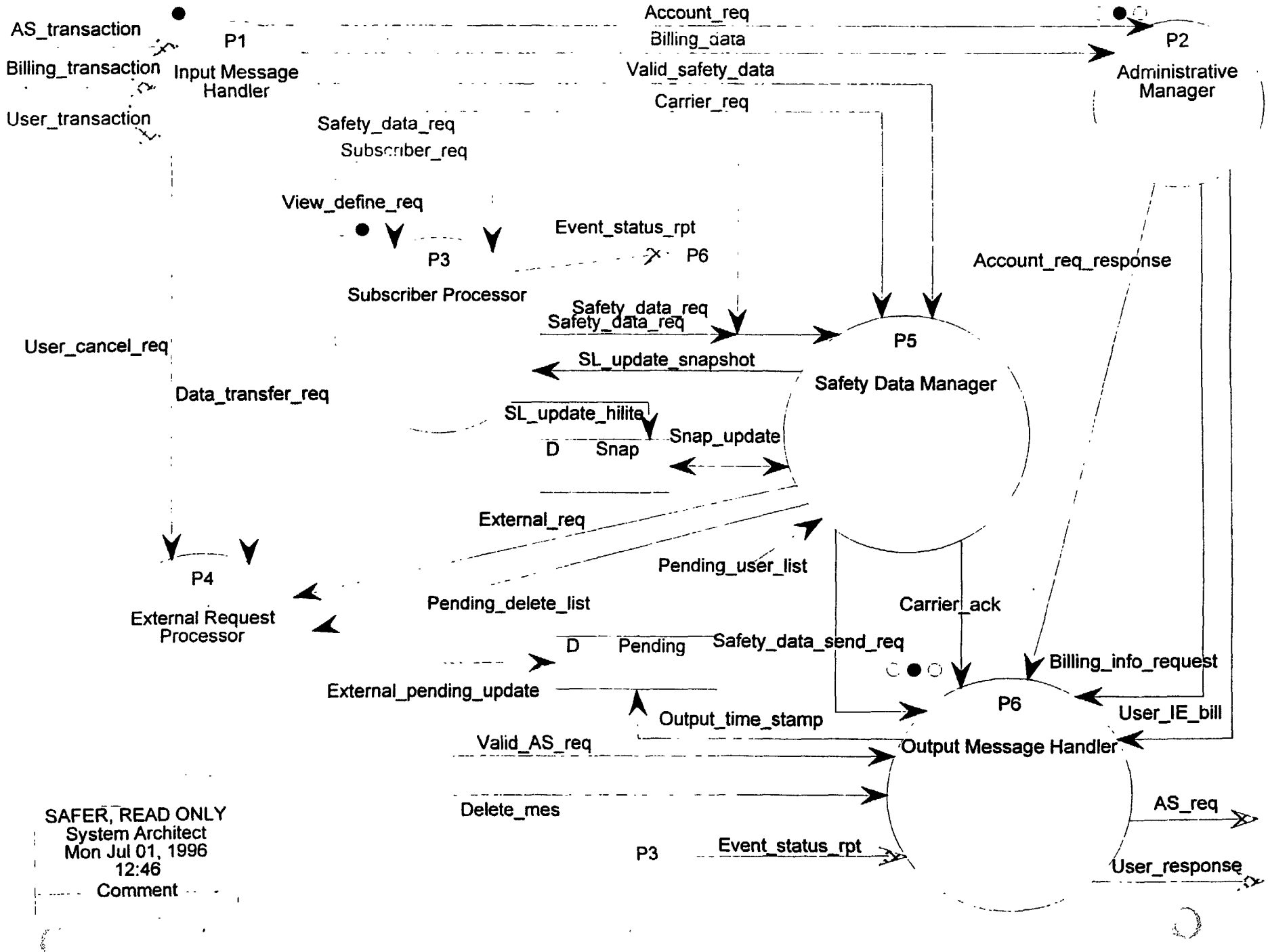
Input: F **Output: T**

View-error-msg

Message sent back to user from P3.2 . Identifies error in a view define request.

View-table-def

This is a dataflow P 3.2.2 writes to the View data store. It is the mechanism to update the system view table when a view define request has been received.


AS_transaction
Billing_transaction
User_transaction
P1
Input Message Handler
Account_req
Billing_data
Valid_safety_data
Carrier_req
P2
Administrative Manager
Safety_data_req
Subscriber_req
View_define_req
P3
Subscriber Processor
Event_status_rpt
P6
Account_req_response
Safety_data_req
Safety_data_req
User_cancel_req
P5
SL_update_snapshot
Safety Data Manager
Data_transfer_req
SL_update_hilite
D Snap
Snap_update
External_req
Pending_user_list
P4
External Request Processor
Pending_delete_list
Carrier_ack
D Pending
Safety_data_send_req
Billing_info_request
External_pending_update
P6
Output_time_stamp
User_IE_bill
Valid_AS_req
Output Message Handler
Delete_mes
AS_req
P3
Event_status_rpt
User_response
SAFER, READ ONLY
System Architect
Mon Jul 01, 1996
12:46
Comment

4.0 External Request Processor

The External Request Processor handles the routing, queuing and formatting of requests to the Authoritative Sources. Incoming requests for data are placed in the Pending data store. Setting the Send-request flag indicates that the request is ready to be formatted for output to the Authoritative Source (by P6). If there are several requests of the same type for a single Authoritative Source, they are bundled together into a single Valid-AS-req. The Network-address of the Authoritative Source is obtained from the Base State Routing data store. Requests which have been issued are monitored for receipt of data. If snapshot or profile data is not received within a predetermined timeframe (the timeframe for each AS is found in Base State Routing), then the request is written out to an AS Nonresponse Log.

Input: T **Output: F**

Data-transfer-req

Request for data to be passed through SAFER to an authoritative source.

External-req

A request to an authoritative source for carrier data processing, request deletes, snapshot or profile data.

Pending-delete-list

A list of snapshot or profile Carrier-unique-id, Req-user-id combinations that is sent to P4. The information is used for Pending data store maintenance.

User-cancel-req

Data needed to cancel a low priority request.

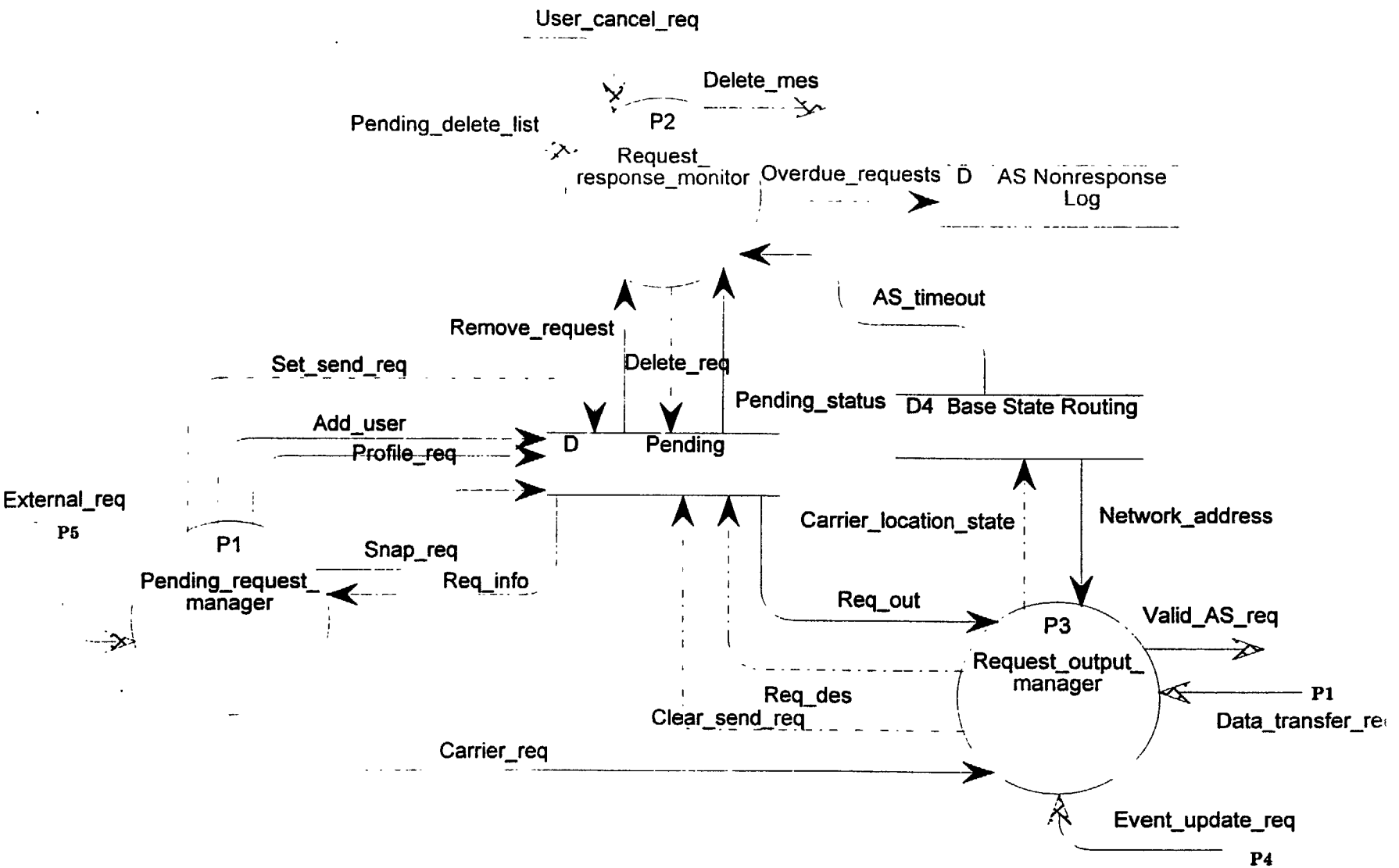
Input: F **Output: T**

Delete-mes

This data flow provides a message to P6 which identifies that a request has been received to delete a previous request and that P4 has already serviced this delete request. Servicing by P4 indicates that P4 has looked for this request in pending and has deleted it if it was found.

External-pending-update

This data flow represents the data which is put into the pending file to maintain a record of requests until the associated data is received and the request has been


User_cancel_req
Delete_mes
P2
Pending_delete_list
Request_
response_monitor
Overdue_requests
D
AS Nonresponse
Log
AS_timeout
Remove_request
Delete_req
Set_send_req
Pending_status
D4 Base State Routing
Add_user
Profile_req
D
Pending
External_req
P5
P1
Carrier_location_state
Network_address
Snap_req
Pending_request_
manager
Req_info
Req_out
P3
Valid_AS_req
Request_output_
manager
P1
Req_des
Clear_send_req
Data_transfer_re
Carrier_req
Event_update_req
P4
External Request
Processor, READ ONLY
System Architect
Thu Nov 02, 1995 22:27
Comment
P6

4.1 Pending-request-manager

This process puts incoming snapshot and profile requests in the Pending data store as they are received from P5 (External-req). The packet-id in the packet-header provides a unique identification number for that request. When a profile request comes in, then each request of that type in the Pending data store is checked (via Req-info) to identify if a profile request was already initiated for that carrier identification. If one already exists, then its priority is checked. If the existing request is of lower priority, then a new request is put in the Pending data store and a Set-send-req is issued. If the existing request is of the same or higher priority, then the new user information is added to the existing request via Add-user and a Set-send-req is not issued. If the request is for snapshots (i.e. a snapshot refresh request), they are added to the Pending data store. If the request is a carrier related request (add/update/delete) then it is sent to the Request-output-manager for appropriate formatting.

Input: T **Output: F**

External-req

A request to an authoritative source for carrier data processing, request deletes, snapshot or profile data.

Req-info

Provides request data to the Pending-request-manager which is necessary to consolidate profile requests from multiple users.

Input: F **Output: T**

Add-user

Provides user information to add to the user list of an existing profile data request.

Carrier-req

Request to add, update or delete carrier information.

Profile-req

This data flow results from adding a request ID to the profile type of External-req, and is used to put the request information in the pending file.

Set-send-req

This data flow sets the Send-req in Pending to indicate that a request is ready to be sent to the Authoritative Source.

Snap-req

Provides information for entry of snapshot requests into the pending file.

4.2 Request-response-monitor

This process monitors the time since a data request has been issued by looking at the Pending-status and compares against the Timeout value for a request of that priority and for that Authoritative source (gets timeout for that AS from Base State Routing). The Pending data store is read and each request is checked to determine how much time has passed since being sent. The time the request was initiated is compared to the allowable response'time for a request from that AS. If the allowable time is exceeded, then the overdue request is removed (Remove-request) from Pending and placed in the AS Nonresponse Log (Overdue-requests). Further action to obtain the data for these requests or to notify users will be the responsibility of the SAFER system operator. This process also handles deletes from the Pending data store. Deletes can be initiated through a User-cancel-request or the Pending-delete-list. The User-cancel-request prompts the process to delete any associated pending entries related to the specified User-transjd. Upon completion of deleting any related entries, a delete message is sent to P6. The Pending-delete-list from P5 provides a list of Carrier-unique-id's for data which has come been received. In response to receiving the Pending delete-list, this process deletes all entries associated with each Carrier-unique-id.

Input: T **Output: F**

AS-timeout

This dataflow provides the allowable time for data to be received from an authoritativ source before the message is resent.

Pending-delete-list

A list of snapshot or profile Carrier-unique-id, Req-user-id combinations that is sent to P4. The information is used for Pending data store maintenance.

Pending-status

Used for polling the requests waiting for data to determine past due requests which have not been serviced.

Remove-request

This data flow supports the movement of a request which has not been sufficiently responded to by receipt of data from the authoritative source to the AS Nonresponse Log.

User-cancel-req

Data needed to cancel a low priority request.

Input F **Output: T**

Delete-mes

This data flow provides a message to P6 which identifies that a request has been received to delete a previous request and that P4 has already serviced this delete request. Servicing by P4 indicates that P4 has looked for this request in pending has deleted it if it was found.

Overdue-requests

Provides all the data in the Pending data srore to record a specific request which has not received a response within a reasonable time period.

4.3 Request-output-manager

This process polls the Pending data store and gets request information for requests which have the Send-req flag set. This process also receives incoming data transfer/event update requests and formats them into Valid-AS-requests for P6. If the Send-req flag is set, then a Req-des is issued to obtain the information for that request (Req-out). This process uses the Carrier-location-state for that request to obtain the corresponding AS-address from Base State Routing. The Valid-AS-request is formatted and issued to P6. A Clear-send-req is issued to clear the Send-req flag in the Pending data store indicating that the request has been sent to P6.

Input: F **Output: F**

Valid-AS-req

Appropriately formatted request for an authoritative source. Includes the authoritative source address and request priority.

Input: T **Output: F**

Carrier-req

Request to add, update or delete carrier information.

Data-transfer-req

Request for data to be passed through SAFER to an authoritative source.

Event-update-req

This is a transaction sent via P4 to all authoritative sources, requesting that they update their local copy of the event table. The event table fields are passed along as parameters at the end of this transaction.

Network-address

This data flow provides the network address from the Authoritative Source which is then passed with the request information to P6 as part of a Valid-AS-request.

Req-out

Provides request information from the Pending data store to the Request-Output-manager so that request information can be formatted into Valid-AS-req for output to P6.

Input: F **Output: T**

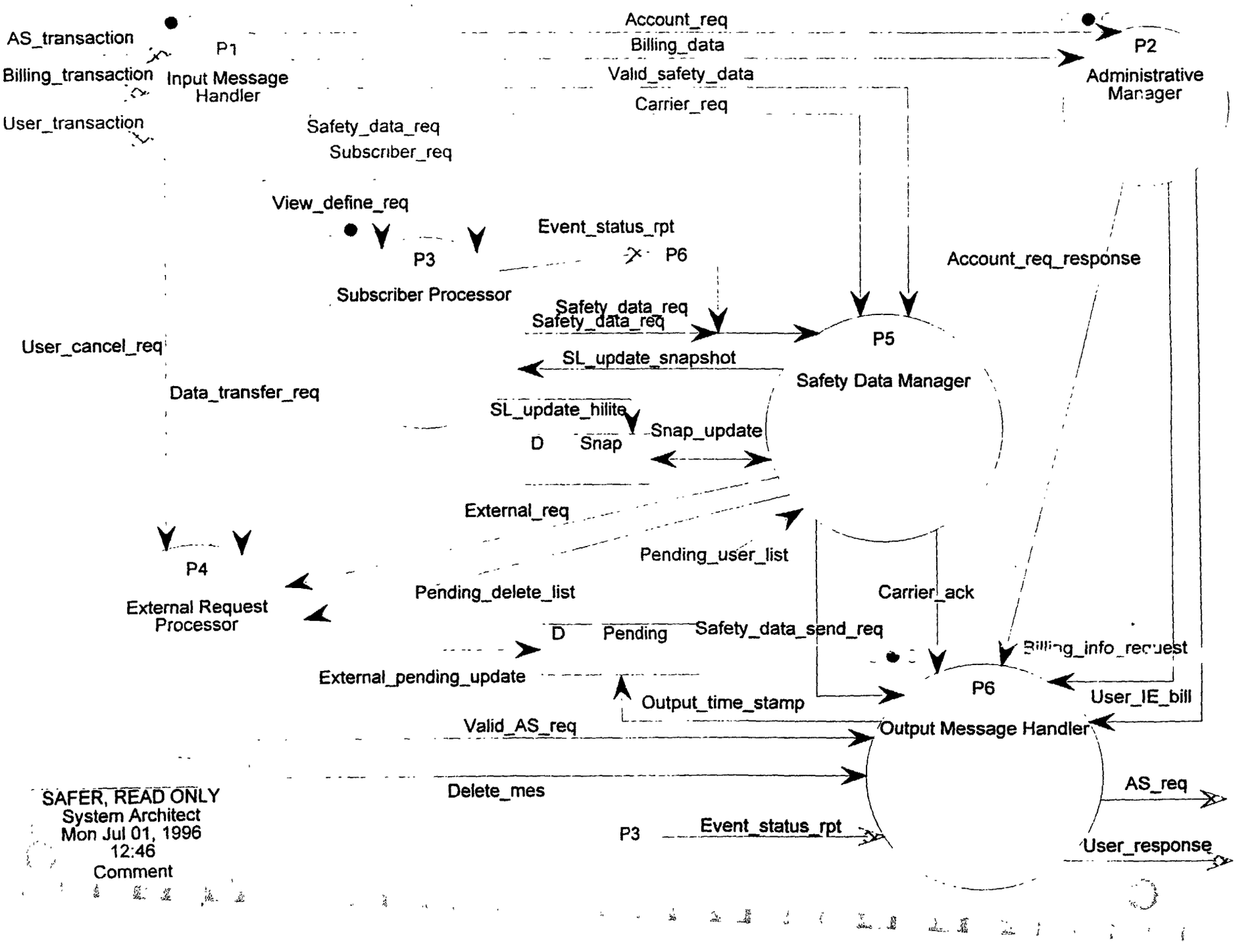
Carrier-location-state

Clear-send-req

This data flow clears the Send-request in Pending to indicate that a request has been issued to P6.

Req-des

Provides the request identification for which request information is needed to generate the Valid-AS-req for P6.


AS_transaction
Billing_transaction
User_transaction
P1
Input Message Handler
Account_req
Billing_data
Valid_safety_data
Carrier_req
P2
Administrative Manager
Safety_data_req
Subscriber_req
View_define_req
P3
Subscriber Processor
Event_status_rpt
P6
Account_req_response
Safety_data_req
Safety_data_req
User_cancel_req
P5
SL_update_snapshot
Safety Data Manager
Data_transfer_req
SL_update_hilite
D
Snap
Snap_update
External_req
Pending_user_list
P4
External Request Processor
Pending_delete_list
Carrier_ack
D
Pending
Safety_data_send_req
Billing_info_request
External_pending_update
P6
User_IE_bill
Output_time_stamp
Valid_AS_req
Output Message Handler
Delete_mes
AS_req
P3
Event_status_rpt
User_response
SAFER, READ ONLY
System Architect
Mon Jul 01, 1996
12:46
Comment

5.0 Safety Data Manager

The Safety Data Manager (P5) is responsible for satisfying all Safety-data-reqs for snapshots and profiles. Snapshots are always found in the Snap data store since SAFER is the authoritative source for snapshots. Profiles might be found in the Profile data store but most likely will result in the generation of an External-req (AS-profile-req) for the data to the authoritative source. The Profile data store is a temporary cache containing profiles that were received because a snapshot changed and a user requested a profile. Profiles are kept in the cache for the remainder of the day to satisfy other users interested in the same data. Snapshots and cache Profiles are returned to the user by a Safety-data-send-req.

The Safety Data Manager generates a new USDOT number for 'ADD' type Carrier-reqs, makes a temporary entry in the Snap data store and returns the USDOT number to the user in a Carrier-ack. 'ADD', 'UPDATE' and 'DELETE' type Carrier-reqs receive base state lookup processing and are forwarded to the authoritative source in an External-req.

Valid-safety-data received by the Safety Data Manager is checked for valid source and added to either the Snap or Profile data store as required. The Carrier-unique-ids for "updated" snapshots are loaded into an SL-update-list, time stamped and sent to P3. The Carrier-unique-ids of profiles are checked against the Pending data store to determine if users are waiting for them. P5 issues Safety-data-send-reqs to return profiles to the waiting users. The Carrier-unique-id, Req-user-id combinations for all refreshed snapshots and any profiles are loaded into a Pending-delete-list and issued to P4. This information is used for Pending data store maintenance.

Periodically the Safety Data Manager issues an External-req (Snap-refresh-req) to P4 to refresh all snapshots with create dates greater than x days old.

Input: T **Output: F**

Carrier-req
Request to add, update or delete carrier information.

Pending-user-list
List of users and related information that are waiting for a particular profile.

Safety-data-req
A request for snapshot or profile data.

Valid-safety-data
Incoming safety data(damended, snapshot or profile) that has been translated, checked for access authorization and has undergone application-level edit checks.

Input: F **Output: T**

Carrier-ack
Acknowledgement for an 'Add' type carrier request(includes the new DOT number)

External-req

A request to an authoritative source for carrier data processing, request deletes, snapshot or profile data.

Pending-delete-list

A list of snapshot or profile Carrier-unique-id, Req-user-id combinations that is sent to P4. The information is used for Pending data store maintenance.

SL-update-snapshot

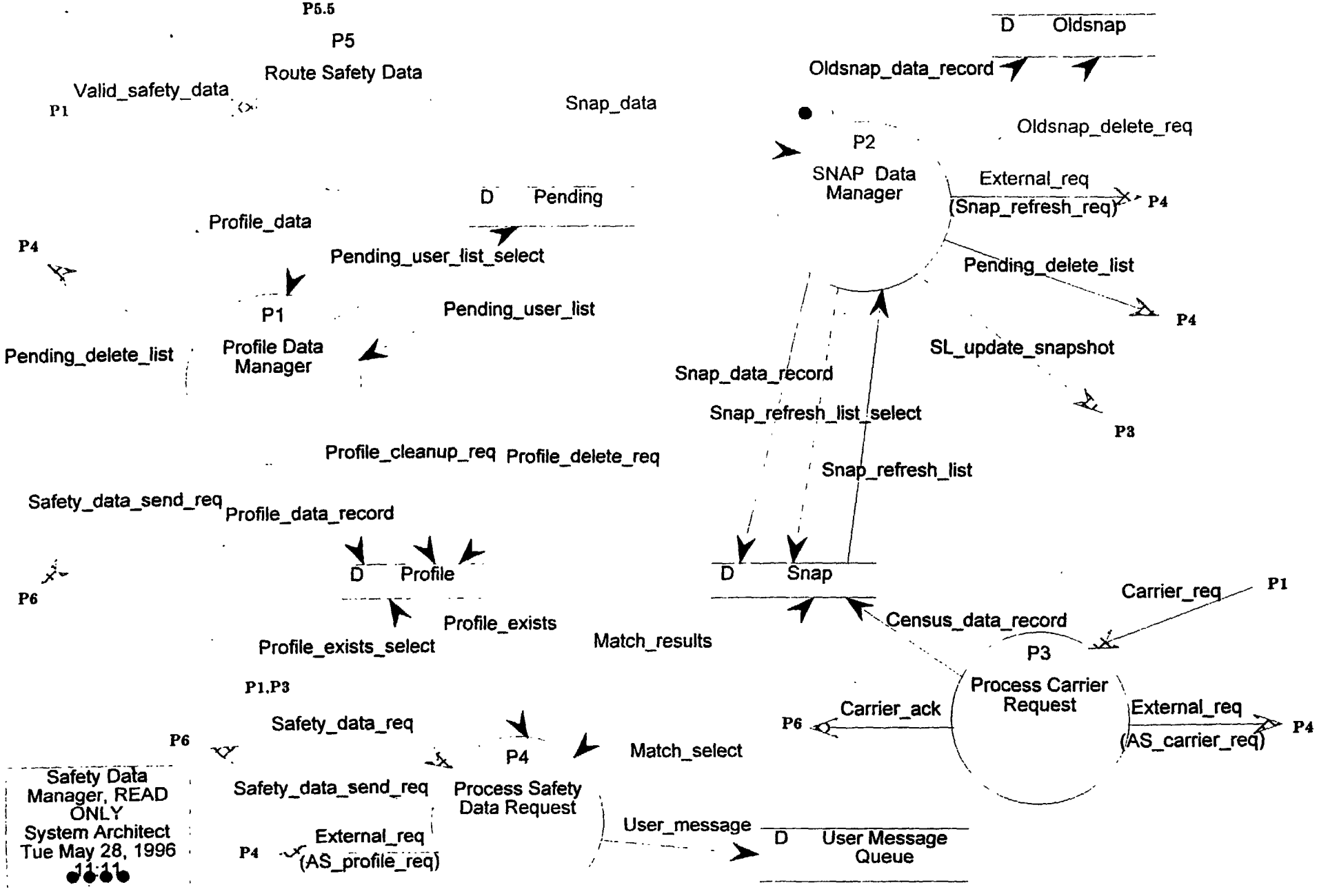
This is the dataflow which P3 receives from P5. It includes a new snapshot, and may include an old snapshot.

Safety-data-send-req

Request to send amended, snapshot, profile or census information to a user.

Input: T **Output: T**

Snap-update
TBD


P5.5
P5
Route Safety Data
Valid_safety_data
P1
Snap_data
D Oldsnap
Oldsnap_data_record
Oldsnap_delete_req
P2
SNAP Data Manager
External_req
(Snap_refresh_req)
P4
D Pending
Profile_data
P4
Pending_user_list_select
Pending_delete_list
P4
P1
Pending_user_list
Pending_delete_list
Profile Data Manager
SL_update_snapshot
Snap_data_record
P3
Snap_refresh_list_select
Profile_cleanup_req
Profile_delete_req
Snap_refresh_list
Safety_data_send_req
Profile_data_record
P6
D Profile
D Snap
Carrier_req
P1
Census_data_record
Profile_exists
Profile_exists_select
Match_results
P3
Process Carrier Request
P1,P3
Carrier_ack
P6
External_req
(AS_carrier_req)
P4
Safety_data_req
P6
P4
Match_select
Safety Data Manager, READ ONLY System Architect Tue May 28, 1996 11:11
Safety_data_send_req
Process Safety Data Request
User_message
External_req
(AS_profile_req)
P4
D User Message Queue

5.1 Profile Data Manager

The Profile Data Manager (P5.1) receives Profile-data from P5.5. It extracts the Carrier-unique-id for each Profile-data-record in the Profile-data and checks that the authoritative source sending the data is the valid source for the Carrier-unique-id. If the source is invalid, a User-message is issued, otherwise P5.1 inserts the Profile-data-record into the Profile data store. It issues a Pending-user-list-select to the Pending data store for each Carrier-unique-id and receives a Pending-user-list containing the Req-user-ids of all users waiting for the profile. Duplicate requests for the same profile by the same user are ignored. It then issues a Safety-data-sendreq to send the profile to the waiting users. The Carrier-unique-id, Req-user-id combinations are loaded into a Pending-delete-list and issued to P4. P4 uses this information for Pending data store maintenance. Periodically P5.1.2 issues a Profile-cleanup-req to delete all entries from the Profile data store that are greater than x hours old and are not waiting to be sent to a user.

Input: T **Output: F**

Pending-user-list
List of users and related information that are waiting for a particular profile.

Profile-data
One or more profiles

Input: F **Output: T**

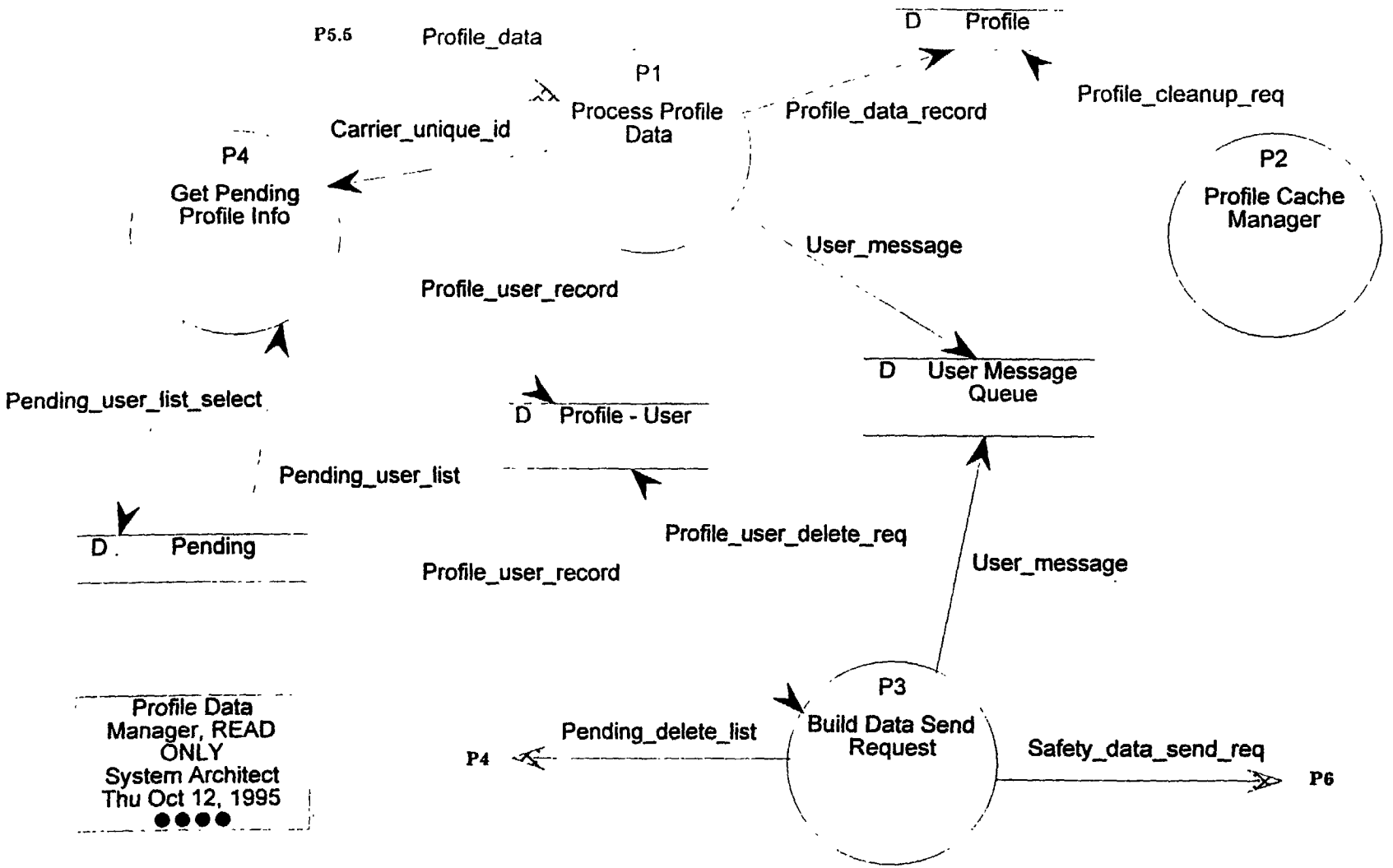
Pending-delete-list
A list of snapshot or profile Carrier-unique-id, Req-user-id combinations that is sent to P4. The information is used for Pending data store maintenance.

Pending-user-list-select
Select request to the Pending data store to obtain a list of users waiting for a particular profile.

Profile-cleanup-req
Request to delete all profiles with create date greater than x hours old and that are not waiting to be sent to a user

Profile-data-record
All the data item values contained in a profile..

Safety-data-send-req
Request to send amended, snapshot, profile or census information to a user.


P5.5
Profile_data
P1
Process Profile Data
D Profile
Profile_cleanup_req
Profile_data_record
Carrier_unique_id
P4
Get Pending Profile Info
P2
Profile Cache Manager
User_message
Profile_user_record
D User Message Queue
Pending_user_list_select
D Profile - User
Pending_user_list
Profile_user_delete_req
D Pending
Profile_user_record
User_message
Profile Data Manager, READ ONLY System Architect Thu Oct 12, 1995
P3
Build Data Send Request
Pending_delete_list
P4
Safety_data_send_req
P6

5.1.1 Process Profile Data

Process Profile Data(P5.1.1) receives Profile-data from P5.5. It extracts the Carrier-unique-id for each Profile-data-record in the Profile-data. The Carrier-unique-id is used to determine the Carrier location-state from the Census area of the Snap data store. The Carrier-location-state together with the Valid-safety-data-type are used to determine the Account-id from the Base State Routing data store. If the Account-ids from Base State Routing and the Profile-data Data-header do not match, a User-message is issued to the Authoritative Source. Otherwise, P5.1.1 sends the Carrier-unique-id to P5.1.4. and inserts the Profile-data-record into the Profile data store.

Input: T **Output: F**

Profile-data
One or more profiles

Input: F **Output: T**

Carrier-unique-id
A unique ID number for a carrier, presently the combination of the USDOT number. issuing authority and the terminal ID.

Profile-data-record
All the data item values contained in a profile.

User-message
TBD

5.1.2 Profile Cache Manager

Once a day the Profile Cache Manager issues a Profile-cleanup-req to delete all profiles from the Profile data store that are greater than x hours old and are not waiting to be sent to a user.

Input: F **Output: T**

Profile-cleanup-req

Request to delete all profiles with create date greater than x hours old and that are not waiting to be sent to a user

5.1.3 Build Data Send Request

Build Data Send Request(P5.1.3) builds Safety-data-send-reqs for all records in the Profile-User data store. It sorts the records in the Profile-User data store on user, overnite flag and media. If a duplicate record(user, overnite flag, media and carrier) is found, the Carrier-unique-id is not added to the Safety-data-send-req and the user is notified (User-message) that the duplicate request has been ignored. It then issues one Safety-data-send-req for each unique combination of user, overnite flag and media. Each Safety-data-send-req contains a list of all Carrier-unique-ids(no duplicates) with the same overnite flag and media that a user has requested. Records are deleted from the Profile-User data store by a Profile-user-delete-req. All Req-user-id, Carrier-unique-id combinations loaded into Safety-data-send-reqs are also loaded into a Pending-delete-list and issued to P4 for Pending data store maintenance.

Input: T **Output: F**

Profile-user-record

User request information associated with a user waiting for a profile.

Input: F **Output: T**

Pending-delete-list

A list of snapshot or profile Carrier-unique-id, Req-user-id combinations that is sent to P4. The information is used for Pending data store maintenance.

Profile-user-delete-req

Request to delete a record from the Profile-User data store for a particular user, profile combination.

Safety-data-send-req

Request to send amended, snapshot, profile or census information to a user.

User-message

TBD

5.1.4 Get Pending Profile Info

Get Pending Profile Info(P5.1.4) receives the Carrier-unique-id for a profile from P5.1.1. It issues a Pending-user-list-select and receives a Pending-user-list from the Pending data store that contains user information (req-header) and original request information (Packet-header) for all users waiting for this profile. The user (user, overnite flag and media) and request information are merged with the Carrier-unique-id to form a Profile-user-record which is inserted into the Profile-User data store.

Input: T **Output: F**

Carrier-unique-id

A unique ID number for a carrier, presently the combination of the USDOT number, issuing authority and the terminal ID.

Pending-user-list

List of users and related information that are waiting for a particular profile.

Input: F **Output: T**

Pending-user-list-select

Select request to the Pending data store to obtain a list of users waiting for a particular profile.

Profile-user-record

User request information associated with a user waiting for a profile.

5.2 SNAP Data Manager

The SNAP Data Manager (P5.2) receives (Snap-data) from P5.5. It extracts the Carrier-unique-id for each Snap-data-record in the Snap-data. It checks that the authoritative source sending the data is the valid source for the Carrier-unique-id. If the source is invalid, a User-message is issued. Otherwise, the old snapshot is moved from the Snap data store to the Oldsnap data store. Next, it inserts the new Snap-data-record into the Snap data store. Receipt of a new snapshot indicates that some or all of a carrier's census or safety information. Therefore, any existing profile will be deleted unless it is waiting to be sent to a user. If Reason-for-send is UPDATE, the Carrier-unique-ids of al records in Snap-data are loaded into an SL-update-list, time stamped and sent to the Subscriber Processor (P3). If Reason-for-send is REFRESH, the Carrier-unique-ids of all records in Snap-data are loaded into a Pending-delete-list and send to P4. Periodically the SNAP Data Manager issues a Snap-refresh-list-select to the Snap data store and receives a Snap-refresh-list containing the Carrier-unique-ids of all snapshots that are greater than x days old. A Snap-refresh-req is issued to P4 to obtain a "refreshed" snapshot from the authoritative source for each Carrier-unique-id in the Snap-refresh-list. P5.2.2 writes a copy of the Snap-refresh-req to the Internal Transaction Log.

Input: T **Output: F**

Snap-data
One or more snapshots.

Snap-refresh-list
A list of carrier IDS and base states for snapshots greater than x days old

Input: F **Output: T**

External-req
A request to an authoritative source for carrier data processing, request deletes, snapshot or profile data.

Oldsnap-data-record
An old snapshot that has been replaced by an updated snapshot.

Oldsnap-delete-req
A request to delete a snapshot in the Oldsnap data store.

Pending-delete-list
A list of snapshot or profile Carrier-unique-id, Req-user-id combinations that is sent to P4. The information is used for Pending data store maintenance.

Profile-deletereq
Request to delete a profile for a particular carrier from the Profile data store.

SL-update-snapshot

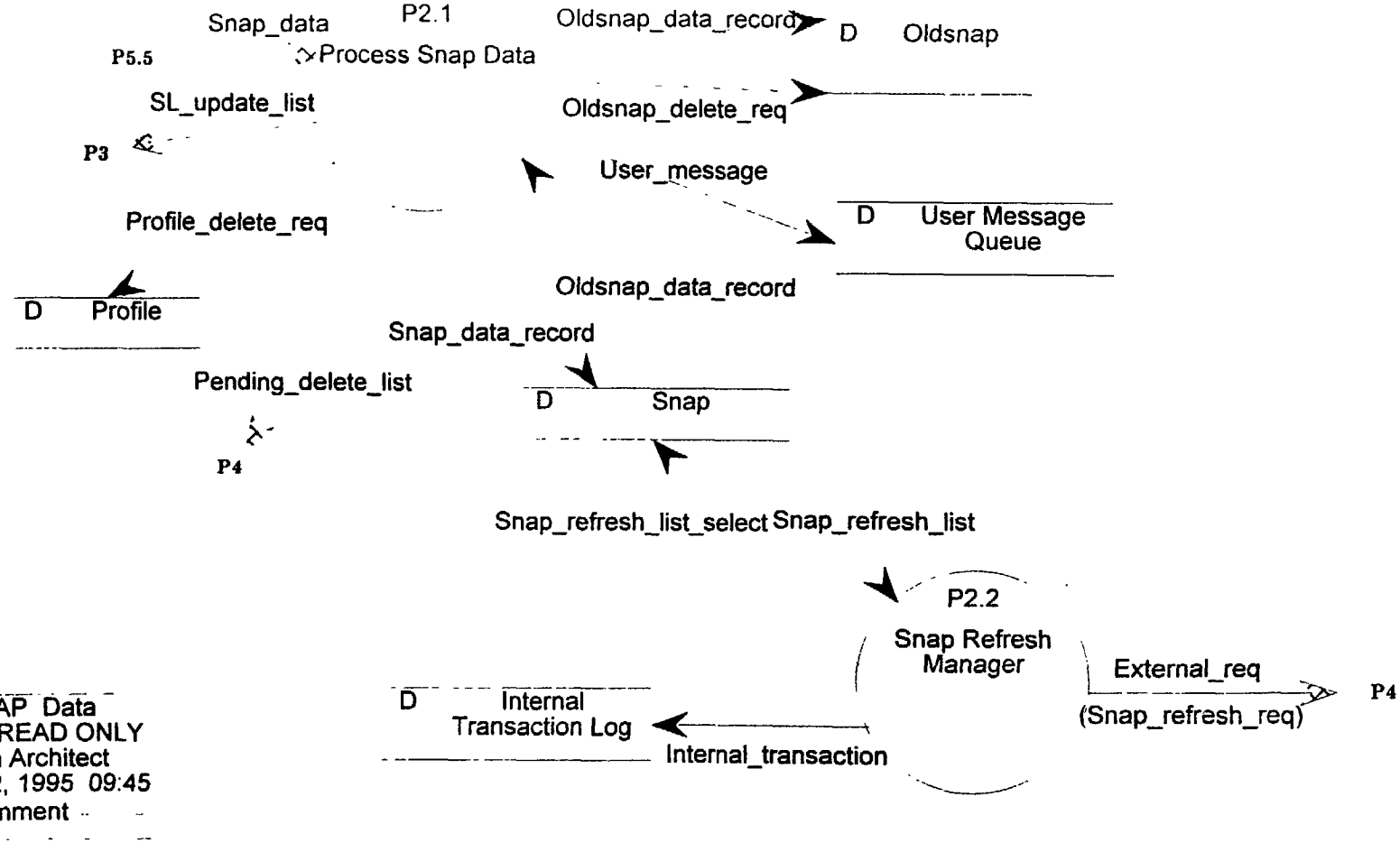
This is the dataflow which P3 receives from P5. It includes a new snapshot, and may include an old snapshot.

Snap-data-record

All fields contained in a snapshot. This includes the Census and Safety Data sections.

Snap-refresh-list-select

Select request to obtain all carrier IDS with base states for snapshots that have a create date greater than x days old.


Snap_data
P2.1
Oldsnap_data_record
D Oldsnap
P5.5
Process Snap Data
SL_update_list
Oldsnap_delete_req
P3
User_message
Profile_delete_req
D User Message Queue
Oldsnap_data_record
D Profile
Snap_data_record
Pending_delete_list
D Snap
P4
Snap_refresh_list_select
Snap_refresh_list
P2.2
Snap Refresh Manager
External_req
P4
(Snap_refresh_req)
D Internal Transaction Log
Internal_transaction
2. SNAP Data Manager, READ ONLY
System Architect
Thu Oct 12, 1995 09:45
Comment

5.2.1 Process Snap Data

Process Snap Data (P5.2.1) receives Snap-data from P5.5. It extracts the Carrier-unique-id for each snapshot in the Snap-data. The Carrier-unique-id is used to determine the Carrier-location-state from the Census area of the Snap data store. The Carrier-location-state together with the Valid-safety-data-type are used to determine the Account-id from the Base State Routing data store. If the Account-ids from Base State Routing and the Snap-data Data-header do not match, a User-message is issued to the Authoritative Source and processing moves to the next snapshot. Otherwise, any existing snapshot for the carrier is deleted from the Oldsnap data store. The existing snapshot for the carrier is copied from Snap to Oldsnap and then deleted from the Snap data store. The new snapshot is then loaded into the Snap data store. Receipt of a new snapshot indicates that the carrier information has changed so any existing profile should be deleted (Profile-delete-req) from the profile cache unless it is waiting to be sent to a user. If the Reason-for-send from the Data-header is REFRESH, the Req-user-id (always SAFER for snapshots) together with the Carrier-unique-id of the new snapshot is loaded into a Pending-delete-list. If the Reason-for-send is UPDATE, the Carrier-unique-id is loaded into an SL-update-list. After all snapshots in the Snap-data have been processed, a Pending-delete-list is issued to P4 if Reason-for-send is REFRESH. If Reason-for-send is UPDATE, the SL-update-list is time stamped and issued to P3.

Input: T **Output: F**

Oldsnap-data-record
An old snapshot that has been replaced by an updated snapshot.

Snap-data
One or more snapshots.

Input: F **Output: T**

Oldsnap-data-record
An old snapshot that has been replaced by an updated snapshot.

Oldsnap-delete-req
A request to delete a snapshot in the Oldsnap data store.

Pending-delete-list
A list of snapshot or profile Carrier-unique-id, Req-user-id combinations that is sent to P4. The information is used for Pending data store maintenance.

Profile-delete-req
Request to delete a profile for a particular carrier from the Profile data store.

SL-update-list
This is the dataflow which P3 receives from P5. It identifies new snapshots which have been received and placed into the Snap data store. It consists of a timestamp and

a set of carrier ID's.

Snap-data-record
All fields contained in a snapshot. This includes the Census and Safety Data sections.

User-message
TBD

5.2.2 Snap Refresh Manager

Periodically the Snap Refresh Manager (P5.2.2) issues a Snap-refresh-list-select to the Snap data store and receives a Snap-refresh-list containing the Carrier-unique-ids of all snapshots that are greater than x days old. It then issues Snap-refresh-reqs to P4 to request updated snapshots from the authoritative source. A copy of each Snap-refresh-req is Ioaded into an Internal-transaction and written to the Internal Transaction Log.

Input: T **Output: F**

Snap-refresh-list

A list of carrier IDS and base states for snapshots greater than x days old

Input: F **Output: T**

External-req

A request to an authoritative source for carrier data processing, request deletes, snapshot or profile data.

Internal-transaction

A copy of an SAFER generated transaction.

Snap-refresh-list-select

Select request to obtain all carrier IDS with base states for snapshots that have a create date greater than x days old.

5.3 Process Carrier Request

Process Carrier Request (P5.3) is responsible for generating new Carrier-unique-ids (USDOT numbers) and forwarding 'ADD', 'UPDATE' and 'DELETE' type Carrierreqs to the authoritative source. If an 'ADD' request is received, the Census area of the Snap data store is checked for duplicate entries on the Carrier-legal-name and DBA-name fields. If no duplicates are found, a new Carrier-unique-id is generated. The new Carrier-unique-id is returned to the user in a Carrier-ack. A Census-data-record is inserted into the Census area of the Snap data store to add the new Carrier unique-id and Carrier-info. This information has a status of pending until a new snapshot is received from the authoritative source. All Carrier-reqs receive base state lookup processing and are forwarded to the authoritative source as External-reqs.

Input: T **Output: F**

Carrier-req

Request to add, update or delete carrier information.

Input: F **Output: T**

Carrier-ack

Acknowledgement for an 'Add' type carrier request(includes the new DOT number)

Census-data-record

TBD

External-req

A request to an authoritative source for carrier data processing, request deletes, snapshot or profile data.

5.4 Process Safety Data Request

Process Safety Data Request (P5.4) receives Safety-data-reqs for snapshots and profiles from Pl and P3. High priority processing is limited to snapshot requests. Snapshot requests for a USDOT number (one or more) match, are returned immediately by issuing a Safety-data-send-req to P6. For a profile request, all the requested profiles might not be available in the Profile cache. In this case, AS-profile-reqs will be built and sent to the authoritative source(s). When the data is received, it will be sent to the user. If the request is for a Value (selected CENSUS fields other than USDOT number) match and one or more carriers are found, the user has the option to receive either Snapshot or Census information. Because the number of carriers returned by a Value match could be large, the user can request Cost-check processing. If requested, the number of carriers and the approximate cost of sending the data, rather than the data itself, will be returned to the user in a User-message. If no carriers were found for a Value match and the user requested Fuzzy match processing (selected Census fields only), the Census data will be queried for fuzzy matches. Fuzzy match processing is limited to low and medium priority non USDOT number requests. Only snapshot or census data (user choice) can be returned for a fuzzy match. If Fuzzy match processing was not selected and no carriers were found, a User-message indicating 'no carriers found' is issued to the user. In certain cases, redaction (blanking) of certain fields may be done if a user is not authorized to receive certain data.

Input: T **Output: F**

Match-results

The results of either an exact match query or a fuzzy match query to the Census area of the Snap data store

Profile-exists

Flag indicating existence or non-existence of a particular profile in the Profile data store

Safety-data-req

A request for snapshot or profile data.

Input: F **Output: T**

External-req

A request to an authoritative source for carrier data processing, request deletes, snapshot or profile data.

Match-select

A select request for either an exact match (ID or VALUE) query or a fuzzy match query to the Census area of the Snap data store.

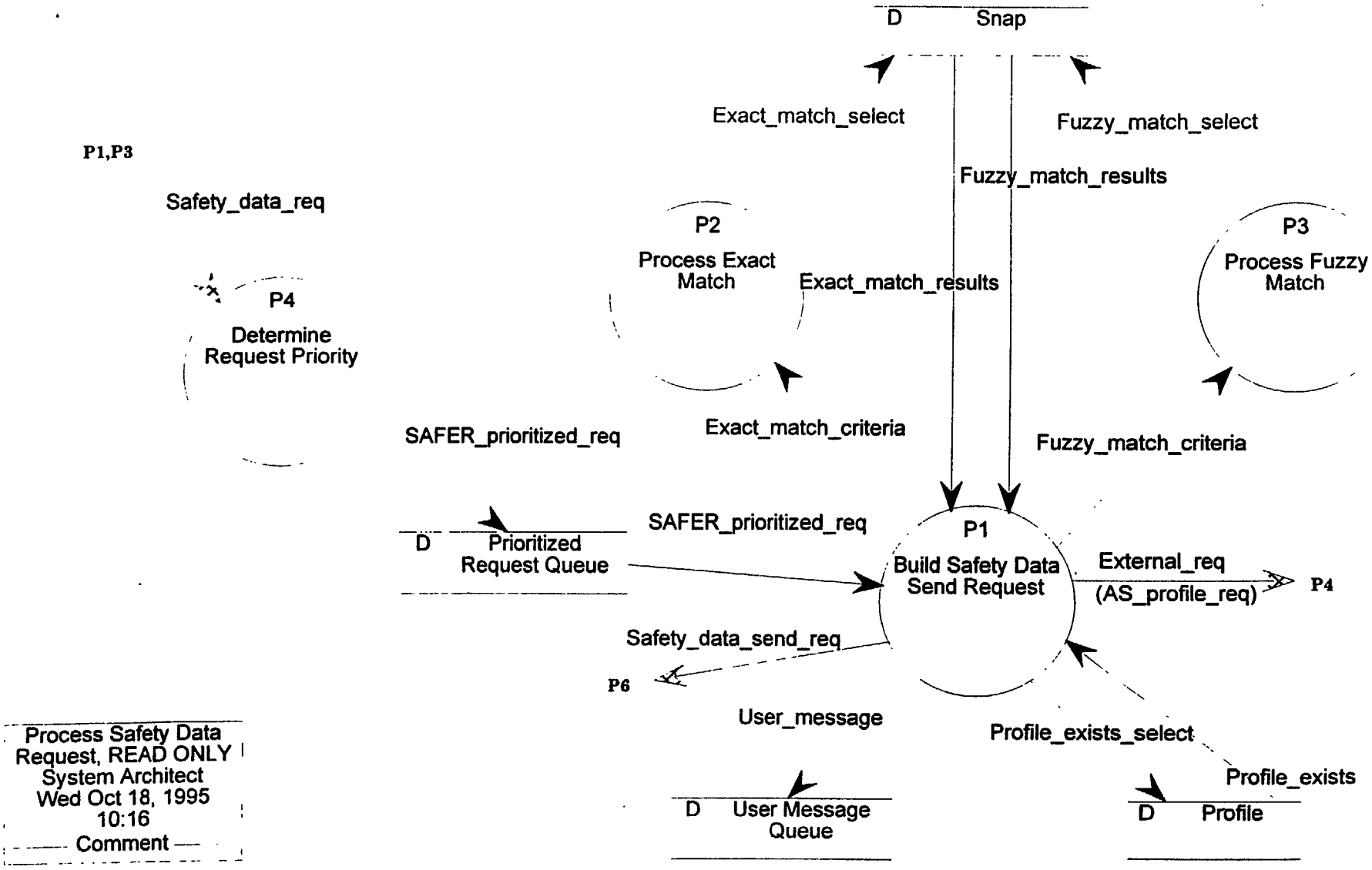
Profile-exists-select

Select request to determine if a particular profile exists in the Profile data store.

Safety-data-send-req

Request to send amended, snapshot, profile or census information to a user.

User-message
TBD


D Snap
Exact_match_select
Fuzzy_match_select
P1,P3
Safety_data_req
Fuzzy_match_results
P2
Process Exact Match
Exact_match_results
P3
Process Fuzzy Match
P4
Determine Request Priority
SAFER_prioritized_req
Exact_match_criteria
Fuzzy_match_criteria
D Prioritized Request Queue
SAFER_prioritized_req
P1
Build Safety Data Send Request
External_req
(AS_profile_req)
P4
Safety_data_send_req
P6
User_message
Profile_exists_select
Profile_exists
D User Message Queue
D Profile
Process Safety Data Request, READ ONLY
System Architect
Wed Oct 18, 1995
10:16
Comment

5.4.1 Build Safety Data Send Request

Build Safety Data Request (P5.4.1) receives Safety-data-reqs for snapshots or profiles from P1 and P3. High priority processing is limited to snapshot requests. Snapshot requests for a USDOT number (one or more) match are returned immediately by issuing a Safety-data-send-req to P6. Profile request are handled by a combination of Safety-data-send-req for profiles found in cache and AS-profile-req for profiles not found in cache. A User-message indicating the total number of profiles (cache and non-cache) found for the query is tagged with the User-transjd of the Safety-data-req and returned to the user. In general, all profiles satisfying a given query will not be returned to the user in a single transmission. The profile count will let the user know how many profiles to expect. If the request is for a Value(selected Census fields other than USDOT number) match and one or more carriers are found, the user has the option to receive either Snapshot or Census information. Because the number of carriers returned by a Value match could be a large number, the user can request Cost-check processing. In this case, the number of carriers and the approximate cost of sending the data rather than the data itself, will be returned to the user in a User-message. If no carriers were found for a Value match and the user requested Fuzzy match processing(selected Census fields only), the Census data will be queried for Fuzzy matches. Fuzzy match processing is limited to low and medium priority non USDOT number requests. Only snapshot or census data(user choice) can be returned for a fuzzy match. If Fuzzy match processing was not selected and no carriers were found, a User-message indicating 'no carriers found' is sent to the user.

Input: T **Output: F**

Exact-match-results

Results of an exact match query on the Census area of the Snap data store.

Fuzzy-match-results

Results of a fuzzy match query on the Census data store

Profile-exists

Flag indicating existence or non-existence of a particular profile in the Profile data store

SAFER-prioritized-req

Safety-data-req with a SAFER determined priority based on user input parameters.

Input: F **Output: T**

Exact-match-criteria

The query number and associated parameter values for an exact match..

External-req

A request to an authoritative source for carrier data processing, request deletes, snapshot or profile data.

Fuzzy-match-criteria
The query number, fuzzy limit and associated parameter values for a fuzzy match.

Profile-exists-select
Select request to determine if a particular profile exists in the Profile data store.

Safety-data-send-req
Request to send amended, snapshot, profile or census information to a user.

User-message
TBD

5.4.2 Process Exact Match

Process Exact Match (P5.4.2) receives the Exact-match-criteria from P5.4.1 The Exact-match-criteria contains the Query-number (pick list number selected by the user) and the Query-parms. For a query corresponding to a USDOT Number select, the Query-parms will contain one or more USDOT numbers. For queries other than a USDOT Number select, Query-parms will contain all the parameters required for the selected query. If, for example, the user selected a query based on Carrier DBA Name and State, the Query-parms might contain 'Monback Trucking' and 'MD'. The query is formulated into an Exact-match-select and issued to the Census area of the Snap data store.

Input: T **Output: F**

Exact-match-criteria

The query number and associated parameter values for an exact match..

Input: F **Output: T**

Exact-match-select

A query issued to the Census area of the Snap data store for an exact match on selected data item values.

5.4.3 Process Fuzzy Match

Process Fuzzy Match (P5.4.3) receives the Fuzzy-match-criteria from P5.4.1. Fuzzy match processing occurs only if exact match processing failed and the user specifically requested a fuzzy match. Only selected data fields(tbd) are available for fuzzy match processing. A Fuzzy-match-select, based on the contents of the Fuzzy-match-criteria, is formulated and issued to the Census area of the Snap data store.

Input: T **Output: F**

Fuzzy-match-criteria

The query number, fuzzy limit and associated parameter values for a fuzzy match.

Input: F **Output: T**

Fuzzy-match-select

A query issued to the Census area of the Snap data store for a fuzzy match on selected data item values. The number of rows returned is limited by the Fuzzy-limit parameter.

5.4.4 Determine Request Priority

Determine Priority (P5.4.4) receives Safety-data-reqs for snapshots or profiles from Pl and P3. It determines the SAFER-priority based on the values for Req-overnite-flag, SDR-type, Query-number, Fuzzy-select and Req-media-type. It then adds the SAFER-priority to the request and puts it into the Safety Data Request Queue.

Input: T **Output: F**

Safety-data-req
A request for snapshot or profile data.

Input: F **Output: T**

SAFER-prioritized-req
Safety-data-req with a SAFER determined priority based on user input parameters.

5.5 Route Safety Data

The Data Router (P5.5) receives Valid-safety-data from PI and depending on the Valid-safety-data type, routes Snap-data to P5.2 or Profile-data to P5.1.

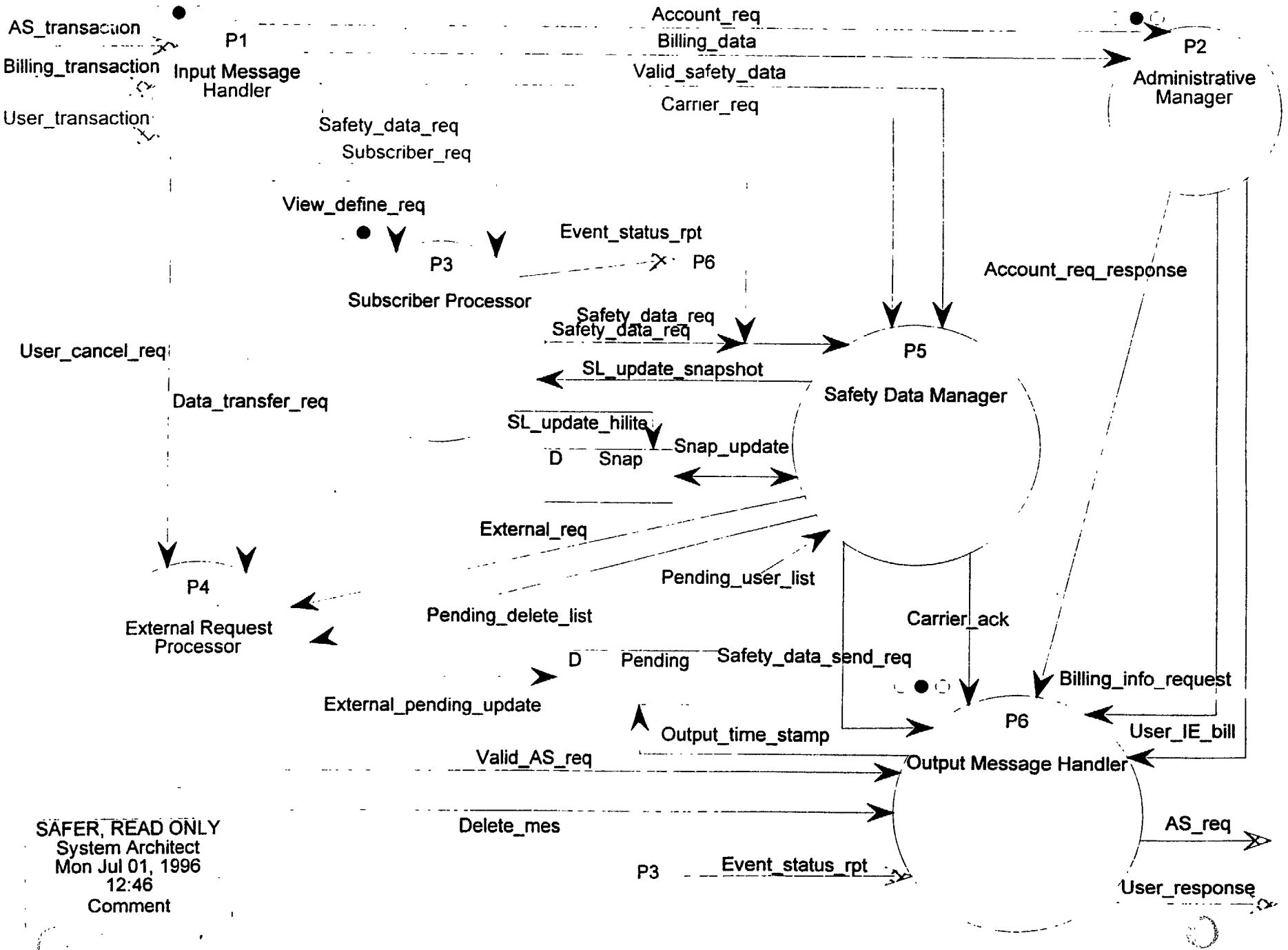
Input: T **Output: F**

Valid-safety-data
Incoming safety data(damended, snapshot or profile) that has been translated, checked for access authorization and has undergone application-level edit checks.

Input: F **Output: T**

Profile-data
One or more profiles

Snap-data
One or more snapshots.


AS_transaction
Billing_transaction
User_transaction
P1
Input Message Handler
Account_req
Billing_data
Valid_safety_data
Carrier_req
P2
Administrative Manager
Safety_data_req
Subscriber_req
View_define_req
Event_status_rpt
P3
P6
Subscriber Processor
Account_req_response
Safety_data_req
Safety_data_req
User_cancel_req
P5
SL_update_snapshot
Safety Data Manager
Data_transfer_req
SL_update_hilite
D
Snap
Snap_update
External_req
Pending_user_list
P4
External Request Processor
Pending_delete_list
Carrier_ack
D
Pending
Safety_data_send_req
Billing_info_request
External_pending_update
P6
Output_time_stamp
User_IE_bill
Valid_AS_req
Output Message Handler
Delete_mes
AS_req
P3
Event_status_rpt
User_response
SAFER, READ ONLY
System Architect
Mon Jul 01, 1996
12:46
Comment

6.0 Output Message Handler

The Output Message Handler provides security controls for data between output transaction generation and distribution, performs error checking, verifies data completeness and integrity, generates error reports, and translates internal formatted messages to standard format for transmission. The Output Message Handler also retrieves data.and prepares output according to user request (EDI, file transfer, CD-ROM). Data is distributed and all transmissions are logged. Checks are also made for duplicate low priority requests. If found, the later request is not processed and the user is informed of the action. Requests for cancellation of previous low priority messages are processed.

Input: T **Output: F**

Account-req-response

A response sent back to the user indicating the status of his request

Billing-info-request

A request made to the VAN at some specified billing interval for billing information for each organization with an account on the system.

Carrier-ack

Acknowledgement for an 'Add' type carrier request(includes the new DOT number)

Delete-mes

This data flow provides a message to P6 which identifies that a request has been received to delete a previous request and that P4 has already serviced this delete request. Servicing by P4 indicates that P4 has looked for this request in pending and has deleted it if it was found.

Event-status-rpt

This is a data report sent to the user on request. It either reflects a user's current subscriber status, and/or system event monitoring status.

Safety-data-send-req

Request to send amended, snapshot, profile or census information to a user.

User-IE-bill

The bill sent to each organization each billing period.

Valid-AS-req

Appropriately formatted request for an authoritative source. Includes the authoritative source address and request priority.

Input: F **Output: T**

AS-req

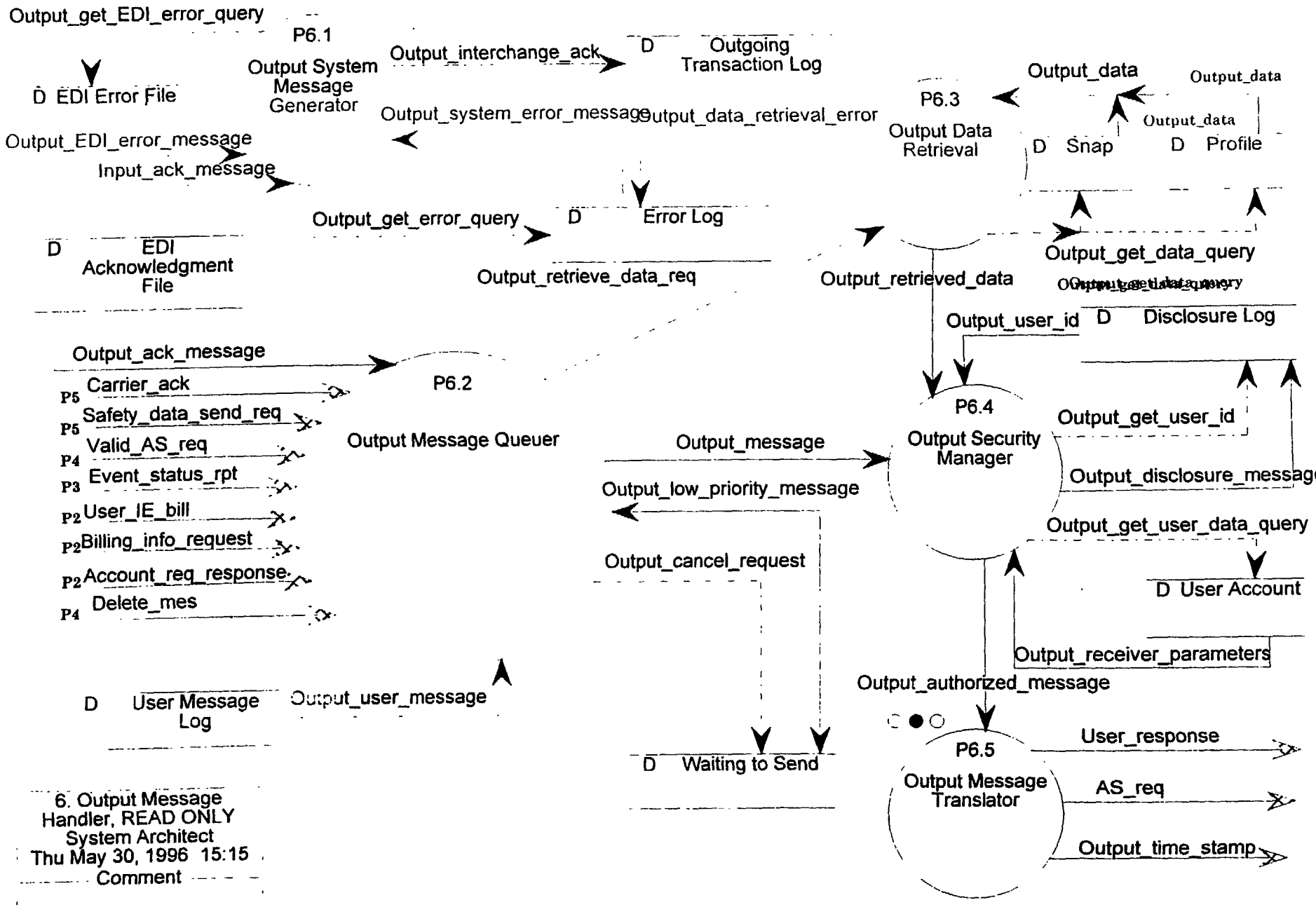
A request for data from an Authoritative Source. This may also route updated carrier information to the appropriate source and send interchange and functional acknowledgments to the Authoritative Source.

Output-time-stamp

Provides the date and time the transaction was sent.

User-response

Response to user request for data (snapshot, profile), account establishment and maintenance, carrier data maintenance, payment for System services, subscriber definition and modification, or forwarded amended data. Also included are interchange and functional acknowledgments.


Output_get_EDI_error_query
P6.1
Output System Message Generator
Output_interchange_ack
D Outgoing Transaction Log
D EDI Error File
Output_system_error_message
Output_data_retrieval_error
P6.3
Output Data Retrieval
Output_data
Output_data
Output_data
D Snap
D Profile
Output_EDI_error_message
Input_ack_message
Output_get_error_query
D Error Log
D EDI Acknowledgment File
Output_retrieve_data_req
Output_retrieved_data
Output_get_data_query
Output_user_id
D Disclosure Log
Output_ack_message
P5 Carrier_ack
P5 Safety_data_send_req
P4 Valid_AS_req
P3 Event_status_rpt
P2 User_IE_bill
P2 Billing_info_request
P2 Account_req_response
P4 Delete_mes
P6.2
Output Message Queuer
Output_message
Output_low_priority_message
Output_cancel_request
P6.4
Output Security Manager
Output_get_user_id
Output_disclosure_message
Output_get_user_data_query
D User Account
Output_receiver_parameters
Output_authorized_message
D User Message Log
Output_user_message
D Waiting to Send
P6.5
Output Message Translator
User_response
AS_req
Output_time_stamp
6. Output Message Handler, READ ONLY
System Architect
Thu May 30, 1996 15:15
Comment

6.1 Output System Message Generator

The Output System Message Generator handles all error messages generated within SAFER, and incoming interchange and functional acknowledgments.

EDI acknowledgments, Input-ack-message, are either interchange or functional acknowledgments. Acknowledgements not reporting errors are used to update the status of transmitted transactions in the Transmission Log through Output-interchange-ack. An error report is generated for acknowledgments reporting errors. Requests for errors generated during output translation are made through Output-get-EDI-error-query and received via Output-EDI-error-message. These errors may be resolved by a SAFER operator or referred to the system manager.

For SAFER messages, the process accesses the Error Log via Output-get-error-query, to respond to user queries and for report or statistics generation. The requested data is returned through Output-system-error-message.

Input: T **Output: F**

Input-ack-message
Input interchange and functional acknowledgments received by SAFER. These may indicate error(s) with the transmission sent by SAFER.

Output-EDI-error-message
Contains the invalid transaction and any error messages.

Output-system-error-message
A message generated by a SAFER process during the processing of a transaction.

Input: F **Output: T**

Output-get-EDI-error-query
Contains the fields necessary to retrieve an error message from the ED1 Error File.

Output-get-error-query
Contains the fields necessary to retrieve an error message from the Error Log.

Output-interchange-ack
If an Interchange Acknowledgment was requested by SAFER, this data flow contains the acknowledgment status.

6.2 Output Message Queuer

The Output Message Queuer retrieves the input message (Output-ack-message, Carrier-ack, Safety data-sendreq, Valid-AS-req, Event-status-rpt, User-IE-bill, Billing-info-request, Account-req-response, Delete-mes, Output-user-message) with the highest priority, based on transaction type, and/or the oldest date and time. Messages are also retrieved from the ED1 Acknowledgment Log via Output-ack-message and from the User Message Log via Output-user-message. Checks are made for the proper time to process the requests in the Waiting to Send data store. If time, the requests are retrieved via Output-low-priority-message.

If the message requires data retrieval, a request, Output-retrieve-data-req, is sent to the Output Message Retrieval process, otherwise the message is passed on to the Output Security Manager, via Output-message.

Low priority requests are checked for cancellations and duplicates. For incoming cancellation requests the process checks the Waiting to Send data store for the request through Output-cancel-request. If found, the request is deleted and the user informed of the action. Low priority messages are checked for duplicates in the Waiting to Send data store. If the new message is a duplicate, a message is generated for the user indicating the later request will be ignored. If the message is not a duplicate and it is not the proper time to process the message, it is written to the Waiting to Send data store via Output-low-priority-message.

Input: T **Output: F**

Account-req-response
A response sent back to the user indicating the status of his request

Billing-info-request
A request made to the VAN at some specified billing interval for billing information for each organization with an account on the system.

Carrier-ack
Acknowledgement for an 'Add' type carrier request(includes the new DOT number)

Delete-mes
This data flow provides a message to P6 which identifies that a request has been received to delete a previous request and that P4 has already serviced this delete request. Servicing by P4 indicates that P4 has looked for this request in pending and has deleted it if it was found.

Event-status-rpt
This is a data report sent to the user on request. It either reflects a user's current subscriber status, and/or system event monitoring status.

Output-ack-message
Interchange and functional acknowledgments generated by SAFER. These may indicate error(s) with the transmission received by SAFER.

Output-user-message
An informational message to the user. This may indicate application-level errors found with a request, it may provide a cost estimate, or it may indicate duplicate requests were found or cancellation requests were processed.

Safety-data-send-req
Request to send amended, snapshot, profile or census information to a user.

User-IE-bill
The bill sent to each organization each billing period.

Valid-AS-req
Appropriately formatted request for an authoritative source. Includes the authoritative source address and request priority.

Input: F **Output: T**

Output-cancel-request
The data needed to delete a low priority request.

Output-message
A message, in internal format, ready for security checks and translation.

Output-retrieve-data-req
A request to retrieve snapshot, census, or profile data for output to requestor.

Input: T **Output: T**

Output-low-priority-message
Contains the data fields to store or retrieve a low priority message.

6.3 Output Data Retrieval

The Output Data Retrieval process receives a retrieval request (snapshot, profile, other) through Output-retrieve-data-req, and issues the database query through Output-get-data-query to retrieve the Iatest copy of a profile, snapshot or other data. The process accepts the retrieved data, Output-data, and structures that data for output message translation. The process returns the data, Output-retrieved-data, to the Output Message Security Manager.

If database access errors occurred, a message, Output-data-retrieval-error, is stored in the Error Log for later processing.

Input: T **Output: F**

Output-data

The data retrieved for a snapshot, profile or census data retrieval request. This may be a message indicating an error occurred in data retrieval.

Output-retrieve-data-req

A request to retrieve snapshot, census, or profile data for output to requestor.

Input: F **Output: T**

Output-data-retrieval-error

A message indicating an error occurred in data retrieval.

Output-get-data-query

A request to retrieve latest snapshot, profile, or census data for output to requestor.

Output-retrieved-data

The data retrieved for a snapshot, census, or profile data retrieval request. This may be a message indicating an error occurred in data retrieval.

6.4 Output Security Manager

The Output Security Manager accepts the message to be sent, Output-message or Output-retrieved-data, and sends a query, Output-get-user-data-query, to the User Account to retrieve the user's privileges. It receives that information through Output-receiver-parameters. The process verifies the receiver is authorized to receive the transaction and the data segments, and removes personal data as needed. It logs personal data transmissions, made as a'result of query by driver name or ID number, to the Disclosure Log through Output-disclosure-message. This includes, but is not limited to: the date and time, type of request, material transmitted, media requested, and identification number, name, and address of the user to whom the disclosure was made. The resulting message, Output-authorized-message, is sent, along with the standard to use, to the Output Message Translator.

On the occasion when information has changed as a result of an amendment request, this process shall also query the Disclosure Log, through Output-get-user-id, for all users (persons or agencies) previously sent information as a result of query by Driver name or ID number, except for FHWA requests. The process uses the result, from Output-user-id, to query, via Output-get-user-data-query, the User Account for the latest names and addresses of these users. A transaction of the amended data is built and forwarded to these users using the result of the query, Output-receiver-parameters.

Input: T **Output: F**

Output-message
A message, in internal format, ready for security checks and translation.

Output-receiver-parameters
Authorization levels for receiver and other identifying information.

Output-retrieved-data
The data retrieved for a snapshot, census, or profile data retrieval request. This may be a message indicating an error occurred in data retrieval.

Output-user-id
Contains the users (persons or agencies) previouysly sent information as a result of query by driver name or ID number.

Input: F **Output: T**

Output-authorized-message
An output transaction that has been verified as authorized for the receiver.

Output-disclosure-message
Contains data necessary to record transmissions with personal data.

Output-get-user-data-query
Contains the user identifier to retrieve parameters for sending data.

Output-get-user-id
Contains the fields needed to retrieve users previously sent personnal data.

6.5 Output Message Translator

The Output Message Translator converts the supplied message (Output-authorized-message) from an internal format to the standard specified by the receiver, and sends User-response or AS-req according to the output format specified by the user (EDI, CD-ROM, file transfer). All transmissions are logged with a date and time, via Output-time-stamp, to the, Pending data store.

Input: T **Output: F**

Output-authorized-message

An output transaction that has been verified as authorized for the receiver.

Input: F **Output: T**

AS_req

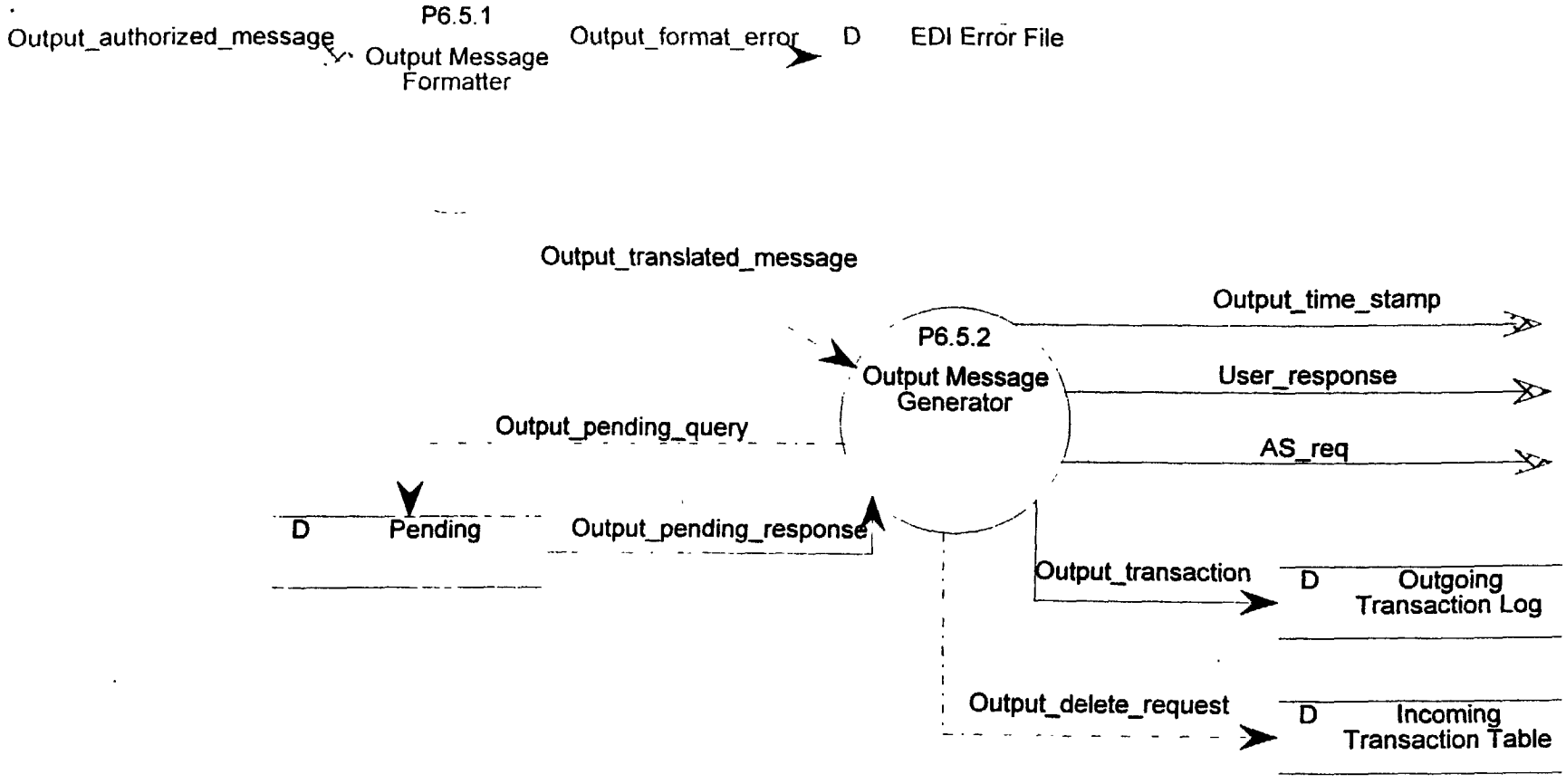
A request for data from an Authoritative Source. This may also route updated carrier information to the appropriate source and send interchange and functional acknowledgments to the Authoritative Source.

Output-time-stamp

Provides the date and time the transaction was sent.

User-response

Response to user request for data (snapshot, profile), account establishment and maintenance, carrier data maintenance, payment for System services, subscriber definition and modification, or forwarded amended data. Also included are interchange and functional acknowledgments.



6.5. Output Message Translator, READ ONLY
System Architect
Tue Aug 29, 1995 15:12
Comment

6.5.1 Output Message Formatter

The Output Message Formatter is a COTS product used to translate internally structured messages, Output-authorized-message, to external, (e.g., EDI) format, Output-translated-message, according to the specified standard. This is the same commercial product used to convert incoming transactions to internal format. Error messages, Output-format-error, are written to the EDI Error File if the supplied message could not be mapped to the specified standard. Otherwise, compliance checking is performed on the input message. If errors are detected, a message is written to the ED1 Error File. Valid messages are structured for transmission.

Input: T **Output: F**

Output-authorized-message

An output transaction that has been verified as authorized for the receiver.

Input: F **Output: T**

Output-format-error

Contains the data necessary to identify the mapping or compliance error found with the transaction.

Output-translated-message

A message converted to external (EDI) format,

6.5.2 Output Message Generator

Messages are transmitted via the user specified media (EDI, file transfer, CD-ROM) through User-response and AS-req. The process logs all transmissions to the Outgoing Transaction Log through Output-transaction, and updates the Pending data store with the date and time the transmission was sent, Output-time-stamp. Information from the packet-header of outgoing transactions, is used to delete entries in the Incoming Transaction Table, via Output-delete-request. A query to the Pending data store is made first to determine if there is more processing to do for the current transaction. If so, the entry is not deleted from the Incoming Transaction Table. The entry is deleted only when all processing for the transaction has been completed.

Input: T **Output: F**

Output-pending-response

Indication if processing for current transaction is complete.

Output-translated-message

A message converted to external (EDI) format.

Input: F **Output: T**

AS-req

A request for data from an Authoritative Source. This may also route updated carrier information to the appropriate source and send interchange and functional acknowledgments to the Authoritative Source.

Output-delete-request

Contains the key information to reconcile input requests to output messages.

Output-pending-query

Query to Pending datastore to determine if processing for current transaction is complete.

Output-time-stamp

Provides the date and time the transaction was sent.

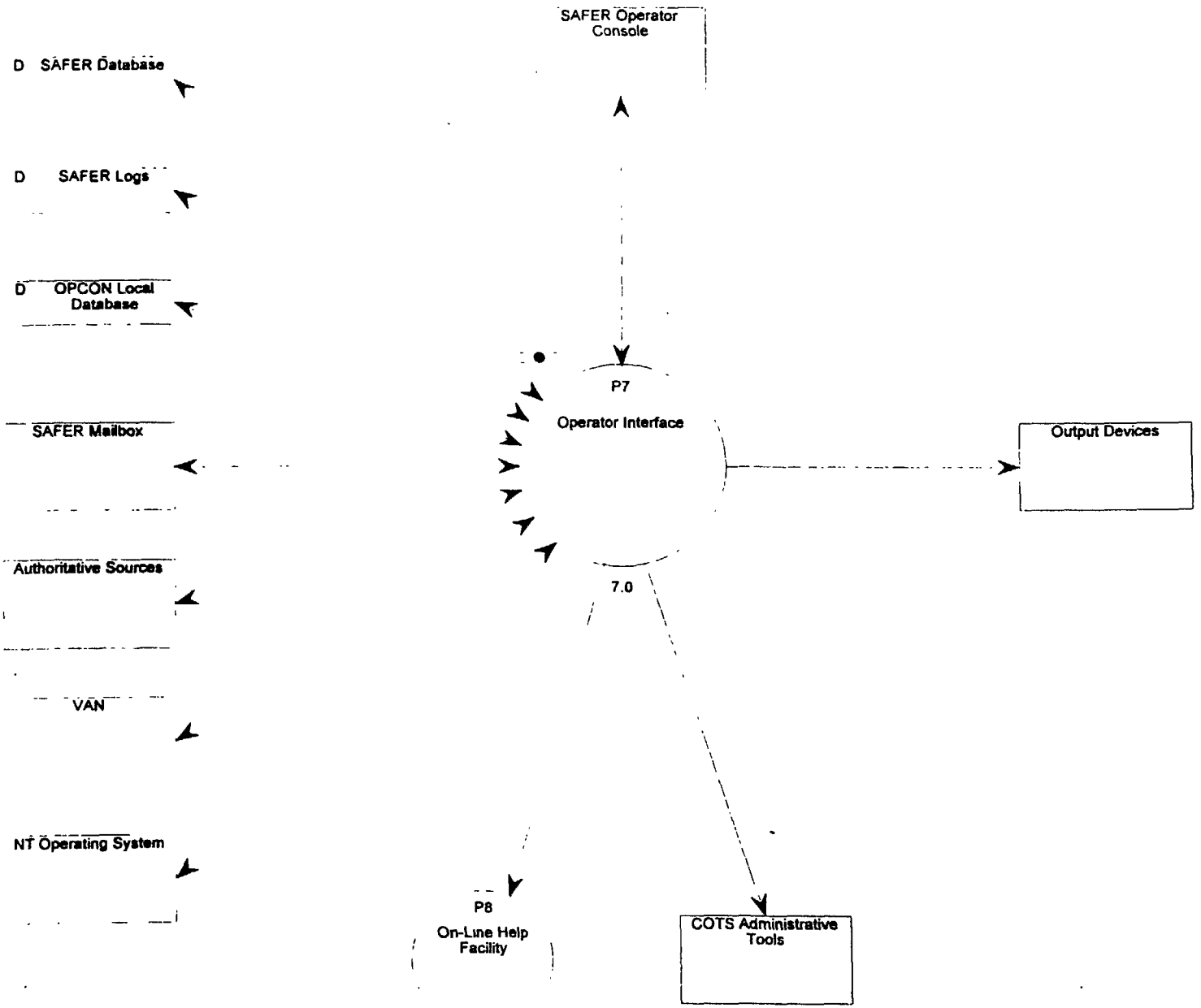
Output-transaction

An image of the outgoing transaction, plus SAFER control numbers, authorization code,, date and time of transmission, and operator and terminal identifiers.

User-response

Response to user request for data (snapshot, profile), account establishment and maintenance, carrier data maintenance, payment for System services, subscriber definition and modification, or forwarded amended data. Also included are interchange and functional acknowledgments.

SAFER Operator Interface - Top Level Diagram



7.0 Operator Interface

The Operator Interface (also known as the Operator Console - OPCON) is the graphical user interface which will support SAFER administrators in performing operations and maintenance functions for SAFER.

The operation and maintenance (O&M) of SAFER will be performed at one or more designated sites. SAFER operators will ensure that the system runs smoothly, and perform backup and recovery operations. Operators will also support SAFER customers, especially those without electronic access to SAFER; customers can request operators to send them inspection information and safety data in various formats and media. SAFER includes an Operator Interface which will be used by authorized SAFER personnel in support of these tasks. It also provides access to billing data and invoices. This interface will include all the capabilities provided by the SAFER Generic User Interface as well.

Its capabilities include managing SAFER data, security, and organization and user accounts. It may also perform some financial management tasks, although these will be primarily achieved through a separate software package. It shall include the ability to perform queries on SAFER data and to create and manage subscriptions and reports. It shall allow operators to view and manage requests and request results. It may additionally provide operators with simplified methods of viewing and managing SAFER system logs and queues. It shall provide the ability to print information presente in OPCON displays.

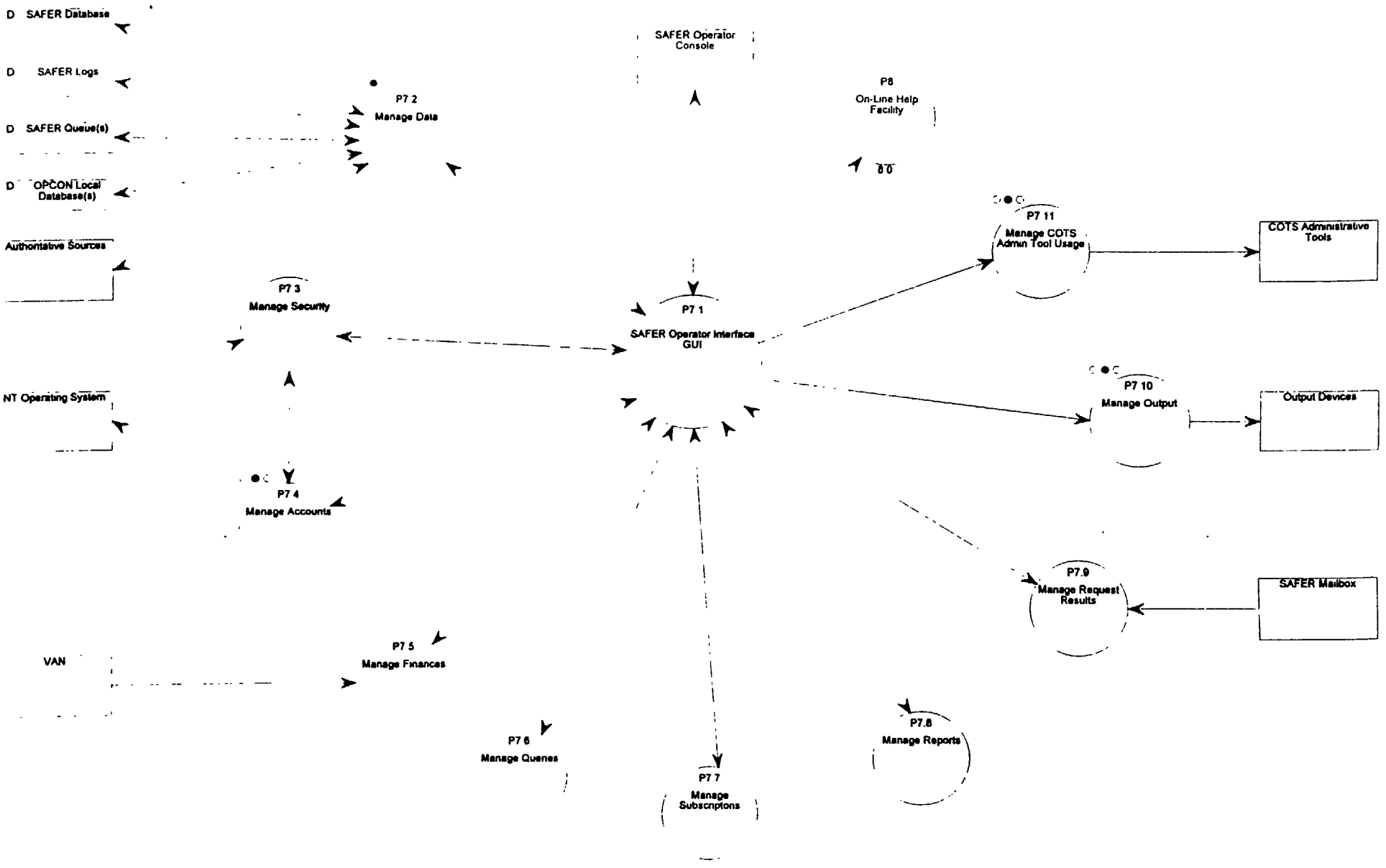
Although many of the user requests to SAFER will be sent electronically, a manual method of access must also be provided for each capability. This allows operators to support and maintain SAFER, as well as users without electronic access.

The SAFER Operator Interface enables SAFER operators and administrators to:
1) link to and communicate with the SAFER System;
2) create and maintain SAFER accounts;
3) make on-line requests for carrier, vehicle, and driver safety and (possibly) credential information (often on behalf of users without electronic access to SAFER);
4) create data subscriptions that define for SAFER the data, the rate, and the media on which data is to be sent to subscribers in an automated, proactive fashion;
5) perform search operations for data when the user's request input is "fuzzy" or ambiguous;
6) access database, network, and system administrative tools to perform operation and maintentance functions on SAFER (e.g. backup, recovery);
7) view and modify SAFER parameter settings; this allows SAFER's behavior to change to reflect business process changes, without requiring code modifications;
8) view SAFER queues, logs , and event tables, for status checking, user request tracking, metrics, and/or trouble-shooting; and
9) create and maintain pre-defined report formats, generate reports from pre-defined formats, and generate customized (ad-hoc) reports.

Input: F	Output: T
Input: T	Output: T
Input: T	Output: T

Input: T	Output: T
Input: T	Output: T
Input: T	Output: T
Input: T	Output: T

SAFER Operator Interface - Main Detailed Flow Diagram



7.01 SAFER Operator Interface GUI

The main window of the SAFER Operator Interface shall have its capabilities organized into various functional groupings. Access to specific functions will be obtained by clicking on any of the menu bar selections according to the standard Windows methodology. Doing so will reveal sub-menu selections, and in some cases additional sub-menus.

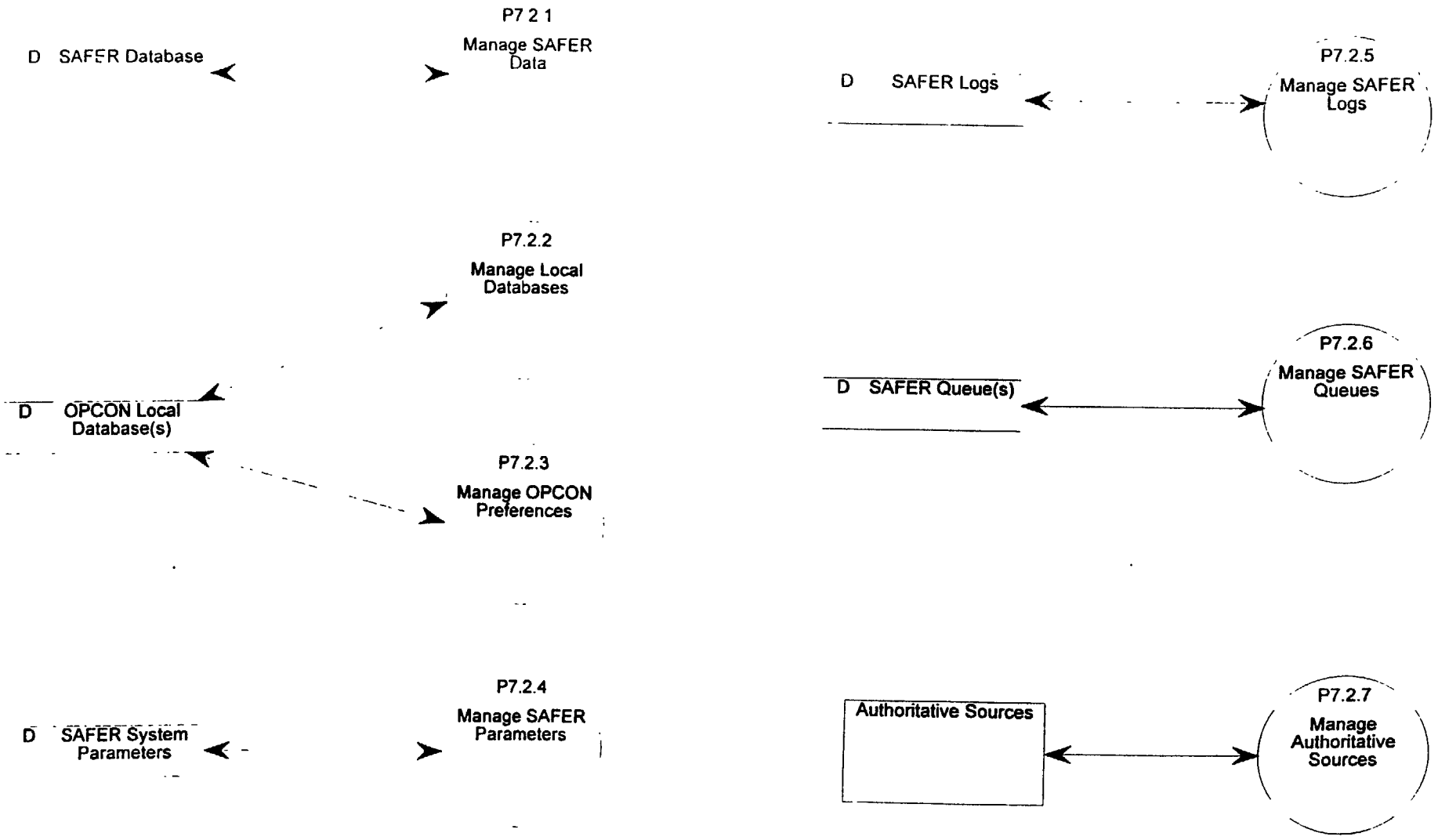
The menu items in this application should generally be sorted alphabetically to help users locate desired items. A letter in each menu item should be underlined in order to enable shortcut key access. Users should be able to select menu items from the keyboard (as an alternative to the mouse) by selecting the Alt key and then the underlined letter of the menu bar item desired. To continue selecting from the menus which appear, they should just select the underscored letters in sequence.

This menu bar should be available on most SAFER Operator Interface windows. An exception to this rule is dialog boxes. Dialog boxes should be programmed to operate modally --- that is, the user should be required to select one of its buttons (operations) before being allowed to Procede The remaining Operator interface windows should not be modal. This will give the user the freedom to view information in one window on the screen, and perform operations on another.

In addition, most Operator Interface windows should be resizable and iconizable (shrinkable). When windows are resized, Delphi automatically adds or removes scroll bars, as appropriate.

If the user selects Exit, OPCON shall close all its windows and log the user off from the SAFER database.

Input: F	Output: T
Input: T	Output: T
Input: T	Output: T
Input: T	Output: T
Input: T	Output: T


D SAFER Database
P7 2 1
Manage SAFER Data
D SAFER Logs
P7.2.5
Manage SAFER Logs
P7.2.2
Manage Local Databases
D OPCON Local Database(s)
P7.2.3
Manage OPCON Preferences
D SAFER Queue(s)
P7.2.6
Manage SAFER Queues
D SAFER System Parameters
P7.2.4
Manage SAFER Parameters
Authoritative Sources
P7.2.7
Manage Authoritative Sources

7.02 Manage Data

Input: T	Output: T
Input: T	Output: T
Input: T	Output: T

7.02.01 Manage SAFER Data

The OPCON shall provide the ability to display, and print SAFER data such as inspection reports, carrier/driver/vehicle snapshots, or profiles.

A carrier snapshot includes: Safety/Compliance Review Data, Work/Cargo/Hazmat Data, Identification Data, Description/Scores Data, and Inspections Data. A driver snapshot may include: Identification Data, and Inspections/Accidents/Violations Data. A vehicle snapshot may include: Identification Data, Inspection Data, and Permits Data.

Input: T Output: T

7.02.02 Manage Local Databases

OPCON shall support one or more local databases to store GUI reference information (e.g. combo bos values) and SAFER query or subscription results temporarily stored on behalf of call-in users.

If multiple local databases are supported (to allow better organization of information), OPCON shall provide the capability to create a new database, open an existing database, and save a database. It shall only allow one database to be open at a time (designated as the current database).

It shall provide a means of reviewing, deleting, archiving, and restoring the contents of the local database(s).

It may also provide a means to update local GUI reference data from a new text (ini) file.

Input: T Output: T

7.02.03 Manage OPCON Preferences

Input: T Output: T

7.02.04 Manage SAFER Parameters

Input: T Output: T

7.02.05 Manage SAFER Logs

Input: T Output: T

7.02.06 Manage SAFER Queues

Input: T Output: T

7.02.07 Manage Authoritative Sources

Input: T Output: T

7.03 Manage Security

SAFER users will require User IDS, Organization IDS, and passwords in order to gain access SAFER. SAFER will assign a unique Organization ID whenever a new Organization Account is created. Organizations will assign their own users User IDS. They must be unique within their organization.

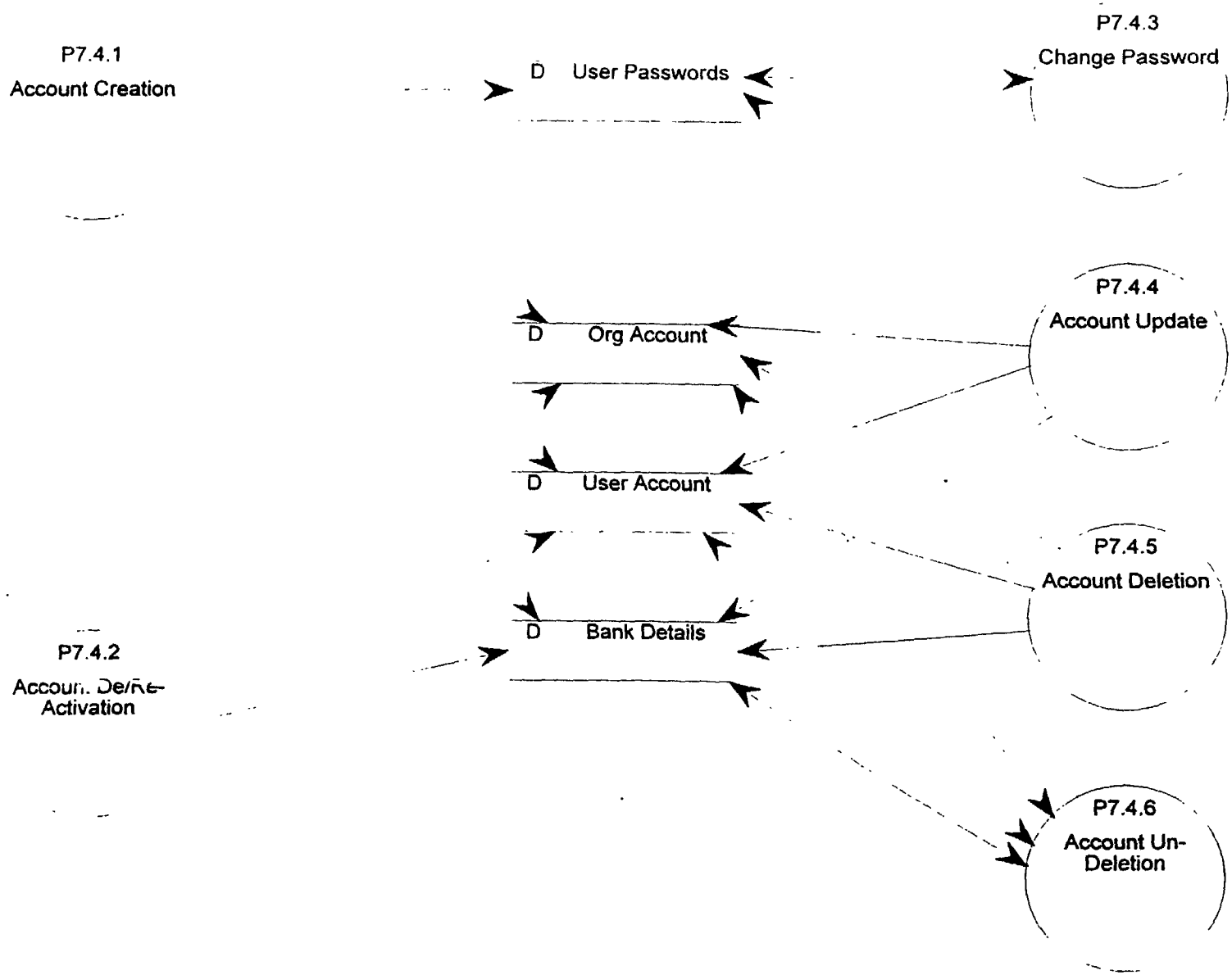
When passwords are typed into the Login screen, the characters typed should not be echoed onto the screen. This is a security measure to reduce the chances of unauthorized access to SAFER.

A check should be performed to ensure that the Organization Account, and the User Account are valid and active, and that the correct password was provided. If any of these checks fail, an error message should be presented to the user.

In addition, since access to OPCON is restricted to SAFER personnel, the user type should be checked. Non-SAFER operators should not be allowed access to OPCON. SAFER operators will be categorized according to the type of interaction they will have with the system. Administrative support people will be sheltered from portions of the Operator Interface which don't concern them. This is done as a security feature, and to simplify the user interface for people who are less computer literate. Certain menus and menu items will not be available to some types of operators.

Input: T	Output: F
Input: T	Output: T

Account Management - Details



7.04 Manage Accounts

SAFER maintains two primary types of accounts: Organizations and Users. A third type of entity is listed here: Banks. SAFER stores all the general data about the banks that any of its member Organizations use for their SAFER financial dealings. Any financial or bank information that is specific to an organization will be stored with that Organization account. All users must be associated with an Organization in SAFER. Billing for the cost of SAFER transactions is generally handled by the organization.

SAFER does accommodate private users (ones who aren't associated with a paying organization). They will be assigned to an organization named "PRIVATE". Private users will be responsible for paying any charges accrued for their SAFER transactions. Private users will have additional financial information stored in SAFER, such as a credit card number.

When an operator selects Users from the Account Maintenance menu a window should appear, containing an overview of all the users defined in the SAFER database. This information should be shown in table format. If there is more data available than can fit on the screen, scroll bars should appear. The operator should be able to sort the table by any of the columns. Three levels of sorting may be provided (primary, secondary, and tertiary). The operator should be able to limit the users which are displayed in the window by specifying values in the Filter Specifications boxes, one associated with each column. The operator should be able to use a wildcard character (e.g. an asterisk) in the filter specification.

If the operator specifies several filters, and then wants to see the whole list again, he should be able to clear the filter fields all at once.

The operator should be able to create a new User record, based upon a previous user's record. If one of the users was selected (i.e. a row in the table was highlighted) on the users overview screen, that user's information should be displayed as default values for the new user record.

The operator should be able to see more detail associated with a particular User. They should also be able to de-activate a user account, re-activate a previously de-activated account, and delete a user account.

Input: F	Output: T
Input: T	Output: T
Input: T	Output: T
Input: T	Output: T
Input: T	Output: T

7.04.01. Account Creation

A SAFER organization account record is uniquely defined by its Organization ID. General account information is stored in SAFER. Contact information is stored, in case an operator needs to contact a representative from the organization.

Each organization account can have one mailing address and one billing address. If a company wants to specify multiple mailing and/or billing addresses, they need to identify different sub-organization names for those groups, and establish a separate SAFER organization account for each such group. A SAFER organization record can also be uniquely defined by its name and sub-organization name. Organizations will have financial information associated with them, for SAFER billing.

A SAFER user account record is uniquely defined by the combination of its Organization ID and User ID, or alternately by a unique NTUserID. Only certain users will have the authority to create another user. Private users will also have financial information associated with them, for SAFER billing.

A SAFER bank record is uniquely defined by its name and branch. Some of the information stored in the SAFER bank record is used to identify the bank and location. Contact information is stored, in case an operator needs to contact the bank about one of its clients (who is also one of our organizations or private users).

Manual verification processes need to be put in place to ensure that SAFER is populated with accurate data.

Input: F Output: T

7.04.02 Account De/Re-Activation

OPCON should allow operators to de-activate user or organization accounts. It should also allow them to re-activate previously de-activated accounts.

Input: F Output: t

7.04.03 Change Password

The operator should be able to change a user's password. Their own User ID and Organization ID will appear as default values. If they are changing their own password they must enter their old password, and then their new password twice (to avoid typos). If they have the authority to set user passwords, and they are changing another user's password, they do not have to specify the old password (e.g. this allows them to reset the password for a user who has forgotten theirs). Validation checks should occur, to ensure that the new password conforms to SAFER password guideles.

Input: T Output: T

7.04.04 Account Update

OPCON should updates the account record in the SAFER database. All changes to the database should be recorded in logs, along with the date and time, and the org/user ID of the person making the change. If the change is being made on behalf of a call-in user, that user/org ID should also be loggged.

Input: F Output: T

7.04.05 Account Deletion

OCPON shall allow operators to delete user or organization accounts from the SAFER database. This will actually only mark a record as deleted. It will only be purged from the SAFER database after a system specified number of days. This will allow time for organizations to change their mind (or correct for an erroneous deletion) and un-delete an account. All changes to the database shall be recorded in logs, along with the date and time, and the user ID of the person making the change.

Input: F Output: T

7.04.06 Account Un-Deletion

The OPCON should allow operators to un-delete SAFER (user or organization) accounts which were previously deleted, but have not yet been purged from the system.

Input: T Output: T

7.05 Manage Finances

Input: T	Output: F
Input: T	Output: T

7.06 Manage Queries

Input: T Output: T

7.07 Manage Subscriptions

Input: T Output: T

7.08 Manage Reports

OCPON shall provide a certain number of pre-defined report formats for reports which are regularly requested by SAFER customers. It may also provide access to an ad-hoc report writing utility to handle custom report requests.

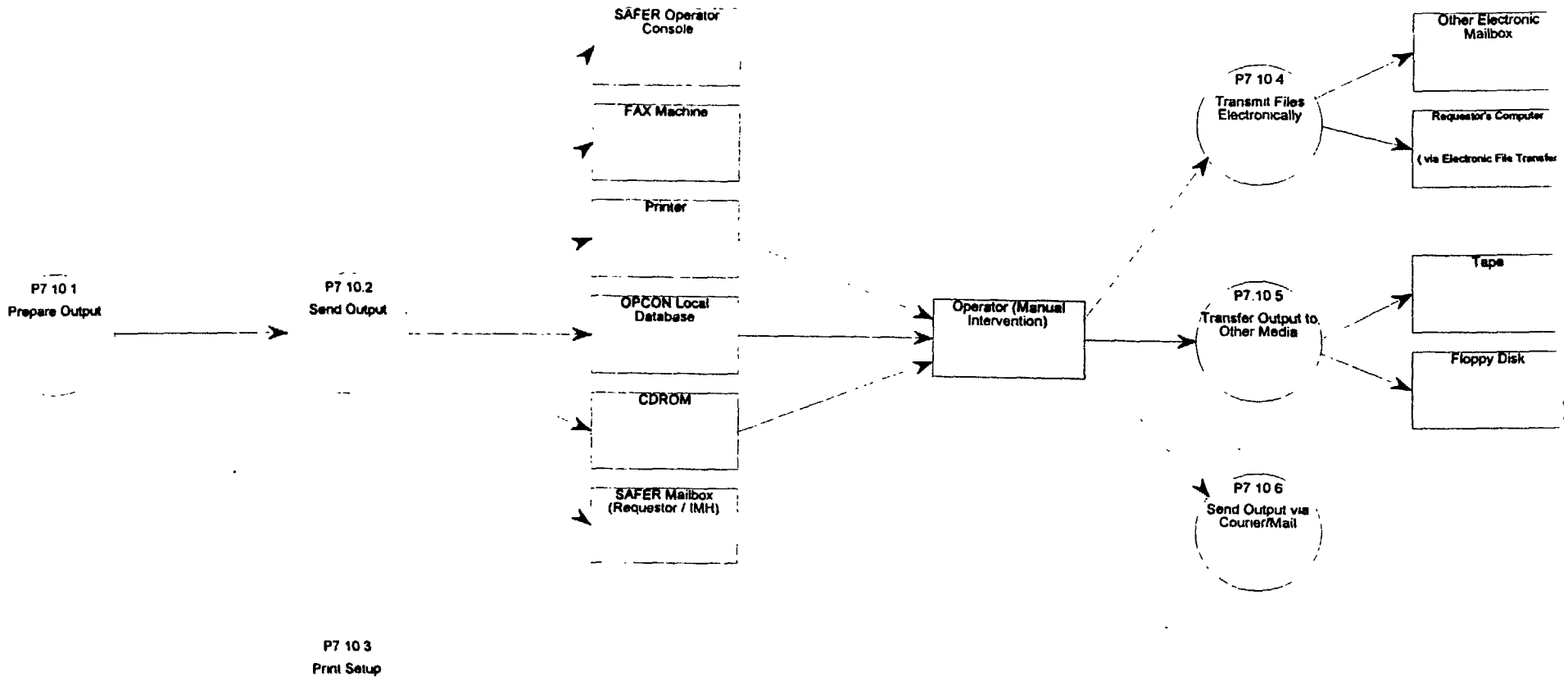
Some of the pre-defined reports which may be included are: System Activity, Billing, Event Status, and Metrics on Authoritative Source interaction with SAFER.

Input: T Output: T

7.09 Manage Request Results

Input: T Output: F

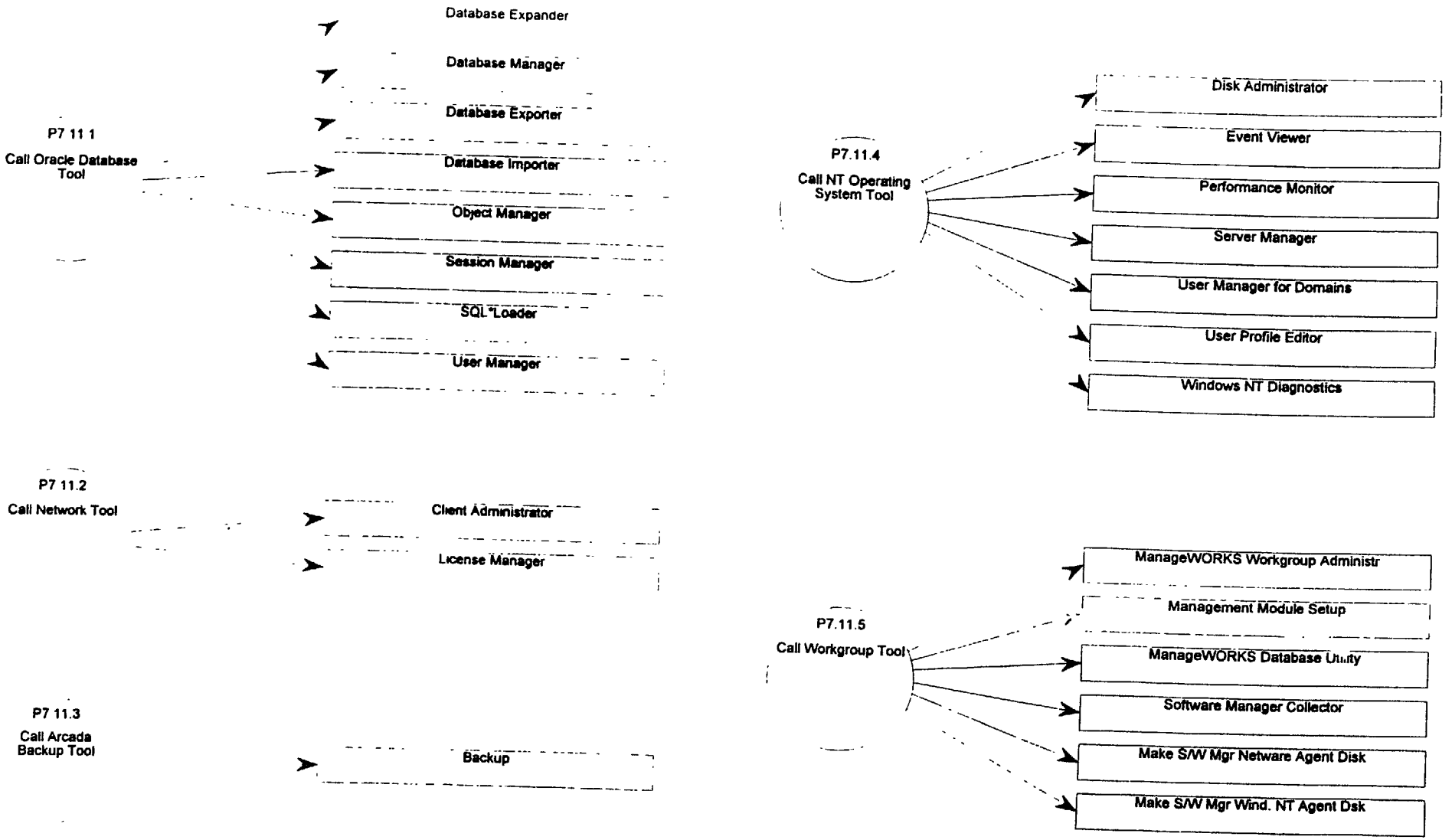
Manage Output - Details



7.10 Manage Output

Input: T	Output: F
Input: F	Output: T

Call COTS Administrative Tools - Details



7.11 Manage COTS Admin Tool Usage

SAFER System Administrators will need to perform tasks such as: diagnostic analysis, repair, backup, recovery and maintenance of SAFER. Sophisticated, and well-maintained software is already available commercially to assist an administrator in performing these tasks. Rather than duplicate these software efforts, SAFER provides links to those commercial off-the-shelf software (COTS) packages.

The Administrative Tools menu groups these packages logically for easy access. System Administrators can still access those packages via the Program Manager, but this menu allows an operator to go between COTS tools and other Operator Interface functions easily.

The initial tool set shall be categorized as follows: Oracle Database tools; NT operating system tools; Arcada backup tools; Network tools; and Workgroup tools (from Digital).

Additional tools may be added, or some current ones may be removed, as operator requirements are further refined.

Input: T	Output: F
Input: F	Output: T

Requirement Name	Build Data Send Req	Build Safety Data Req	Get Pending Profile	Input Message Edit	Input Message Handle	Input Message Trans	Output Security Man	Pending Request_ Man	Process Exact Match	Process Fuzzy Match	Process Profile	Process Cache Man	Request Output_ Man	Request Response_ Man	SAFER	Subscriber Create	Subscriber New Data	Subscriber Process	Subscriber Send Data	Subscriber User Data	Subscriber User Proc	Subscriber View Proc	View Definition
Amended Data Forwarding							X																
Aggregate Data Req. Processing		X														X			X		X		
Aggregate Data Request Input		X														X					X		
Amended Privacy Data					X																		
Authoritative Source Ack.													X										
Authoritative Src. Data Receipt								X						X									
Automated Subscriber Update																						X	
Base State Routing													X										
Carrier Add/Update								X					X										
Carrier Self-Update								X					X										
Cost Check		X																					
Determination of Event																	X				X		
Duplicate Profile Requests	X		X								X												
Event Monitoring Report																						X	X
Forwarding Carrier Updates								X					X										
Fuzzy Searching		X								X													
Inter-User Data Exchange													X										
Inter-User Data Routing													X										
Maximum Carriers Returned										X													
Multiple Match Format		X																					
Pending Profile Data	X		X					X						X									
Pre-Formatted Queries									X	X													
Privacy Considerations				X																			
Privacy Logging Contents							X																
Privacy Logging Requirements							X																
Profile Cache											X	X											
Profile Data Extraction		X						X	X				X										
Profile Processing Definition								X					X										
Request for Carrier Update								X					X										

Requirement Name	Build Data Send Req	Build Safety Data Send	Get Pending Profile	Input Message Edito	Input Message Handl	Input Message Trans	Output Security Man	Pending_req uest_man	Process Exact Match	Process Fuzzy Match	Process Profile Data	Process Profile Cache Manag	Request_output_mana	Request_response_mo	SAFER	Subscriber Create	Subscriber New Data	Subscriber Process	Subscriber Send Data	Subscriber User Mat	Subscriber User Pro	Subscriber View Pro	View Definition
Snapshot Data Extraction		X							X														
Snapshot Refresh								X					X										
Snapshot Update Format				X		X																	
Subscriber List Input																X				X	X		
Subscriber List Output								X					X						X		X		
Subscriber List Trigger Process																	X	X		X	X		
Subscriber Status Report																X					X		
Threshold/Event Translation																						X	X
Types of Events Monitored													X							X	X	X	X
Update Event Table													X		X							X	X
User Documentation									X						X								
User Input For Profiles									X														
User Input For Snapshots									X														

Issue	Build Data Send Req	Build Safety Data S	Determine Requester P	Process Profile Dat	Request_response_mo	Subscriber Create	Subscriber New Data	Subscriber Send Dat	Subscriber User Mat	View Definition
007						X		X		
008						X		X		
020						X				X
022						X				
031								X		
036							X			
039						X		X		
040						X			X	
044					X					
051						X				
053	X				X					
071					X					
072					X					
084				X						
085			X							
086		X								
087		X								
091		X								
098	X				X					
100			X							

Process	ADD_PROF_REQUESTS	ADD_SNAP_REQUESTS	ADD_USERS	DELETE_PROFILE_ENTRY	EVENT_LISTMOV	FORMAT_OUTPUT_REQUESTS	GET_AS_ADDRESS	LOG_OVERDUE_REQUESTS	OVERDUE_REQUEST_PROCESSING	P2:ACCOUNT_MANAGER	P3:SL_MONITOR	PENDING_DELETE_REQUESTS	PROCESS_CARRIER_REQUESTS	PROCESS_DATA_REQUESTS	PROCESS_EVENT_REQUESTS	REQUEST_OUTPUT_MANAGER	REQUEST_RESPONSE_MONITOR	SEARCH_PENDING	SL_DATAREQ	SL_FORMREQ	SL_HILITE	SL_LISTMOV	SL_NEWOLD	SL_START	SL_TEMPWRITE	SL_USER	USER_CANCEL_REQUEST
Account Manager										X																	
Pending_request_manager	X	X	X															X									
Request_output_manager						X	X						X	X	X	X											
Request_response_monitor				X				X	X			X					X										X
Subscriber Create																						X		X			
Subscriber New Data																					X		X				
Subscriber Send Data																			X	X							
Subscriber User Match																									X	X	
Subscriber User Processing											X																
View Definition					X																						

CVO LIBRARY

Safety and Fitness Electronic Records (SAFER) System

Physical Architecture Document

Working Draft

January 31, 1997

Prepared for

Federal Highway Administration

Prepared by:

The Johns Hopkins University
Applied Physics Laboratory

Table of Contents

List of Figures

Figure **Section 3: Page**

1.0 Introduction

This Physical Architecture Document includes the products developed during the physical analysis of the Safety and Fitness Electronic Records (SAFER) System. It defines the configuration of the physical entities and modules that are intended to accomplish the required data and control processing identified in the Logical Architecture Document. This document, along with the companion Operational Concept and Logical Architecture Documents, describe the SAFER Architecture proposed by the Johns Hopkins Applied Physics Laboratory (APL).

1.1 System Overview

SAFER fits within the larger context of existing and planned information systems related to Commercial Vehicle Operations (CVO) and Intelligent Transportation Systems (ITS). SAFER supports the overall CVO Safety Assurance mission to improve the safety of commercial vehicle operations throughout North America.

The SAFER System will provid fixed and mobile commercial vehicle inspection sites, other systems, and users with electroni acces sto the data residing within existing and planned Federal and State motor carrier safety information systems.

The SAFER System will provide information within seconds on a motor carrier 's safety risk rating, roadside inspection history, and accident record, thereby ensuring that enforcement officers working at the roadside will have the most recent information available when deciding which vehicles and drivers should be inspected. The system will also provide insurers and shippers with electronic access to the safety information they need to support their business operations. The system is being designed to support other Intelligent Transportation System (ITS) applications such as electronic verification, checking safety credentials at the time of vehicle registration, and other commercial vehicle administrative processes.

SAFER development will adhere to the ITS CVO Information Systems Architecture being developed by JHU/AP L for FHWA.

The SAFER System will provide users with either a summary of a carrier's safety record ("snapshot" or a more detailed report ("profile"). The system will be both re-active (i.e., responding to specific requests) and pro-active (i.e., allowing users to request that they be informed when the snapshot changes substantially). Users will be able to request information for specific carriers, or for carriers meeting certain selection criteria. Users will specify the desired response time and delivery mechanism.

Initially, information will be available for interstate carriers only. The system will be expanded to provide access to State information systems and intrastate carriers. Expanded system functions will also include allowing users to request industry-wide safety statistics, and provisions for providing larger user sites with change updates rather than complete updates.

To carry out its functions, SAFER will need access to motor carrier identification information for both interstate and intrastate carriers. Over time, SAFER will become the authoritative source for certain forms of motor carrier identification information.

To support the long term operations and maintenance of the SAFER System, users will be charged

for SAFER services. The rate structure will be developed in concert with the system design.

1.2 Approach

The Physical Architecture definition process is responsible for design partitioning and physical entity/subsystem allocations. The Physical Architecture partitioning and physical entity definitions are based upon the Logical Architecture which identified each of the functions required to implement the SAFER System along with the functional relationships and interfaces to the outside world. The Physical Architecture utilizes the data flow diagrams and data dictionary provided for by the Logical Architecture as inputs for the physical partitioning process and analysis. The output of this process is a set of structure charts that represent the physical partitioning of the logical system.

The format and content of the Physical Architecture Document reflect the structured analysis approach originally developed by Tom DeMarco Data and control flows are integrated within a single set of diagrams rather than partitioned onto separate structure chart diagrams.

The structured analysis model was generated using Popkin's System Architect Computer Aided Software Engineering (CASE) tool.

1.3 Document Organization

Following the introductory section, Section 3 provides a description of the internal message passing system that will be used to support communications among the various SAFER software components (processes). The section begins by identifying the processes involved in the SAFER System and then describes the message passing architecture that will facilitate communication among them. Section 3 provides a description of the modules into which the the logical architecture data flow diagrams have been physically partitioned. The section begins by presenting the top-level SAFER structure chart that is logically equivalent to the SAFER top-level data flow diagram (Section 3, Page 3) o the Logical Architecture Document. Each of the subsequent structure charts with their corresponding module definitions are mapped to the Logical Architecture via a Process Matrix that is presented in Section 7. Section 4 presents a description of System Library Modules which are shared among the various System processes. Section 5 and 6. define the System files and database tables, respectively.

2.0 Inter-process Communications

The physical SAFER system is composed of several separate processes that may be distributed across more than one networked node. This architecture facilitates easily accomplished reconfiguration in order to scale the system for various levels of performance over time. It also allows redundant instances of processes on several nodes, increasing reliability to the level required.

The physical processes that will be created are defined in the top-level (Level 1) data flow diagram (DFD) (Section 3, Page 3) of the Logical Architecture Document (LAD). These processes are:

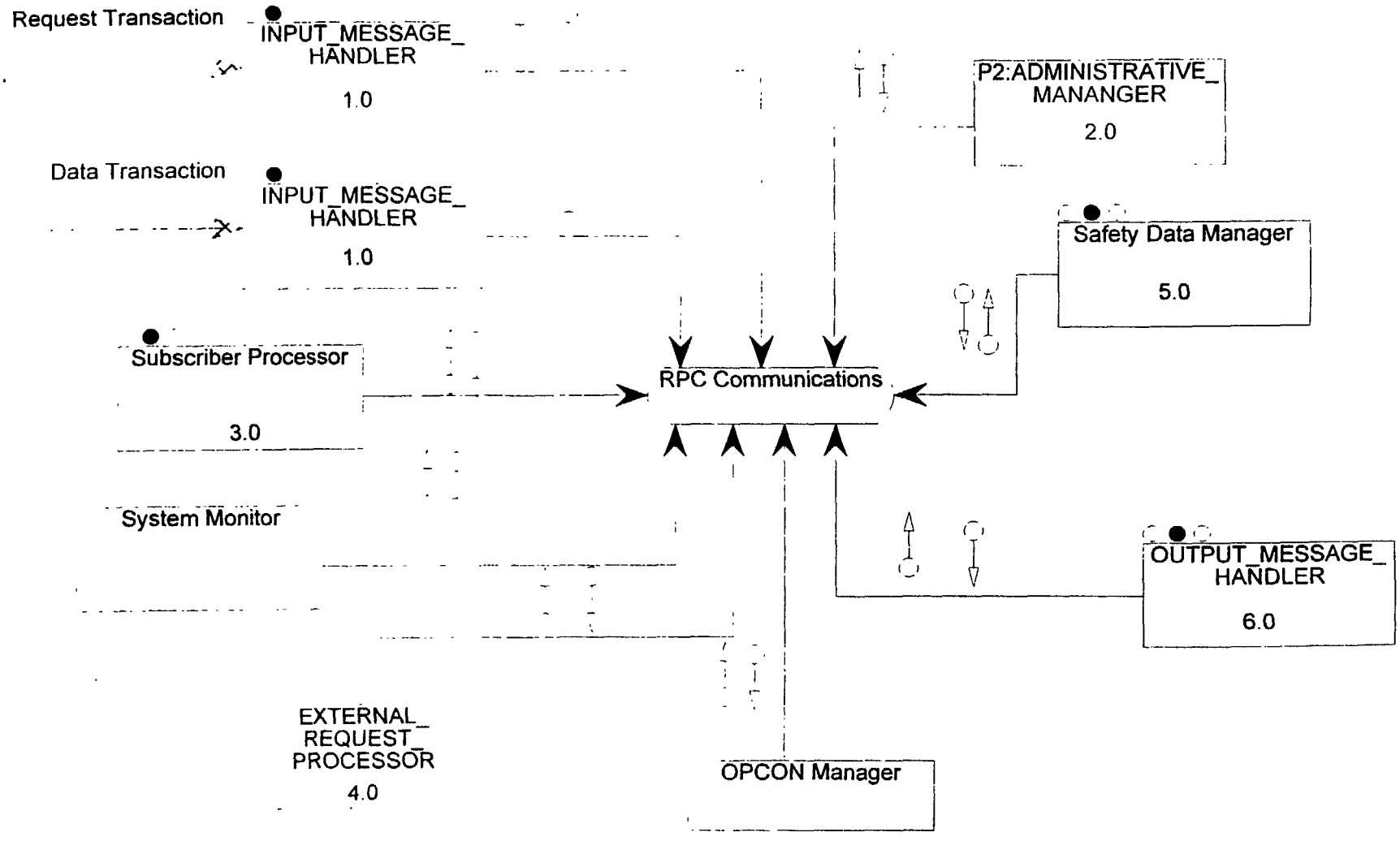
- The Input Message Handler
- The Safety Data Manager
- The Subscriber Processor
- The External Request Processor
- The Administrative Manager
- The Output Message Handler

Several other processes, categorized as utilities to be scheduled for periodic execution, are also defined in the physical architecture, but those listed are the primary operational presence of the system.

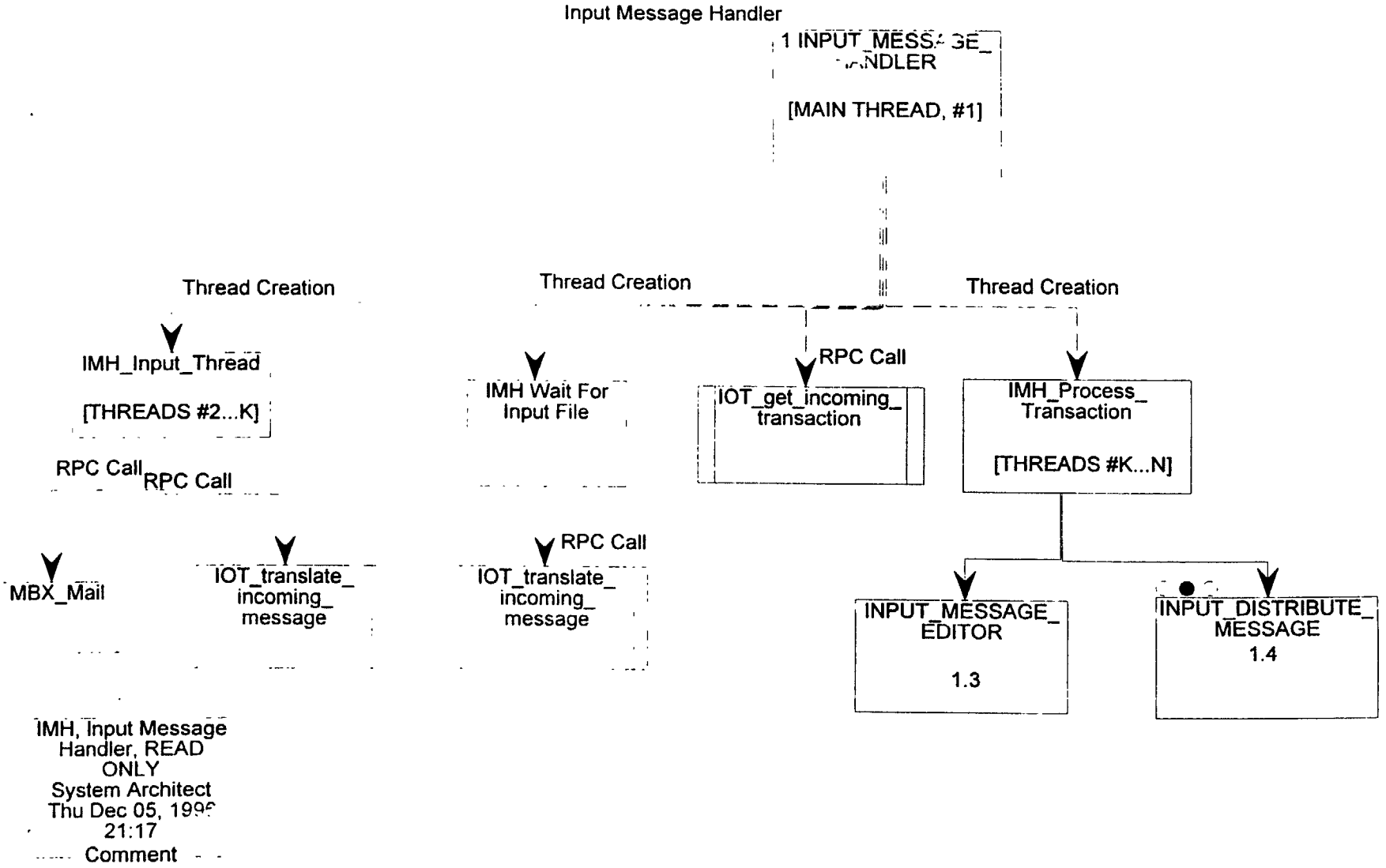
The data flows shown on the Level 1 DFD of the LAD are the principal messages sent between these processes. The mechanism for sending these messages and for receiving results and status back is the Microsoft NT's standard remote procedure call (RPC) call facility. The NT system is designed to support client-server architectures through this facility. It is essentially the same as the Open Software Foundation's Distributed Computing Environment RPC facility, and thus closely adheres to cross platform standards.

An RPC facility allows a "data flow" message passing architecture to be implemented using the familiar "call and return" structured programming paradigm. Each message and response pair becomes a simple call to a subroutine passing input parameters (the sent message) and returning output parameters (the returned message). Each data flow arrow on the Level 1 LAD DFD becomes a call to an appropriately named function. Each process that receives a data flow becomes an RPC server providing those functions and each process from which a data flow originates becomes an RPC client, making calls to the servers. In the physical design, each server is defined as a "library" of calls that can be made by a client, while a client is shown making calls asynchronously to those functions.

By utilizing the RPC mechanism for inter-process messaging, the advantages of independent processes for cohesive subsets of functionality may be achieved with the convenience of familiar structured programming techniques.



SAFER, READ ONLY
System Architect
Mon Jun 03, 1996 14:28
Comment



Waits for file to be created in special directory passes it to IOT. Used for update files that do not come in through a mail box or client server.

1.0 INPUT-MESSAGE-HANDLER

Return Value :

INPUT-MESSAGE-HANDLER accepts incoming interchanges, translates the input into internal format, determines if the sender has the proper authorization level, performs application-level edit checks, and sends a valid transaction to the appropriate process.

PDL :

```
Initialization:
  Call initialization service for initialization information
  For each Mail Box that is to be used:
    Call MBX-Init using Mail Box Initialization information.
    Start up IMH-Input-Thread with resulting MBX-Mail-Box.

 Initialize Worker Threads:
   For 1 to IMH-MAX-TRANSACTIONS:
     Create & Suspend IMH-Process-Transaction Thread


Operation:

  Do Forever:
    EDI-Get-Transaction(Packet-header, SAF-Transaction-Count. SAF-Transaction-List)
    Wait For An IMH-Process-Transaction Thread to Suspend Itself
    Start an IMH-Process-Transaction Thread using Packet-header, SAF-Transaction-Count &
SAF-Transaction-List
```

Files Used :

Database Tables Used :

Input Variables :

Output Variables:

1.1 NW-RECEIVE-TRANSMISSION

Return Value :

NW-INTERFACE executes a 'script' to access the mailbox and retrieve incoming interchanges.

PDL :

Access network

Read mailbox

Write interchange to File-Receive

Files Used :

File-Receive

Database Tables Used :

Input Variables :

Output Variables :

1.2 EDI-Translate-File

Return Value :

EDI-Translate-File accepts files containing SAFER system requests and initiates ED1 translation of their contents. It may also initiate other translation procedures if SAFER is required to support them. The actual tranlation is done by an Commercial, Off the Shelf (COTS) server or utility.

EDI-Translate-File and EDI-Get-Transaction are responsible for: compliance checking, error processing, providing an audit trail, moving the inbound interchange into an internal format, and for generating functional acknowledgments.

This module will use a commercial translation product to analyze each interchange's compliance with the specified standard. At a minimum, the translator should convert incoming (and outgoing) messages according to the specified standard (ANSI X12, EDIFACT, industry standard). It must also be able to check : minimum/maximum field size, presence of required fields, optional and conditional requirements met, proper sequence of data elements within data segment and segments within a transaction set, and correct data types. It should also be able to generate acknowledgments and log incoming and outgoing messages. It should verify that the number of functional groups specified in the Interchange Control Trailer were received, and the Interchange Control Number is the same in both the Interchange Control Header and Trailer. The control number must be the same in the Functional Group Header and Trailer, and the number of transactions received must match the number specified in the Functional Group Trailer. The Transaction set control number must be the same in both the Transaction Set Header and Trailer. The number of segments received for the transaction must match the number specified in the Transaction Set Trailer.

Acknowledgments will be generated, as part of the validation process, for each interchange, if requested, and for each functional group, and sent to the translator generated File-EDI-Acknowledgment for later processing. Valid transactions will be sorted by type. mapped to internal application structures, and written to a flat file, File-Translator-Output. It is assumed all valid incoming transactions are logged to a translator generator audit log, File-Incoming-Transaction-Log. Valid translated messages are sent to the INPUT-MESSAGE-EDITOR for any application-leve checks. If errors are found with a message, it is assumed the translator will generate an error log, File-Translator-Error.

PDL :

Submit MBX-Message-File to COTS EDI translator. Return to caller.

When translation is done, place results on EDI-Transaction-Queue. Translation and placing of results on queue are to be done asynchronously to the initiation of the translation. The translation process, performed by a COTS EDI translator, may be summarized as follows:

Open MBX-Message-File
Read interchange

Validate interchange

If valid interchange Then
  Validate transaction

```
  If valid transaction Then
    Sort transaction by type
    Map transaction to application structures
    Write translated transaction to File-Translator-Output
    Log message, for audit trail, to File-Incoming-Transaction-Log
  Else
    Build transaction error structure
    Build functional group error structure
  Endif

Else
  Build interchange error structure
Endif

If interchange acknowledgment requested Then
  Build interchange acknowledgment
  Write to File-EDI-Acknowledgment
Endif

If interchange error Then
  Write message to File-Translator-Error
Endif

If functional acknowledgment requested Then
  Build functional acknowledgment
  Write to File-EDI-Acknowledgment
Endif

If functional group or transaction error Then
  Write message to File-Translator-Error
Endif
```

Files Used :

File-EDI-Acknowledgment
File-Incoming-Transaction-Log
File-Receive
File-Translator-Error
File-Translator-Output

Database Tables Used :

Input Variables :

MBX-Message-File

Output Variables :

1.2 IOT-translate-incoming-message Return Value :

IOT-Translate-incoming-message accepts files containing SAFER system requests and initiates ED1 translation of their contents. It may also initiate other translation procedures if SAFER is required to support them. The actual tranlation is done by a Commercial, Off the Shelf (COTS) server or utility.

IOT-translate-incoming-message and IOT-get-transaction are responsible for: compliance checking. error processing, providing an audit trail, moving the inbound interchange into an internal format, and for generating functional acknowledgments.

This module will use a commercial translation product to analyze each interchange's compliance with the specified standard. At a minimum, the translator should convert incoming (and outgoing) messages according to the specified standard (ANSI X12, EDIFACT, industry standard). It must also be able to check : minimum/maximum field size, presence of required fields, optional and conditional requirements met, proper sequence of data elements within data segment and segments within a transaction set, and correct data types. It should also be able to generate acknowledgments and log incoming and outgoing messages. It should verify that the number of functional groups specified in the Interchange Control Trailer were received, and the Interchange Control Number is the same in both the Interchange Control Header and Trailer. The control number must be the same in the Functional Group Header and Trailer, and the number of transactions received must match the number specified in the Functional Group Trailer. The Transaction set control number must be the same in both the Transaction Set Header and Trailer. The number of segments received for the transaction must match the number specified in the Transaction Set Trailer.

Acknowledgments will be generated, as part of the validation process, for each interchange, if requested, and for each functional group, and sent to the translator generated File-EDI-Acknowledgment for later processing. Valid transactions will be sorted by type, mapped to internal application structures, and written to a flat file, File-Translator-Output. It is assumed all valid incoming transactions are logged to a translator generator audit log, File-Incoming-Transaction-Log. Valid translated messages are sent to the INPUT-MESSAGE-EDITOR for any application-leve checks. If errors are found with a message, it is assumed the translator will generate an error log, File-Translator-Error.

PDL :

Submit MBX-Message-File to COTS ED1 translator. Return to caller.

When translation is done, place results on EDI-Transaction-Queue. Translation and placing of results on queue are to be done asynchronously to the initiation of the translation. The translation process, performed by a COTS ED1 translator, may be summarized as follows:

```
Open MBX-Message-File
Read interchange

Validate interchange

If valid interchange Then
  Validate transaction
```

```
  If valid transaction Then
    Sort transaction by type
    Map transaction to application structures
    Write translated transaction to File-Translator-Output
    Log message, for audit trail, to File-Incoming-Transaction-Log
  Else
    Build transaction error structure
    Build functional group error structure
  Endif

Else
  Build interchange error structure
Endif

If interchange acknowledgment requested Then
  Build interchange acknowledgment
  Write to File-EDI-Acknowledgment
Endif

If interchange error Then
  Write message to File-Translator-Error
Endif

If functional acknowledgment requested Then
  Build functional acknowledgment
  Write to File-EDI-Acknowledgment
Endif

If functional group or transaction error Then
  Write message to File-Translator-Error
Endif
```

Files Used :

File-EDI-Acknowledgment
File-Incoming-Transaction-Log
File-Receive
File-Translator-Error
File-Translator-Output

Database Tables Used :

Input Variables :

MBX-Message-File

Output Variables :

1.3 INPUT-MESSAGE-EDITOR

Return Value :

This module verifies the user is authorized for the transaction, performs any application-level edit checks, and records the transaction for transaction reconciliation.

PDL :

```
Initialize Input-valid-message flag to TRUE

For I = 1 to SAF-Transaction-Count:
  /* validate sender's account data */
  INPUT_VALIDATE~SENDER(Pacet-header SAF-Transaction-List[ I ]

  If Input-valid-user Then
    If Interchange-test-indicator = TRUE Then
      Call INPUT-TEST-MODE(Pacet-header, SAF-Transaction-List[ I ]
    Else
      /* perform application-level edit checks*/
      Call INPUT-APPLICATION-EDIT-CHECKS(Pacet-header SAF-Transaction-List[ I ]
  Endif
Else
  Reset Input-valid-message flag to FALSE
Endif
```

Files Used :

File-Translator-Output

Database Tables Used :

Input Variables :

Packet-header
SAF-Transaction-Count
SAF-Transaction-List

Output Variables :

Input- application-msg-struct+
Input-valid-message+
User-message-struct

Input-application-msg-struct	Data Structure
Input-valid-message	Data Element
Packet-header	Data Structure
User-message-struct	Data Structure

1.3.1 INPUT-VALIDATE-SENDER

Return Value :

This module helps to maintain integrity of SAFER data by limiting System access to authorized users. Data tables with valid users, Table-Org-Account and Table-User-Account, are maintained and contain the user the transactions sets he is authorized to use, his access privileges, the transaction standard he has agreed to use, validation flag, and his account number and balance. These fields are maintained through transactions to establish and update user accounts. The set of transactions is established by SAFER based on the type of user. A test database and automatic test procedures will be provided by SAFER for new users. When a new user has demonstrated the ability to send and receive transactions correctly, the user will be considered validated, and is then authorized to send and receive his set of transactions.

The Organization and User Accounts are queried for the user ID found in the message and if found, the user account data is retrieved. This may include, but is not limited to, sender privileges, authorized transaction sets, and account number and balance. If the user ID is not found in the tables, an Input-new-user flag is set. If the user ID is found, checks are made on the status of his account. If the user's account is not valid for the user or is overdrawn, a message is generated for the user. A check is made on his access violation count, and if equals or exceeds the System limit, the transaction is not authorized, and a message is generated for the user. A check will also be made on the validation flag. If the user has not been validated and the interchange test indicator is set for production, a message will be generated for the user.

PDL :

```
Initialize Input-new-user flag to FALSE
Initialize Input-valid-user flag to TRUE

Query Organization and User Accounts

If User ID is not found Then
  Reset Input-new-user flag to TRUE
Else
   If Account-id is not equal to one supplied by user or
    Account-$-value is insufficient Then
    Reset Input-valid-user to FALSE
    Set SAFER-type to USER MESSAGE
    Generate a user message
   Endif

   If User-validation-flag = FALSE and Interchange-test-indicator = PRODUCTION Then
      Reset Input-valid-user flag to FALSE
      Increment User-access-violation-count
      Set SAFER-type to USER MESSAGE
      Generate a user message
    Endif

    If User-access-violation-count equals or exceeds maximum Then
      Reset Input-valid-user flag to FALSE
       Set SAFER-type to USER MESSAGE
```

```
        Generate a user message
    Endif
Endif
```

Files Used :

Database Tables Used :
Table-Org-Account
Table-User-Account

Input Variables :
Application-senders-code+
Interchange-sender-id+
Interchange-test-indicator

Output Variables :
Input-new-user+
Input-valid-user+
User-access-violation-count+
User-message-struct+
User-transactions

Application-senders-code	Data Element
Input-new-user	Data Element
Input-valid-user	Data Element
Interchange-sender-id	Data Element
Interchange-test-indicator	Data Element
User-access-violation-count	Data Element
User-message-struct	Data Structure
User-transactions	Data Element

1.3.2 INPUT-TEST-MODE

Return Value :

This module is an automated test procedure to validate new users. Users must be able to send and receive specified test transaction sets correctly before they will be allowed to send and receive production data. Once the user has successfully passed these 'tests', the User-validated-flag in the User Account database will be reset to TRUE.

PDL :

```
Verify current transaction set

If no errors Then
  Set flag for this transaction set
  Set SAFER-type to USER MESSAGE
  Generate message for user
Endif

If all transaction sets have been successfully processed Then
  Update Tabel-User-Account User-validated-flag to TRUE
Endif
```

Files Used :

Database Tables Used :

Tabel-User-Account

Input Variables :

Input-translated-message-struct

Output Variables :

User-message-struct

Input-translated-message-struct	Data Structure
User-message-struct	Data Structure

1.3.3 INPUT_APPLICATION_EDIT_CHECKS Return Value :

Application-level editing may include semantic checks (reasonableness, null values, unusual or large monetary amounts, duplicate or missing data, invalid data combinations, out of range data), and any special editing required to map the input elements into internal format required by the application that could not be done by the INPUT-MESSAGE-TRANSLATOR. This may include converting time fields in the incoming transactions to Greenwich Mean Time and determining message priority.

Checks are made on the user and the transactions sets a user is authorized to use. If a new user has been identified and the transaction set is not to open a new account, an access violation has occurred This is logged in the Access Log. If the user known to the System, but the transaction is not authorized for the user, the transaction is not authorized. The access violation count is incremented and a message is generated for the user. An updated access violation count is sent to the User Account. For transactions with errors, an error message is generated for the user. Otherwise, internal control numbers are assigned and the transaction is logged. Certain constraints with respect to privacy access are determined and stored into the request header.

PDL :

```
Dofor each transaction

 Initialize Input-valid-message flag to TRUE

 Get transaction set identifier code

   Set User-privacy = 'n'
   Set Privacy-log-flag' = 'y'
   If user-type = '2' (enforcement, non-federal), set User-privacy = 'y'
   If user-type = '1'(enforcement, federal), set User-privacy = 'y' and
     Set Privacy-log-flag' = 'no'
   Insert User-privacy' and Privacy-log-flag' into the Request header.


 If Input-new-user and not new account transaction Then
   Reset Input-valid-message flag to FALSE
   Write message to File-Access-Log
 Elseif transaction not in list of sets User-transactions authorized for user Then
   Reset Input-valid-message flag to FALSE
   Set SAFER-type to USER MESSAGE
   Generate user message
   Increment Access Violation Count
   Update Table-User-Account
  Else
    Perform application-level edit checks

     If the query type is Driver ID or driver name, do:
      If User-privacy field for the requesting user = 'n', reject the request (an error is found) log
       in access violations list, and notify user that he is unauthbrized to make this request.
      If User-privacy field for the requesting user = 'y', User is authorized, (an error is not found)
```

(continue to process request).

If error found Then
Reset Input-valid-message flag to FALSE
Generate user message
Call SEND-USER-MESSAGE
Else
Call INPUT-DETERMINE-PRIORITY

Generate control numbers
Call INPUT-LOG-TRANSACTION
Endif

Endif

Enddo

Files Used :
File-Access-Log

Database Tables Used :
Table-User-Account

Input Variables:
Input-new-user+
Input-translated-message-struct+
User-access-violation-count+
User-transactions

Output Variables :
Input-application-msg-struct+
Input-valid-message+
User-message-struct

Input-application-msg-struct	Data Structure
Input-new-user	Data Element
Input-translated-message-struct	Data Structure
Input-valid-message	Data Element
User-access-violation-count	Data Element
User-message-struct	Data Structure
User-transactions	Data Element

1.3.3.1 INPUT-DETERMINE-PRIORITY

Return Value :

This module sets the message priority based on the SAFER type and request options.

PDL :

```
If SAFER-type = SAFETY DATA REQ Then
  If SDR-type = SNAP and Query-no = 1 Then
    Set SAFER-priority to HIGH
  Elseif Req-overnite-flag = ON or Req-medi-type = CDROM Then
    Set SAFER-priority to LOW
  Else
    Set SAFER-priority to MEDIUM
  Endif
Elseif SAFER-type = CARRIER-req Then
  Set SAFER-priority to MEDIUM
Endif
```

Files Used :

Database Tables Used :

Input Variables :

Fuzzy-select +
Query-no +
Req-media-type +
Req-overnite-flag +
SAFER-type +
SDR-type

Output Variables :

SAFER-priority

Fuzzy-select	Data Element
Query-no	Data Element
Req-media-type	Data Element
Req-overnite-flag	Data Element
SAFER-priority	Data Element
SAFER-type	Data Element
SDR-type	Data Element

1.3.3.2 INPUT-LOG-TRANSACTION

Return Value :

This module updates the translator generated File- Incoming-Transaction-Log with SAFER internal control numbers. Key information for each incoming transaction is stored in the File-Incoming-Transaction-Table for message reconciliation.

PDL :

Update File-Incoming-Transaction-Log

Record transaction in File-Incoming-Transaction-Table for reconciliation

Files Used :

File-Incoming-Transaction-Log
File-Incoming-Transaction-Table

Database Tables Used :

Input Variables:

Functional-group-id+
Interchange-id+
Packet-header+
Transaction-set-header

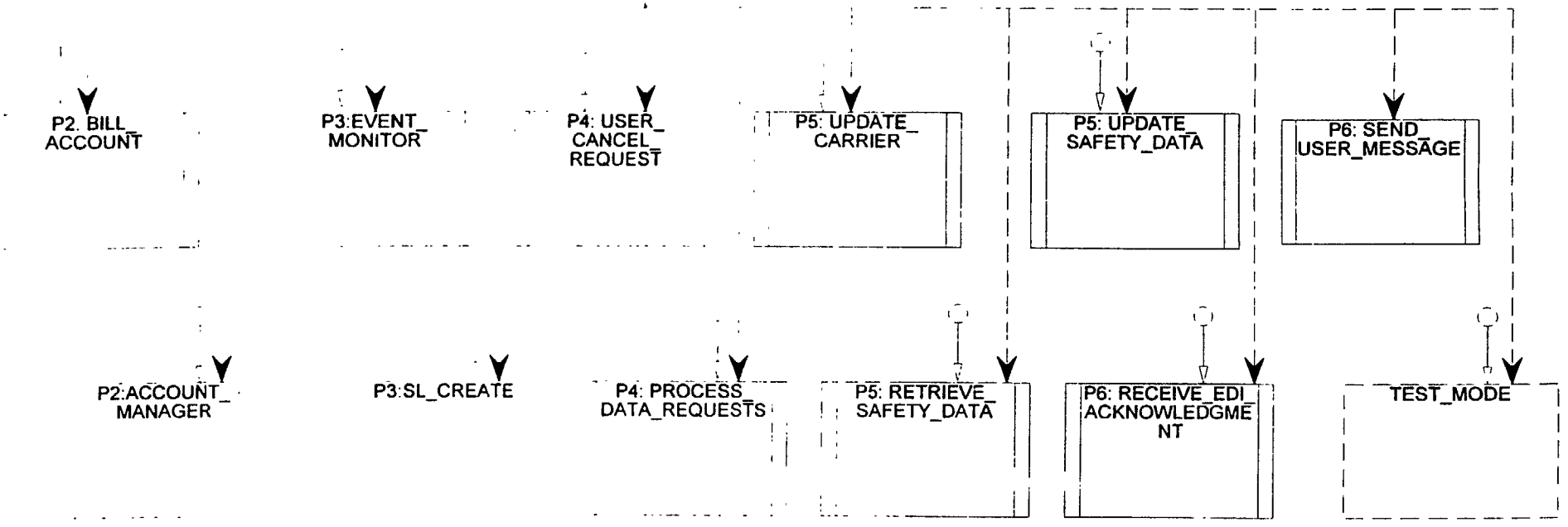
Output Variables :

Functional_group-id	Data Structure
Interchange-id	Data Structure
Packet-header	Data Structure
Transaction-set-header	Data Structure

Input Distribute Message

INPUT_DISTRIBUTE_
MESSAGE

1.4



IMH, Input Distribute
Message, READ ONLY
System Architect
Mon Jun 03, 1996 13.22
Comment

1.4 INPUT-DISTRIBUTE-MESSAGE

Return Value :

INPUT-DISTRIBUTE-MESSAGE receives the next message, and its priority, to send, determines the process, and invokes the method.

PDL :

```
If Interchange-test-indicator = Test Then
 Call TEST-MODE
Elseif SAFER-type = Billing Data Request Then
 Call P2: BILL-ACCOUNT
Elseif SAFER-type = Account Request Then
  Call P2: ACCOUNT-MANAGER
Elseif SAFER-type = Subscriber List Update Then
  Call P3: SL-CREATE
Elseif SAFER-type = Event Defined Request Then
  Call P3: EVENT-MONITOR
Elseif SAFER-type = User Cancel Request Then
  Call P4: USER-CANCEL-REQUEST
Elseif SAFER-type = Data Transfer Request Then
  Call P4: PROCESS-DATA-REQUESTS
Elseif SAFER-type = SafetyData Request Then
  Call P5: RETRIVE-SAFETY-DATA
  If request contains UPDATE-SUBSCRIPTION = TRUE Then
    Call P3: SL-CREATE
Elseif SAFER-type = Valid Safety Data Request Then
  Call P5: UPDATE-SAFETY-DATA
Elseif SAFER-type = Carrier Request Then
  CallP5: UPDATE-CARRIER
EIseif SAFER-type = ED1 Acknowledgment Then
   Call P6: RECEIVE-EDI-ACKNOWLEDGMENT
Elseif SAFER-type = USER MESSAGE Then
    Call P6: SEND-USER-MESSAGE
Endif
```

Files Used :

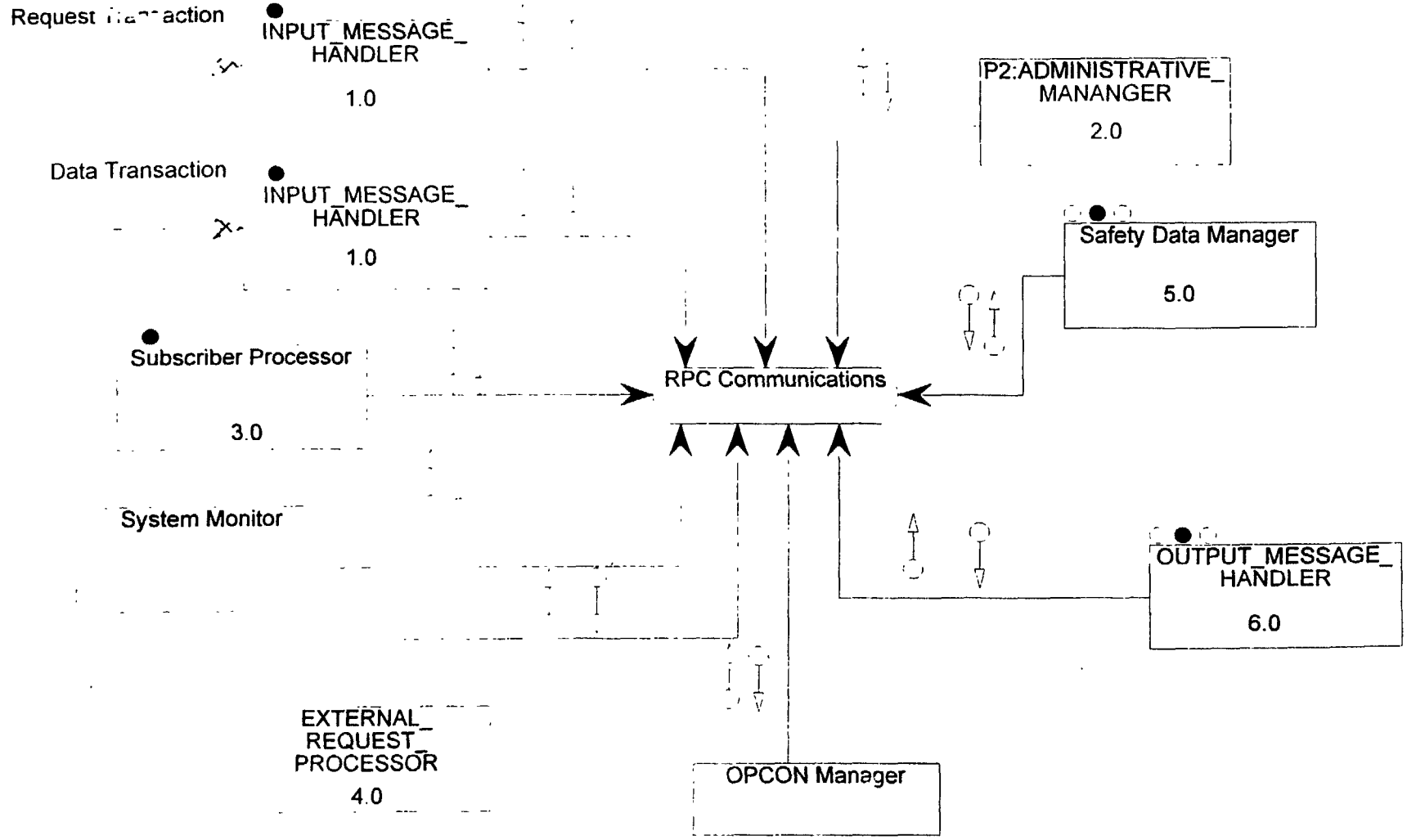
Database Tables Used :

Input Variables :

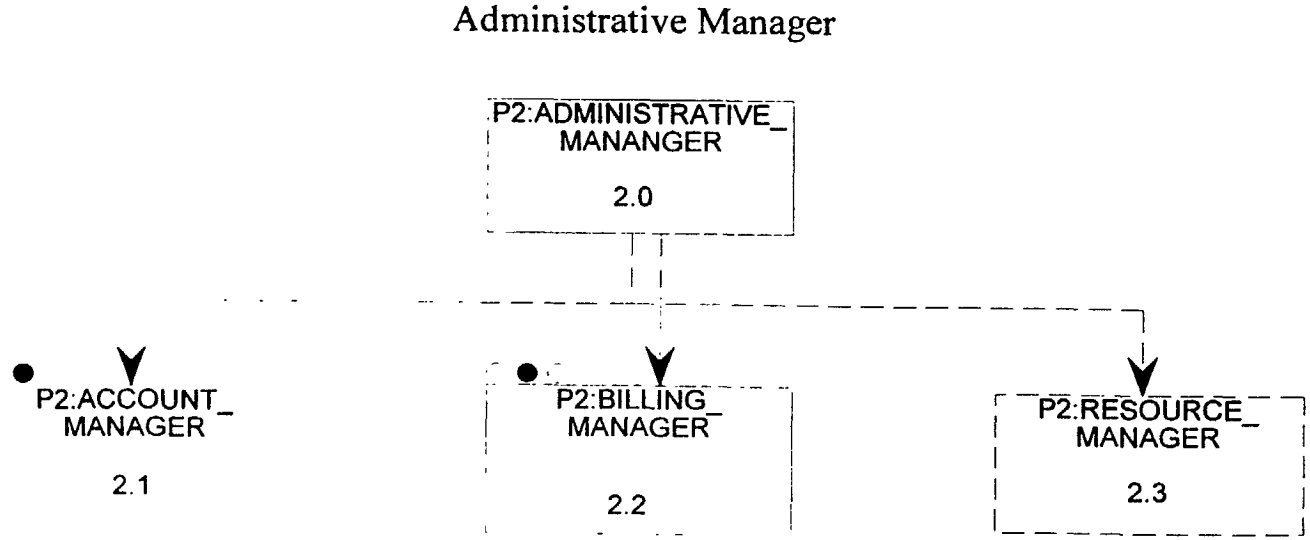
Input-application-msg-struct+
User-message-struct

Output Variables :

Input-application-msg-struct	Data Structure
User-message-struct	Data Structure



SAFER, READ ONLY
System Architect
Mon Jun 03, 1996 14:28
Comment


Administrative Manager
P2:ADMINISTRATIVE_
MANANGER
2.0
P2:ACCOUNT_
MANAGER
2.1
P2:BILLING_
MANAGER
2.2
P2:RESOURCE_
MANAGER
2.3

2.0 P2:ADMINISTRATIVE-MANAGER Return Value :

The Administrative Manager (AM) is responsible for managing:
Accounts - It must facilitate and control the creation, deletion and update of account information. Accounts are established and maintained at the organization and user levels. An organization can be associated with one or more users; a user can be associated with one or more organizations;
Billing operations - It must be able to request monthly billing information from the VAN, process that information based on the resource expenditure rates of each organizational user and bill each accordingly. These functions will be provided by a Commercial-Off-The-Shelf (COTS) financial/billing product;
System Resources - It must provide system resource management most likely in the form of a COTS data archive and retrieval product. The product must be capable of performing online backup and file management functions without disrupting data exchange services.

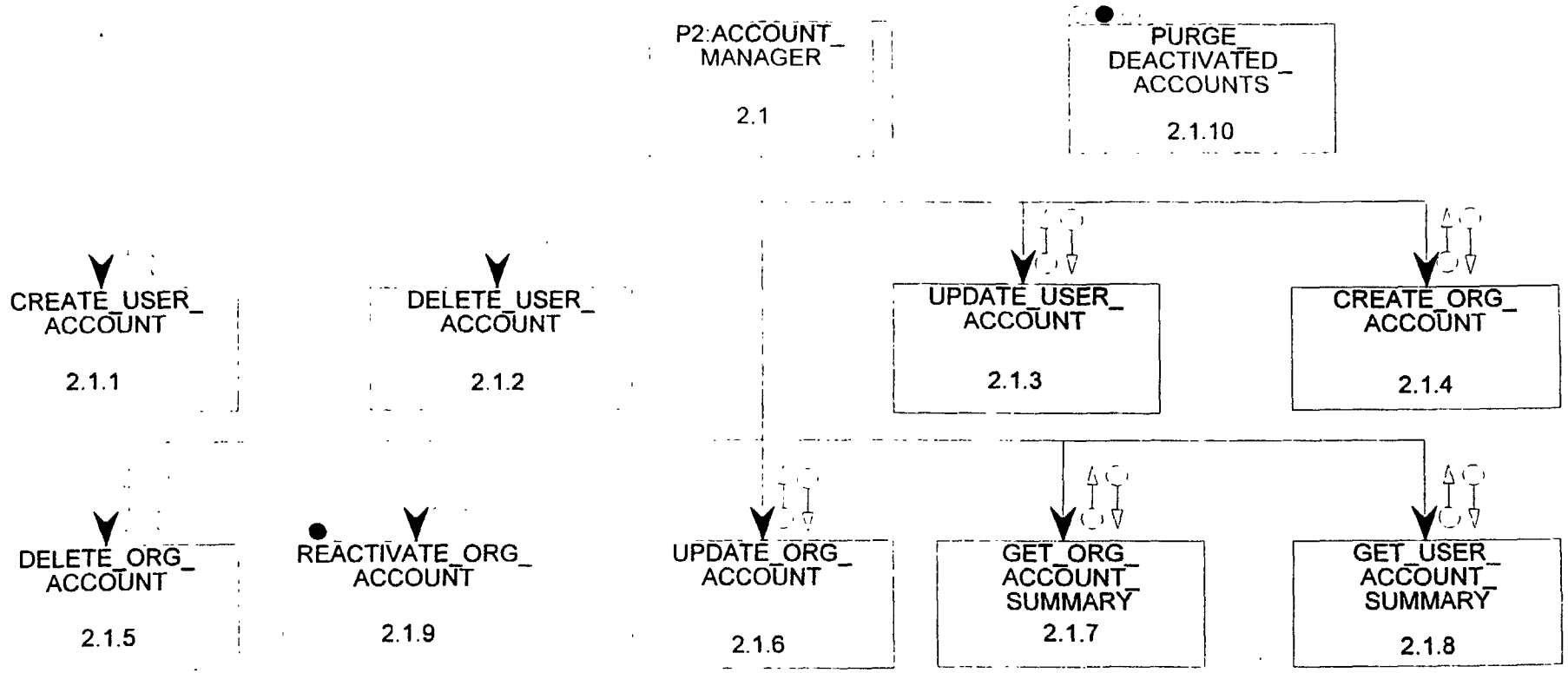
PDL :

Files Used :

Database Tables Used :

Input Variables :

Output Variables :



P2:ACCOUNT_
MANAGER, READ ONLY
System Architect
Wed Dec 04, 1996 12:13
Comment

2.1 P2:ACCOUNT-MANAGER

Return Value :

The Account Manager is responsible for managing user and organizational accounts - It must facilitate and control the creation, deletion, update and providing summaries of account information. Accounts are established and maintained at the organization and user levels. An organization can be associated with one or more users; a user can be associated with one or more organizations.

PDL :

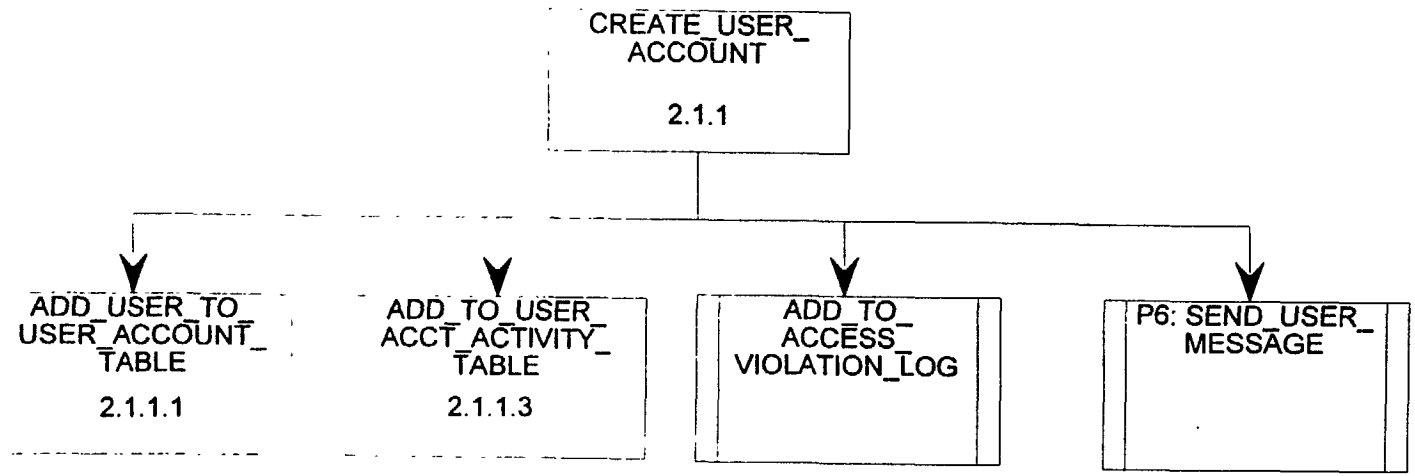
```
Receive Account-req from P1:IMH
If Req-type = Create-account-req
Then Call CREATE-ORG-ACCOUNT 0
Elseif Req-type = Delete-account-req
Then Call DELETE-ORG-ACCOUNT 0
Elseif Req-type = Update-account-req
Then Call UPDATE-ORG-ACCOUNT 0
Elseif Req-type = Org-account-info-req
Then Call GET-ORG-ACCOUNT-SUMMARY ()
Elseif Req-type = Create-user-req
Then Call CREATE-USER-ACCOUNT 0
Elseif Req-type = Delete-user-req
Then Call DELETE-USER-ACCOUNT 0
Elseif Req-type = Update-user-req
Then Call UPDATE-USER-ACCOUNT 0
Elseif Req-type = User-account-info-req
Then Call GET-USER-ACCOUNT-SUMMARY ()
Elseif Req-type = Org-account-info-req
Then Call GET-ORG-ACCOUNT-SUMMARY 0
Endif
```

Files Used :

Database Tables Used :

Input Variables :

Output Variables :


CREATE_USER_ACCOUNT
2.1.1
ADD_USER_TO_USER_ACCOUNT_TABLE
2.1.1.1
ADD_TO_USER_ACCT_ACTIVITY_TABLE
2.1.1.3
ADD_TO_ACCESS_VIOLATION_LOG
P6: SEND_USER_MESSAGE

2.1.1 CREATE-USER-ACCOUNT

Return Value :

The CREATE-USER-ACCOUNT module is responsible for creating an account in response to a user account create request. It performs the following operations: it determines if the account, identified by Account-id, exists in Org Account table. If not, it sends an error message to the requester indicating that the requested account does not exist; Else, it determines if accounts had already been created for the requested user(s) in the User Account table.If some exists, send an error message to the requester indicating that those accounts already exists; If there are still user account(s) to create, and this is not to create a private user, checks to ensure that the user-id of requester that has the User-account-createqriv or the Super-user privilege. If not, logs the user-id and his attempt to access violation log, and sends an error message to the requester indicating that only a user with the User-account-createqriv or Super-user privileges can create a user account; Else, it adds the user(s) to the User Account table based on Account-id and User-id(s), sends a successful user add message to the requester and logs the creations in user account activity table.

Errors encountered when creating users, for example, the user to be created already exists or invalid data in user information, do not terminate the execution of the module, the module shall be able to skip those user(s) and continue processing other users in the list.

PDL :

```
Select Account-id, User-ids from User-Account where Account-id = Input-Account-id
If Account-id = NULL
Then Call P6:SEND-USER-MESSAGE ("Requested Org Account Does Not Exist")
Else
   Requester-exist = False
   Do i = 1 to   Number-Of-Returned-User-id's_for-Account-id
    If  (Requesting-user-id = Returned-User-id (i)
       Then
         Requester-exist = True
         leave
     Endif
   Enddo

   If Requester-exist = False
    Then
       Call P6:SEND-USER-MESSAGE ("User-id of Requester Does Not Exist")
    Else
      Set existing-user-list = empty
      Do While More User-id's in Input-User-List
        Do i = 1 to Number-Of-Returned-User-id's-for-Account-id
          If (Current-Input-User-id = Returned-User-id (i)
          Then
            Add Current-input-User-id to Existing-user-list
            Remove Current-input-User-id from Input-User-List
[] Endif
        Enddo
      Enddo
      If Existing-user-list is not empty
```

Then
Call P6:SEND-USER-MESSAGE (“These requested User Accounts Already Exists”)
Endif
If Input-User-list is empty
Then Done(Do not continue)
Endif
If Input-Account-id != PRIVATE
Then
[] Select Usergrivilege from User-Account table User-id = Requesting-User-id
[] If Requesting-user-id does not have the User-account-create-priv or the Super-user privilege
Then
Call ADD-TO-ACCESS-VIOLATION-LOG()
[] Call P6:SEND-USER-MESSAGE (“Insufficient privilege for requested operation”)
Done(Do not continue)
❑ Endif
Endif

Call ADD-USER-TO-USER-ACCOUNT-TABLE 0
Call ADD-TO-USER-ACCT-ACTIVITY-TAEBLE 0
[]Call P6:SEND-USER-MESSAGE (“An Account was successful created for”, User-id)

Endif
Endif

Files Used :

Database Tables Used :
User-Account

Input Variables :
User-list, Account-id,
Requesting-user-id,
Packet-header

Output Variables :
User-message-struct

Account-id	Data Element
Packet-header	Data Structure
Requesting-user-id	Data Element
User-list	Data Structure
User-message-struct	Data Structure

2.1.1.1 ADD-USER-TO-USER-ACCOUNT-TABLE

Return Value :

Add an entry to database table User-Account.

PDL :

An INSERT database command.

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

2.1.1.3 ADD-TO-USER-ACCT-ACTIVITY-TABLE

Return Value : void

Add an entry to database table Account-Activity.

PDL :

An INSERT database command.
The routine check logical consistency before doing an insert..
- if the routine is called to insert creation activity, the input parameter for new record has to be filled (NOT NULL).
- if the routine is called to insert deletion activity, the input parameter for old record has to be filled.
- if the routine is called to insert update activity, both parameters for old and new record have to be filled.

Note: SDB-WriteOrgLog 0 is the routine to write log to ORG-ACCOUNT-ACTIVITY.

Files Used :

acct.h, SDB-Write0rgLog.c

Database Tables Used : .

ORG-ACCOUNT-ACTIVITY

Input Variables :

SAFER-acct-transaction-pt
Org-Account-pt
Org-Account-pt

Output Variables :

2.1.10 PURGE-DEACTIVATED-ACCOUNTS Return Value :

Periodiclly this module is called upon to look through SAFER organizational accounts and user accounts, and remove those that have been deactivated as a result of DELETE requests for over certain period.

The module looks through organization accounts first. If any org account needs to be removed, it first checks to see if its bank is referenced by other orgs and delete the bank if it is not referenced; then it deletes all users associated with the org from user account table; at last the org account is removed from the org account table.

The module then looks through all user accounts to see if any users need to be removed. If found any, delete the users from the user account table.

PDL :

```
Select account-ids and inactive-start-date from org-account table where org account-status =
DELETED
For each Account-id in the return list
 If inactive-start-date > X number of days
 Then
   get bank id for the org
   If the reference count on the bank = 1
   Then delete the bank from Bank table
   Endif
   delete all users for the org account id from User-account table
   delete the org account from Org-account table
 Endif

Select user ids and inactive-start-date from User-account table where user account status =
DELETED
For each user id in the return list
 If inactive-start-date > X number of days
 Then delete the user from User-account table
 Endif
```

Files Used :

Database Tables Used :

Input Variables :

Inactive-start-date

Output Variables :

Inactive-start-date	Data Element

DELETE_USER_ACCOUNT
2.1.2
ADD_TO_ACCESS_VIOLATION_LOG
P6: SEND_USER_MESSAGE
ADD_TO_USER_ACCT_ACTIVITY_TABLE
2.1.1.3
DEACTIVATE_USER_ACCOUNT
2.1.5.4

2.1.2 DELETE-USER-ACCOUNT

Return Value :

The DELETE-USER-ACCOUNT module is responsible for deleting an account in response to a user account delete request. It performs the following operations: it determines if the account, identified by Account-id, exists in Org Account table. If not, it sends an error message to the requester indicating that the requested organizational account does not exist; Else, it determines if an account had already been created for the requested user(s) in the User Account table and is associated with the supplied Account-id. If no, it send an error message to the requester indicating that the supplied user account(s) do not exist; Else, it determines if the requester user id is in the user list. If yes, send an error message indicating that a user cannot delete itself; Else, it checks to ensure that the user-id of requester the has the User-account-deleteqriv or Super-user privilege for the supplied Account-id. If not, it logs the requester user id and the attempt to access violation log, and sends an error message to the requester indicating that only a user with the User-account-delete-priv or Super-user privileges can delete a user account; Else, it deactivates the user(s) in the User Account table based on Account-id and User-id(s), sends a message to the requester stating the user accounts are deactivated and will be permanently removed in certain days (In a separate process: After certain days if those user accounts are still deactivated, delete those from User Account table), and logs the deletion in user account activity table.

Errors encountered when deleting users, for example, the user to be deleted does not exist or can not be deleted, do not terminate the execution of the module, the module shall be able to skip those user(s) and continue processing other users in the list.

PDL :

```
Select Account-id, User-id's from User-Account where Account-id = Input-Account-id
Obtain the COUNT of returned User-id's
If Account-id = NULL
Then Call P6:SEND-USER-MESSAGE ("Requested Org Account Does Not Exist")
Else

User-Exists = FALSE
  Do i = 1 to Number-Of-Returned-User-id's_for-Account-id
    If (Requesting-user-id = Returned-User-id (i)
    Then
      User-Exists = TRUE
      Leave
    Endif
  Enddo
  If (User-Exits) = FALSE
  Then
    Call P6:SEND-USER-MESS4GE ("User-id of Requester Does Not Exist")
  Else

    Do While More User-id's in Input-User-List
      User-Exists = FALSE
      Do i = 1 to Number-Of-Returned-User-id' s-for-Account-id
```

```
        If (Current-Input-User-id = Returned-User-id (i)
        Then
          User-Exists = TRUE
          Leave
[] Endif
     Enddo
     If (User-Exists) is FALSE
     Then
[][] Call P6:SEND-USER-MESSAGE ("User " Current-Input-User-id, "does not exist for
account", [][] ❑ Acount-id)
❑ Else
[]  Select User-privilege from User-Account table User-id = Requesting-User-id
[]  If Requesting-user-id has the User-account-deleteqriv or the Super-user-privilege
[] T h e n
            If Current-Input-User-id = Requester-User-Id
            Then
              Call P6:SEND-USER-MESSAGE("User can delete itself')
            Else
             Deactivate the user account
             Call ADD-TO-USER-ACCT-ACTIVlTY-TABLE ()
[]      Call P6:SEND-USER-MESSAGE ("User", Current-Input-User-id, " was successful
deleted [] C @from account", Account-id)
              Endif
           Else
              Call ADD-TO-ACCESS-VIOLATION-LOG()
[]      Call P6:SEND-USER-MESSAGE ("Insufficient privilege for requested operation")

[]  Endif
    Endif
  Enddo
 Endif
Endif
```

Files Used :

Database Tables Used :
User-Account

Input Variables :
Account-id, Target-user-id,
Requesting-user-id,
Packet-header

Output Variables :
User-message-struct

Account-id	Data Element

Packet-header	Data Structure
Requesting-user-id	Data Element
Target-user-id	Data Element
User-message-struct	Data Structure

2.1.2.1 DEL-USER-FROM-UA-TABLE

Return Value :

Delete an entry from User-Account table.

PDL :
An DELETE database command.

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

UPDATE_USER_ACCOUNT
2.1.3
P6: SEND_USER_MESSAGE
ADD_TO_ACCESS_VIOLATION_LOG
ADD_TO_USER_ACCT_ACTIVITY_TABLE
2.1.1.3
UPDATE_USER_ACCOUNT_TABLE
2.1.3.1

2.1.3 UPDATE-USER-ACCOUNT

Return Value :

The UPDATE-USER-ACCOUNT module is responsible for updating an account in response to a user account update request. It performs the following operations: it determines if the account, identified by Account-id, exists in the Org Account table. If not, it sends an error message to the requester indicating that the requested organizational account does not exist; Else, it determines if the requesting user has an account that is associated with the supplied organizational Account-id. If no, it sends an error message to the requester indicating that the requester does not have a valid user account; Else, it checks to ensure that the user-id of the requester the has the User-account-updateqriv or has the Super-user privilege for the supplied Account-id. If not, it logs the requester user id and the attempt in access violation log and sends an error message to the requester indicating that only a user with the User-account-updateqriv or Super-user privilege can update a user account; Else, it verifies that the supplied fields for update are updatable; If not, it sends an error message to the requester indicating that the requested field(s) for update are not updatable; Else, it updates the user account information in User-Account, based on User-id and Account-id, sends a successful user update message to the requester and logs the update activity in user account activity table.

Errors encountered when updating users, for example, the user to be updated does not exist or trying to update unupdateable fields, do not terminate the execution of the module, the module shall be able to skip those user(s) and continue processing other users in the list.

PDL :

```
Select Account-id, User-id's from User-Account where Account-id = Input-Account-id
Obtain the COUNT of returned User-id's
If Account-id = NULL
Then Call P6:SEND-USER-MESSAGE ("Requested Org Account Does Not Exist")
EIse
   User-Exists = FALSE
   Do i = 1 to Number-Of-Returned-User-id's_for-Account-id
      If (Requesting-User-id = Returned-User-id (i)
      Then
         User-Exists = TRUE
         Leave
      Endif
   Enddo
   If (User-Exits) = FALSE
   Then
      Call P6:SEND-USER-MESSAGE ("User-id of Requester Does Not Exist")
   Else
     Do While More User-id's in Input-User-List
       User-Exists = FALSE
       Do i = 1 to Number-Of-Returned-User-id's-for-Account-id
         . If (Current-Input-User-id = Returned-User-id (i)
          Then
             User-Exists = TRUE
             Leave
• Endif
```

```
        Enddo
        If (User-Exists) is FALSE
        Then
[][] Call P6:SEND-USER-MESSAGE ("User ", Current-Input-User-id, "does not exist for
account", [][][]Account-id
[]      Else
[] Select User-privilege from User-Account User-id = Requesting-User-id
             If Requester is updating other users' info or updating some of its own info that requires
special permission
             Then
[]     If Requesting-user-id does NOT have the User-account-deletegriv or the Super-user-
[][][]  [][]privilege
[]      Then
                  Call ADD-TO-ACCESS-VIOLATION-LOGO
3          Call P6:SEND-USER-MESSAGE ("Insufficient privilege for requested operation")
               Else
[]
[]         Obtain attributes of the fields to be updated from the RDBMS and verify that the
requested 3 3 C []fields can be updated
[]      If Updatable-fields are "updatable"
[]      Then
                  Call UPDATE-USER-ACCOUNT-TABLE 0
                  Call ADD-TO-USER-ACCT-ACTIVITY-TABLE0
[]          Call P6:SEND-USER-MESSAGE ("User", Current-Input-User-id,  was successful
[][][][]updated")
                Else
                  Call P6:SEND-USER-MESSAGE ("Update of one or more requested fields is not
[][][][]permitted")
        Endif
[]  Endif
       Endif
     Enddo
   Endif
Endif
```

Files Used :

Database Tables Used :
User-Account

Input Variables :
Account-id, Requesting-user-id,
Target-user-id, Updatable-account-fields,
Packet-header

Output Variables :
User-message-struct

Account-id	Data Element
Packet-header	Data Structure
Requesting-user-id	Data Element
Target-user-id	Data Element
Updatable-account-fields	Data Structure
User-message-struct	Data Structure

2.1.3.1 UPDATE-USER-ACCOUNT-TABLE

Return Value :

Update an entry in User-Account table.

PDL :

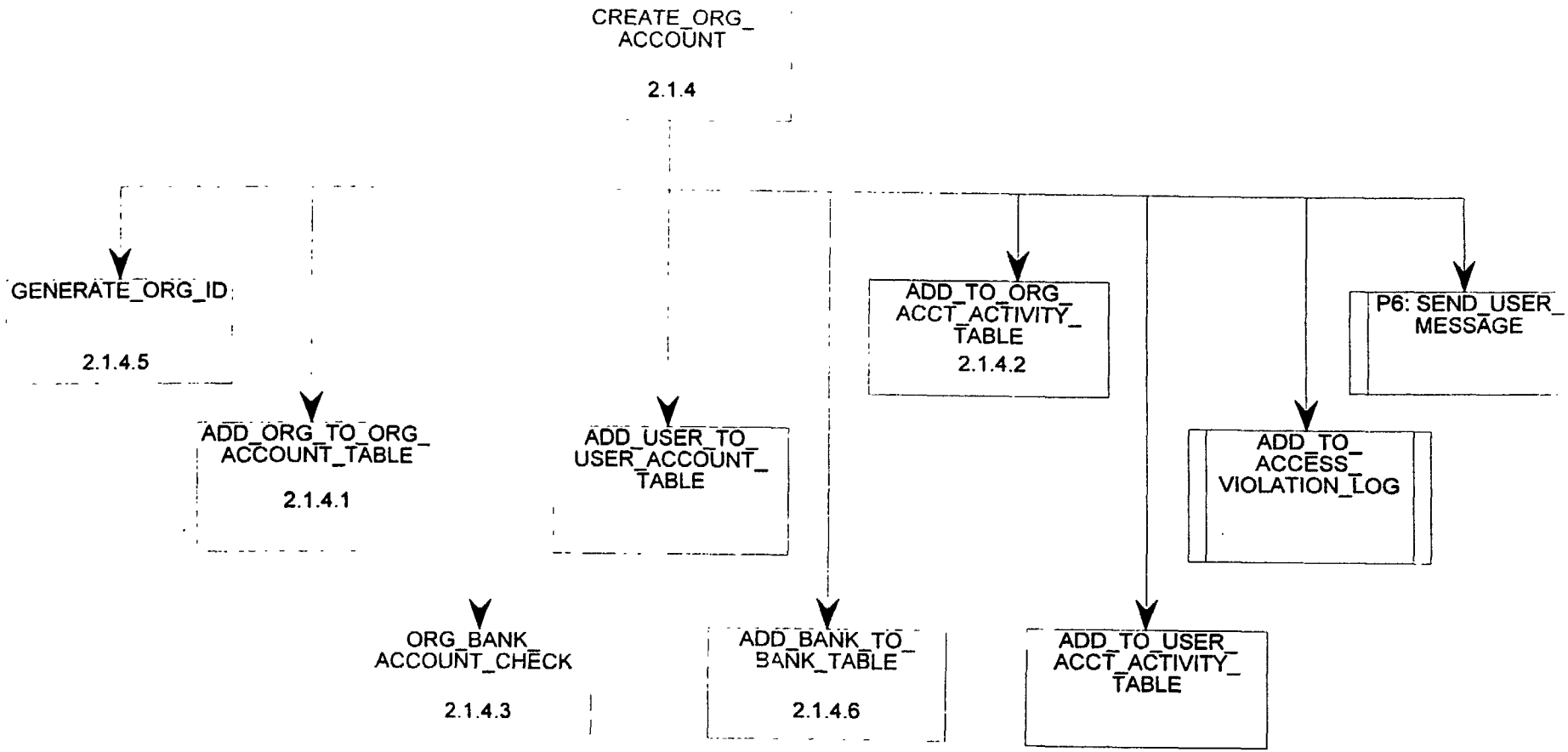
An ALTER database command.

Files Used :

Database Tables Used :

Input Variables :

Output Variables :


CREATE_ORG_ ACCOUNT
2.1.4
GENERATE_ORG_ID
2.1.4.5
ADD_ORG_TO_ORG_ ACCOUNT_TABLE
2.1.4.1
ADD_USER_TO_ USER_ACCOUNT_ TABLE
ADD_TO_ORG_ ACCT_ACTIVITY_ TABLE
2.1.4.2
P6: SEND_USER_ MESSAGE
ADD_TO_ ACCESS_ VIOLATION_LOG
ORG_BANK_ ACCOUNT_CHECK
2.1.4.3
ADD_BANK_TO_ BANK_TABLE
2.1.4.6
ADD_TO_USER_ ACCT_ACTIVITY_ TABLE

2.1.4 CREATE-ORG-ACCOUNT

Return Value :

The CREATE-ORG-ACCOUNT module is responsible for creating an account in response to an organizational account create request. It performs the following operations:
It determines if an account had already been created for the requesting organization in the Org Account table. If yes, it sends an error message to the requester indicating that the account had already been created: Else, it checks to ensure that the user id of the requester is in the user list of the Create Account request and has the Org-account-create-priv or that the user id of the requester has the Super-user privilege. If not, it sends an error message to the requester indicating that only a user with the Org-account-create-priv or Super-user privileges can create an account; Else, it issues an Org-bank-account-check-req (structure in Billing-info-req) to determine if the organization's specified bank has sufficient funds to cover the dollar value of the SAFER account to be created as specified in Account-$-value. If the response from the bank, Org-bank-check-response (structure in Billing-data) validates that sufficient funds are available, it makes the user known to the operating system via the OS-user-account-req, updates the Base State Routing data store if the requesting organization is a Base State, updates the Org Account with the Account-id and the organization information, adds the bank the org is associated with to bank table if it does not exist, adds the users, specified in the user list, to the User Account with the Account-id as a key field, sends a successful account create message with the Account-id to the requester via the Account-req-response and logs the creation in Org Account Activity table; Else, it issues a Failed-bank-check-msg to the user.

PDL :

Retrieve Org-name, Suborg-name, Bank-info, Requesting-user-id and Input-Userliat from Create-account-req

Select Org-name (and Suborg-name if provided from input) from Org-Account where Org-name = Input-Org-name (and Suborg-name = Input-Suborg-name)
If Org-name != NULL
Then CaIl P6:SEND-USER-MESSAGE ("Requested org Account id Already Esists")
Else
User-Exists = FALSE
I=O
Do i = While More User-name's in Input-Userlist
I=I+ 1
If (Requesting-user-id = Userlist-User-id (i)
Then
User-Exists = TRUE
[] Extract User-privilege for Userlist-User-id (i)
Endif
Enddo
Number-of-User-id's-in-Userlist = i
If (User-Exits) = FALSE
Then
Call P6:SEND-USER-MESSAGE ("User-id of Requester Not Specified in the Userlist")
Else
If Requesting-user-id has the User-account-createqriv or the Super-user privilege

```
        Then
[ ] CallORG-BANK-ACCOUNT-CHECK(Funds-Status)
[] If Funds-Status = "sufficient funds"
[] Then
❑    If Org-Type = "Authoritative Source"
[]   Then CALL UPDATE-BASE-STATE-ROUTING-TABLE 0
            Endif
[] Call ADD-ORG-TO-ORG-ACCOUNT-TABLE 0
            Call ADD-TO-ORG-ACCT-ACTIVITY-TABLE 0
            If (Bank-name, Bank-branch0 is NOT in Bank table
             Call   ADD-BANK-TO-BANK-TABLE(Bankjnfo).
            Endif

          Do I   1 to Number-of-User-id's_in-Userlist
            Select User-id from User-Account where Account-id = Input-Account-id and User-
id = Userlist-User-id[i]
            If User-id != NULL
            Then CALL SEND-USER-,ESSAGE("User id has been taken, failed to create an
account for Userlist-User-id[i]")
            Else
             Call ADD-USER-TO-USER-ACCOUNT-TABLE (Userlist-User-id (i))
             Call ADD-TO-USER-ACCT-ACTIVITY-TABLE 0
             Call P6:SEND-USER-MESSAGE ("An Acount was successful created for",
[] Userlist-User-id (i))
            Endif
          Enddo

[] Else
[]   Call P6:SEND-USER-MESSAGE ("Insufficient ORG Funds Message")
[] Endif

      Else
[] Call P6:SEND-USER-MESSAGE ("Insufficient privilege for requested operation")
      Endif
   Endif
Endif
```

Files Used :

Database Tables Used :

Org-Account, User-Account, Bank

Input.Variables :

Create-account-req, Packet-header

Output Variables :

User-message-struct

Create-account-req	Data Structure
Packet-header	Data Structure
User-message-struct	Data Structure

2.1.4.1 ADD-ORG-TO-ORG-ACCOUNT-TABLE

Return Value :

Add an entry to Org-Account table.

PDL :

An INSERT database command.

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

2.1.4.3 ORG-BANK-ACCOUNT-CHECK

Return Value :

PDL :

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

2.1.4.4 ADD-TO-ORG-ACCT-ACTIVITY-TABLE

Return Value :

Add an entry to Account-Activity table.

PDL :

An INSERT database command.

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

2.1.4.5 GENERATE-ORGID Return Value :

This module generates an organization id for the requester. The organization id consists of a stream of characters followed by an underscore and then a number. The character stream takes the short name of the organization if provided with the request, otherwise it takes the first word of the organization name. The maximum length of the character stream is 16.

PDL :

```
Receives Org-account-info-req
Retrieve Shortname if provided, Org-Name
If (Shortname provided)
  Set Org-id-str = Shortname (Truncated after 16th character)
Else
  Set Org-id-str = first word of Org-name (Truncated after 16th)
Endif
Search Org-Account table with Org-id-str and find the max Org-id-num for it
If (Org-id-str not found)
  Org-id-num = 1
Else
  increment Org-id-num by 1
Endif

  Append Org-id-str with a underscore, then the incremented Org-id-num.
```

Files Used :

Database Tables Used :
Org-Account

Input Variables :
Org-account-info-req

Output Variables :
Account-id

Account-id	Data Element
Org-account-info-req	Data Structure

.

2.1.4.6 ADD-BANK-TO-BANK-TABLE

Return Value :

Add a new bank to database BANK table.

PDL :
An INSERT database operation.

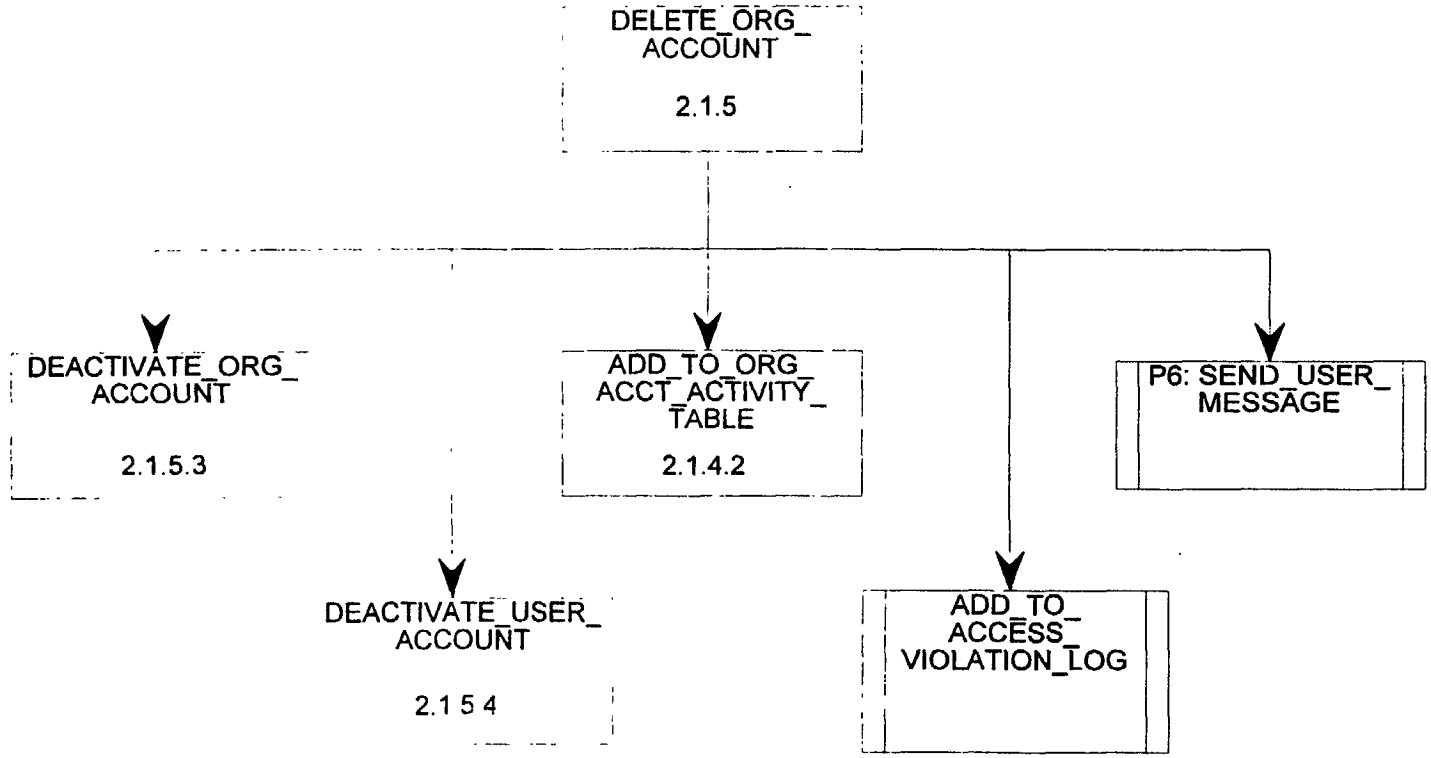
Files Used :

Database Tables Used :
BANK

Input Variables :
Bank-info

Output Variables :

Bank-info	Data Structure


DELETE_ORG_ ACCOUNT
2.1.5
DEACTIVATE_ORG_ ACCOUNT
2.1.5.3
ADD_TO_ORG_ ACCT_ACTIVITY_ TABLE
2.1.4.2
P6: SEND_USER_ MESSAGE
DEACTIVATE_USER_ ACCOUNT
2.1.5.4
ADD_TO_ ACCESS_ VIOLATION_LOG

2.1.5 DELETE-ORG-ACCOUNT Return Value :

The DELETE-ORG-ACCOUNT module is responsible for deleting an account in response to an organizational account delete request. It performs the following operations: it determine if the account, identified by Account-id, exists in Org Account table. If not, it send an error message to the requester indicating that the requested account does not exist; Else, it checks to ensure that the user id of the requester is in the User Account table, is associated with Account-id and has the Org-account-delete-priv or that the user-id of the requester has the Super-user privilege. If not, it logs the requester user id and the attempt in access violation log and sends an error message to the requester indicating that only a user with the Org-account-delete-priv or Super-user privileges can delete an account; Else, it deactivates all user accounts associated with og and send a user message indicating that all its org account is deactivated and will be permanently removed in certain days (In a separate process: After certain day if the org account is still deactivated, it deletes the bank the org is associated with if it is only referenced by this org, deletes all entries in User Account and Org Account tables associated with Account-id). logs the deletion in organization account activity table.

PDL :

```
Select Account-id, User-id's from Org-Account where Account-id = Input-Account-id
Obtain the COUNT of returned User-id's
If Account-id = NULL
Then Call P6:SEND-USER-MESSAGE ("Requested Org Account Does Not Exist, Can't
Delete")
Else
   User-Exists = FALSE
   Do i = 1 to Number-Of-Returned-User-id's_for--Account-id
      If (Requesting-User-id = Returned-User-id (i)
      Then
         User-Exists = TRUE
         Leave
      Endif
   Enddo
   If (User-Exits) = FALSE
   Then
      Call P6:SEND-USER-MESSAGE ("User-id of Requester Does Not Exist")
   Else

      Select User-privilege from User-Account where User-id = Requesting-User-id
      If Requesting-user-id has the Org-account-delete-priv or the Super-user-privilege
      Then
           Deactivate Org Account for Account-id
[] Deactivate all User Accounts with Account-id
[] Call ADD-TO-ORG-ACCT-ACTIVITY-TABLE 0
[] Call P6:SEND-USER-MESSAGE ("Org Account", Account-id, "was deactivated and will be
permanently removed in certain days.)[]

      Else
       ADD-TO-ACCESS-VIOLATION-LOG()
```

```
        Call P6:SEND-USER-MESSAGE ("Insufficient privilege for requested operation")
      Endif
    Endif
Endif
```

Files Used :

Database Tables Used :
Org-Account, User-account, Bank

Input Variables :
Delete-account-req, Packet-header

Output Variables :
User-message-struct

Delete-account-req	Data Structure
Packet-header	Data Structure
User-message-struct	Data Structure

2.1.5.1 DEL-ORG-FROM-OA-TABLE

Return Value :

PDL :

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

2.1.5.2 DELETE-BANK-FROM-BANK-TABLE

Return Value :

Delete an entry from database Bank table.

PDL :
A DELETE database operation.

Files Used :

Database Tables Used :
Bank

Input Variables :
Bank-name +
Bank-branch

Output Variables :

Bank-branch	Data Element
Bank-name	Data Element

2.1.5.3 DEACTIVATE-ORG-ACCOUNT Return Value :

This module marks DELETED to org account status and all the org's user account status if the module is called upon in response to an org delete request. The module marks DEACTIVED to org account status and all the org's user account status if the module is called upon in response to an org deactivate request.

Before a user account status is changed, always save the current status so that when the org account is undeleted, users' previous (pre-delete) account status is restored. The reason behind this is, we want the user accounts that were marked DELETED or DEACTIVED before the org account deletion happens remain DELETED or DEACTIVATED in the case when the org account needs to be undeleted or reactivated.

PDL :

```
If input Status-flag = DELETED
  Update Org-Account table :
     set Acct-status = DELETED where org Account-id = input-account-id
  Call DEACTIVATE-USER-ACCOUNT;

Else if input Status-flag = DEACTIVATED
  Update Org-Account table:
    set Acct-status = DEACTIVATED where org Account-id = input-account-id
   Call DEACTIVATE-USER-ACCOUNT;
Endif
```

Files Used :

Database Tables Used :

Input Variables:

Prev-status

Output Variables:

Prev-status	Data Element

2.1.5.4 DEACTIVATE-USER-ACCOUNT Return Value :

The module marks DELETED or DEACTIVATED to a user account status insponse to a delete or deactivate user(or org) account request.

Before user account status is changed, save it to prev-status. Reference DEACTIVATE-ORG-ACCOUNT for explainations.

PDL :

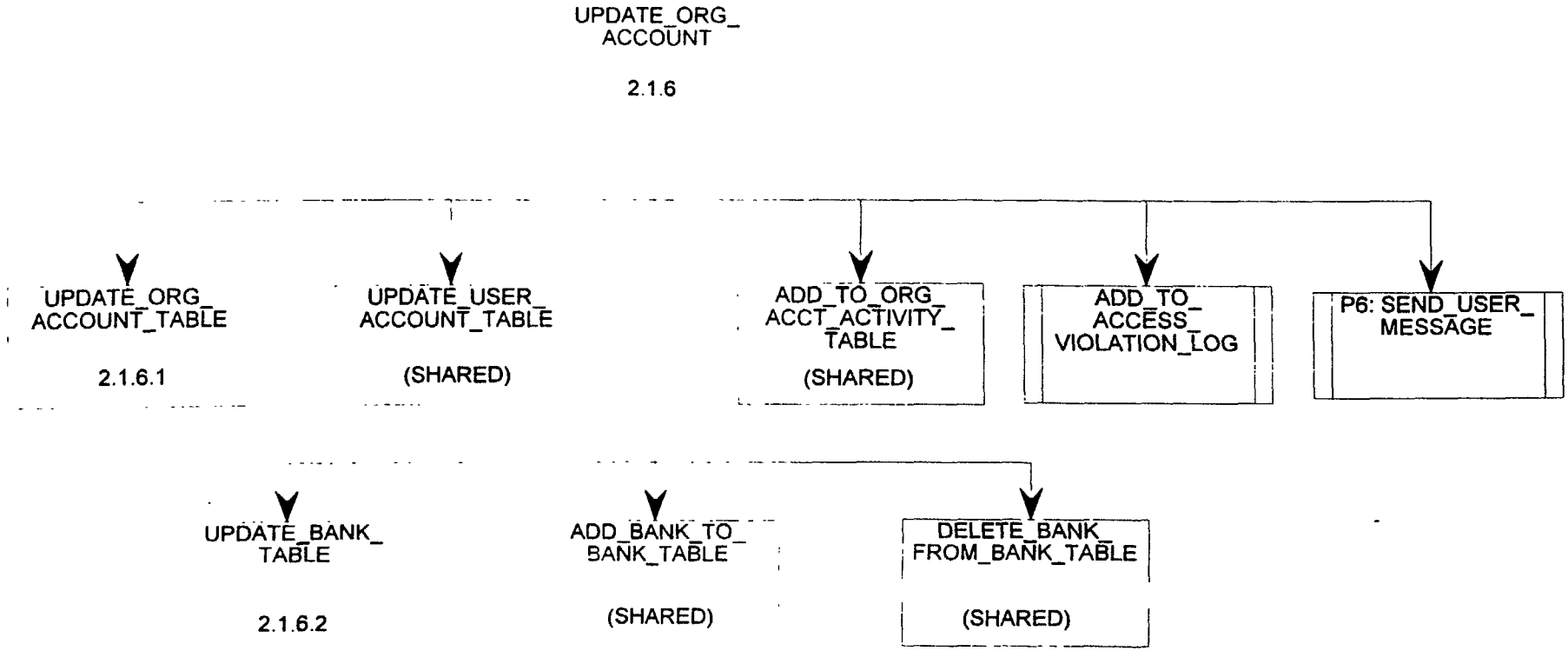
```
If input user-id = NULL
 If input Status-flag = DELETED
  Update User-Account table :
    set Prev-status = Acct-status
    set Acct-status = DELETED where org Account-id = input-account-id
 Else if input Status-flag = DEACTIVATED
  Update User-Account table :
   set Prev-status = Acct-status
   set Acct-status = DEACTIVATED where org Account-id = input-account-id
 Endif
Else -- input user id is not null
 If input Status-flag = DELETED
  Update User-Account table :
    set Prev-status = Acct-status
    set Acct-status = DELETED where org Account-id = input-account-id and User-id = input-user-id
 Else if input Status-flag = DEACTIVATED
  Update User-Account table :
   set Prev-status = Acct-status
   set Acct-status = DEACTIVATED where org Account-id = input-account-id and user-id = input-user-id
 Endif
Endif
```

Files Used :

Database Tables Used :

Input Variables :

Output Variables :


UPDATE_ORG_
ACCOUNT
2.1.6
UPDATE_ORG_
ACCOUNT_TABLE
2.1.6.1
UPDATE_USER_
ACCOUNT_TABLE
(SHARED)
ADD_TO_ORG_
ACCT_ACTIVITY_
TABLE
(SHARED)
ADD_TO_
ACCESS_
VIOLATION_LOG
P6: SEND_USER_
MESSAGE
UPDATE_BANK_
TABLE
2.1.6.2
ADD_BANK_TO_
BANK_TABLE
(SHARED)
DELETE_BANK_
FROM_BANK_TABLE
(SHARED)

2.1.6 UPDATE-ORG-ACCOUNT

Return Value :

The UPDATE-ORG-ACCOUNT module is responsible for updating an account in response to an organizational account update request. It performs the following operations: it determines if the account, identified by Account-id, exists in Org Account table. If not, it send an error message to the indicating that the requested account does not exist; Else, it checks to ensure that the user-id of the requester is in the User Account table, is associated with Account-id and has the Org-account-update-priv or that the user-id of the requester has the Super-user privilege. If not, it logs the requester user id and the attempt to the access violation log, and sends an error message to the requester indicating that only a user with the Org-account-update-priv or Super-user privileges can update an account; Else, it verifies that the requested fields are updatable. If not, it sends an invalid field update request message to the user. Else, it checks if the org is having a new bank(or branch), if yes, adds the new bank to Bank table if it does not exist and deletes the old bank from Bank table if no other org is referencing it; it updates the Org Account associated with Account-id, updates the users on the user update list and send successful account update messages to the requester.

PDL :

```
Select Account-id from Org-Account where Account-id = Input-Account-id
Obtain the COUNT of returned User-id's
If Account-id = NULL
Then Call P6:SEND-USER-MESSAGE ("Requested Org Account Does Not Exist")
Else
   User-Exists = FALSE
   Do i = 1 to Number-Of-Returned-User-id' s-for-Account-id
      If (Requesting-User-id = Returned-User-id (i)
      Then
         User-Exists = TRUE
         Leave
      Endif
   Enddo
   If (User-Exits) = FALSE
   Then
      Call P6:SEND-USER-MESSAGE ("User-id of Requester Does Not Exist")
   Else
      Select User-privilege from User-Account User-id = Requesting-User-id
      If Requesting-user-id has the User-account-deleteqriv or the Super-user-privilege
      Then
         Obtain attributes of the fields to be updated from the RDBMS and verify that the
requested [][] [] Fields can be updated
[]If Updatable-fields are "updatable"
[]Then
            If (Bank-name or Bank-branch changed)
              Retrieve Current Bank-name/branch to Saved-Bank-Name and Saved-Bank-
Branch
              Search Bank table with new Bank-name/Branch combo
              If (Combo not found)
```

```
            Call ADD-BANK-TO BANK-TABLE()
          Endif
          Get reference-count in Org Account table with Saved-Bank-Name and Saved-Bank-
Branch combo
          If (reference-count = 1)
            Call DELETE-BANK-FROM-BANK-TABLE0
          Endif
        Call UPDATE-ORG-ACCOUNT-TABLE 0
        Call UPDATE-USER-ACCOUNT-TABLE 0
        Call UPDATE-BANK-TABLE0

[] Call P6:SEND-USER-MESSAGE ("The requested fieids were successfully updated")
[]Else
[] Call P6:SEND-USER-MESSAGE ("Update of one or more requested fields is not
❑ ❑ [] @permitted")
[]Endif
    Else
        Call ADD-TO-ACCESS-VIOLATION-LOG()
[]Call P6:SEND-USER-MESSAGE ("Insufficient privilege for requested operation")
    Endif
  Endif
Endif
```

Files Used :

Database Tables Used :
Org-Account, User-Account. Bank

Input Variables:
Update-account-req, Packet-header

Output Variables :
User-message-struct

Packet-header	Data Structure
Update-account-req	Data Structure
User-message-struct	Data Structure

2.1.6.1 UPDATE-ORG-ACCOUNT-TABLE

Return Value :

PDL :

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

2.1.6.2 UPDATE-BANK-TABLE

Return Value :

Update an entry in Bank table.

PDL :
An Update database operation.

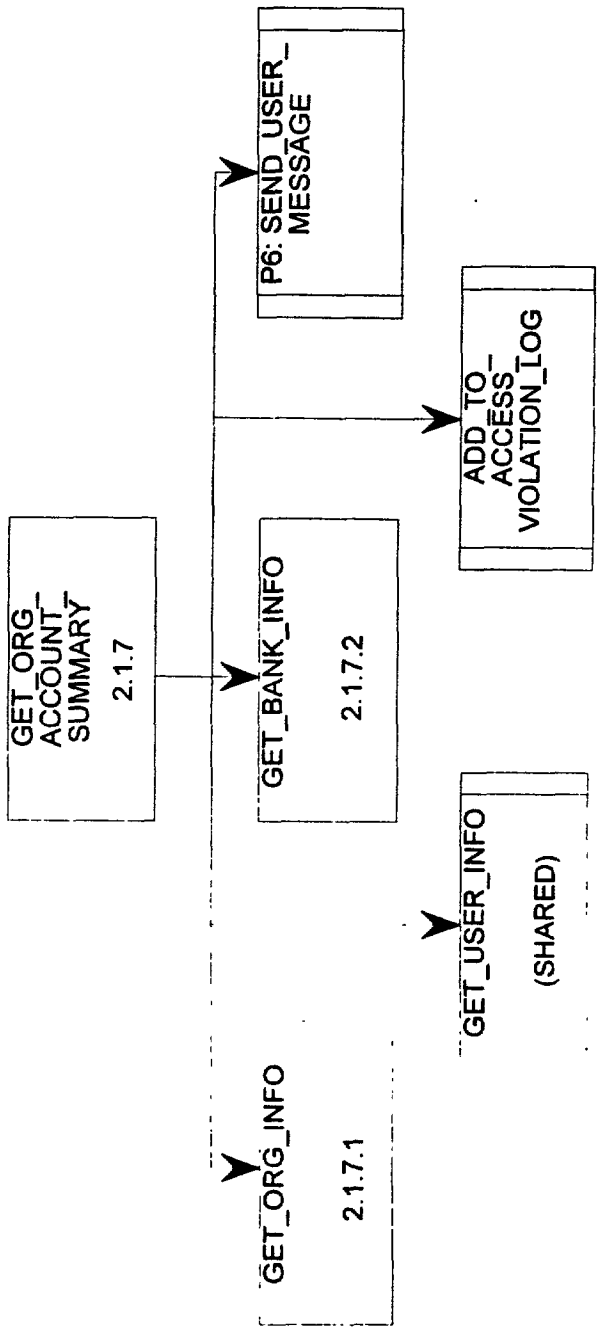
Files Used :

Database Tables Used :
Bank

Input Variables :
Bank-info

Output Variables :

Bank-info	Data Structure


GET_ORG_ACCOUNT_SUMMARY
2.1.7
GET_BANK_INFO
2.1.7.2
GET_ORG_INFO
2.1.7.1
GET_USER_INFO
(SHARED)
P6: SEND_USER_MESSAGE
ADD_TO_ACCESS_VIOLATION_LOG

2.1.7 GET-ORG-ACCOUNT-SUMMARY

Return Value :

This module handles requests on organizational account summaries. It performs the following operations: It checks if the org account identified by account-id exists in Org Account table and the requester has an account with the org, if not, send an error message to the requester; Else, it checks if the requester has the superuser privilege. if not, log the requester user id and the attempt to access violation log and send an error message: Else, it queries the database for the organization and its user information and sends the information back to the requester.

PDL :

```
Receive Org-account-info-req, Packet header
Extract account-id and requesting-user-id from Org-account-info-req
Select Account-id from Org-Account table where Account-id = input-account-id
If Account-id = NULL
Then Call P6:SEND-USER-MESSAGE("Org account does not exist.");
Else
 Get requester-info from User-Account table where Account-id = input-erg-id and User-id =
requesting-user-id
 If database NOT FOUND error
 Then Call P6:SEND-USER-MESSAGE("Requester is not associated with the org. can not
request the org account summary.");
 Else
   If requester-info indicates no superuser privilege
   Then
     Call ADD-TO-ACCESS-VIOLATION-LOG()
     Call P6:SEND-USER-MESSAGE("The requester does not have the privilege to request org
account summary.");
   Else
     Call GET-ORG-INFO()
     Call GET-ORG-BANK-INFO()
     Call GET-USER-INFO()
     Set SAFER-type = ACCOUNT DATA SEND REQ
     load SAFER-type to packet-header
     load returned data to Account-req-response
     Call P6:SEND-ACCOUNT-SUMMARY()
   end if
 end if
end if
```

Files Used :

Requesting-user-id
Account-id

Output Variables :
Org-account-struct +
User-account-struct

Account-id	Data Element
Org-account-info-req	Data Structure
Org-account-struct	Data Structure
Packet-header	Data Structure
Requesting-user-id	Data Element
User-account-struct	Data Structure

2.1.7.1 GET-ORGINFO **Return Value :**

Data base operation:

Query Org-Account table with Account-id and return information about the org.

PDL :

Select Org from Org-Account where Account-id = input-account-id

Files Used :

Database Tables Used :

Org-Account

Input Variables :

Account-id

Output Variables :

Org-account-struct

Org-bank-info

Account-id	Data Element
Org-account-struct	Data Structure
Org-bank-info	Data Structure

2.1.7.2 GET-BANK-INFO

Return Value :

Database operation:

Query Bank table with Bank-name and Bank-branch and return information about the bank.

PDL :

Select bank from Bank where Bank-name = input-bank-name and Bank-branch = input-bank-branch

Files Used :

Database Tables Used :

Bank

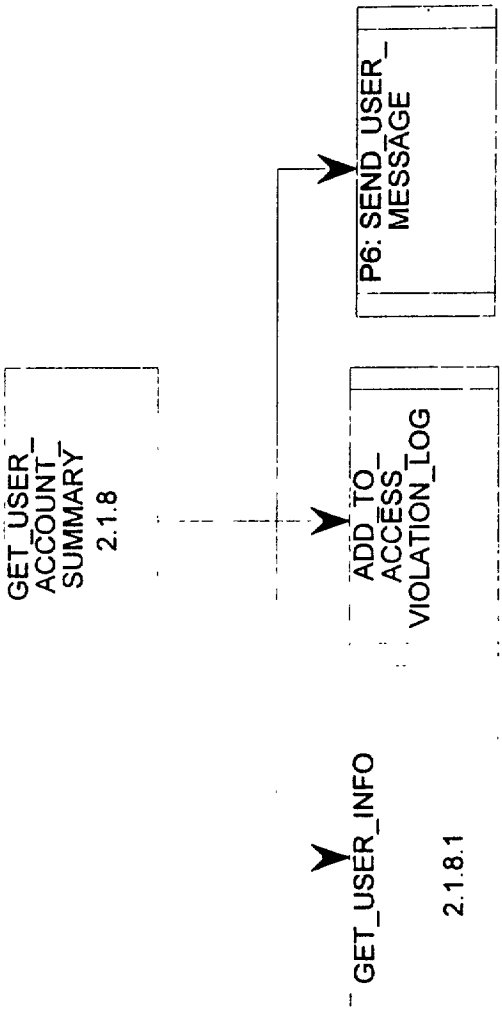
Input Variables :

Bank-name
Bank-branch

Output Variables :

Org-bank-info

Bank-branch	Data Element
Bank-name	Data Element
Org-bank-info	Data Structure


GET_USER_ACCOUNT_SUMMARY 2.1.8
P6: SEND_USER_MESSAGE
ADD_TO_ACCESS_VIOLATION_LOG
GET_USER_INFO 2.1.8.1

2.1.8.1 GET-USER-INFO

Return Value :

Data base operation:

Query User-Account table with Account-id and User-id.

Return information about the user if User-id is provided, and return information about all users of the org identified by Account-id if User-id is not provided.

PDL :

If User-id = NULL
Then
Select users from User-Account where Account-id = input-account-id
Else
Select user from User-Account where Account-id = input-account-id and User-id = input-user-id
end if

Files Used :

Database Tables Used :

User,4ccount

Input Variables :

Account-id
User-id

Output Variables :

User-account-struct

Account-id	Data Element
User-account-struct	Data Structure
User-id	Data Element

2.1.9 REACTIVATE-ORG-ACCOUNT

Return Value :

The REACTIVATE-ORG-ACCOUNT is responsible for reactivating a previously deactivated org account or recovering an accidentally deleted org account in response to an org account reactivation request. It performs the following operations: it determines if the account, identified by Account-id, exists in Org Account table. If not, it sends an error message to the requester indicating that the requested account does not exist in SAFER; Else, it checks if the account is deactivated or deleted. If not, it sends a message to the requester indicating that the org account is currently active; Else, it checks to ensure the requester is a user of the org and has the Org-account-delete privlilege. If not, it logs the requester's user id and the attempt into the access violation log and sends a message back to the requester. Else, it sets the org account status to ACTIVE and sets all its previously active user accounts to ACTIVE again. Logs any activations in org (and user) account activity logs.

PDL :

```
Obtain Account-id and Requester-user-id from Reactivate-org-account-req
Select Account-id, Acct-status from Org-Account table where account-id = input-account-id
If Account-id = NULL
Then
  Call P6:SEND-USER-MESS4GE("Org account does not exist.");
  Exit;
Else if Acct-status is not DEACTIVATED or DELETED
Then
  CALL P6:SEND-USER-MESSAGE("Org account is currently active.");
  Exit;
Else
  Select Requester's Org-Delete-Priv from User-Account table
  If no Org-Delete-Priv
  Then
    CALL ADD-TO-ACCESS-VIOLATION-LOG()
    CALL P6: SEND-USER-MESSAGE("Insufficient privileges");
    Exit
  Else
    CALL REACTIVATE-ORG(Account-id)
    CALL REACTIVATE-USERS(Account-id)
    CALL ADD-TO-ORG-ACCT-ACTIVITY
  End if
End if
```

Files Used :

Database Tables Used :

Input Variables :

Reactivate-erg-req

Packet_header

Output Variables :

User-message-struct

Packet-header	Data Structure
Reactivate-erg-req	Data Structure
User-message-struct	Data Structure

2.1.9.1 REACTIVATE-ORG

Return Value :

This module sets an org account status to ACTIVE. It also checks on the user accounts associated with the org. If any of its user accounts was active before the org account is deactivated or deleted, it sets the user account status to ACTIVE.

PDL :

```
Set Acct-Status = ACTIVE where org Account-id = input-account-id

Select User-id, Prev-Acct-Status from User-Account table where Account-id = input-account-id
For each selected user
Do
  If Prev-Acct-Status = ACTIVE
  Then
    Set Acct-Status = ACTIVE
  End if
End do
```

Files Used :

Database Tables Used :

Org-Account, User-Account

Input Variables :

Account-id

Output Variables :

Account-id	Data Element

.

2.1.9.2 REACTIVATE-USERS

Return Value :

This module reactivates all user accounts for the specified org that were previously active by setting the Acct-Status field to ACTIVE.

PDL :

Select all users from User-Account table where Prev-Acct-Status = ACTIVE and Account-id = input-account-id
For each selected user
Do
  set Acct-Status = ACTIVE
End do

Files Used :

Database Tables Used :

User-Account

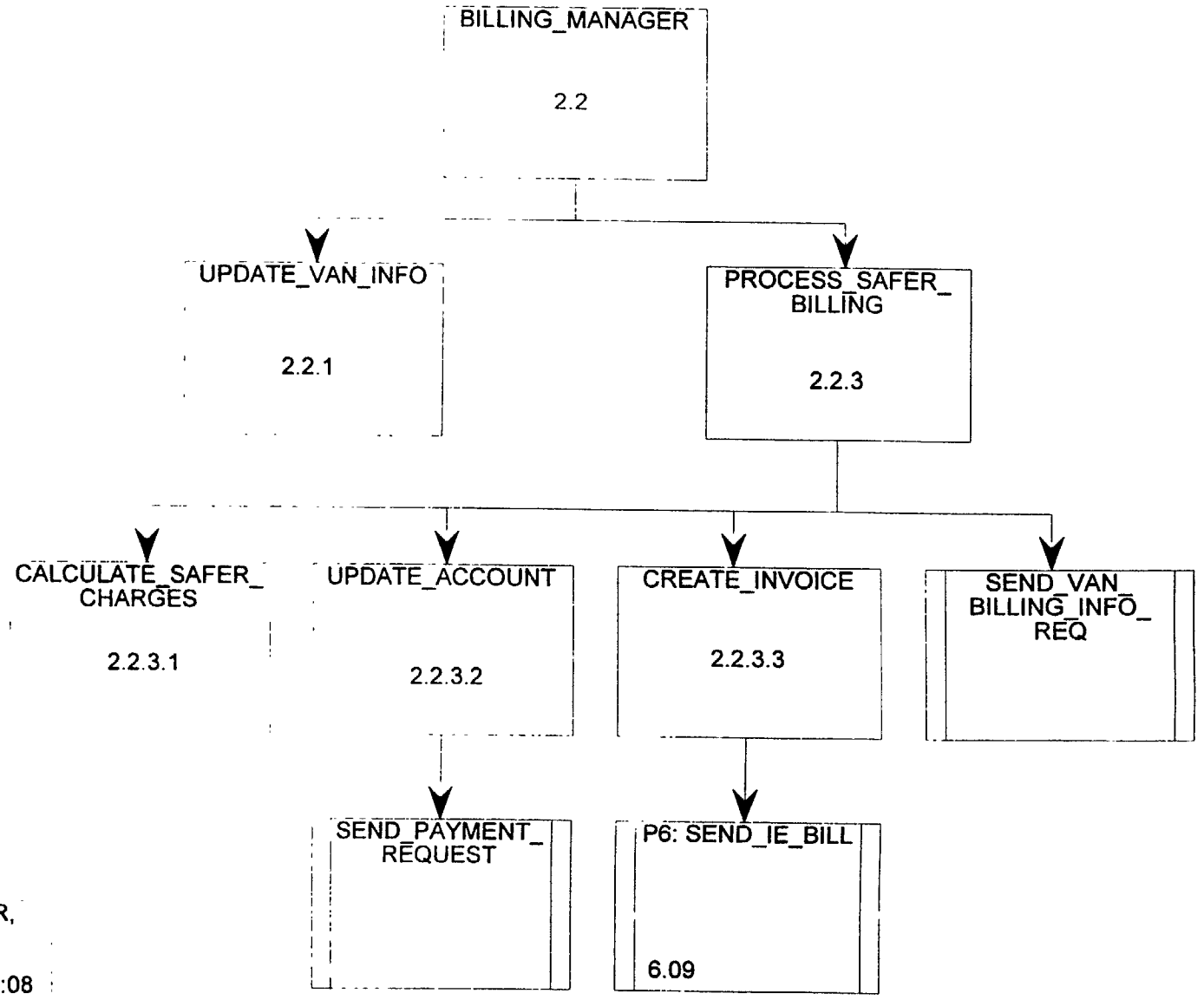
Input Variables :

Account-id

Output Variables :

Account-id	Data Element

Billing Manager



BILLING_MANAGER,
READ ONLY
System Architect
Thu May 30, 1996 14:08
Comment

2.2 BILLING-MANAGER

Return Value :

The Billing Manager is responsible for managing billing operations. It must be able to request monthly billing information from the VAN, process that information based on the resource expenditure rates of each organization/user and bill each accordingly. These functions may be provided by a Commercial-Off-The-Shelf (COTS) financial/billing product.

PDL :

```
If receive Van-billing-info then
   Call UPDATE-VAN-INFO
Endif /* receive billing info */
If receive Date-event-trigger then
   For all org ids Do
      Call PROCESS-SAFER-BILLING
   Enddo /* orgs */
Endif /* date event trigger *l
If receive Payment-transaction then
   Call UPDATE-ACCOUNT
Endif /* Payment-transaction */
```

Files Used :

Database Tables Used :

Input Variables :

Van-billing-info,
Payment-transaction,
Date-event-trigger

Output Variables :

Date-event-trigger	Data Structure
Payment-transaction	Data Structure
Van-billing-info	Data Structure

2.2.1 UPDATE-VAN-INFO

Return Value :

This module receives new Van-billing-info, updates the Van Billing Information data store, and sends a notification of the new data back to the calling module.

PDL :

```
If old van-billing-info exists in data store Van Billing Information Then
   Delete old van-billing-info
Endif /* old van info */
Write new van-billing-info to data store Van Billing Information
Set FLAG-NEW-VAN-INFO = TRUE
```

Files Used :

Database Tables Used :

Input Variables :

Van-billing-info

Output Variables:

Van-billing-info	Data Structure

2.2.3 PROCESS-SAFER-BILLING

Return Value :

This module calculates all charges incurred by an organization/user, and then uses that information to adjust accounts and create organization/user invoices. This module also updates account information upon receipt of a payment-transaction.

PDL :

```
If receive payment-transaction, then
   Call UPDATE-ACCOUNT
Else /* not a payment */
   Call CALCULATE-SAFER-CHARGES
   Call UPDATE-ACCOUNT
   Call CREATE-INVOICE
Endif
```

Files Used :

Database Tables Used :

Input Variables :

[Org-name I User-name]

Output Variables:

Org-name	Data Element
User-name	Data Element

2.2.3.1 CALCULATE-SAFER-CHARGES

Return Value :

Calculates charges based on successful transactions during billing cycle.

PDL :

Retrieve all org transactions from data store User Billing Account
Retrieve rate information from data store Rates
Apply rates to transactions

Files Used :

Database Tables Used :

Input Variables :

[Org-name I User-name]

Output Variables :

Debit-amount

Debit-amount	Data Element
Org-name	Data Element
User-name	Data Element

2.2.3.2 UPDATE-ACCOUNT

Return Value :

This module acts on the organization/user account balance. In the case of an incoming payment, it will credit the current account balance as well as the previous months balance. The previous months balance update facilitates identifying those organizations/users who did not pay their previous bill. In the case of bill creation, it will debit the account. If an organization is utilizing electronic funds transfer or credit card, an electronic request will be sent to the creditor.

PDL :

```
If receive incoming Payment-transaction, then
   Credit account
   Log credit in data store Credit Log
Else
   Debit account
   If electronic funds transfer or credit card user then
      Call SEND-PAYMENT-REQUEST
   Endif /* credit card, eft */
Endif
```

Files Used :

Database Tables Used :

Input Variables :

[Org-name 1 User-name]+
[Debit-amount 1 Credit-amount]

Output Variables :

Credit-amount	Data Element
Debit-amount	Data Element
Org-name	Data Element
User-name	Data Element

2.2.3.3 CREATE-INVOICE

Return Value :

This module creates and sends an invoice to an organization/user for their safer charges for the current billing cycle. If requested by the organization/user, their VAN charges may also be included on the invoice.

PDL :

Retreive organization/user location and balance information from data store Org Account
Retreive organization/user billing info from data store User Billing Account
Retreive organization/user credit info from data store Credit Log
Retreive van billing info from data store Van Billing Information
Assemble all billing information into invoice format
Store copy of invoice in data store Invoice Log
Send invoice to P6:SEND-IE-BILL

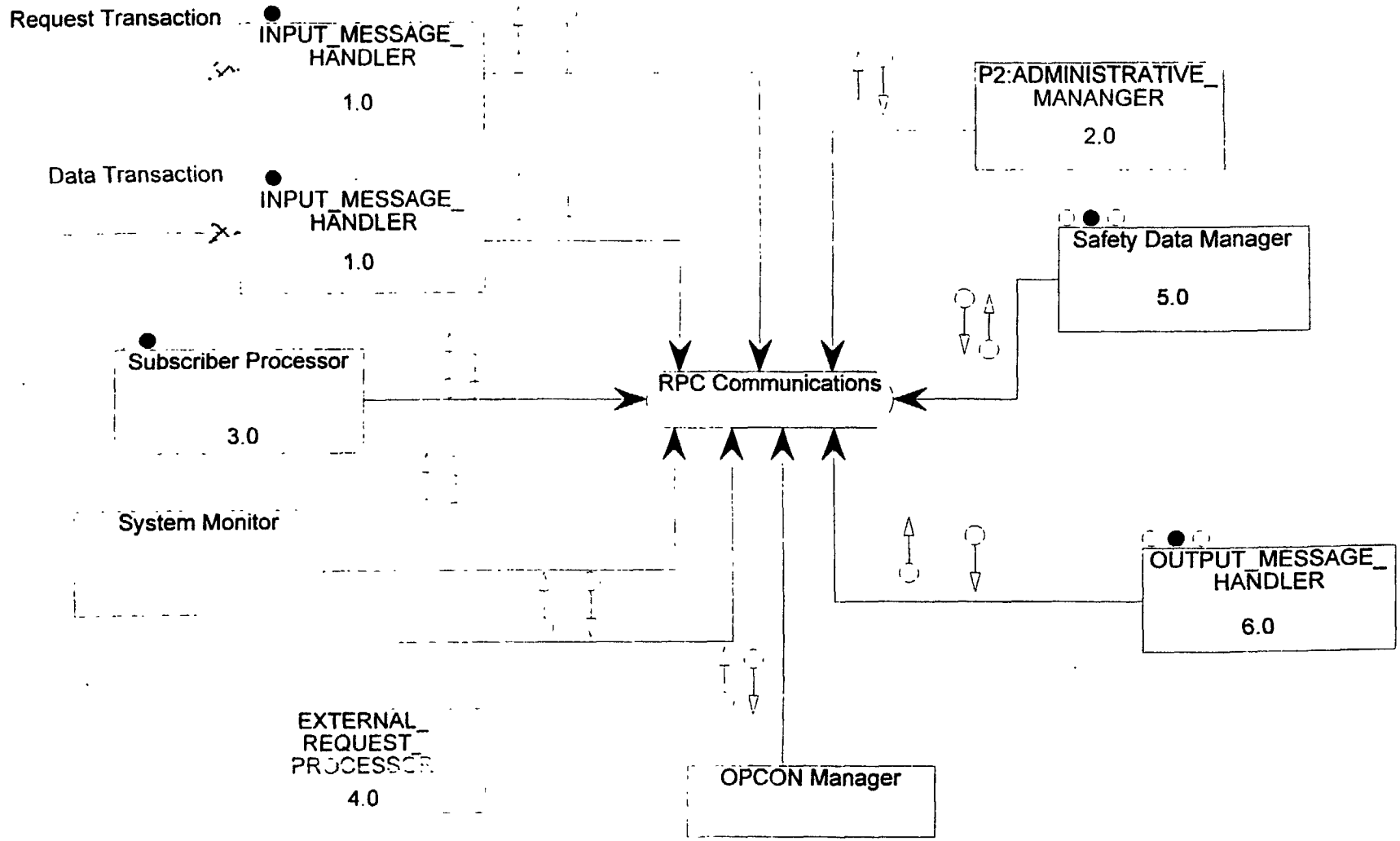
Files Used :

Database Tables Used :

Input Variables :
[Org-name I User-name]

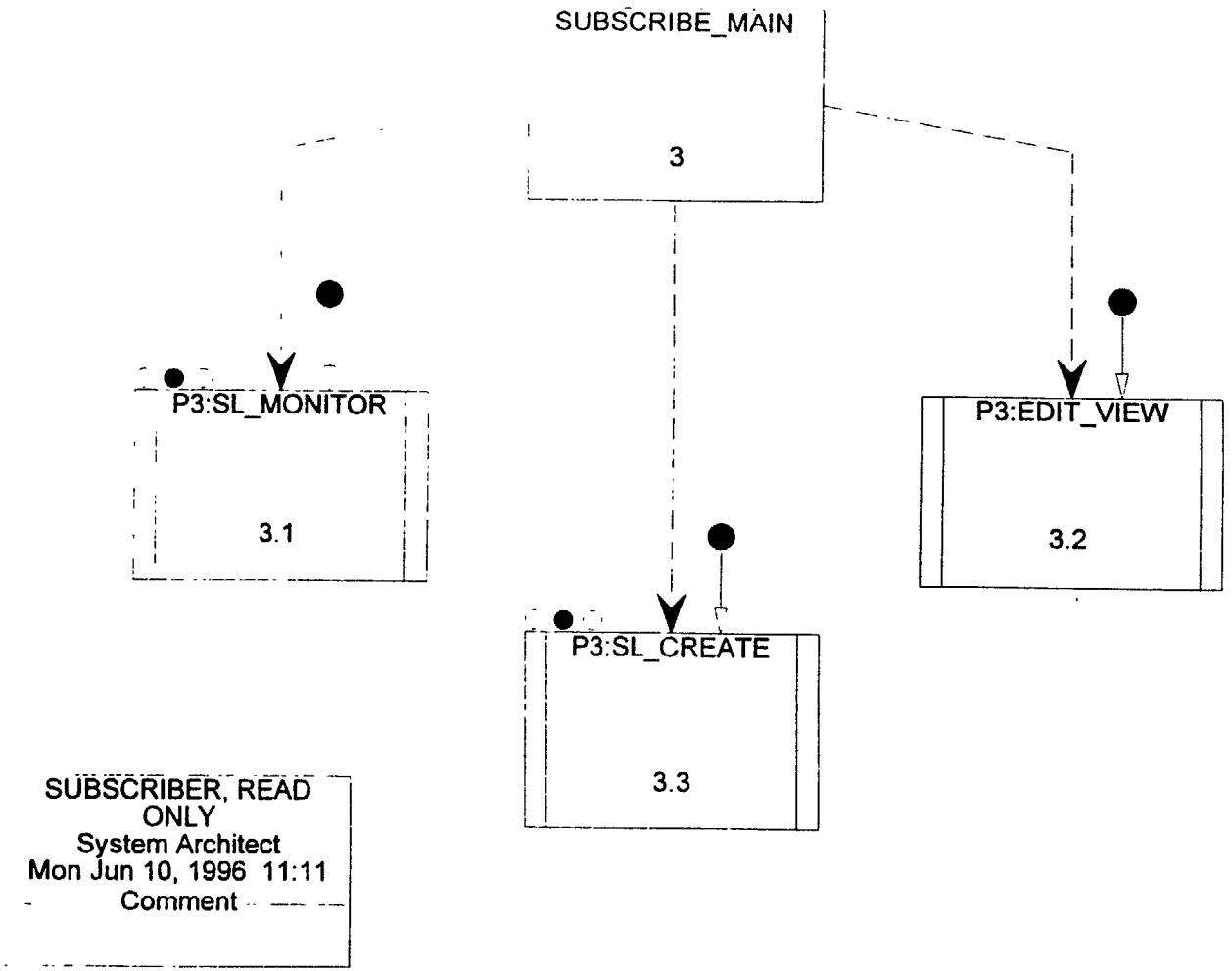
Output Variables :
User-IE-bill-msg-struct

Org-name	Data Element
User-IE-bill-msg-struct	Data Structure
User-name	Data Element



SAFER, READ ONLY
System Architect
Mon Jun 03, 1996 14:28
- - Comment

SUBSCRIBER TOP LEVEL PROCESSING STRUCTURE



3.0 SUBSCRIBE-MAIN Return Value :

This is the shell of a main control module for all processing involving the Subscriber processor. Beneath it encompasses creating subscriber and event lists, comparing old and new images of snapshots, and actually formatting safety data requests for output to users. Depending on the type of call received, it EDIT-VIEW, SL-Create or SL-Monitor.

PDL :

Listen to receive RPC from other SAFER processes.
If Receive SL-update-list from Safety data manager module
then call P3:SL-MONITOR.
If Receive Subscriber-req from Input Handler module
then call p3:SL-CREATE.
If Receive view-define-req from Input Handler module
then call P3.EDIT-VIEW.

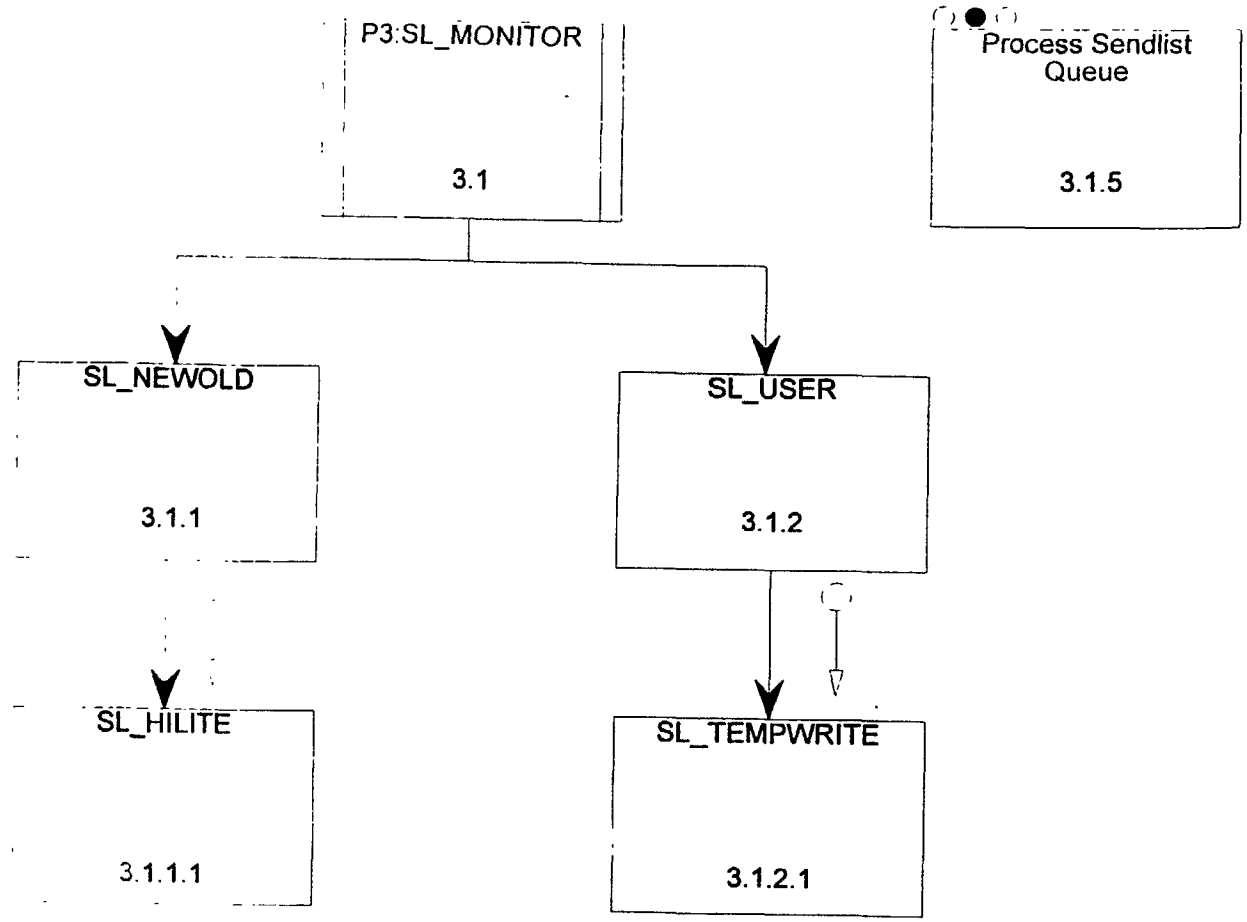
Files Used :

Database Tables Used :
Table-Snap, Table-Subscriber, Table-Event

Input Variables :

Output Variables :

SUBSCRIBER LIST MONITOR STRUCTURE



SL_MONITOR, READ ONLY
System Architect
Tue Dec 03, 1996 09:57
Comment

3.1 P3:SL-MONITOR

Return Value :

This is the RPC module which functions as the main control module for on-going subscriber list processing. It encompasses receiving snapshots, comparing old and new images of snapshots, and actually formatting safety data requests for output to users.

PDL :

```
If receive RPC call then
   call SL-newold.
   If data has changed then
      call SL-USER.
   endif /* data has changed */
endif
```

Files Used :

Database Tables Used :
Table-Subscriber, Table-Snap

Input Variables :
Packet-header +
Req-header +
SL-user-seq +
Event-status-flag +
SL-process-type +
SL-output-type +
SL-baseline-data +
[Carrier-unique-id] I
Query-criteria] +
(Event-name)+
SL-update-complete-flag+
SL-user-ready-flag+
(Snap-data-record-struct)

Output Variables :
Packet-header +
SL-user-seq +
SL-status-flag +
Req-header +
SL-process-type +
SL-output-type +
SL-process-date +
SL-baseiine-data +
[(Carrier-unique-id)]
Query-criteria] +
{Event-name} +
Packet-header +
SL-create-date + Event-update-source +
Event-update-date +
Event-mask

Carrier-unique-id	Data Element
Event-mask	Data Element
Event-name	Data Element
Event-status-flag	Data Element
Event-update-date	Data Structure
Event-update-source	Data Element
Packet-header	Data Structure
Query-criteria	Data Structure
Req-header	Data Structure
SL-baseline-data	Data Element
SL-create-date	Data Element
SL-output-type	Data Element
SL-process-date	Data Element
SL-process-type	Data Element
SL-status-flag	Data Element
SL-update-complete-flag	Data Element
SL-user-ready-flag	Data Element
SL-user-seq	Data Element
Snap-data-record-struct	Data Structure

3.1.1 SL-NEWOLD

Return Value :

This module compares a new and old snapshot of a carrier. From the fields which have changed, it determines which events have triggered, and defines the change highlight flags/event mask.

PDL :

```
If only new version is present,
  Do /*an add*/
  set all event mask fields to ON and
  Write to working data store (carrier events).
  Enddo /*an add*/
If both versions are present /*old and new*/
Then For each event name in event set
  Do /*event list*/
  If new event field != old event field and event-type = change,
    then mark the event ON in the event mask. Set hilite flag
  If new event field != old event field and event-type = interval,
    Do /* interval type*/
      Then determine the old and new intervals.
      If the intervals are different,
      then mark the event ON in the event mask. Set hilite flag
    Enddo /*interval type*/
  End /*event list*/
```

Files Used :

Database Tables Used :
Table-New-events-working, Table-Snap

Input Variables :
SL-ID +
[{Carrier-unique-id}]
Query-criteria] +
(Event-name) +
Event-mask +
{Snap-data-record-struct}

Output Variables :
SL-ID +
SL-user-seq +
Req-header +
SL-process-type +
SL-output-type +
[{Carrie' r-unique-id} I
Query-criteria] +
{Event-name) +
Event-mask

Carrier-unique-id	Data Element
Event-mask	Data Element
Event-name	Data Element
Query-criteria	Data Structure
Req-header	Data Structure
SL-ID	Data Element
SL-output-type	Data Element
SL-process-type	Data Element
SL-user-seq	Data Element
Snap-data-record-struct	Data Structure

3.1.1.1 SL-HILITE

Return Value :

This module stores the updated SAFER change highlight flags (event mask) back into the snapshot table.

PDL :

```
For each event in event list
 Do
   If new event field = ON
     then mark the event ON in the event mask. Set hilite flag
   If new event field != old event field and event-type = interval,
     Do /* interval type*/
       Then determine the old and new intervals.
       If the intervals are different,
       then mark the event ON in the event mask. Set hilite flag
     Enddo /*interval type*/
 Enddo /*event list*/
Write updated hilite fileds to Snap data store.
```

Files Used :

Database Tables Used :

Table-Snap

Input Variables :

Req-header +
Packet-header +
Carrier-unique-id +
Snap-create-date +
{Hilite-field}

Output Variables :

Carrier-unique-id +
Snap-create-date +
{Hilite-field}

Carrier-unique-id	Data Element
Hilite-field	Data Element
Packet-header	Data Structure
Req-header	Data Structure
Snap-create-date	Data Element

.

3.1.2 SL-USER

Return Value :

This module takes the new snapshot, and determines if a subscriber list is monitoring that event. If so, it evaluates the 'where' clause of the subscriber list against the characteristics of the carrier.

PDL :

```
For each event mask in working data store
  Do /*working data*/
  Retrieve each SL-event-mask from temp data store.
  For each subscriber list in subscriber data store
    Do /*subscribers*/
    Extract a Subscriber wantiist from the data store.
    Compare event masks from working data store and wantlist.
    Write entry to Sendiist Queue (Call SL-TEMPWRITE).
    Enddo /*subscribers*/
  Enddo /*working data*/
Set SL-user-ready-flag= ON
Delete working data store
```

Files Used :

Database Tables Used :

Table-New-events-working

Input Variables :

SLJD +
SL-user-seq +
Req-header+
SL-process-type +
SL-output-type +
[{Carrier-unique-id) I
Query-criteria] +
{Event-name) +
Event-mask

Output Variables :

SL-ID +
SL-user-seq +
Req-user-id +
SL-process-date +
(Carrier-unique-id

Carrier-unique-id	Data Element
Event-mask	Data Element
Event-name	Data Element
Query-criteria	Data Structure

Req-header	Data Structure
Req-user-id	Data Element
SL-ID	Data Element
SL-output-type	Data Element
SL-process-date	Data Element
SL-process-type	Data Element
SL-user-seq	Data Element

3.1.2.1 SL-TEMPWRITE

Return Value :

This module writes entries to the temporary table (Sendlist Queue) which describes outputs that need to be sent as a result of subscriber list events.

PDL :

For each user event received
Do /*both on*/
If SL-process-type 0, then set user's SLqrocess-date to TODAY
If SL-process-type >0, then if SL-process-date< TODAY
Then re-compute SL-process-date and write to data store via SL-reset-date.
If Match-type = carrier ID, then compare working carrier id to all in wantlist.
If a match, then write entry to Sendlist data store via SL-matchlist.
If match-type = field value, then perform appropriate Query-number criteria to the working
data.
If a match, then write entry to Sendlist data store via SL-matchlist.
Enddo /*both on*/

Files Used :

Database Tables Used :
Table-Sendlist-queue

Input Variables :
SL-ID +
SL-user-seq +
Req-user-id +
SL-process-date +
{Carrier-unique-id }

Output Variables:
Req-user-id +
SL-process-date +
SL-ID +
SL-user-seq +
{Carrier-unique-id }

Carrier-unique-id	Data Element
Req-user-id	Data Element
SL-ID	Data Element
SL-process-date	Data Element
SL-user-seq	Data Element

3.1.4.99 PROCESS SENDLIST-QUEUE

Return Value :

PDL :

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

3.1.5 Process Sendlist Queue

Return Value :

This module reads through the sendlist table, and sends data if entry has date >= current date.

PDL :

```
BEGIN
   START:
      Wait until time to execute then
         Call SL-DATAREQ
   GOTO START
END
```

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

3.1.5.1 SL-DATAREQ

Return Value :

This module takes the required fields and formats a standard RPC Safety Data Request to be sent to P5. The safety data request is fromatted exactly as if it came in the front door from a user. Certain fields, (cost check, fuzzy match, etc are hard coded to appropriate values).

PDL :

```
For each user in Sendlist Queue data store.
  Do /*sendlist Queue*/
  Extract from the Subscriber data store various required user request fields.
  If SL-process-date = TODAY
    Then do /*process date today*/
    Extract various required user request fields from the Subscriber data store.
    Construct a system request.
    Set Fuzzy-select = OFF
    Set Fuzzy-limit = 0
    Set Cost-check = OFF
    Set Value-match-data-type =SNAP
    Set Query-number = 1
    Set {Query-parm} = (Carrier-unique-ID)
    Set SDR-type = SL-output-type
    Issue RPC to Send a Safety data request to P5:Retrieve-Safety-data.
    Format a message for internal logging noting Safety-data-req
    Delete user entry in Sendlist Queue
    Enddo /*process date today*/
  Enddo /*Sendlist Queue*/
```

Files Used :

Database Tables Used :
Table-Sendlist-queue, Table-Subscriber

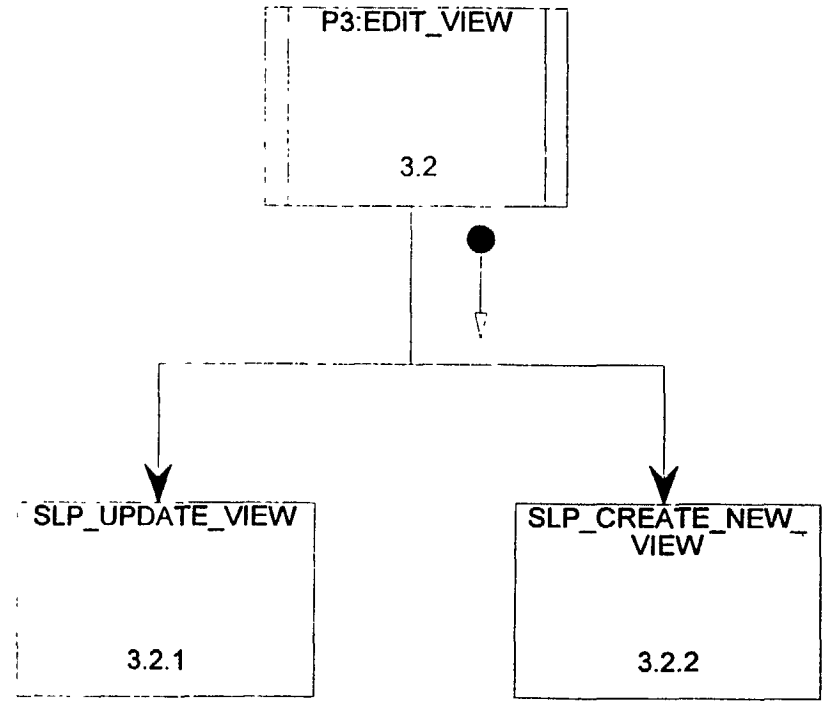
Input Variables :
Req-header +
Packet-header+
SDR-type +
Cost-check +
Value-match-data-type +
Fuzzy-select +
Fuzzy-limit +
Query-criteria

Output Variables :
Packet-header+
SDR-type +
Req-header +
Cost-check +
Value-match-data-type +

Fuzzy-select +
Fuzzy-limit +
Query-criteria

Cost-check	Data Element
Fuzzy-limit	Data Element
Fuzzy-select	Data Element
Packet-header	Data Structure
Query-criteria	Data Structure
Req-header	Data Structure
SDR-type	Data Element
Value-match-data-type	Data Element

VIEW PROCESSING STRUCTURE



EDIT_VIEW, READ ONLY
System Architect
Mon Jul 01, 1996 09:28
- Comment

3.2 P3:EDIT-VIEW

Return Value :

This module creates and maintains the SAFER view table. This is the table which indicates which snapshot fields are being monitored for changes and possible transmission to users in conjunction with a Subscriber list.

PDL :

```
Receive RPC call
if received view name exists in view table then
   call slp-update-view
else
   call slp-create-new-view
endif
```

Files Used :

Database Tables Used :

Input Variables :

Output Variables:

3.2.1 SLP-UPDATE-VIEW

Return Value :

PDL :

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

3.2.1.1 EVENT-LISTMOV

Return Value :

This is a basic routine to get event list data out of the database and into the correct event processing application format. It is possible that this could not be an actual module, but SQL, data base 'find ' with a view, etc. It is important, however that it be done in a consistent manner. It is anticipated that SAFER will undergo planned changes, and ensuring that event list access is always updated correctly and completely is a must.

PDL :

For each Event list ID in the process list received,
  Do /*event id*/
  Retrieve a view of event list via event-verify.
  If an error occurs,
    Do /*an err*/
    set all error fields to ON
    Enddo /*an err*/
  Place filelds into standard system Event Iist data structure.

  Enddo /*event id*/

Files Used :

Database Tables Used:
Table-Event

Input Variables :
Packet-header +
Event-vers +
Event-update-date +
Event-update-source +
{Event-no +
Event-name +
Event-type +
Event-check-type + Event-active-flag +
Event-range-set)

Output Variables :
Event-vers +
Event-list +
Event-update-source +
{Event-check-type + Event-active-flag}

Event-active-flag	Data Element
Event-check-type	Data Element
Event-list	Data Structure
Event-name	Data Element
Event-no	Data Element

Event-range-set	Data Structure
Event-type	Data Element
Event-update-date	Data Structure
Event-update-source	Data Element
Eventgers	Data Element
Packet-header	Data Structure

3.2.2 SLP-CREATE-NEW-VIEW

Return Value :

PDL :

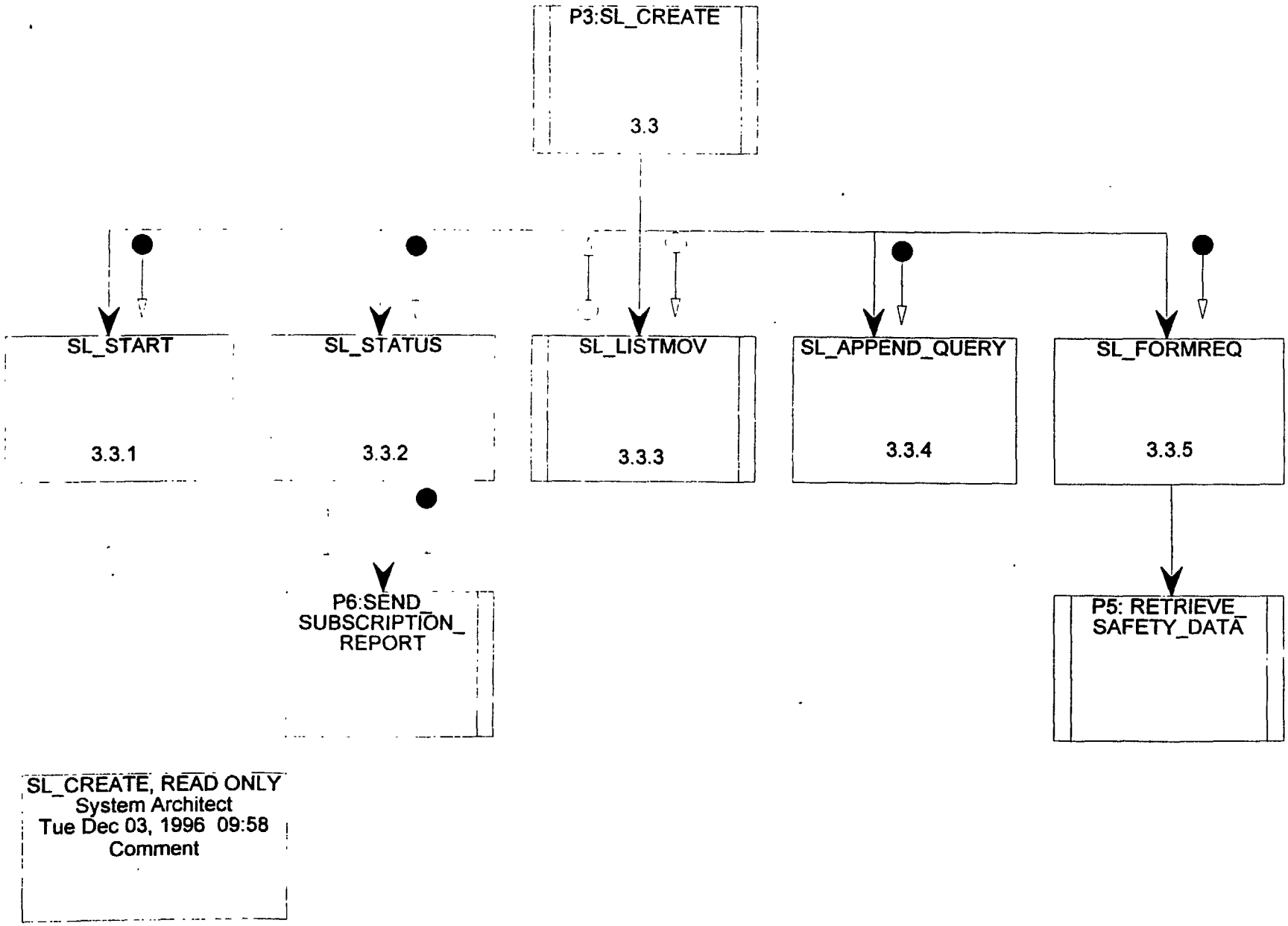
Files Used :

Database Tables Used :

Input Variables :

Output Variables :

SUBSCRIBER LIST CREATE STRUCTURE



3.3 P3:SL-CREATE

Return Value :

This is the RPC module which receives the request to create a subscription, validates the subscription request and places it into the data store.

PDL :

Receive Subscriber request fields via RPC from
Extract a current event set from the data store(Table-Event) .
If Event-status-flag = YES then
Do /*status check*/
Call SL-STATUS /* to Extract a user's current status from Subscriber data store and Format an Event status report */
End /*status check*/
Else
Do /*Create list*/
Call SL-start /* for validation */
If match type=Field value, then if Query-number specified is not allowed,
Then call PG:Send-user-message.
Zero SL-process-date.
If SL-process-type >0, then compute user's SL-process-date.
Set Event-update-source = USER
Set SL-create-date = NOW
If no error messages have been issued,
Then write the entire subscriber definition to the data store (Subscriber).
If SL-baseline-data = YES then send baseline data to SL-FORMREQ for a baseline safety data request.
End /*Create list*/

Files Used :

Database Tables Used :
Table-Subscriber, Table-Event

Input Variables :
Packet-header +
Req-header+
SL-user-seq +
Event-status-flag +
SL-process-type +
SL-output-type +
SL-baseline-data +
[(Carrier-unique-id)
Query-criteria] +
{Event-name}

OutputVariables :
Packet-header +
SL-user-seq +
SL-status-flag+
Req-header +

SL-process-type +
SL-output-type +
SL-process-date +
SL-baseline-data +
[{Carrier-unique-id}]
Query-criteria] +
(Event-name) +
Packet-header +
SL-create-date + Event-update-source +
Event-update-date +
Event-mask

Carrier-unique-id	Data Element
Event-mask	Data Element
Event-name	Data Element
Event-status-flag	Data Element
Event-update-date	Data Structure
Event-update-source	Data EIement
Packet-header	Data Structure
Query-criteria	Data Structure
Req-header	Data Structure
SL-baseline-data	Data Element
SL-create-date	Data Element
SL-output-type	Data Element
SL-process-date	Data Element
SL-process-type	Data Element
SL-status-flag	Data Element
SL-user-seq	Data Element

3.3.1 SL-START

Return Value :

This is the module which validates the subscription request.

PDL :

```
For each event-name in the request /* events in request */
 Do
   For each event-name in the event set /* All possible */
     Do /* try to match events*/
       Compare event name to name from the event set.
       If a match, mark in the event mask.
       If no match, write user error message.
     Enddo /* try to match events*/
  Enddo /* All events in request */
 If match-type = carrier ID, then
   Do for all carrier IDS user specified /*numeric check*/
     verify that all entries are numeric.
     If not numeric, format a user error message.
   Enddo /*numeric check*/
```

If match type = Field value, then if Query-no specified is not allowed, then write user error message

Files Used :

Database Tables Used :
Table-Subscriber

Input Variables :
Packet-header +
Req-header+
SL-user-seq +
Event-status-flag +
SL-process-type +
SL-output-type +
SL-baseline-data +
[(Carrier-unique-id)]
Query-criteria] +-
{Event-name}

Output Variables :
Packet-header +
SL-user-seq +
SL-status-flag +
Req-header +
SLgrocess-type +
SL-output-type +
SL-process-date +
SL-baseline-data +
[(Carrier-unique-id)]
Query-criteria] +

(Event-name) +
Packet-header +
SL-create-date + Event-update-source +
Event-update-date +
Event-mask

Carrier-unique-id	Data Element
Event-mask	Data Element
Event-name	Data Element
Event-status-flag	Data Element
Event-update-date	Data Structure
Event-update-source	Data Element
Packet-header	Data Structure
Query-criteria	Data Structure
Req-header	Data Structure
SL-baseline-data	Data Element
SL-create-date	Data Element
SL-output-type	Data Element
SL-process-date	Data Element
SL-process-type	Data Element
SL-status-flag	Data Element
SL-user-seq	Data Element

3.3.2 SL-STATUS

Return Value :

This module, upon request, creates and formats an Events status report, specifying the current features of the specified user subscription, and possible events to monitor. The actual sending is done via P6.

PDL :

Extract a user's current status via Subscriber-wantlist(Cal1 SL-listmove)
Format an Event status report, and
Send via RPC to P6:SEND-SUBSCRIBER-REPORT.

Files Used :

Database Tables Used :
Table-Subscriber

Input Variables :
SL-user-seq +
SL-status-flag+
Req-header

Output Variables:
Packet-header +
SL-user-seq +
SL-status-flag+
Req-header +
SL-process-type +
SL-output-type +
SL-process-date +
SL-baseline-data +
[{Carrier-unique-id) I
Query-criteria] +
(Event-name} +
Packet-header +
SL-create-date + Event-update-source +
Event-update-date +
Event-mask

Carrier-unique-id	Data Element
Event-mask	Data Element
Event-name	Data Element
Event-update-date	Data Structure
Event-update-source	Data Element
Packet-header	Data Structure

Query-criteria	Data Structure
Req-header	Data Structure
SL-baseline-data	Data Element
SL-create-date	Data Element
SL-output-type	Data Element
SL-process-date	Data Element
SL_process-type	Data Element
SL-status-flag	Data Element
SL-user-seq	Data Element

3.3.3 SL-LISTMOV

Return Value :

This is a basic routine to get subscription data out of the database and into the correct subscriber processing application format. It is possible that this could not be an actual module, but SQL, data base 'find ' with a view, etc. It is important, however that it be done in a consistent manner. It is anticipated that SAFER will undergo planned changes, and ensuring that subscription access is always updated correctly and completely is a must.

PDL :

For each Subscriber list ID in the process list received,
  Do /*subscribe id*/
  Retrieve a view of subscriber list from subscriber datastore.
  If an error occurs,
    Do /*an err*/
    set all error fields to ON
    Enddo /*an err*/
  Place fields into standard system Subscriber list data structure.

  Enddo /*subscribe id*/

Files Used :

Database Tables Used :

Table-Subscriber

Input Variables:

SL-user-seq +
SL-status-flag +
Req-header

Output Variables :

Packet-header +
SL-user-seq +
SL-status-flag +
Req-header +
SL-process-type +
SL-output-type +
SL-process-date +
SL-baseline-data +
[{Carrier-unique-id}
Query-criteria] +
{Event-name) +
Packet-header +
SL-create-date + Event-update-source +
Event-update-date +
Event-mask

Carrier-unique-id	Data Element
Event-mask	Data Element

Event-name	Data Element
Event-update-date	Data Structure
Event-update-source	Data Element
Packet-header	Data Structure
Query-criteria	Data Structure
Req-header	Data Structure
SL-baseline-data	Data Element
SL-create-date	Data Element
SL-output-type	Data Element
SL-process-date	Data Element
SL-process-type	Data Element
SL-status-flag	Data Element
SL-user-seq	Data Element

3.3.4 SL-APPEND-QUERY

Return Value :

This module appends query criteria (for example DOT # 123456) to a subscribers existing query parameter list.

PDL :

```
BEGIN
   Check for existing parameter in table subscriber-parms
   If parameter is not already in table then
      Insert new parameter into table subscriber-parms
   Endif.
END
```

Files Used :

Database Tables Used :
Table-Subscriber-Parms

Input Variables :
SL-ID

Output Variables :

SL-ID	Data Element

3.3.5 SL-FORMREQ

Return Value :

This module examines the table of snapshots to be sent and formats safety data requests for P5.

PDL :

Extract an various required user request, fields from the Subscriber data store.
Construct a system request.
Set Fuzzy-select = OFF
Set Fuzzy-limit = 0
Set Cost-check = OFF
Set Value-match-data-type =SNAP
Set Query-number = SL-match-type
Set (Query-parm) = SL-query-criteria
Set SDR-type = SL-output-type
Via RPC, Send the Safety data request fields to P5.
Format a message for internal logging noting Safety-data-req

Files Used :

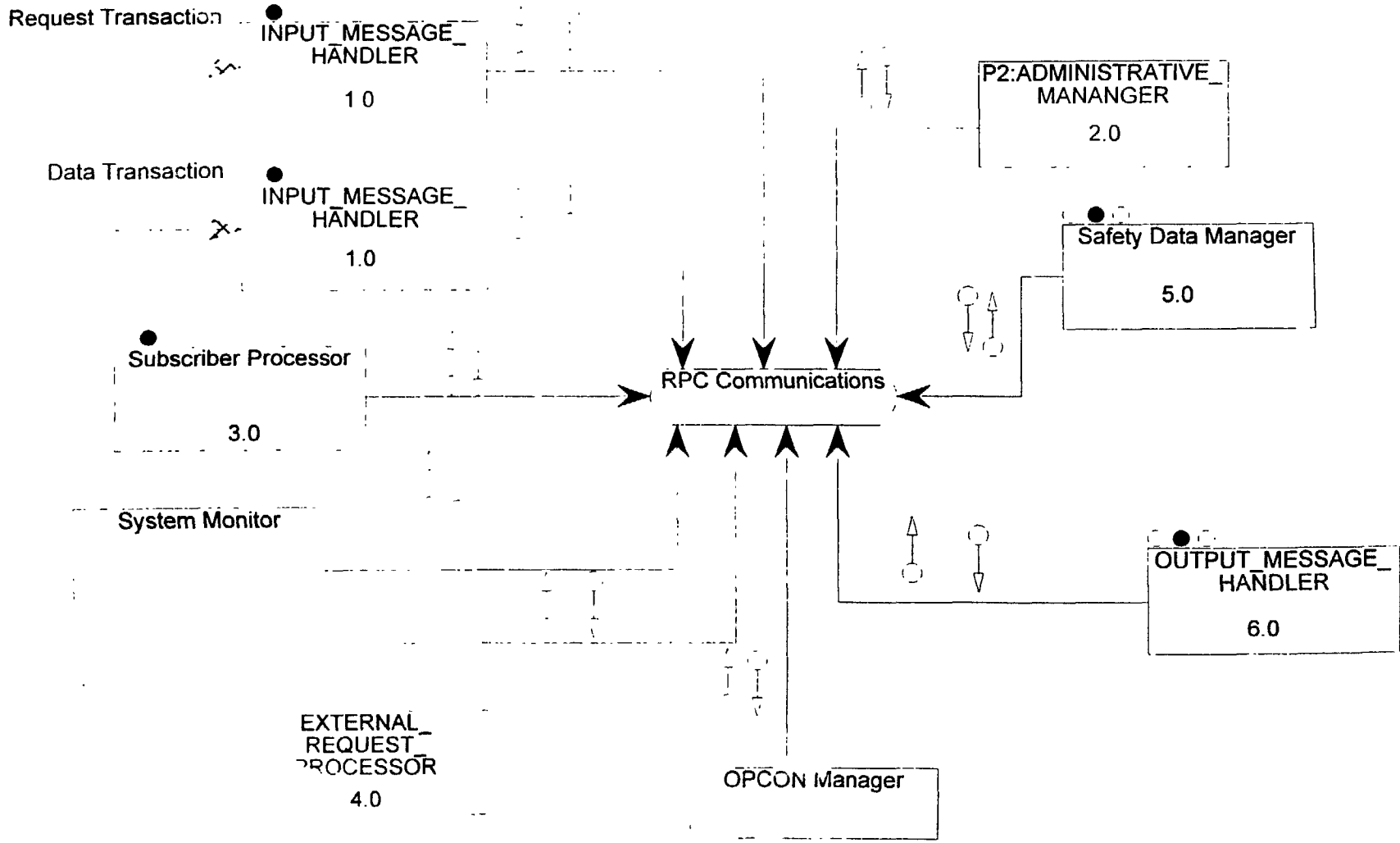
Database Tables Used :
Table-Sendlist-queue, Table-Subscriber

Input Variables :
SL-ID +
SL-user-seq +
Req-user-id +
(Carrier-unique-id}

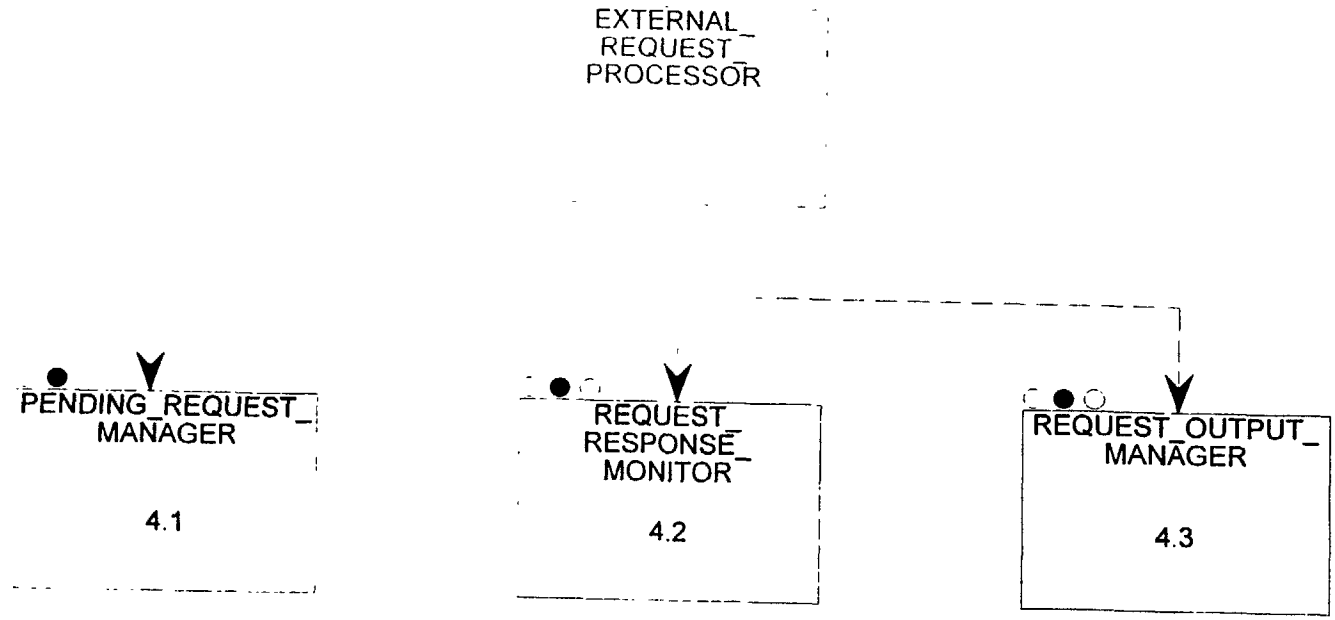
Output Variables :
Packet-header+
SDR-type +
Req-header +
Cost-check +
Value-match-data-type +
Fuzzy-select +
Fuzzy-limit +
Query-criteria

Carrier-unique-id	Data Element
Cost-check	Data Element
Fuzzy-limit	Data Element
Fuzzy-select	Data Element
Packet-header	Data Structure
Query-criteria	Data Structure

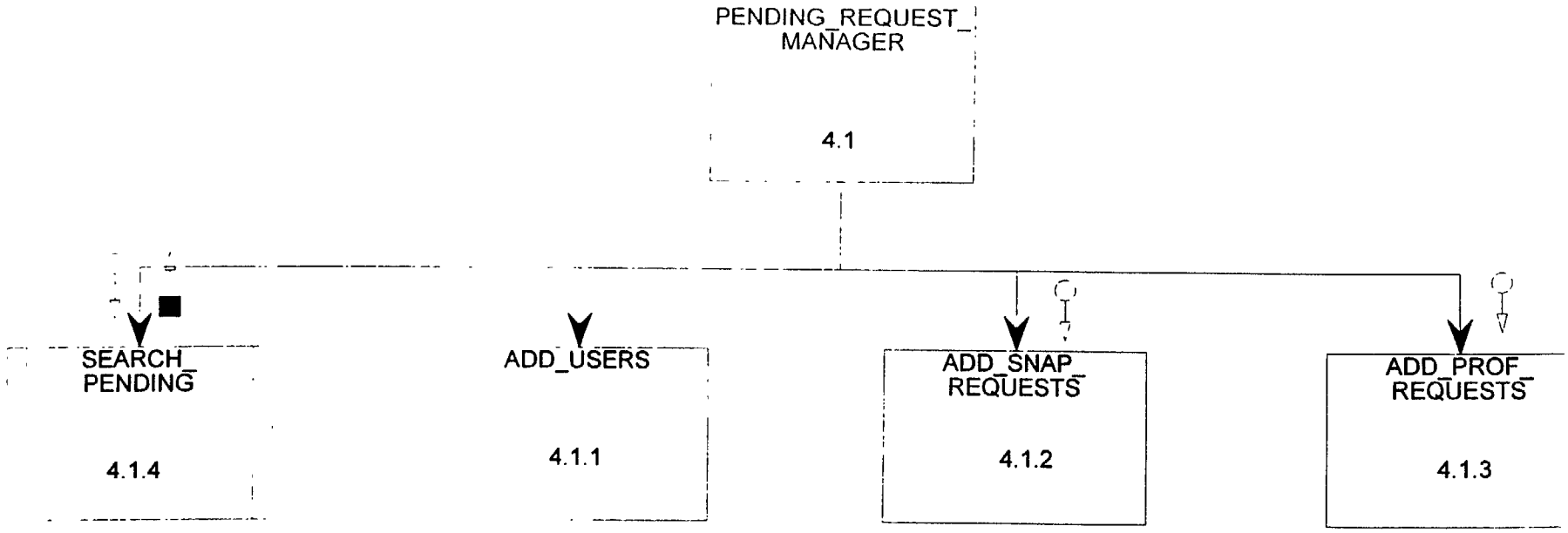
Req-header	Data Structure
Req-user-id	Data Element
SDR-type	Data Element
SLJD	Data Element
SL-user-seq	Data Element
Value-match-data-type	Data Element



SAFER, READ ONLY
System Architect
Mon Jun 03, 1996 14:28
Comment



External Request
Processor, READ ONLY
System Architect
Mon Jun 03, 1996 14:17
Comment



Pending_Request_
Manager, READ ONLY
System Architect
Mon Jun 03, 1996 13:51
Comment

4.1 PENDING-REQUESTMANAGER

Return Value :

This module accepts incoming External-req's and identifies the type of request. For carrier add/update/delete requests, this module sends these requests to the REQUEST-OUTPUT-MANAGER. For profile requests, the pending datastore is checked to verify if it will be a new request or if the user will be added to an existing request. Each snapshot request is treated as a new request and added to pending. After PENDING-REQUEST-MANAGER has verified that their are no existing requests for the carrier-unique-id's, the request information is sent to ADD-SNAP-REQUESTS or ADD-PROF-REQUESTS based on the type of request to be added.

PDL :

```
Receive External-req
  If (request type = carrier) OR (request type = event update)
    then
     send request to REQUEST-OUTPUTJIANAGER
  If (request type = profile)
    call SEARCH-PENDING passing the carrier-unique-id
    If (request information is returned from SEARCH-PENDING)
       Then
          If (request is of lower priority than incoming request)
          send request information to ADD-PROF-REQUEST
          Else send request information to -ADD-USERS
     Else send request information to ADD-PROF-REQUEST
  If (request type = snap)
    then send request information to ADD-SNAP-REQUEST
  If (request type = delete)
    then Call Pending-Delete-Requests
```

Files Used :

Database Tables Used :

Table-Pending-Profile
Table-Pending-Snapshot

Input Variables :

External-req +
Reqjnfo

Output Variables :

Carrier-req +
Profile-req +
Reqjnfo

4.1.1 ADD-USERS

Return Value :

This module is called when the pending request manager has determined that a profile request already exists for a carrier-unique-id and the user should be added to the existing user table.

PDL :

Get the request information

Put the User-info into the pending user table with the Packet-number as a reference

Files Used :

Database Tables Used :

Table-Pending-Profile

Table-Pending-User

Input Variables :

Req-info

Output Variables :

User-info +

Packet-number

Packet-number	Data Element
User-info	Data Element

4.1.2 ADD-SNAP-REQUESTS

Return Value :

This module accepts information from AS requests for snapshot data and puts it into the pending datastore. After the request information is in the datastore, associated send-req flag is set to indicate that the request output manager should send this data as a Valid-AS-req to P6.

PDL :

```
Get snapshot request information
  Call SEARCH-PENDING passing the Packet-number
  If (nothing is returned)
    Then put the request information into the request table
  Put the carrier-unique-id, and Packet number into the snapshot table
```

Files Used :

Database Tables Used :

Table-Pending-Snapshot
Table-Pending-Request

Input Variables :

Snap-req

Output Variables:

Se t-send-req

4.1.3 ADD-PROF-REQUESTS

Return Value :

This module adds profile request information to the tables in the pending datastore. After inputting the data the set-send-req flag is set to indicate that the data is ready to be formatted into a Valid-AS-req.

PDL :

```
Get Profile request information
  call SEARCH-PENDING passing the Packet-number
  If (nothing is returned)
    Then [enter request information into the Table-Pending-Request]
  enter profile information into the Table-Pending-Profile
```

Files Used :

Database Tables Used :
Table-Pending-Profile
Table-Pending-Requests

Input Variables :
Profile-req

Output Variables:
Set-send-req

4.1.4 SEARCH-PENDING

Return Value :

This module accepts a key identifier (packet-id, carrier-location-state) and returns request information for that key.

PDL :

```
Retrieve incoming key fields
  If (Packet-number)
     Then
     Get the request information for that Packet-number and
     return the data to the calling module
  If (carrier-unique-id)
    Then
     Get each requests which is pending for that carrier-unique-id
     and return the data to the calling module
```

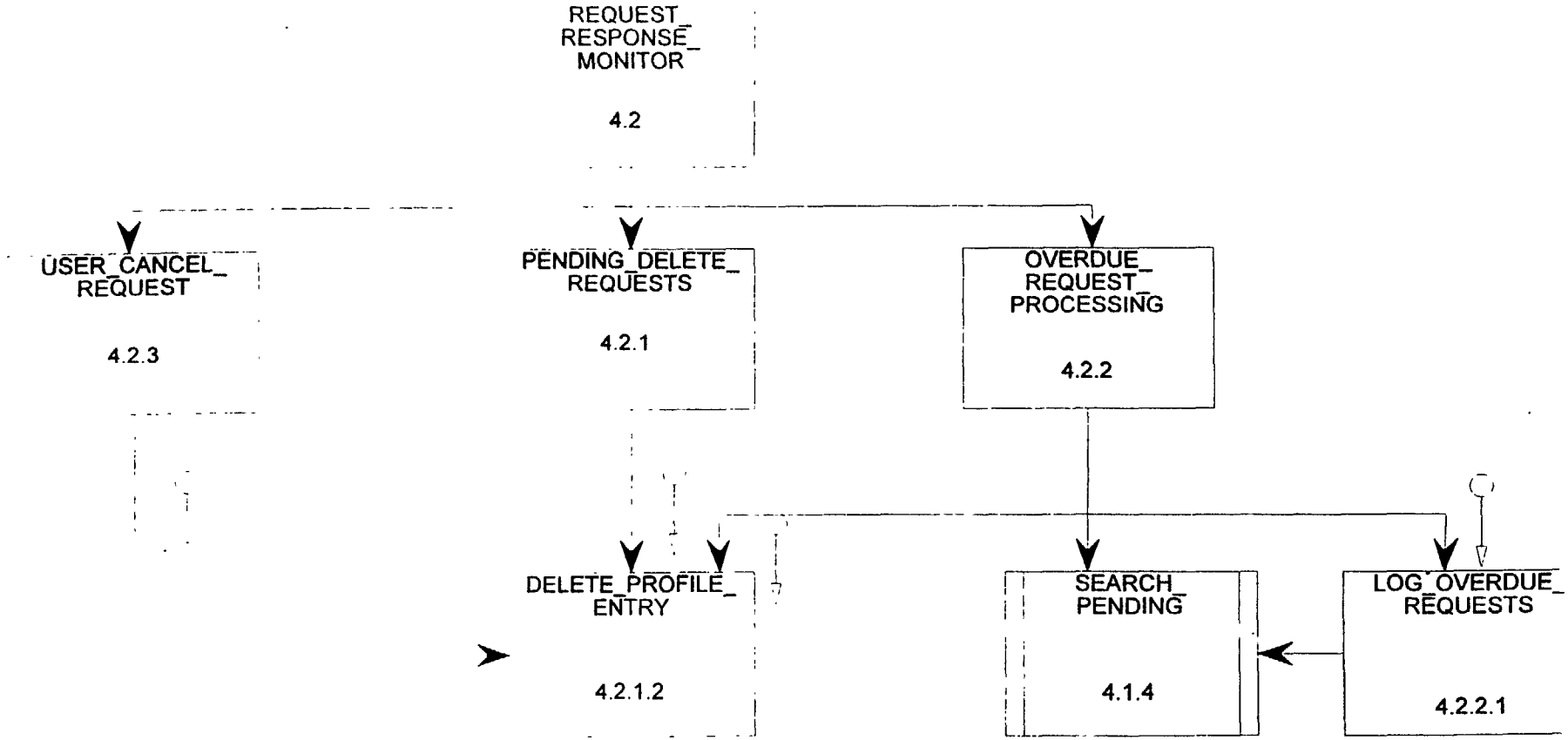
Files Used :

Database Tables Used :
Table-Pending-Profile
Table-Pending-Snapshot
Table-Pending-Request
Table-Pending-User

Input Variables:
[Packet-number I
Carrier-unique-id]

Output Variables :
Req-info

Carrier-unique-id	Data Element
Packet-number	Data Element


REQUEST_
RESPONSE_
MONITOR
4.2
USER_CANCEL_
REQUEST
4.2.3
PENDING_DELETE_
REQUESTS
4.2.1
OVERDUE_
REQUEST_
PROCESSING
4.2.2
DELETE_PROFILE_
ENTRY
4.2.1.2
SEARCH_
PENDING
4.1.4
LOG_OVERDUE_
REQUESTS
4.2.2.1

4.2 REQUEST-RESPONSE-MONITOR

Return Value :

This module monitors the status of requests for data. When the data is received the associated requests are deleted. If a request is not satisfied within an allotted time limit (specified for each As in the base state routing table), these overdue requests are logged. In addition, this module handles deletion of requests in response to a User-cancel-req.

PDL :

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

4.2.1 PENDING-DELETE-REQUESTS

Return Value :

This module accepts designation carrier-unique-id and deletes corresponding entries from the pending datastore.

PDL :

```
Receive Pending-delete-list
Do (for each carrier-unique-id)
  call SEARCH-PENDING passing carrier-unique-id
  If (data returned is of type = snap)
    then remove entries in the snap table corresponding with the received request information
  If (data returned is of type = profile)
    then send the request information to DELETE-PROFILE-ENTRY
Enddo
```

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

4.2.1.2 DELETE-PROFILE-ENTRY

Return Value :

This module accepts request packet-number, or carrier-unique-id and deletes all related entries in the pending datastore.

PDL :

```
Receive Packet-number
 call SEARCH-PENDING with the Packet-number
        delete entries from Table-Pending-Profile that
        have that packet number
              delete the corresponding request entry for that packet-number
        SEARCH-PENDING pass packet number and look for
        associated entries in the Table-Pending-Users
        if (no other requests are pending for that user)
          delete the corresponding user information
Receive Carrier-unique-id
 call SEARCH-PENDING with the Carrier-unique-id
        delete entries from Table-Pending-Profile that
        have that id
        Save packet-number for each deleted profile entry
        Do (for each packet-number saved)
            Call SEARCH-PENDING with the packet-number
            If (profile request information is not returned, i.e.
              no corresponding entries in the Table-Pending-Profile)
            Then
            delete the corresponding request entry for that packet-number
        SEARCH-PENDING pass packet number and look for
        associated entries in the Table-Pending-Users
        if (no other requests are pending for that user)
          delete the corresponding user information
```

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

4.2.2 OVERDUE-REQUEST-PROCESSING

Return Value :

This module searches through the pending file for data requests which have not been responded to within a specified time limit for that AS. When an overdue request has been identified, its packet-number is sent to LOG-OVERDUE-REQUESTS and a delete request is issued to remove it from the pending datastore.

PDL :

```
Do
Get time-tag, packet-number, and carrier-location-state from pending
Get Timeout from Base State Routing
If ((current time - time-tag) > Timeout)
  Then [issue a delete request with packet-id to DELETE-PROFILE-ENTRY
        and send the packet-number to LOG-OVERDUE-RESPONSE
        module to initiate logging and send a message to the
        SAFER operator]
  Else
        Repeat for next pending entry
```

Files Used :

Database Tables Used :

Table-Pending, Table-Base-State-Routing

Input Variables :

Time-tag +
Timeout +
Packet-number

Output Variables :

Packet-number +
User-message-info

Packet-number	Data Element
Time-tag	Data Element
Timeout	Data Element
User-message-info	Data Element

4.2.2.1 LOG-OVERDUE-REQUESTS

Return Value :

This module accepts packet-numbers for overdue requests and retrieves the request information via search pending. This request information is then output to the AS-non-response log.

PDL :

Get Packet-number
Call SEARCH-PENDING
format the request information
add Time-removed to the request
write request information out to the AS-nonresponse-log

Files Used :

AS-Nonresponse-log

Database Tables Used :

Input Variables :

Packet-number

Output Variables :

Overdue-requests

Packet-number	Data Element

4.2.3 USER-CANCEL-REQUEST

Return Value :

This module accepts User-cancel-requests and initiates the deletion of all related profile entries.

PDL :

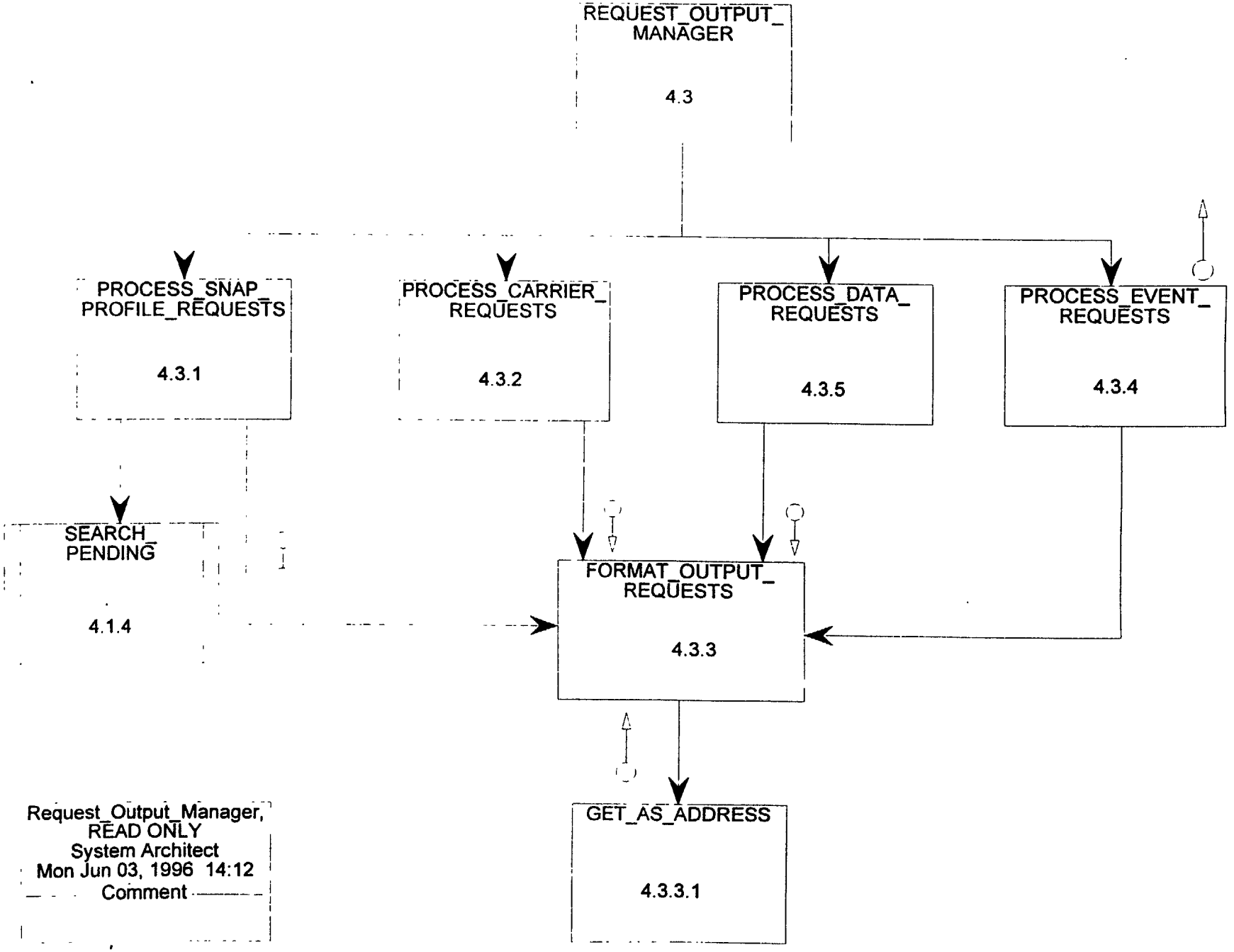
```
Get User-cancel-req
  call SEARCH-PENDING passing the User-transjd
  If request information is returned, send packet-number
  to DELETE-PROFILE-ENTRY
Send Delete-mes to P6
```

Files Used :

Database Tables Used :

Input Variables :

Output Variables :


REQUEST_OUTPUT_
MANAGER
4.3
PROCESS_SNAP_
PROFILE_REQUESTS
4.3.1
PROCESS_CARRIER_
REQUESTS
4.3.2
PROCESS_DATA_
REQUESTS
4.3.5
PROCESS_EVENT_
REQUESTS
4.3.4
SEARCH_
PENDING
4.1.4
FORMAT_OUTPUT_
REQUESTS
4.3.3
GET_AS_ADDRESS
4.3.3.1
Request_Output_Manager,
READ ONLY
System Architect
Mon Jun 03, 1996 14:12
Comment

4.3 REQUEST-OUTPUT-MANAGER

Return Value :

This module provides the formatting of requests for output by P6. In addition, it retrieves the network address associated with the specified Authoritative Source. Snapshot and Profile request informationis retrieved from the pending datastore. Carrier, event and data transfer requests are received directly.

PDL:

Files Used :

Database Tables Used :
Table-Pending-Snapshot
Table-Pending-Profile
Table-Pending-Request

Input Variables :
Carrier-req +
Req-out +
Event-update-req +
Network-address

Output Variables :
Valid-AS-req

Network-address	Data Element

4.3.1 PROCESS-SNAP-PROFILE-REQUESTS Return Value :

Indicators to send snapshot and profile requests are handled by retrieving the request information from the pending datastore based on the packet-id. The packet-id is provided to SEARCH-PENDING and the resulting datastore information is formatted and provided to FORMAT-OUTPUT REQUESTS for addition of the AS network address and sending out as a Valid-AS-request.

PDL :

Retrieve Packet-numbers for any pending datastore entries which have the send-flag set.
call SEARCH-PENDING passing the Packet-number
Send returned request information to FORMAT-OUTPUT-REQUEST

Files Used :

Database Tables Used :

Table-Pending-Snapshot
Table-Pending-Profile
Table-Pending-Request

Input Variables :

Output Variables :

Snap-request-info +
Profile-request-info

4.3.2 PROCESS-CARRIER-REQUESTS

Return Value :

This module handles incoming carrier add/update/delete requests. The incoming requests are identified, formatted, queued and sent to FORMAT-OUTPUT-REQUESTS for addition of the AS network address and sending out as a Valid-AS-request.

PDL :
Receive Carrier-req
Add priority for Carrieryeq
Identify the request type (Add/Update/Delete)
Verify priority
Send the necessary request information to FORMAT-OUTPUT-REQUEST

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

4.3.3 FORMAT-OUTPUT-REQUESTS

Return Value :

This module accepts incoming AS requests. The carrier-location-state or AS name is used to retrieve network address from the base state routing table. This address is applied to the request and a Valid-AS-request is sent to P6.

PDL :

Receive incoming request information
Format the information as expected by P6
Verify/Add request priority
Passing Carrier-location-state Call GET-AS-ADDRESS
Add the returned network address in the appropriate field
Send Valid-AS-req to P6

Files Used :

Database Tables Used :

Table-Base-State-Routing

Input Variables :

Req-info

Output Variables :

Valid-AS-req

4.3.3.1 GET-AS-ADDRESS

Return Value :

This module accepts carrier-location-state and returns the corresponding network address.

PDL :

Receive the Carrier-location-state
Search Base State Routing for the AS for that state
return the Network-address for that AS

Files Used :

Database Tables Used :

Table-Base-State-Routing

Input Variables :

Carrier-location-state

Output Variables :

Network-address

Carrier-location-state	Data Element
Network-address	Data Element

4.3.4 PROCESS-EVENT-REQUESTS

Return Value :

This module identifies incoming event-update requests and builds a request for each AS. These individual requests are formatted then sent to FORMAT-OUTPUT-REQUESTS for addition of the AS network address and sending out as a Valid-AS-request message.

PDL:

```
Receive incoming Event-update-req
Do (until end of table)
Get an AS Network-address from Base State Routing
If (the Network-address is unique)
  send the request information with that address to FORMAT-OUTPUT-REQUESTS
Enddo
```

Files Used :

Database Tables Used :

Table-Base-State-Routing

Input Variables :

Event-update-req +
Network-address

Output Variables :

Event-req-info

Network-address	Data Element

4.3.5 PROCESS-DATA-REQUESTS

Return Value :

This module handles incoming data transfer requests. The incoming requests are identified, formatted, queued and sent to FORMAT-OUTPUT-REQUESTS for addition of the AS network address and sending out as a Valid-AS-request.

PDL :

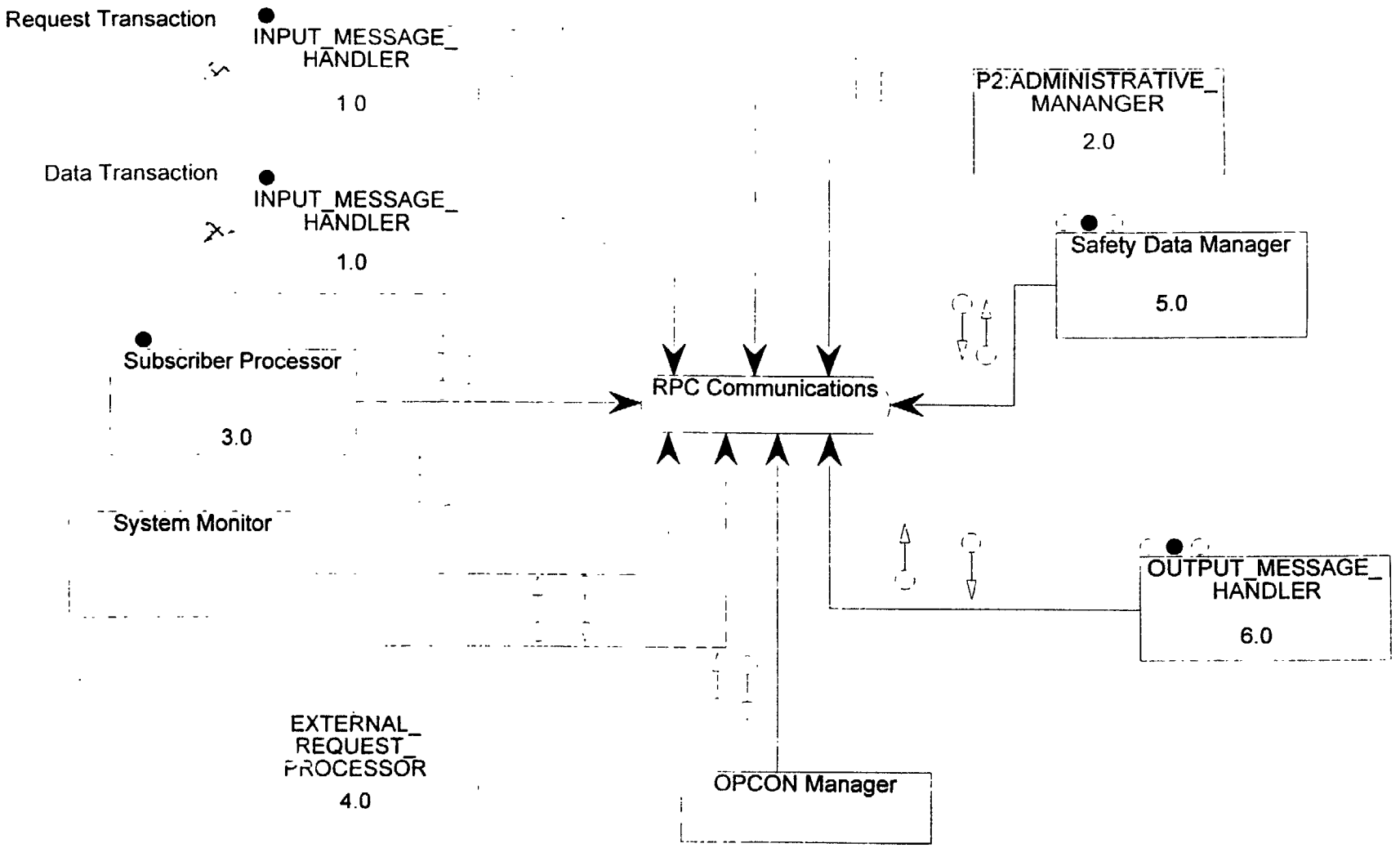
Receive Data-transfer-request
Verify priority
call FORMAT-OUTPUT-REQUEST

Files Used :

Database Tables Used :

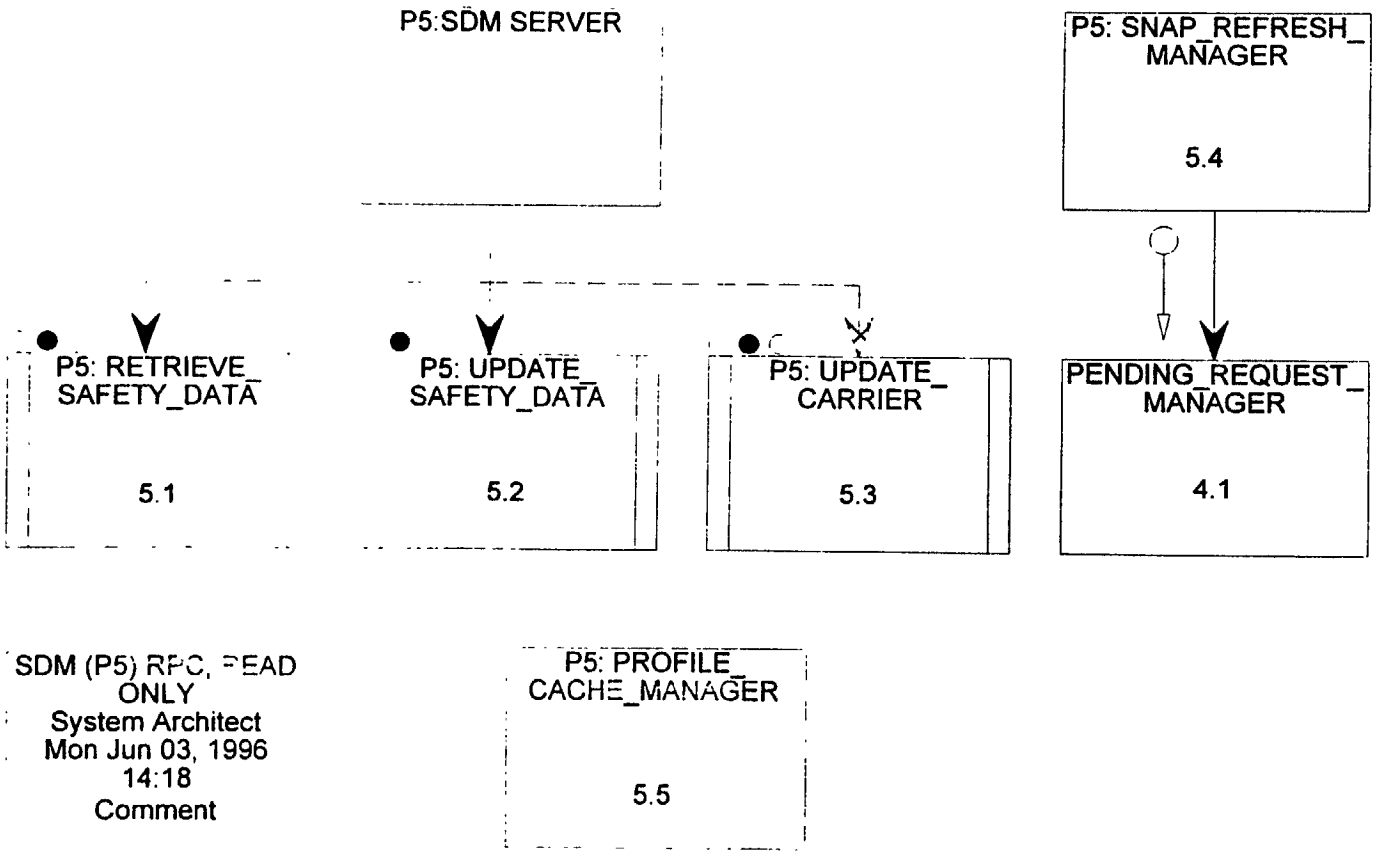
Input Variables :

Output Variables :



SAFER, READ ONLY
System Architect
Mon Jun 03, 1996 14:28
Comment

SAFETY DATA MANAGER



5.0 P5:SDM SERVER

Safety Data Manager shell.

PDL :

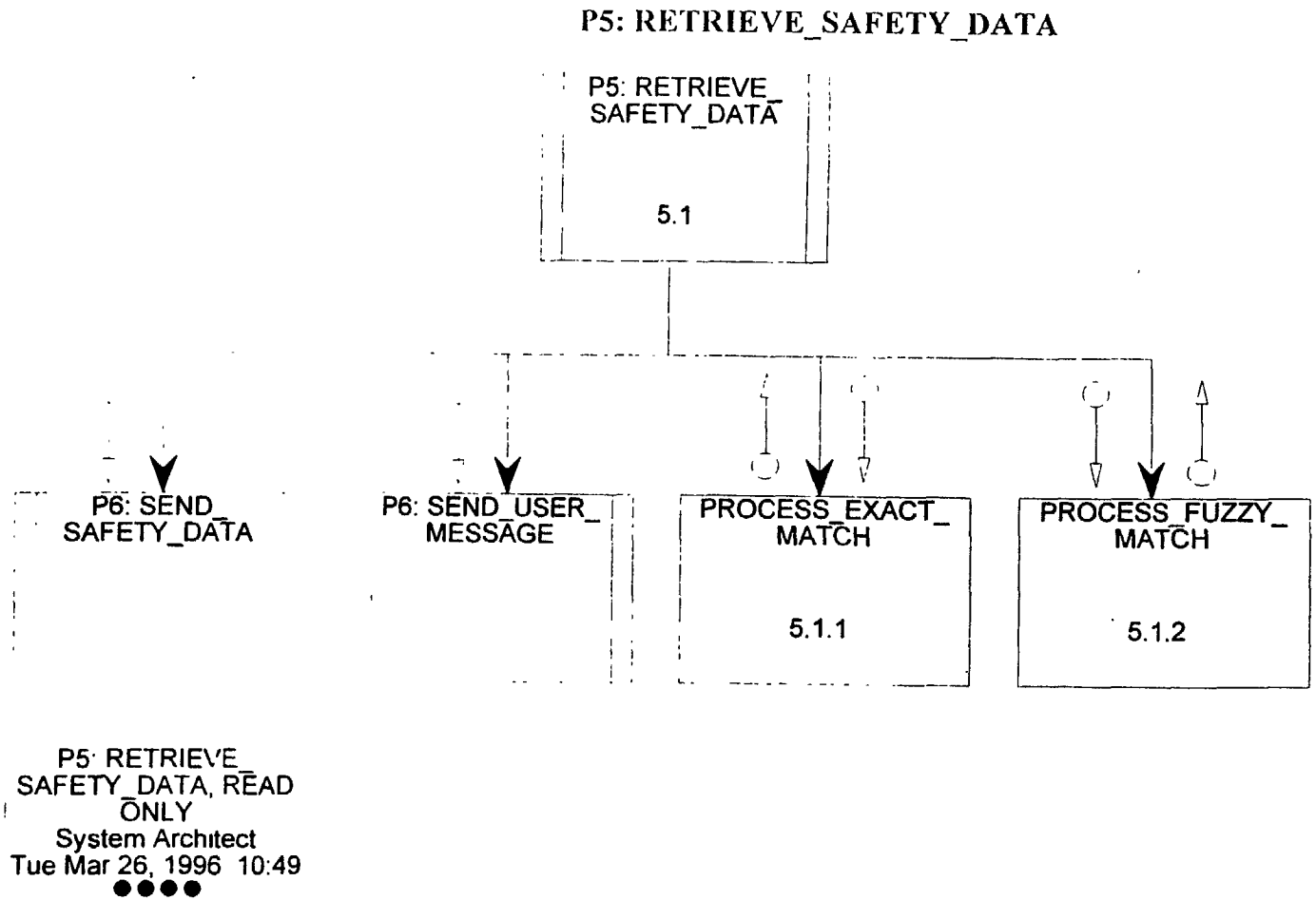
Files Used :

Database Tables Used :

Input Variables :

Output Variables :

Return Value :


P5: RETRIEVE_SAFETY_DATA
P5: RETRIEVE_ SAFETY_DATA
5.1
P6: SEND_ SAFETY_DATA
P6: SEND_USER_ MESSAGE
PROCESS_EXACT_ MATCH
5.1.1
PROCESS_FUZZY_ MATCH
5.1.2
P5: RETRIEVE_ SAFETY_DATA, READ ONLY
System Architect
Tue Mar 26, 1996 10:49

5.1 P5: RETRIEVE-SAFETY-DATA

Return Value :

RETRIEVE-SAFETY-DATA is responsible for processing all snapshot and profile request.

PDL :

```
Receive Safety-data-req
Extract SDR-type, Packet-header, Req-header, Fuzzy-select, Value-match-data-type and
    Query-criteria from Safety-data-req
Extract Req-user-id, Req-media-type and Req-priority from Req-header
Extract Query-number from Query-criteria
Load Query-criteria into Exact-match-criteria
Call PROCESS-EXACTMATCH(Exact-match-criteria ,Exact-match-results)
Extract Match-count from Exact-match-results
If Query-number = 1(USDOT) and Match-count > 0 Then
  If SDR-type = SNAP Then
    Set SDSR-type = SNAP
    Set SAFER-type = SAFETY DATA SEND REQ
    Load SAFER-type into Packet-header
    Load SDSR-type, Packet-header and Req-header into Safety-data-send-req
    Dofor all Carrier-unique-ids in Exact-match-results
      Add Carrier-unique-id to Safety-data-send-req
    Enddo
    Call P6: SEND-SAFETY-DATA (Safety-data-send-req)
  Elseif SDR-type = PROFILE Then
    Set SAFER-type = USER MESSAGE
    Load SAFER-type into Packet-header
    Set User-message-info = Match-count PROFILES FOUND FOR User-trans-id
    Call P6: SEND-USER-MESSAGE (Packet-header, User-id, User-message-info)
    Set Local-flag = FALSE
    Set AS-flag = FALSE
    Dofor all Carrier-unique-ids in Exact-match-results
      Issue Profile-exists-select to determine if profile for this carrier is in profile cache
      If Profile-exists = TRUE Then
        Add Carrier-unique-id to Safety-data-sendreq
        Set Local-flag = TRUE
      Elseif Profile-exists = FALSE Then
        Retrieve Carrier-location-state for this Carrier-unique-id from Table-Snap
        Add Carrier-unique-id and Carrier-location-state to AS-profile-req
        Set AS-flag = TRUE
      Endif
    Enddo
    If Local-flag = TRUE Then
      Set SAFER-type = SAFETY DATA SEND REQ
      Set SDSR-type = PROFILE
      Load SDSR-type, Packet-header and Req-header into Safety-data-send-req
      Call P6: SEND-SAFETY-DATA (Safety-data-send-req)
    Endif
    If AS-flag = TRUE Then
```

```
        Set AS-req-type = PROFILE
        Set AS-req-originator = SAFER
        Set AS-req-media-type = ELECTRONIC
        Set AS-req-priority = LOW
        Load AS-Req-type, AS-req-originator, AS-req-media-type and AS-req-priority
             into AS-req-header
        Set SAFER-type = AS PROFILE REQ
        Load SAFER-type into Packet-header
        Load Packet-header, AS-req-header and Req-header into AS_profile-req
        Call P4: Pending-Request-Manager (External-req) to issue AS-profile-req as External-
req
      Endif
    Endif
  Elseif Query-number > 1 and Match-count > 0 Then
    Extract Cost-check from Safety-data-req
    If Cost-check is ON Then
      Set User-id = Req-user-id from Req-header
      Set SAFER-type = USER MESSAGE
      Load SAFER-type into Packet-header
      Extract Match-count from Exact-match-results
      Calculate Estimated -transmission-cost based on Match-count and
           Value-match-data-type
      Load Match-count and Estimated-transmission-cost into User-message-info
      Set SAFER-type = USER MESSAGE
      Load SAFER-type into Packet-header
      Set User-id = Req-user-id
      Call P6: SEND-USER-MESSAGE (Packet-header, User-id, User-message-info)
    Elseif Cost-check is OFF Then
      If Value-match-data-type = SNAP Then
        Set SDSR-Type = SNAP
      Elseif Value-match-data-type = CENSUS Then
        Set SDSR-type = CENSUS
      Endif
      Set SAFER-type = SAFETY DATA SEND REQ
      Load SAFER-type into Packet-header
      Load SDSR-type, Packet-header and Req-header into Safety-data-send-req
      Dofor all Carrier-unique-ids in Exact-match-results
        Add Carrier-unique-id to Safety-data-send-req
      Enddo
      Call P6: SEND-SAFETY-DATA (Safety-data-send-req)'
    Endif
  Elseif.Query-number > 1 and Match-count = 0 and Fuzzy-select = ON Then
    Extract Fuzzy-limit from Safety-data-req
    Load Exact-match-criteria and Fuzzy-limit into Fuzzy-match-criteria
    Call PROCESS-FUZZY-MATCH (Fuzzy-match-criteria ,Fuzzy-match-results)
    Dofor all Carrier-unique-ids in Fuzzy-match-results
      Add Carrier-unique-id to Safety-data-send-request
```

```
    Enddo
  Elseif Match-count = 0 and Fuzzy-select = OFF Then
      Set User-message-info = NO EXACT MATCH FOUND
      Set SAFER-type = USER MESSAGE
      Load SAFER-type into Packet-header
      Set User-id = Req-user-id
      Call P6: SEND-USER-MESSAGE (Packet-header, User-id, User-message-info)
  Endif
```

Files Used :

Database Tables Used :

Input Variables :
Safety-data-send-struct

Output Variables :

Safety-data-send-struct	Data Structure

5.1.1 PROCESS-EXACT-MATCH

Return Value :

The PROCESS-EXACT-MATCH module is responsible for submitting ID(USDOT number) and VALUE match queries to the SAFER database.

PDL :

Files Used :

Database Tables Used :
Table-snap

Input Variables :
Exact-match-criteria

Output Variables :
Exact-match-results

5.1.1.1 SDM-REDACT-SNAPSHOT

Return Value :

This module is responsible for (when appropriate) redacting (blanking) certain data fields which a user is not authorized to receive. Examples of fields are driver name, and SSN.

PDL :

Extract the 'User-privacy from the request header.

If Req-type = 'ds' or 'ir' (driver snapshot or inspection report), then do:
If User-privacy = 'no' then set
Driver-name = "and SSN= " (must blank both)

If Req-type = 'cs' (Carrier snapshot), then do:
If User-privacy = 'n' then If Tax-ID-type = 'S' then set
Tax-ID-number = ' ' (must blank SSN-type id)

Files Used :

Database Tables Used :

Input Variables :

Safety-data-send-struct
Req-header

Output Variables :

Safety-data-send-struct

Req-header	Data Structure
Safety-data-send-struct	Data Structure

5.1.2 PROCESS-FUZZY-MATCH

Return Value :

The PROCESS-FUZZY-MATCH module is responsible for submitting FUZZY match queries to the SAFER database.

PDL :

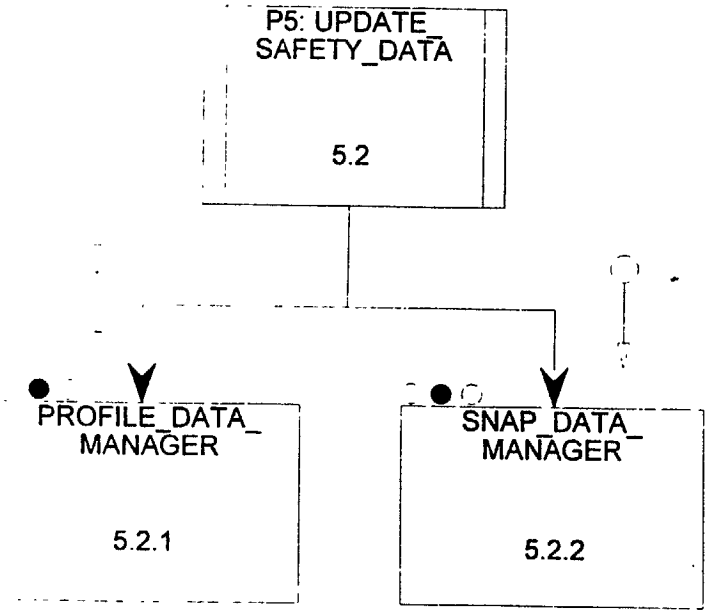
Files Used :

Database Tables Used :
Table-Snap

Input Variables :
Fuzzy-match-criteria

Output Variables :
Fuzzy-match-results

P5: UPDATE_SAFETY_DATA



P5: UPDATE_SAFETY_
DATA, READ ONLY
System Architect
Thu Oct 12, 1995 11:25
Comment

5.2 P5: UPDATE-SAFETY-DATA

Return Value :

UPDATE-SAFETY-DATA is the control module for obtaining Valid-safety-data from the IPC Message Queue and adding the data to the SAFER database.

PDL :

```
Receive Valid-safety-data
Extract Data-header from Valid-safety-data
Extract Valid-safety-data-type from Data-header
If Valid-safety-data-type = PROFILE Then
  Extract Profile-data from Valid-safety-data
  Call PROFILE-DATA-MANAGER(Profile-data) to add profiles to database
Elseif Valid-safety-data-type = SNAP Then
  Extract Snap-data from Valid-safety-data
  Call SNAP-DATA-MANAGER(Snap-data) to add snapshots to database
Endif
```

Files Used :

Database Tables Used :

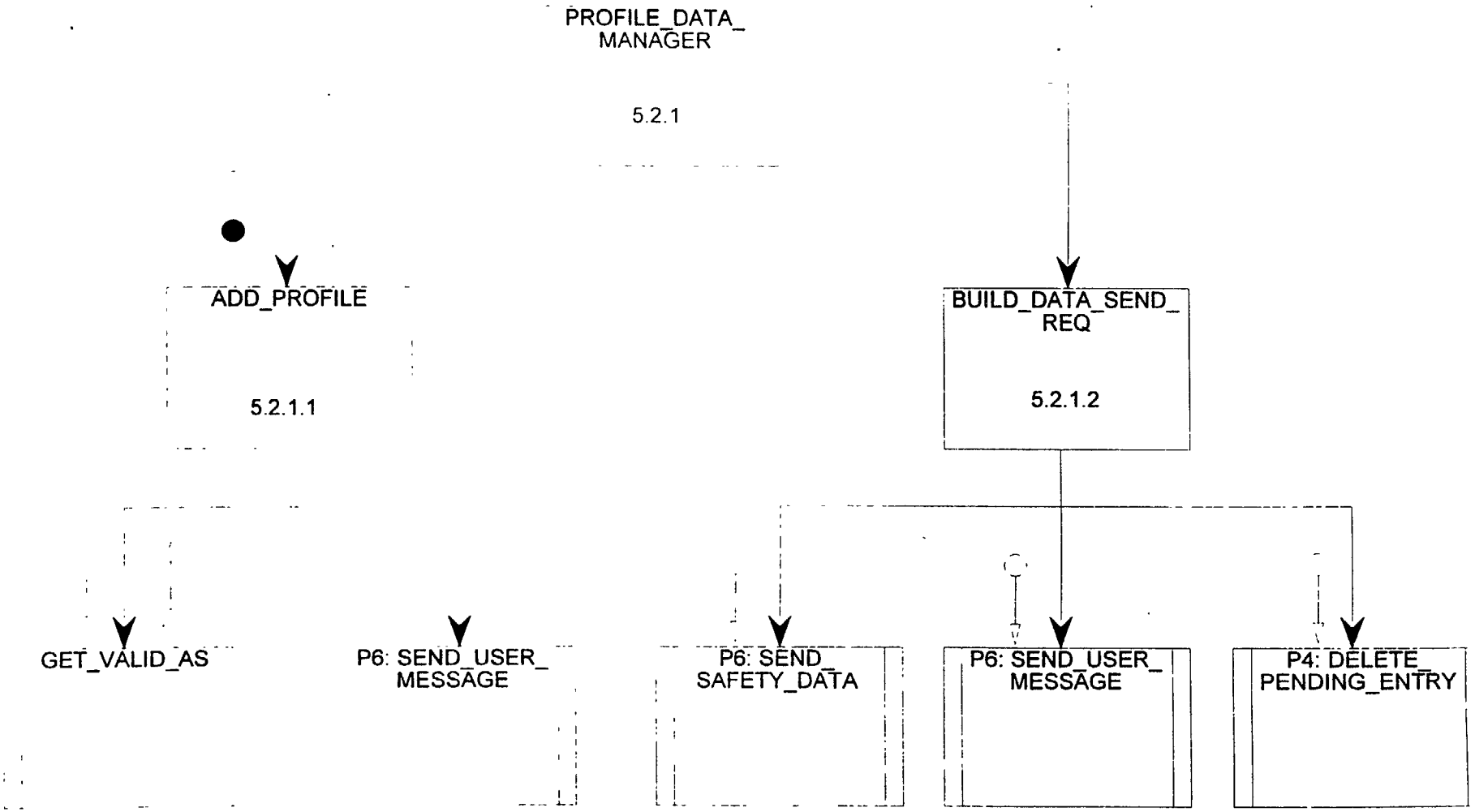
Input Variables :

Valid-safety-data-struct

Output Variables :

Valid-safety-data-strum?	Data Structure

PROFILE DATA MANAGER



5.2.1 PROFILE-DATA-MANAGER

Return Value :

PROFILE-DATA-MNAGER is the control module for adding profiles to the SAFFER database.

PDL :

```
Receive Profile-data
Set Req-type = PROFILE
Create temporary Table-Profile-User
Dofor each Profile-data-record in Profile-data
  Extract Carrier-unique-id from Profile-data-record
  Call ADD-PROFILE(Packet-header, Data-header, Profile-data-record) to verify AS and add
  profile to SAFER database
  If Return-code = SUCCESS Then
    Select Packet-header and Req-header from PENDING for this Carrier-unique-id
      Begin
        Insert Carrier-unique-id, Packet-header and Req-header into Table-Profile-User
      End
  Endif
Enddo
Call BUILD-DATA-SEND-REQUEST to send profiles to all waiting users
Delete temporary Table-Profile-User
```

Files Used :

Database Tables Used :

Table-Profile-Users

Input Variables :

Profile-data

Output Variables :

5.2.1.1 ADD-PROFILE

Return Value :

The ADD-PROFILE module adds profiles to the SAFER database.

PDL :

```
Receive Packet-header, Data-header and Profile-data-record
Extract Valid-safety-data-type and Account-id from Data-header
Extract Carrier-unique-id from Profile-data-record
Call GET-VALID-AS(Carrier-unique-id, Valid-safety-data-type, Account-id) to obtain
  Account-id from Base State Routing for this Carrier-unique-id
If Account-ids from Base State Routing and Data-header are different Then
  Set SAFER-type = USER MESSAGE
  Load SAFER-type into Packet-header
  Set User-id = Account-id from Packet-header
  Set User-message-info = INVALID AS FOR CARRIER UNIQUE ID Carrier-unique-id
  Call P6: SEND-USER-MESSAGE(Packet-header, User-id, User-message-info)
  Set Return-code = FAILURE
Else
  Insert Profile-data-record into Table-Profile
  Set Return-code = SUCCESS
Endif
Return Return-code
```

Files Used :

Database Tables Used :

Table-Profile, Table-Base-State-Routing

Input Variables :

Packet-header
Data-header
Profile-data-record

Output Variables :

Return-code

Data-header	Data Structure
Packet-header	Data Structure
Return-code	Data Element

5.2.1.2 BUILD-DATA-SEND-REQ

Return Value :

The BUILD-DATA-SEND-REQ module builds Safety-data-send-reqs for all records in the Table-Profile-Users.

PDL :

```
Initialize old values of Req-user-id, Req-overnite-flag, Req-media-type and Carrier-unique-id
Set SDSR-type = PROFILE
Set SAFER-type = SAFETY DATA SEND REQ
Select new values of Req-user-id, Req-overnite-flag, Req-media-type, Packet-header and
  Carrier-unique-id from Table-Profile-Users Order by Req-user-id, Req-overnite-flag, Req-
media-type
    Begin
    If new value of Req-user-id or Req-overnite-flag or Req-media-type differs from old
      value Then
      If old values of Req-user-id, Req-overnite-flag and Req-media-type are not
        initialization values Then
        Load SAFER-type into Packet-header
        Load Packet-header and Req-header into Safety-data-send-req
        Call P6: SEND-SAFETY-DATA (Safety-data-send-req)
      Endif
      Load SDSR-type and new values of Req-user-id, Req-overnite-flag and
        Req-media-type into Req-header
    Else (old and new values are the same)
      If old and new value of Carrier-unique-id are different Then
        Add new Carrier-unique-id to Safety-data-send-req
        Add Req-user-id and Carrier-unique-id to Pending-delete-list
      Else
        Set SAFER-type = USER MESSAGE
        Load SAFER-type into Packet-header
        Set User-id = Req-user-id
        Set User-message-info = DUPLICATE PROFILE REQUEST CANCELLED
        Call P6: SEND-USER-MESSAGE (Packet-header, User-id, User-message-info)
      Endif
    Endif
    Set old values of Req-user-id, Req-overnite-flag, Req-media-type, Packet-header
      and Carrier-unique-id to new values
    End
Set Req-type = PROFILE
Load Req-type into Pending-delete-list
Call P4: DELETE-PENDING-ENTRY (Pending-delete-list)
```

Files Used :

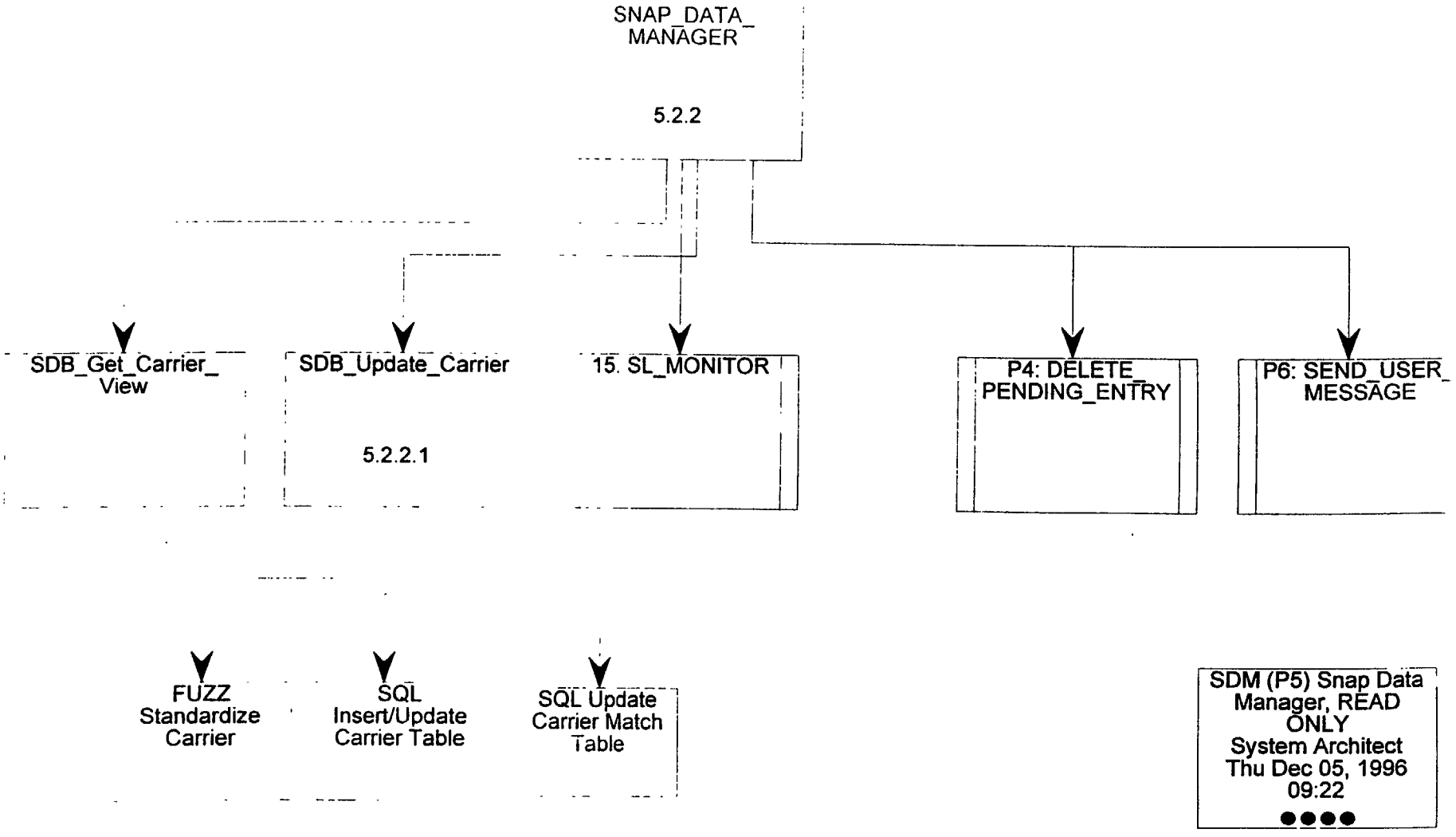
Database Tables Used :

Table-Profile-Users

Input Variables :

Output Variables :

SNAP DATA MANAGER



5.2.2 SNAP-DATA-MANAGER

Return Value :

SNAP-DATA-MANAGER is the control module for adding UPDATED and REFRESHED snapshots to the SAFER database.

PDL :

```
Receive Snap-data
Extract Packet-header and Data-header from Snap-data
Extract Account-id, Reason-for-send and Valid-safety-data-type from Data-header
Set Req-user-id = SAFER
Dofor each Snap-data-record in the Snap-data
  Extract Carrier-unique-id
  Call GET-VALID-AS(Carrieruniquejd, Valid-safety-data-type, Account-id) to obtain
    Account-id from Base State Routing for this Carrier-unique-id
  If Account-ids from Base State Routing and Data-header are different Then
    Set SAFER-type = USER MESSAGE
    Load SAFER-type into Packet-header
    Set User-id = Account-id from Data-header
    Set User-message-info = INVALID AS FOR CARRIER UNIQUE ID Carrier-unique-id
    Call P6: SEND-USER-MESSAGE (Packet-header, User-id, User-message-info)
  Else
    Call ADD-SNAP-RECORD(Reason-for-send, Snap-data-record) to add snapshot to
      SAFER database
    If Reason-for-send = UPDATE Then
      Add Carrier-unique-id to SL-update-list
    Elseif Reason-for-send = REFRESH Then
      Add Req-user-id and Carrier-unique-id to Pending-delete-list
    Endif
    Build Profile-delete-req for this Carrier-unique-id
    Issue Profile-delete-req to Profile data store to delete possible Profile-data-record that
        is not waiting to be sent to a user
  Endif
Enddo
If Reason-for-send = UPDATE Then
  Timestamp the SL-update-list
  Call P3: FULFILL-SUBSCRIPTION (SL-update-list)
Elseif Reason-for-send = REFRESH Then
  Set Req-type = SNAP
  Load Req-type into Pending-delete-list
  Call P4: DELETE-PENDING-ENTRY (Pending-delete-list)
Endif
```

Files Used :

Database Tables Used :

Table-Base-State-Routing

Input Variables :

Snap-data

Output Variables :

5.2.2.1 ADD-SNAP-RECORD

Return Value :

ADD-SNAP-RECORD adds new snapshots to the Snapshot table. If Reason-for-send is UPDATE, a copy of the old snapshot is placed in the Oldsnap table.

PDL :

```
Receive Reason-for-send, Snap-data-record
Extract Carrier-unique-id from Snap-data-record
If Reason-for-send = UPDATE Then
  Delete any existing record for this Carrier-unique-id from Table-Oldsnap
  Get Oldsnap-data-record for this Carrier-unique-id from Table-Snap
  Insert Oldsnap-data-record into Table-Oldsnap
Endif
Delete Snapshot for this Carrier-unique-id from Table-Snap
Insert new Snap-data-record for this Carrier-unique-id into Table-Snap
```

Files Used :

Database Tables Used :

Table-Oldsnap
Table-Snap

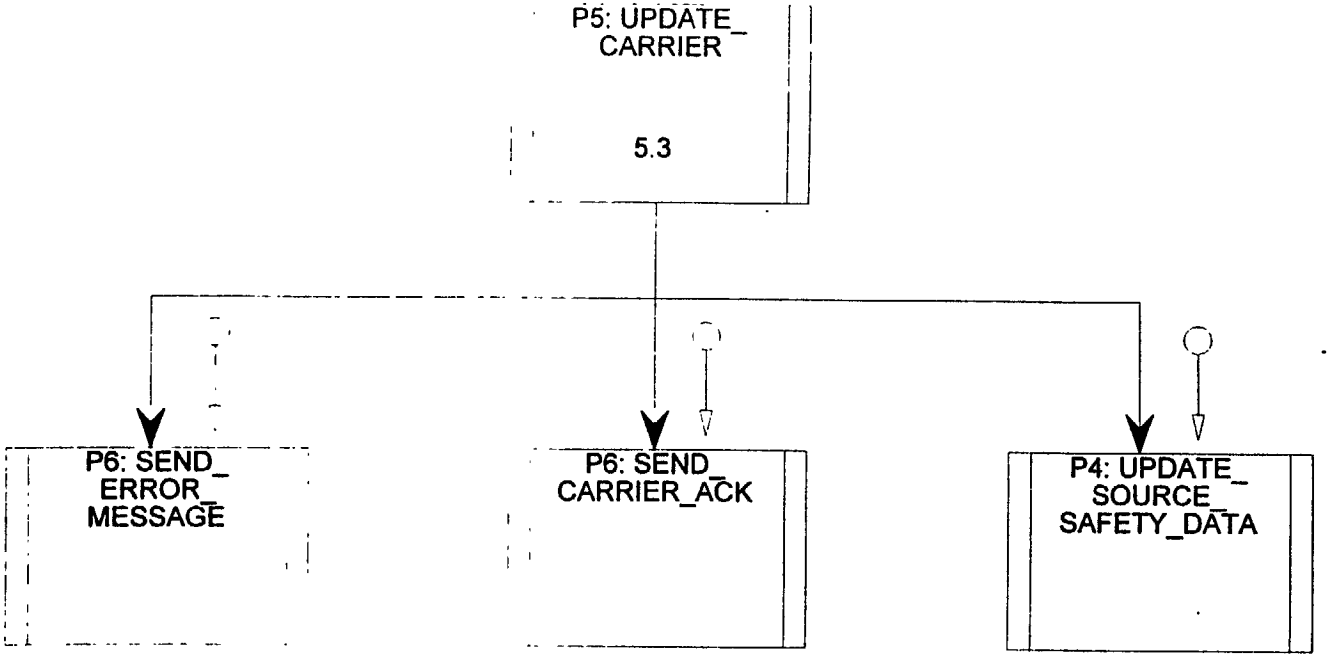
Input Variables :

Reason-for-send
Snap-data-record

Output Variables :

Reason-for-send	Data Element

P5: UPDATE_CARRIER



P5: UPDATE_
CARRIER, READ ONLY
System Architect
Thu Oct 12, 1995 11:42
-- Comment --

5.3 P5: UPDATE-CARRIER

Return Value :

UPDATE-CARRIER is responsible for processing all Carrier-reqs.

PDL :

```
Receive Carrier-req from Pl
Extract Carrier-req-type and Req-header from Carrier-req
If Carrier-req-type = ADD Then
  Generate new Carrier-unique-id
  Extract Carrier-info and Packet-header from Carrier-req
  Extract the Census-data-item-values from the Cairier-info for data fields that are common
to
      carrier and census
  Set Carrier-status = PENDING
  Load Carrier-unique-id, Census-data and Carrier-status into Census-data-record
  Insert Census-data-record into Census area of Snap data store
  If insert was successful Then
    Set Message-text = NEW USDOT GENERATED
    Load Packet-header, Req-header, Message-text and Carrier-unique-id into Carrier-ack
    Issue Carrier-ack to P6
    Extract Carrier-location-state from Carrier-info
    Set AS-req-type = CARRIER
    Set AS-req-originator = SAFER
    Set AS-req-media-type = ELECTRONIC
    Set AS-req-priority = LOW
    Load AS-req-type, AS-req-originator, AS-req-media-type and AS-req-priority into
        AS-req-header
    Set Process-id = P5
    Generate next P5 Packet-number
    Load Process-id and Packet-number into Packet-header
    Load Packet-header, AS_req_header, Req-header, Carrier-loaction-state.
        Carrier-req-type, Carrier-unique-id and Carrier-info into AS-carrier-req
    Issue AS-carrier-req as External-req to P4
    Insert AS-carrier-req as Internal-transaction into Internal Transaction Log data store
  Else
    Set Error-message = NEW USDOT REQUEST UNSUCCESSFUL
    Issue Error-message to Error-log
  ENDIF
Else (UPDATE or DELETE)
  Extract Carrier-unique-id from Carrier-req
  Retrieve Carrier-location-state for this Carrier-unique-id from the Census area of the Snap
      data store
  Set AS-req-type = CARRIER
  Set AS-req-originator = SAFER
  Set AS-req-media-type = ELECTRONIC
  Set AS-req-priority = LOW
  Load AS-req-type, AS-req-originator, AS-req-media-type and AS-req-priority into
      AS-req-header
```

```
Set Process-id = P5
Generate next P5 Packet-number
Load Process-id and Packet-number into Packet-header
If Carrier-req-type = DELETE Then
  Load Packet-header, AS-req-header, Req-header, Carrier-location-state,
      Carrier-req-type, and Carrier-unique-id into AS-carrier-req
Elseif Carrier-req-type = UPDATE Then
  Load Packet-header, AS-req-type, Req-header, Carrier-location-state, Carrier-req-type,
      Carrier-unique-id and Carrier-info into AS-carrier-req
Endif
Issue AS-carrier-req as an External-req to P 4
Issue AS-carrier-req as Internal-transaction into Internal Transaction Log data store
Endif
```

Files Used :

Database Tables Used :

Input Variables :

Carrier-req-struct

Output Variables :

Carrier-req-struct	Data Structure

5.4 P5: SNAP-REFRESH-MANAGER

Return Value :

The SNAP-REFRESH-MANAGER is responsible for generating update requests for all stale snapshots in the SAFER database.

PDL :

```
Get System-date-time from system
If Snap refresh is due Then
   Select Carrier-unique-id from Table-Snap where Snap-create-date greater than x days old
      Begin
         Load Carrier-unique-id into Snap-refresh-req
      End
   Set AS-req-type = SNAP
   Set AS-req-originator = SAFER
   Set AS-req-media-type = ELECTRONIC
   Set AS-req-priority = LOW
   Load AS-req-type, AS-req-originator, AS-req-media-type and AS-req-priority into AS-req-
header
   Set SAFER-type = SNAP REFRESH REQ
   Set Process-id = P5
   Generate next P5 Packet-number
   Set Time-stamp = System-data-time from system
   Load SAFER-Type, Process-id, Packet-number and Time-stamp into Packet-header
   Load Packet-header, AS-req-header and Snap-refresh-list into Snap-refresh-req
   Call P4:UDATE_SOURCE_SAFETY_DATA(External_req) t o issue Snap-refresh-req as
External-req
   Issue Snap-refresh-req as Internal-transaction to Internal Transaction Log
Endif
```

Files Used :

Database Tables Used :

Table-Snap

Input Variables :

Output Variables :

5.5 P5: PROFILE-CACHE-MANAGER

Return Value :

Once a day the Profile Cache Manager issues a Profile-cleanup-req to delete all profiles from the Profile data store that are greater than x hours old and are not waiting to be sent to a user.

PDL :

```
Get System-date-time from system
If Profile cleanup is due Then
  Build Profile-cleanup-req
  Issue Profile-cleanup-req to delete all profiles from Profile data store with
      Profile-create-date greater than x hours old and that are not waiting to be sent to a user
Endif
```

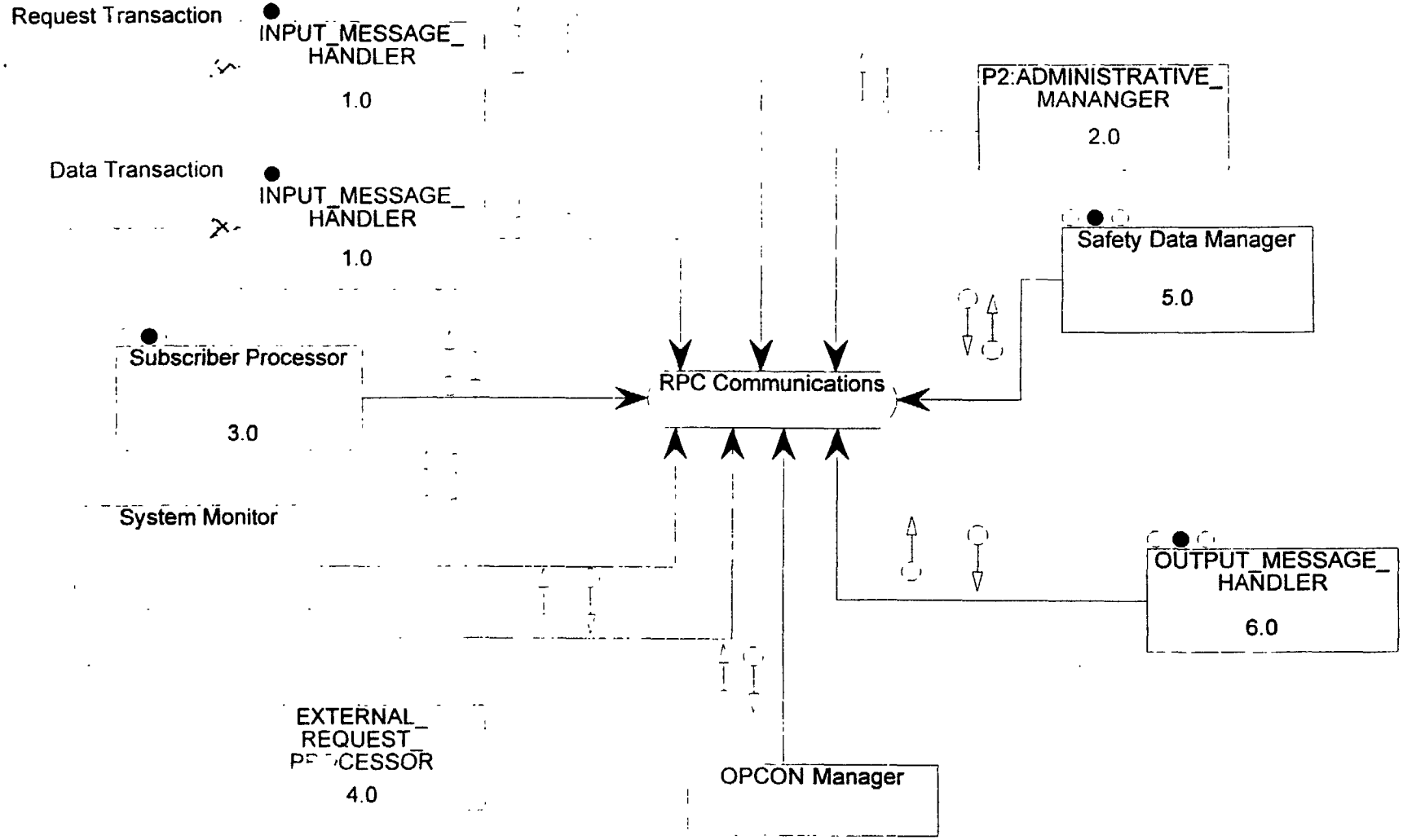
Files Used :

Database Tables Used :

Table-Profile

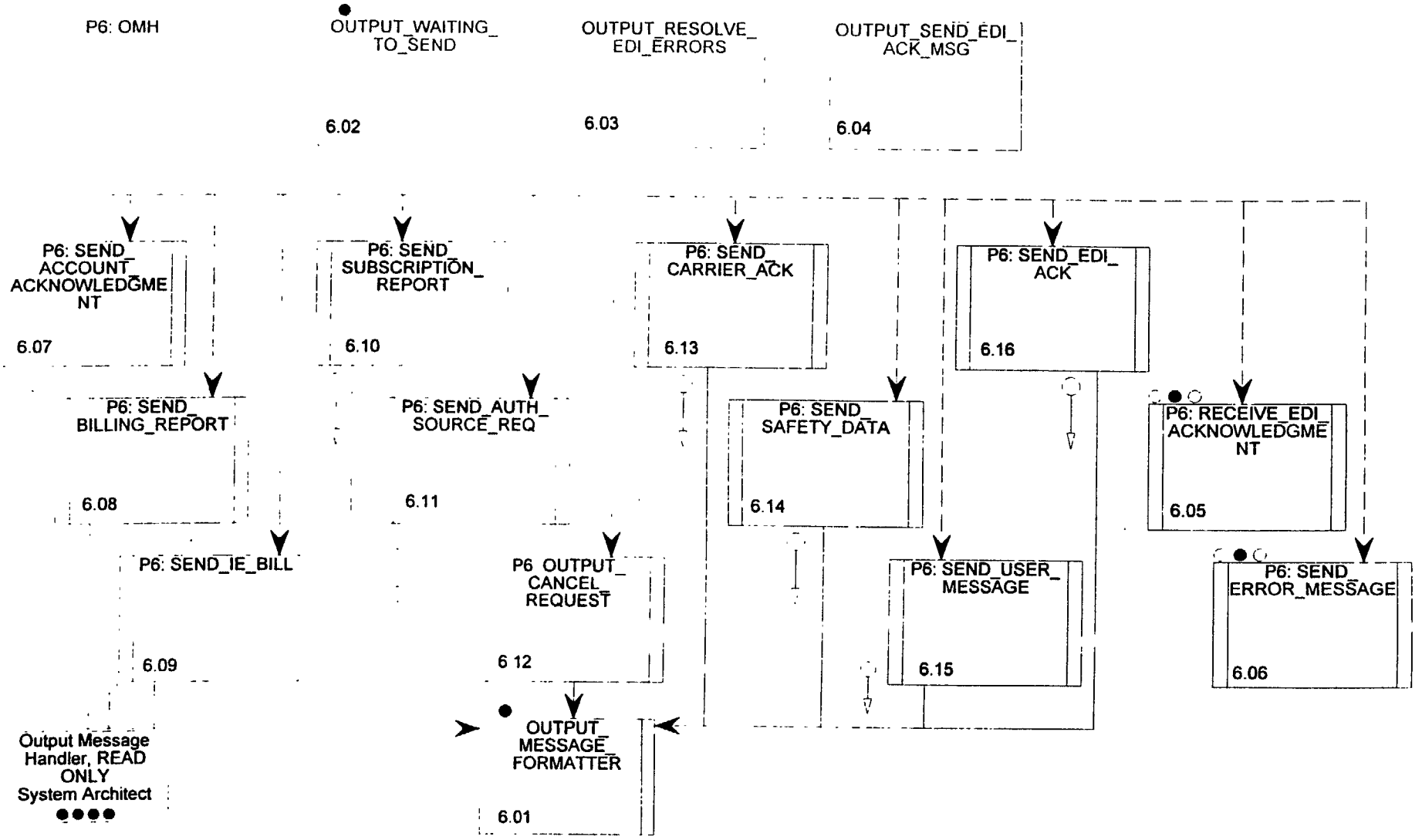
Input Variables :

Output Variables :



SAFER, READ ONLY
System Architect
Mon Jun 03, 1996 14:28
Comment

Output Message Handler



6.0 OUTPUT-MESSAGE-HANDLER **Return Value :**

Output Message Handler shell.

PDL :

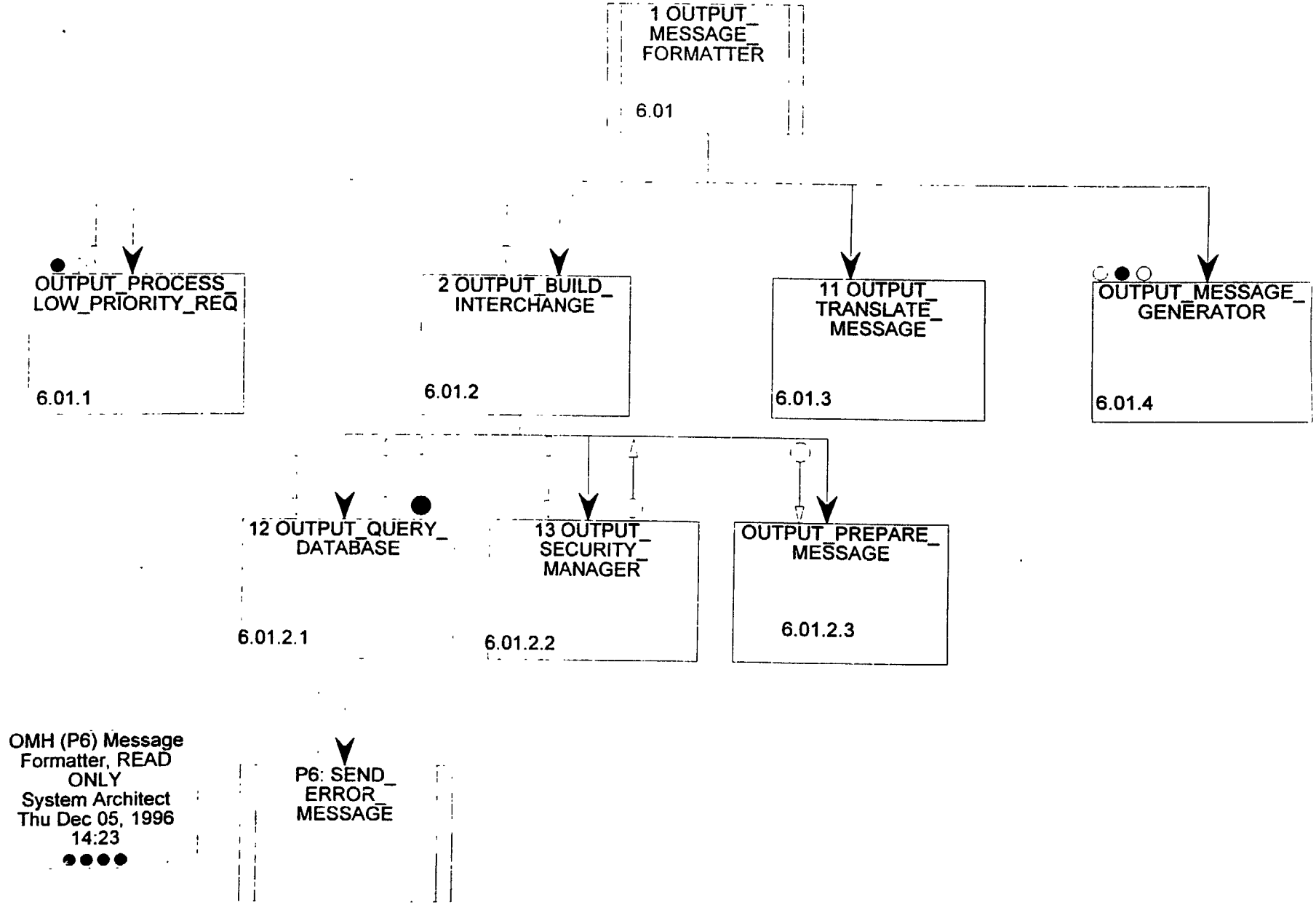
Files Used :

Database Tables Used :

Input Variables :

Output Variables :

Output Message Formatter



6.01 OUTPUT-MESSAGE-FORMATTER **Return Value :**

This module is responsible for processing low priority and cancellation messages, for building and translating transactions from the application messages, and sending the interchange to the appropriate user.

PDL :

If application message has a low priority or
application message is a cancel request Then
Call OUTPUT-PROCESS-LOW-PRIORITY-REQ
Else
Call OUTPUT-BUILD-INTERCHANGE to prepare the application message for translation

Call OUTPUT-TRANSLATE-MESSAGE to create the interchange

Call OUTPUT-MESSAGE-GENERATOR to send the interchange
Endif

Files Used :

Database Tables Used :

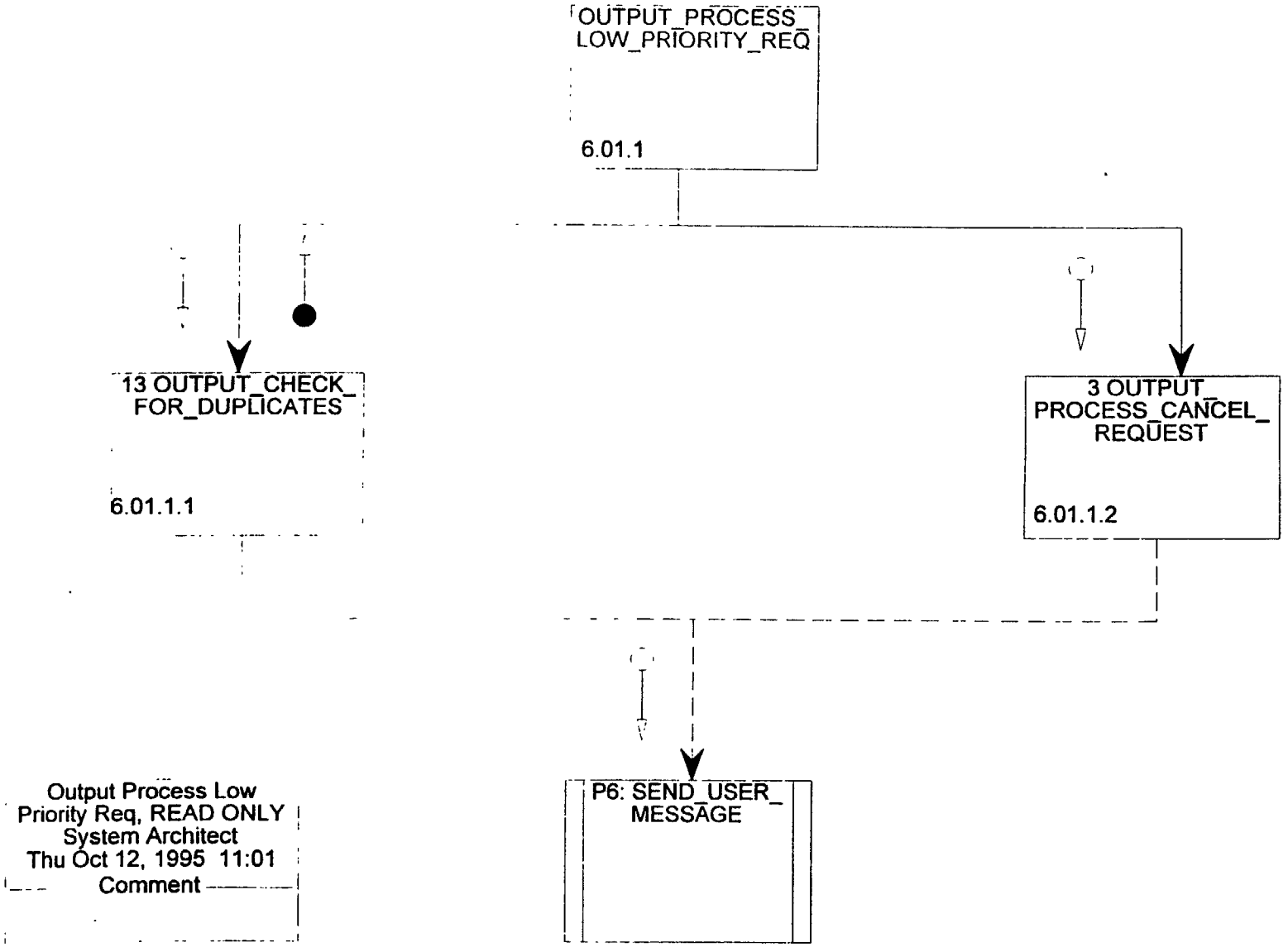
Input Variables :

Output-ack-message-struct+
Output-application-msg-struct+
Output-user-message-struct

Output Variables :

Output-ack-message-struct	**Data Structure**
Output-application-msg-struct	**Data Structure**
Output-user-message-struct	**Data Structure**

Output Process Low Priority Req



6.01.1 OUTPUT-PROCESS-LOW-PRIORITY-REQ

Return Value :

OUTPUT-PROCESS-LOW-PRIORITY-REQ checks for duplicates, handles the 'overnight' requests, and cancellations in the overnight requests.

PDL :

Call OUTPUT-CHECK-FOR-DUPLICATES

If application message is a Cancel Request
and not a duplicate request Then
Call OUTPUT-PROCESS-CANCEL-REQUEST
Endif

Files Used :

Database Tables Used :

Input Variables :

[User-cancel-req-struct
Packet-header+
Req-header+
User-request-struct]

Output Variables :

Packet-header	**Data Structure**
Req-header	**Data Structure**
User-cancel-req-struct	**Data Structure**
User-request-struct	**Data Structure**

6.01.1.1 OUTPUT-CHECK-FOR-DUPLICATES

Return Value :

OUTPUT-CHECK-FOR-DUPLICATES checks low priority messages for duplicates in File-Waiting to-Send. If the new message is a duplicate, a message is for the user indicating the later request will be ignored. Otherwise, the message is appended to File-Waiting-to-Send.

PDL :

```
Initialize Output-duplicate-message flag to FALSE

Search File-Waiting-to-Send for current message

If found Then
  Generate a message for the user
  Call P6: SEND-USER-MESSAGE
  Reset Output-duplicate-message flag to TRUE
Else
  Reset priority to normal
  Append user request to File- Waiting-to-Send
Endif
```

Files Used :

File-Waiting-to-Send

Database TabIes Used :

Input VariabIes :

Packet-header+
Req-header+
User-request-struct

Output Variables :

Output-duplicate-message

Output-duplicate-message	**Data Element**
Packet-header	**Data Structure**
Req_header	**Data Structure**
User-request-struct	**Data Structure**

6.01.1.2 OUTPUT-PROCESS-CANCEL-REQUEST

Return Value :

OUTPUT-PROCESS-CANCEL-REQUEST processes any incoming cancellation requests. It checks File-Waiting-to-Send for the request, and if found, the request is deleted and the user informed of the action.

PDL :

```
Search for User-transjd in File-Waitingto Send-to-Send

If found Then
  Delete request from File-Waiting-to-Send
  Generate message for user
  Call SEND-USER-MESSAGE
Else
  Generate message for user
  Call SEND-USER-MESSAGE
Endif
```

Files Used :

File-Waiting-to-Send

Database Tables Used :

Input Variables :

User-cancel-req-struct

Output Variables :

User-cancel-req-struct	**Data Structure**

6.012 OUTPUT-BUILD-INTERCHANGE

Return Value :

OUTPUT-BUILD-INTERCHANGE receives the next message to process, performs data retrieval as needed, performs security checks, and prepares the message for the translation process.

PDL :

```
If message requires data retrieval Then
  Call OUTPUT-QUERY-DATABASE
Else
  Initialize Output-database-error to FALSE
Endif

If no database errors Then
  Call OUTPUT-SECURITY-MANAGER to perform security checks

  Call OUTPUT-PREPARE-MESSAGE to place message into translator format
Endif
```

Files Used :

Database Tables Used :

Input Variables :

[Output-ack-message-struct]
Packet-header+
Req-header+
User-request-struct
Output-user-message-struct

Output Variables :

Output-ack-message-struct	**Data Structure**
Output-user-message-struct	**Data Structure**
Packet-header	**Data Structure**
Req-header	**Data Structure**
User-request-struct	**Data Structure**

6.01.2.1 OUTPUT-QUERY-DATABASE

Return **Value :**

OUTPUT-QUERY-DATABASE receives a retrieval request (snapshot, profile, other) and issues the database query to retrieve the latest copy of a profile, snapshot or other data. If database access errors occurred, a message is generated.

PDL :

```
Initialize Output-database-error flag to FALSE

Build database query

Access database

If error Then
  Generate error message
  Call P6: SEND-ERROR-MESSAGE
  Reset Output-database-error flag to TRUE
Else
  Retrieve data

  If data not found Then
    Generate error message
    Call P6: SEND-ERROR-MESSAGE
  Endif
Endif
```

Files Used :

Database Tables Used :

Table-Snap
Table-Profile

Input Variables :

Safety-data-send-struct

Output Variables :

Output-database-error+
Output-retrieved-data-struct

Output-database-error	**Data Element**
Output-retrieved-data-struct	**Data Strudture**
Safety-data-send-struct	**Data Structure**

6.01.2.2 OUTPUT-SECURITY-MANAGER

Return Value :

OUTPUT-SECURITY-MANAGER determines if the outgoing transaction needs to be logged for privacy compliance purposes. If that is the case, it logs date/time, information about the requestor, and an image of the actual transaction. It logs this privacy-related data to the File-Disclosure-Log (synonym: Privacy-log).

PDL :

If the Privacy-log-flag' = 'y', then Do:
(the transaction must be logged.)

Extract User-last-name + User-first-name + User-MI + User-address from Table-user-account using Req-userid

Store Driver-id-number, License-state, Time-stamp, Req-type, and Req-userid into Privacy-log-struct.

Move the transaction to Privacy-transjmage

Call Write-log-entry (File-Disclosure-Log, Privacy-log-struct)

Files Used :

File-Disclosure-Log

Database Tables Used :

Table-User-Account

Input Variables :

[Output-message-struct |
Output-retrieved-data-struct]

Output Variables :

Output-auth-msg-struct
Privacy-log-struct

Output-auth-msg-struct	**Data Structure**
Output-message-struct	**Data Structure**
Output-retrieved-data-struct	**Data Structure**
Privacy-log-struct	**Data Structure**

6.01.2.3 OUTPUT-PREPARE-MESSAGE

Return Value :

The module performs any reformatting of the message required for the translator.

PDL :
Reformat message for translator

Files Used :
File-Translator-Input

Database Tables Used :

Input Variables :
Output-auth-msg-struct

Output Variables :

Output-auth-msg-struct	**Data Structure**

6.01.3 OUTPUT-TRANSLATE-MESSAGE

Return Value :

The OUTPUT-TRANSLATE-MESSAGE is a COTS product used to translate internally structured messages to external (e.g., EDI) format, according to the specified standard. This is the same commercial product used to convert incoming transactions to internal format. It does compliance checks to ensure the outgoing messages meet the requirements of the specified standard. Error messages are written to the translator generated ED1 error log, File-EDI-Error-Log, if the supplied message could not be mapped to the specified standard. The module also generates the control numbers and segments needed for the interchange and writes the interchange to a translator generated file, File-Send. It is assumed the translator writes an entry to an audit file for each interchange output.

PDL :

```
Access File-Translator-Input

Retrieve standards table for this message

Translate message according to standard

Perform compliance edit

If Errors Then
  Write message to File-EDI-Error-File
Else
  Create interchange envelope control numbers and segments
  Write interchange to File-Send
  Log interchange to File-Outgoing-Transaction-Log for audit trail
Endif
```

Files Used :

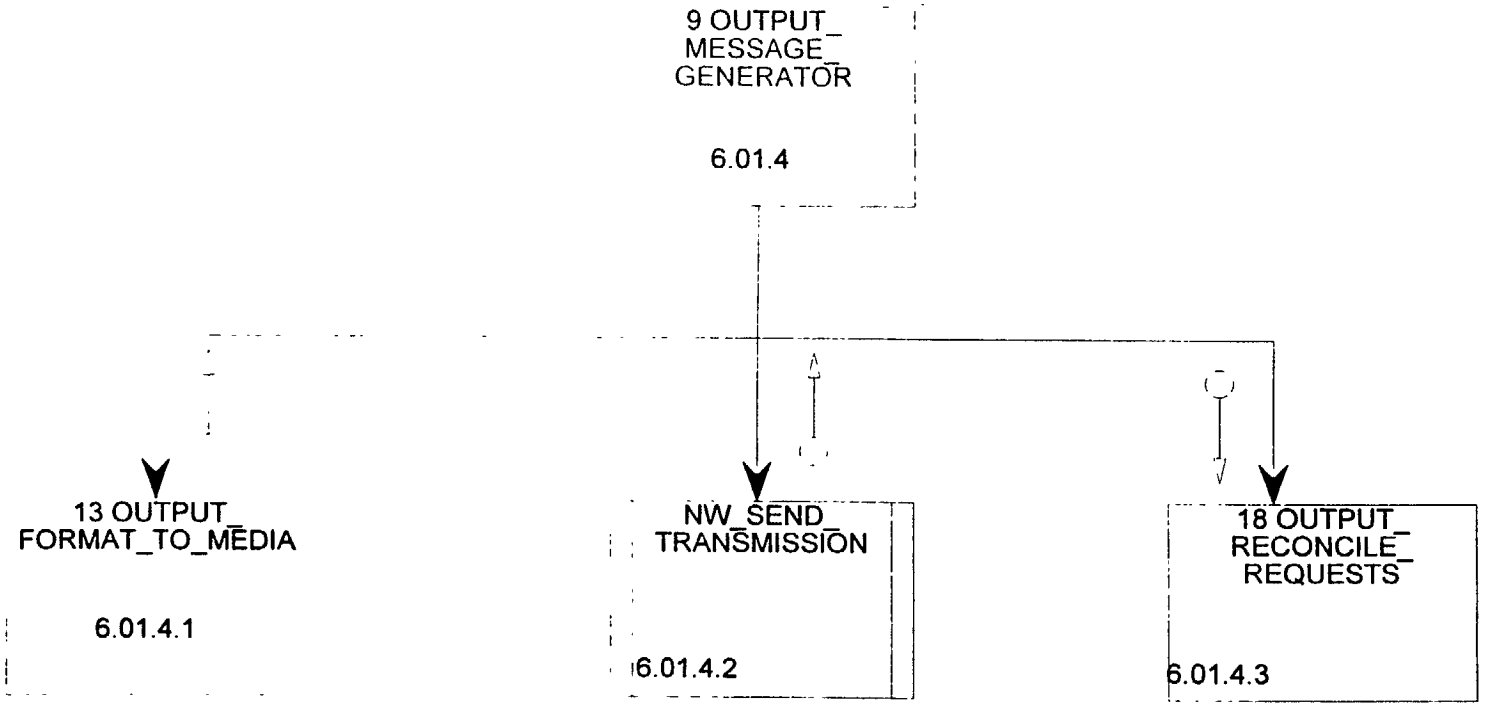
File-EDI-Error
File-Outgoing-Transaction- Log
File-Send
File-Translator-Input

Database Tables Used :

Input Variables :

Output Variables :

Output Message Generator



OMH (P6) Message
Generator, READ ONLY
System Architect
Thu Dec 05, 1996 09:42
Comment

6.01.4 OUTPUT-MESSAGE-GENERATOR

Return Value :

This module sends the interchange to the appropriate user, in the requested format, and reconciles outbound interchanges with requests received.

PDL :

```
Access File-Send

If non-ED1 Format requested Then
  Call OUTPUT-FORMAT-TO-MEDIA
Else
  Call P6: NW-SEND-TRANSMISSION to send interchange to network mailbox
Endif

Call OUTPUT-RECONCILE-MESSAGE to reconcile requests
```

Files Used :

File-Send

Database Tables Used :

Input Variables :

Output Variables :

6.01.4.1 OUTPUT-FORMAT-TO-MEDIA

Return Value :

This module writes the interchange, from the translator generated File-Send, to the user specified media (file transfer, CD-ROM).

PDL :

Determine user specified media

Write interchange to media

Return key information for request reconciliation

Files Used :

File-Send

Database Tables Used :

Input Variables :

Output Variables :

Packet-header+
Req-header

Packet-header	**Data Structure**
Req-header	**Data Structure**

6.01.4.2 NW-SEND-TRANSMISSION

Return Value :

NW-SEND-TRANSMISSION executes a 'script' to access the mailbox and send outgoing interchanges.

PDL :

Access network

Send Interchange

Return key information for request reconciliation

Files Used :

File-Send

Database Tables Used :

Input Variables :

Output Variables :

Packet-header+
REq-header+
SAFER-date+
SAFER-time

Packet-header	**Data Structure**
Req-header	**Data Structure**
SAFER-date	**Data Element**
SAFER-time	**Data Element**

6.01.4.3 **OUTPUT-RECONCILE-REQUESTS** **Return Value :**

This module determines if an outbound interchange has satisfied the user's original request. A check is made to Table-Pending, based on the user supplied transaction ID, and if not found, the request is considered to be complete and is removed from the File-Incoming-Transaction-Table. The translator generated File-Outgoing-Transaction-Log is updated with the date and time the interchange was sent.

PDL :

Access Table-Pending

Search for user supplied transaction ID

If not found Then
Remove from the File-Incoming-Transaction-Table
Endif

Update File-Outgoing-Transaction-Log

Files Used :

File-Incoming-Transaction-Table
File-Outgoing-Transaction-Log

Database Tables Used :

Table-Pending

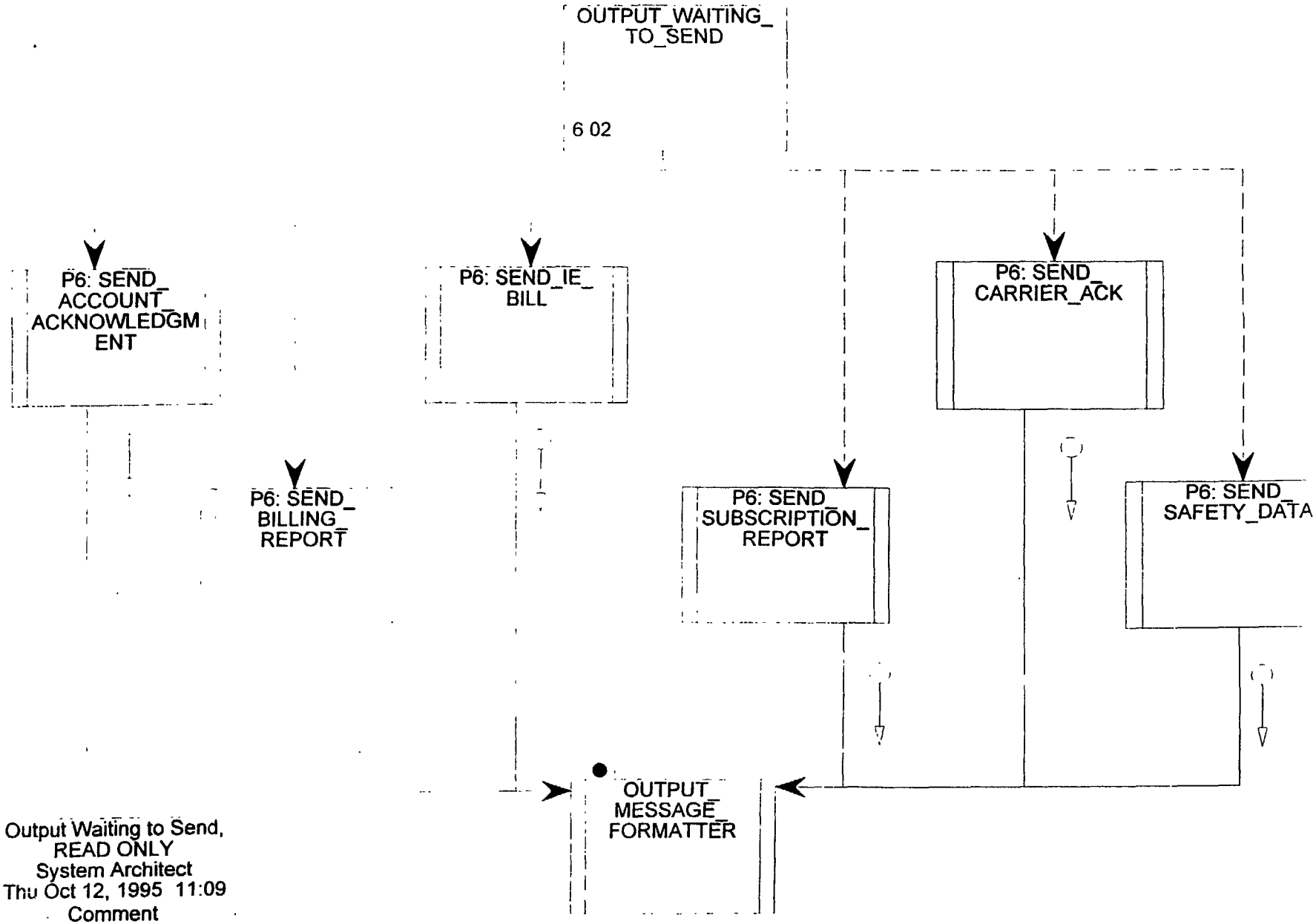
Input Variables :

Packet-header +
Req-header+
SAFER-date+
SAFER-time

Output Variables :

Packet-header	**Data Structure**
Req-header	**Data Structure**
SAFER-date	**Data Element**
SAFER-time	**Data Element**

Output Waiting to Send



6.02 OUTPUT-WAITING-TO-SEND **Return Value :**

This module places all held messages into the output processing stream when the proper time has arrived.

PDL :

```
If proper time to send Then
 Access  File-Waiting-to-Send

 If SAFER-type = Account Request Response Then
   Call P6: SEND-ACCOUNT-ACKNOWLEDGMENT
 elseif SAFER-type = Billing Info Request Then
   Call P6: SEND-BILLING-REPORT
 elseif SAFER-type = User IE Bill Then
   Call P6: SEND-IE-BILL
 elseif SAFER-type = Carrier Acknowledgment Then
    Call P6: SEND-CARRIER-ACK
 elseif SAFER-type = Event Status Report Then
    Call P6: SEND-SUBSCRIPTION-REPORT
 elseif SAFER-type = Safety Data Then
    Call P6: SEND-SAFETY-DATA
Endif
```

Files Used :

File-Waiting-to-Send

Database Tables Used :

Input Variables :

Output Variables :

6.03 OUTPUT_RESOLVE_EDI_ERRORS **Return Value :**

OUTPUT-RESOLVE-EDI-ERRORS provides a means to correct incorrect outbound transactions. The facility will most likely be provided by the commercial translator.

PDL :

Open File- EDI-Error

Read next ED1 Error message

Present data entry panels

If problem resolved for dutbound transaction Then
  Resubmit transaction for outbound processing
Endif

Files Used :

File-EDI-Error

Database Tables Used :

Input Variables :

Output Variables :

6.04 OUTPUT-SEND-EDI-ACK-MSG

Return Value :

This module accesses the translator generated File-EDI-Acknowledgment, and if any acknowledgments are ready, they are sent to the user. Note: the translator may have the capability to automatically send acknowledgments.

PDL :

AccessFIle- EDI-Acknowledgment

Retrieve ED1 acknowledgment message

Call P6: SEND-EDI-ACK

Files Used :

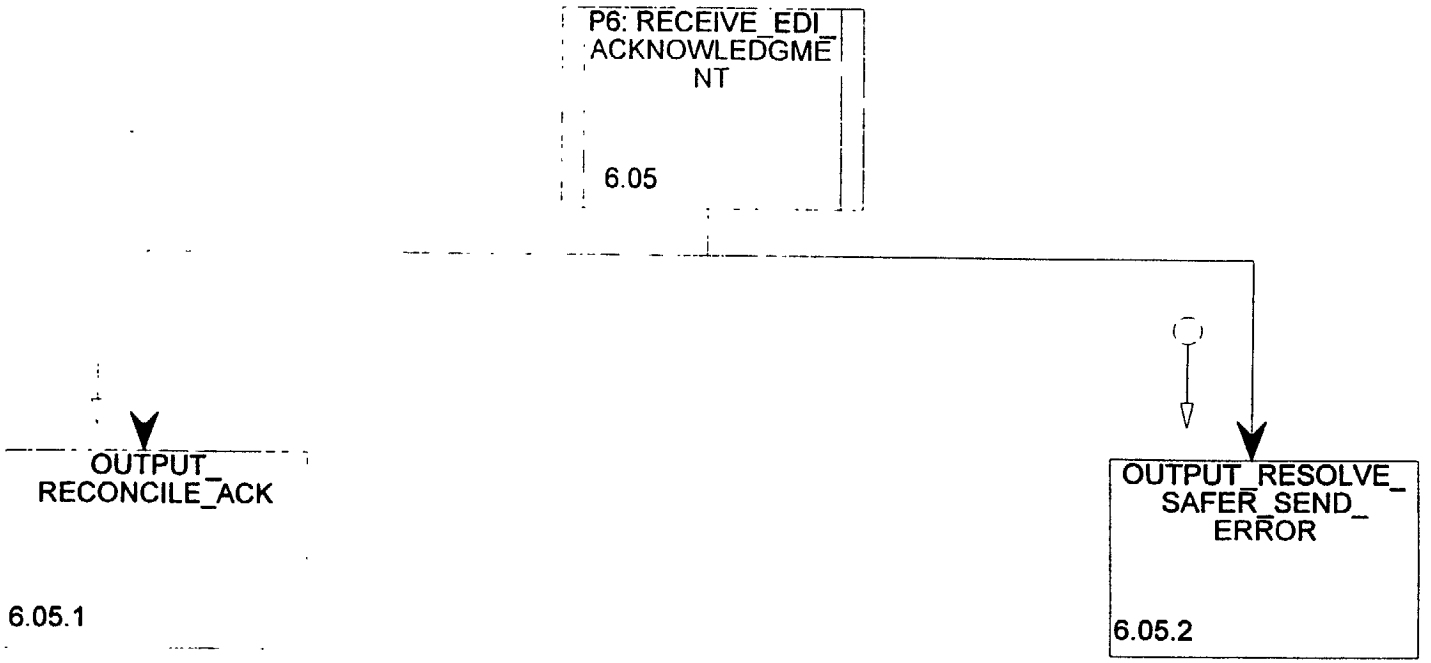
FIle-EDI-Acknowledgment

Database Tables Used :

Input Variables :

Output Variables :

P6: Receive EDI Acknowledgment



OMH (P6) Receive EDI Ack, READ ONLY
System Architect
Thu Oct 12, 1995 11:13
Comment

6.05 P6: RECEIVE-EDI-ACKNOWLEDGEMENT

Return Value :

This module handles incoming EDI acknowledgments, either interchange or functional acknowledgments. Acknowledgments not reporting errors are used to update the status of transmitted interchanges in the translator generated File-Output-Transaction-Log. If errors were found, an opportunity for correction is made available.

PDL :

```
Initialize Output-retrieve-error flag to FALSE

Open File-EDI-Acknowledgment

If error free acknowledgment Then
  Call OUTPUT-RECONCILEACK to reconcile acknowledgment
Elseif acknowledgment with errors Then
  Call OUTPUT-RESOVLE-SAFER-SEND-ERROR to resolve errors
Endif

Delete acknowledgment from EDI Acknowledgment Log
```

Files Used :

File-EDI-Acknowledgment

Database Tables Used :

Input Variables :

Input-ack-message-struct

Output Variables :

Input-ack-message-struct	Data Structure

6.05.1 OUTPUT-RECONCILE-ACK

Return Value :

OUTPUT-RECONCILE-ACK matches incoming acknowledgments to transmissions sent by SAFER and updates the translator generated File-Outgoing-Transaction-Log. **A** report is generated for the SAFER operator to review all outgoing transmissions that have not been acknowledged by the receiver.

PDL :

Extract key fields from incoming acknowledgment

Search File-Outgoing-Transaction-Log for these keys

Set acknowledgment flag

Search File-Outgoing-Transaction-Log for outstanding acknowledgments

If found Then
Display report to operator
Endif

Files Used :

File-Outgoing-Transaction-Log

Database Tables Used :

Input Variables :

Input-ack-message-struct

Output Variables :

Input-ack-message-struct	**Data Structure**

6.05.2 OUTPUT-RESOLVE-SAFER-SEND-ERROR **Return Value :**

Incoming ED1 acknowledgments are either interchange or functional acknowledgments. An error report is generated for acknowledgments with errors. The SAFER operator is presented with the opportunity to resolve the error (invalid code, invalid receiver id, etc.) or refer the problem to the System Administrator.

PDL :

Generate error report from acknowledgment

Present data entry panels for problem correction

If problem resolved Then
Resubmit transaction for output processing
Endif

Files Used :

Database Tables Used :

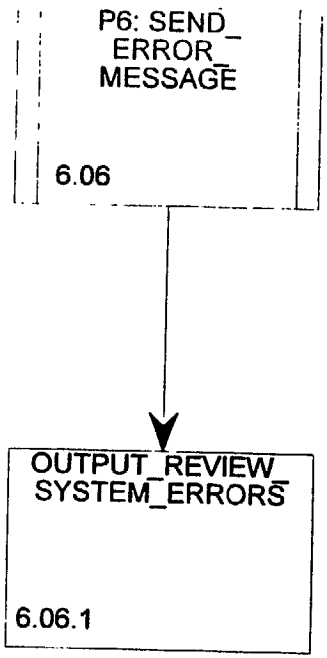
Input Variables :

Input-ack-message-struct

Output Variables :

Input-ack-message-struct	**Data Structure**

P6: Send Error Message



OMH (P6) Send Error
Message, READ ONLY
System Architect
Fri Nov 03, 1995 07:56
Comment

6.06 P6: SEND-ERROR-MESSAGE

Return Value :

This module receives SAFER error messages, determines the severity, and if high, immediately notifies the SAFER operator. All messages are placed into an File-Error-Log for review by a SAFER operator.

PDL :

```
Determine message severity

If high Then
  Notify SAFER operator
Endif

Enter error message into File-Error-Log
```

Files Used :

File-Error-Log

Database Tables Used :

Input Variables :

Packet-header+
SAFER-date+
SAFER-error-code+
SAFER-time

Output Variables:

Packet-header	**Data Structure**
SAFER-date	**Data Element**
SAFER-error-code	**Data Element**
SAFER-time	**Data Element**

6.06.1 OUTPUT-REVIEW-SYSTEM-ERRORS

Return Value :

This module allows the SAFER operator to review SAFER error messages stored in the File-Error-Log. It responds to user queries and provides report or statistics generation.

PDL :

Retrieve error data from File-Error-Log based on user query

If user requested data found Then
  Generate user report
Else
  Display a message to user
Endif

Files Used :

File-Error-Log

Database Tables Used :

Input Variables :

Output Variables :

.

6.07 P6: SEND-ACCOUNT-ACKNOWLEDGMENT

Return Value :

This module receives and invokes processing for responses sent back to the user indicating the **status** of his request.

PDL :

Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables :

Account-response-msg-struct

Output Variables :

Account-response-msg-struct	**Data Structure**

6.08 P6: SEND-BILLING-REPORT Return Value :

This module receives and invokes processing for requests made to the VAN at some specified billing interval for billing information for each organization with an account on the system,

PDL :
Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables :
Billing-info-msg-struct

Output Variables :

Billing-info-msg-struct	**Data Structure**

6.09 P6: SEND-IEBILL **Return Value :**

This module receives and invokes processing for bills sent to each organization each billing period.

PDL :

Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables :

User-IE-bill-msg-struct

Output Variables :

User_IE_bill_msg_struct	**Data Structure**

6.10 P6: SEND-SUBSCRIPTION-REPORT

Return Value :

This module receives and invokes processing for a data reports sent to the user on request. It either reflects a user's current subscriber status, and/or system event monitoring status.

PDL :

Receive message

Call OUTPUT-MESSAGE-FORMATTER to **process** message

Files Used :

Database Tables Used :

Input Variables :

Event-status-msg-struct

Output Variables :

Event-status-msg-struct	**Data Structure**

6.11 P6: SEND-AUTH-SOURCE-REQ

Return Value :

This module receives and invokes processing for appropriately formatted requests for an authoritative source. Includes the authoritative source address and request priority.

PDL :

Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables :

Valid-AS-req-struct

Output Variables :

Valid-AS-req-struct

Valid-AS-req-struct	**Data Structure**

6.12 P6: OUTPUT-CANCEL-REQUEST

Return Value :

This module receives and invokes processing for user requested cancellations of previously transmitted low priority requests.

PDL :

Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables :

CanceI-req-struct

Output Variables :

Cancel-req_struct	**Data Structure**

6.13 P6: SEND-CARRIER-ACK

Return Value :

This module receives and invokes processing for an acknowledgement for an 'Add' type carrier request (includes the new DOT number).

PDL :

Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

Carrier-ack-msg-struct	**Data Structure**

6.14 P6: SEND-SAFETY-DATA

Return Value :

This module receives and invokes processing for requests to send snapshot, profile or census information to a user.

PDL :
Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables :
Safety-data-send-struct

Output Variables :

Safety-data-send-struct	**Data Structure**

6.15 P6: SEND-USER-MESSAGE

Return Value :

This module receives and invokes processing for messages to be sent to a SAFER user.

PDL :

Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables :

User-message-struct

Output Variables :

User-message-struct	**Data Structure**

6.16 P6: SEND-EDI-ACK

Return Value :

This module accepts ED1 acknowledgments and releases them to the output stream.

P D L :

Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

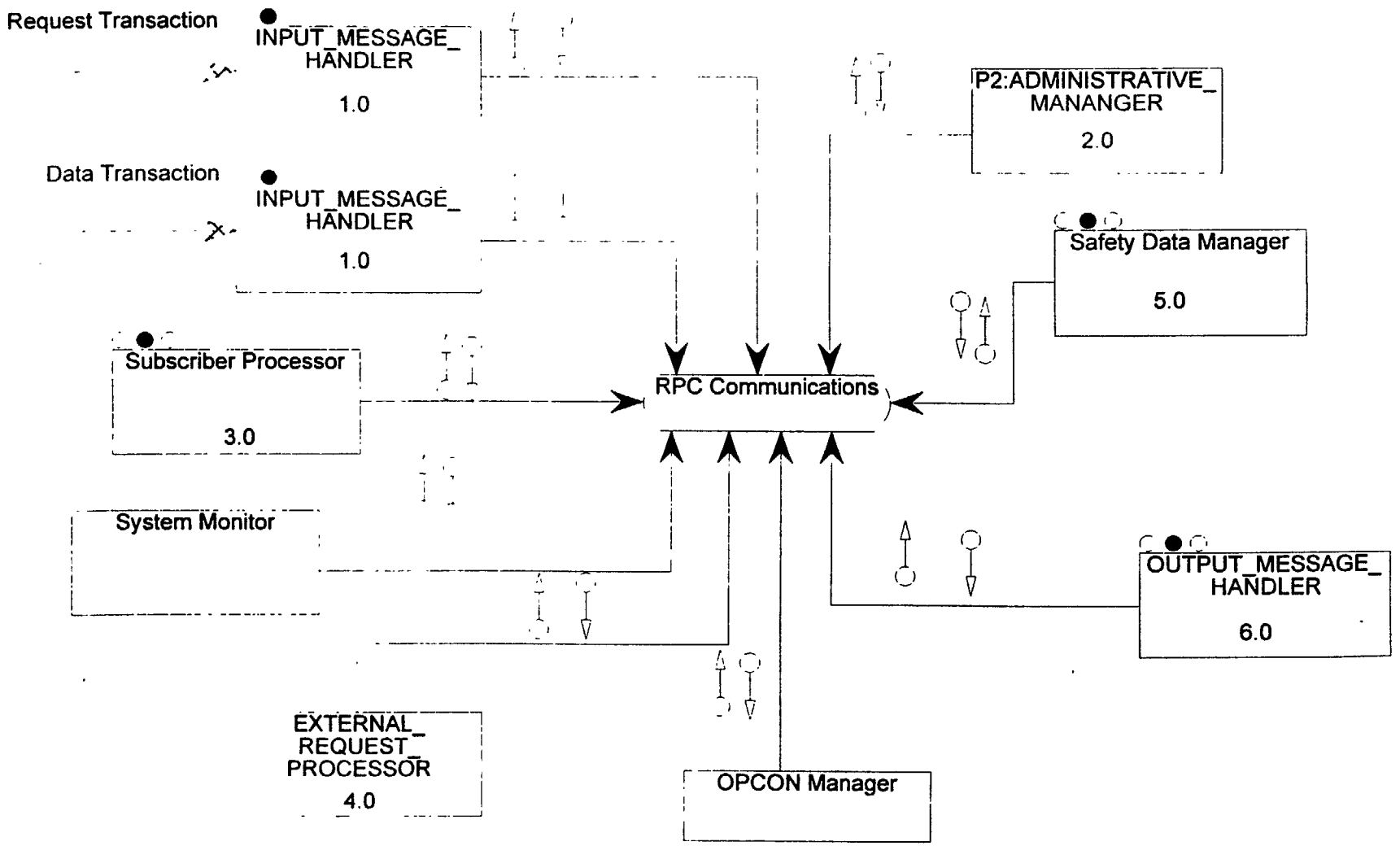
Database Tables Used :

Input Variables :

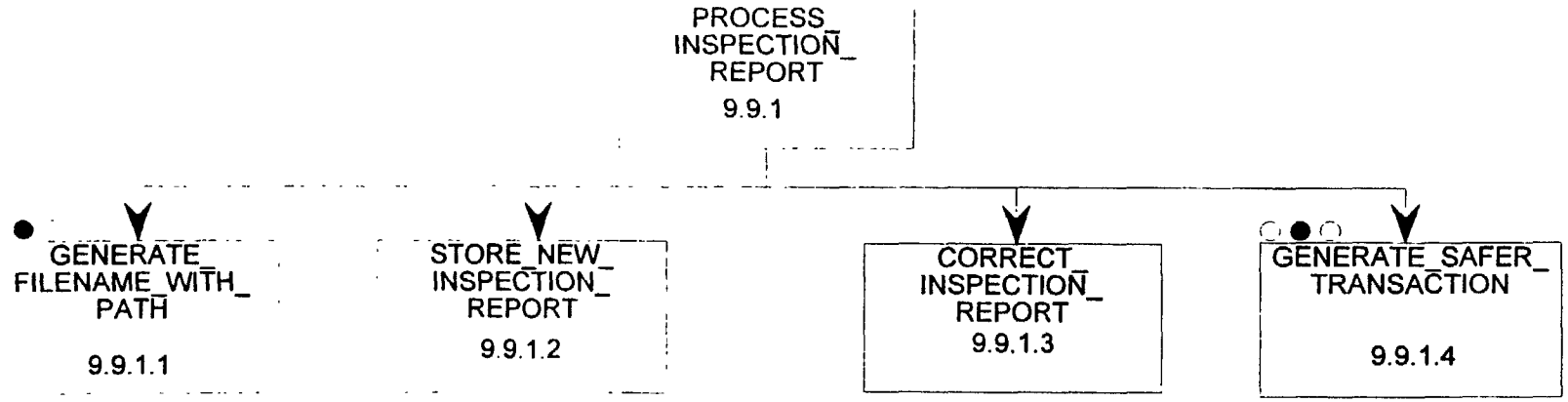
Output-ack-message-struct

Output Variables :

Output-ack-message-struct	**Data Structure**



SAFER, READ ONLY
System Architect
Mon Jun 03, 1996 14:28
Comment



PROCESS_INSPECTION_
REPORT, READ ONLY
System Architect
Sat Dec 21, 1996 09:50
Comment

9.9.1 PROCESS-INSPECTION-REPORT Return Value :

This module processes incoming inspection reports which can be either new reports or correction reports. The reports are stored in a location specified by SAFER. But if such a location is not valid or is not specified at all, the reports will be saved in the system's temp directory and a message be posted to indicate error, and later on the operator should move the reports to the correct location manually.

When an inspection report comes in, a filename based on the inspection date and report number is generated. If a file with the generated filename exists in the inspection report location, it will be replaced by the new one; if a file does not exist, the inspection report will be saved in the location with the generated name. If any error happens while writing inspection reports to disk, it posts a message to the console to indicate the writing error.

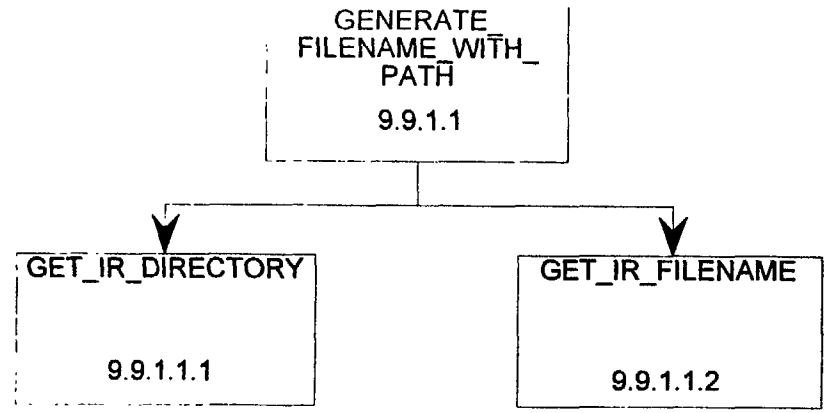
PDL :

Files Used :

Database Tables Used :

Input Variables :

Output Variables :



GENERATE_FILENAME_
WITH_PATH, READ ONLY
System Architect
Thu Dec 05, 1996 13:08
Comment

9.9.1.1 GENERATE-FILENAME-WITH-PATH

Return Value :

This module generates a filename with full path for an incoming inspection report. First, it generates a filename by concatenating the inspection date with an underscore and then the inspection report number. The date has a fixed format of MMDDYY, and leading zeros should be used in the cases when months and/or dates are less than 10. However, for the inspection report number, all leading zeros should be taken out. Second, it reads in the path for the inspection report location from SAFER initialization file if there exists an entry. If not, it gets the system's temp directory, uses it as the path and sets the Temp-Dir flag. Third, it appends the filename to the path, thus completes the generation of a filename with complete path for the incoming inspection report.

PDL :

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

9.9.1.1.1 GET-IR-DIRECTORY Return Value :

This module gets a directory path to inspection report location. It first checks the SAFER initialization file to see if an entry named “Inspection Report Location” exists and the value of it is a valid directory path. If yes, it returns the path; Else, it gets the system’s temp directory path and returns it, also, sets the Temp-Dir flag.

PDL :

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

9.9.1.1.2 GET-IR-FILENAME

Return Value :

This module generates a filename for the incoming inspection report by concatenating the inspection date with an underscore and then the inspection report number. The date has a fixed format of MMDDYY, and leading zeros should be used in the cases when months and/or dates are less than 10. However, for the inspection report number, all leading zeros should be taken out.

PDL :

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

9.9.1.2 STORE-NEW-INSPECTION-REPORT

Return Value :

This module stores a new inspection report onto the disk. It checks to see if Temp-Dir flag is set. If yes, after successfully writing the report to disk, it posts a message to the console indicating an Inspection Report Location error encountered and an inspection report has been saved to the TEMP directory. It also needs to remind SAFER operators to manually move the inspection report file to the correct location. If writing the inspection report to disk fails,' post a message to the console to indicate writing errors.

PDL :

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

9.9.1.3 CORRECT-INSPECTION-REPORT

Return Value :

This module takes in a resent inspection report and overrides the existing one. Post a message to the console if writing to the disk fails.

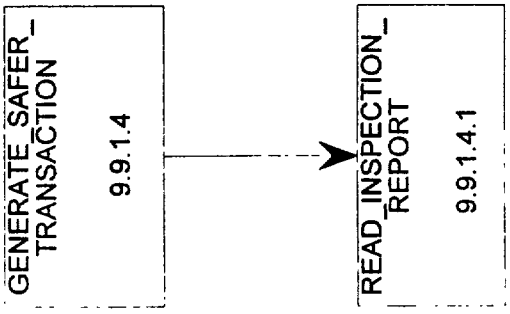
PDL :

Files Used :

Database Tables Used :

Input Variables :

Output Variables :


GENERATE_SAFER_TRANSACTION
9.9.1.4
READ_INSPECTION_REPORT
9.9.1.4.1

9.9.1.4 GENERATE-SAFER-TRANSACTION

Return Value :

This module reads an incoming inspection report and generates a SAFER transaction. In the initial phase, one inspection report composes of several files but it is sent to SAFER as one zipped file. So this module first creates a subdirectory under the system's temp directory, unzips the inspection report into the subdirectory, reads in data from the unzipped files into appropriate SAFER structure, then removes the subdirectory with the unzipped files from temp directory. It posts error messages when it fails to do any subdirectory/files manipulations.

PDL :

Files Used :

Database Tables Used :

Input Variables:

Output Variables :

ADD-USER-TO-USER-ACCOUNT-TABLE

Return Value :

Add an entry to database table User-Account.

PDL :

An INSERT database command.

Files Used :

Database Tables Used :

Input Variables :

Output Variables :

EVENT-LISTMOV Return Value :

This is a basic routine to get event list data out of the database and into the correct event processing application format. It is possible that this could not be an actual module, but SQL, data base 'find with a view, etc. It is important, however that it be done in a consistent manner. It is anticipated that SAFER will undergo planned changes, and ensuring that event list access is always updated correctly and completely is a must.

PDL :

For each Event list ID in the process list received,
Do /*event id*/
Retrieve a view of event list via event-verify.
If an error occurs,
Do /*an err*/
set all error fields to ON
Enddo /*an err*/
Place filelds into standard system Event list data structure.

Enddo /*event id*/

Files Used :

Database Tables Used :
Table-Event

Input Variables :
Packet-header +
Event-vers +
Event-update-date +
Event-update-source +
{Event-no +
Event-name +
Event-type +
Event-check-type + Event-active-flag +
Event-range-set}

Output Variables :
Event-vers +
Event-list +
Event-update-source +
(Event-check-type + Event-active-flag}

Event-active-flag	Data Element
Event-check-type	Data Element
Event-list	Data Structure
Event-name	Data Element
Event-no	Data Element

Event-range-set	Data Structure
Event-type	Data Element
Event-update-date	Data Structure
Event-update-source	Data Element
Event-vers	Data Element
Packet-header	Data Structure

GET-VALID-AS Return Value :

GET-VALID-AS obtains the authoritative source Account-id for a Carrier-unique-id.

PDL :
Receive Carrier-unique-id and Valid-safety-data-type
Select Carrier-location-state for Carrier-unique-id from Table-Snap
Select Account-id for Carrier-location-state and Valid-safety-data-type from Table-Base-State-Routing
Return Account-id

Files Used :

Database Tables Used :
Table-Census
Table-Base-State-Routing

Input Variables :
Carrier-unique-id
Valid-safety-data-type

Output Variables :
Account-id

Account-id	Data Element
Carrier-unique-id	Data Element
Valid-safety-data-type	Data Element

NW-RECEIVE-TRANSMISSION Return Value :

NW-INTERFACE executes a 'script' to access the mailbox and retrieve incoming interchanges.

PDL :
Access network

Read mailbox

Write interchange to File-Receive

Files Used :
File-Receive

Database Tables Used :

Input Variables :

Output Variables :

NW-SEND-TRANSMISSION Return Value :

NW-SEND-TRANSMISSION executes a 'script' to access the mailbox and send outgoing interchanges.

PDL :
Access network

Send Interchange

Return key information for request reconciliation

Files Used :
File-Send

Database Tables Used :

Input Variables :

Output Variables:
Packet-header+
Req-header+
SAFER-date+
SAFER-time

Packet-header	Data Structure
Req-header	Data Structure
SAFER-date	Data Element
SAFER-time	Data Element

OUTPUT-MESSAGE-FORMATTER Return Value :

This module is responsible for processing low priority and cancellation messages, for building and translating transactions from the application messages, and sending the interchange to the appropriate user.

PDL :

```
If application message has a low priority or
  application message is a cancel request Then
   Call OUTPUT-PROCESS-LOW-PRIORITY-REQ
Else
  Call OUTPUT-BUILD-INTERCHANGE to prepare the application message for translation

   Call OUTPUT-TRANSLATE-MESSAGE to create the interchange

   Call OUTPUT-MESSAGE-GENERATOR to send the interchange
Endif
```

Files Used :

Database Tables Used :

Input VariabIes :

Output-ack-message-struct+
Output-application-msg-struct+
Output-user-message-struct

Output Variables :

Output-ack-message-struct	Data Structure
Output-application-msg-struct	Data Structure
Output-user-message-struct	Data Structure

P2: BILL-ACCOUNT Return Value :

Request for a usage report to include, but not be limited to: account id, message count, login count, connect time, data amount.

PDL :

Files Used :

Database Tables Used :

Input Variables :
Billing-struct

Output Variables :

P5: RETRIEVE-SAFETY-DATA Return Value :

RETRIEVE-SAFEIY-DATA is responsible for processing all snapshot and profile request.

PDL :

```
Receive Safety-data-req
Extract SDR-type, Packet-header, Req-header, Fuzzy-select, Value-match-data-type and
    Query-criteria from Safety-data-req
Extract Req-user-id, Req-media-type and Req-priority from Req-header
Extract Query-number from Query-criteria
Load Query-criteria into Exact-match-criteria
Call PROCESS-EXACT-MATCH(Exact-match-criteria ,Exact-match-results)
Extract Match-count from Exact-match-results
If Query-number = l(USDOT) and Match-count > 0 Then
  If SDR-type = SNAP Then
    Set SDSR-type = SNAP
    Set SAFER-type = SAFETY DATA SEND REQ
    Load SAFER-type into Packet-header
    Load SDSR-type, Packet-header and Req-header into Safety-data-send-req
    Dofor all Carrier-unique-ids in Exact-match-results
      Add Carrier-unique-id to Safety-data-send-req
    Enddo
    Call P6: SEND-SAFETY-DATA (Safety-data-send-req)
  Elseif SDR-type = PROFILE Then
    Set SAFER-type = USER MESSAGE
    Load SAFER-type into Packet-header
    Set User-message-info = Match-count PROFILES FOUND FOR User-trans-id
    Call P6: SEND-USER-MESSAGE (Packet-header, User-id, User-message-info)
    Set Local-flag = FALSE
    Set A3-flag = FALSE
    Dofor all Carrier-unique-ids in Exact-match-results
      Issue Profile-exists-select to determine if profile for this carrier is in profile cache
      If Profile-exists = TRUE Then
        Add Carrier-unique-id to Safety-data-send-req
        Set Local-flag = TRUE
      Elseif Profile-exists = FALSE Then
        Retrieve Carrier-location-state for this Carrier-unique-id from Table-Snap
        Add Carrier-unique-id and Carrier-location-state to AS-profile-req
        Set AS-flag = TRUE
      Endif
    Enddo
    If Local-flag = TRUE Then
      Set SAFER-type = SAFETY DATA SEND REQ
      Set SDSR-type = PROFILE
      Load SDSR-type, Packet-header and Req-header into Safety-data-send-req
      Call P6: SEND-SAFETY-DATA (Safety-data-send-req)
    Endif
    If AS_flag = TRUE Then
```

```
        Set AS-req-type = PROFILE
        Set AS-req-originator = SAFER
        Set AS-req-media-type = ELECTRONIC
        Set AS-req-priority = LOW
        Load AS-Req-type, AS-req-originator, AS-req-media-type and AS-req-priority
             into AS-req-header
        Set SAFER-type = AS PROFILE REQ
        Load SAFER-type into Packet-header
        Load Packet-header, AS-req-header and Req-header into AS-profile-req
        Call P4: Pending-Request-Manager (External-req) to issue AS-profile-req as External-
req
      Endif
    Endif
  Elseif Query-number > 1 and Match-count > 0 Then
    Extract Cost-check from Safety-data-req
    If Cost-check is ON Then
      Set User-id = Req-user-id from Req-header
      Set SAFER-type = USER MESSAGE
      Load SAFER-type into Packet-header
      Extract Match-count from Exact-match-results
      Calculate Estimated -transmission-cost based on Match-count and
           Value-match-data-type
      Load Match-count and Estimated-transmission-cost into User-message-info
      Set SAFER-type = USER MESSAGE
      Load SAFER-type into Packet-header
      Set User-id = Req-user-id
      Call P6: SEND-USER-MESSAGE (Packet-header, User-id, User-message-info)
    Elseif Cost-check is OFF Then
      If Value-match-data-type = SNAP Then
        Set SDSR-Type = SNAP
      Elseif Value-match-data-type = CENSUS Then
        Set SDSR-type = CENSUS
      Endif
      Set SAFER-type = SAFETY DATA SEND REQ
      Load SAFER-type into Packet-header
      Load SDSR-type, Packet-header and Req-header into Safety-data-send-req
      Dofor all Carrier-unique-ids in Exact-match-results
        Add Carrier-unique-id to Safety-data-send-req
      Enddo
      Call P6: SEND-SAFETY-DATA (Safety-data-send-req)
    Endif
  Elseif.Query-number > 1 and Match-count = 0 and Fuzzy-select = ON Then
    Extract Fuzzy-limit from Safety-data-req
    Load Exact-match-criteria and Fuzzy-limit into Fuzzy-match-criteria
    Call PROCESS-FUZZY-MATCH (Fuzzy-match-criteria, Fuzzy-match-results)
    Dofor all Carrier-unique-ids in Fuzzy-match-results
      Add Carrier-unique-id to Safety-data-send-request
```

```
    Enddo
  Elseif Match-count = 0 and Fuzzy-select = OFF Then
      Set User-message-info = NO EXACT MATCH FOUND
      Set SAFER-type = USER MESSAGE
      Load SAFER-type into Packet-header
      Set User-id = Req-user-id
      Call P6: SEND-USER-MESSAGE (Packet-header, User-id, User-message-info)
  Endif
```

Files Used:

Database Tables Used :

Input Variabies :
Safety-data-send-struct

Output Variables :

Safety-data-send-struct	Data Structure

P5: UPDATE-CARRIER Return Value :

UPDATE-CARRIER is responsible for processing all Carrier-reqs.

PDL :

```
Receive Carrier-req from Pl
Extract Carrier-req-type and Req-header from Carrier-req
If Carrier-req-type = ADD Then
  Generate new Carrier-unique-id
  Extract Carrier-info and Packet-header from Carrier-req
  Extract the Census-data-item-values from the Carrier-info for data fields that are common
to
      carrier and census
  Set Carrier-status = PENDING
  Load Carrier-unique-id, Census-data and Carrier-status into Census-data-record
  Insert Census-data-record into Census area of Snap data store
  If insert was successful Then
    Set Message-text = NEW USDOT GENERATED
    Load Packet-header, Req-header, Message-text and Carrier-unique-id into Carrier-ack
    Issue Carrier-ack to P6
    Extract Carrier-location-state from Carrier-info
    Set AS-req-type = CARRIER
    Set AS-req-originator = SAFER
    Set AS-req-media-type = ELECTRONIC
    Set AS-req-priority = LOW
    Load AS-req-type, AS-req-originator, AS-req-media-type and AS-req-priority into
      AS-req-header
    Set Process-id = P5
    Generate next P5 Packet-number
    Load Process-id and Packet-number into Packet-header
    Load Packet-header, AS-req-header, Req-header, Carrier-loaction-state,
      Carrier-req-type, Carrier-unique-id and Carrier-info into AS-carrier-req
    Issue AS-carrier-req as External-req to P4
    Insert AS-carrier-req as Internal-transaction into Internal Transaction Log data store
  Else
    Set Error-message = NEW USDOT REQUEST UNSUCCESSFUL
    Issue Error-message to Error-log
  ENDIF
Else (UPDATE or DELETE)
  Extract Carrier-unique-id from Carrier-req
  Retrieve Carrier-location-state for this Carrier-unique-id from the Census area of the Snap
      data store
  Set AS-req-type = CARRIER
  Set AS-req-originator = SAFER
  Set AS-req-media-type = ELECTRONIC
  Set AS-req-priority = LOW
  Load AS-req-type, AS-req-originator, AS-req-media-type and AS-req-priority into
    AS-req-header
```

```
  Set Process-id = P5
  Generate next P5 Packet-number
  Load Process-id and Packet-number into Packet-header
  If Carrier-req-type = DELETE Then
    Load Packet-header, AS-req-header, Req-header, Carrier-location-state,
         Carrier-req-type, and Carrier-unique-id into AS-carrier-req
  Elseif Carrier-req-type = UPDATE Then
    Load Packet-header, AS-req-type, Req-header, Carrier-location-state, Carrier-req-type,
         Carrier-unique-id and Carrier-info into AS-carrier-req
  Endif
  Issue AS-carrier-req as an External-req to P4
  Issue AS-carrier-req as Internal-transaction into Internal Transaction Log data store
Endif
```

Files Used :

Database Tables Used :

Input Variables:
Carrier-req-struct

Output Variables :

Carrier-req-struct	Data Structure

UPDATE-SAFETY-DATA Return Value :

UPDATE-SAFETY-DATA is the control module for obtaining Valid-safety-data from the IPC Message Queue and adding the data to the SAFER database.

PDL :

```
Receive  Valid-safety-data
Extract Data-header from Valid-safety-data
Extract Valid-safety-data-type from Data-header
If Valid-safety-data-type = PROFILE Then
  Extract Profile-data from Valid-safety-data
  Call PROFILE_DATA-MANAGER(Profile_data) to add profiles to database
Elseif Valid-safety-data-type = SNAP Then
  Extract Snap-data from Valid-safety-data
  Call SNAP-DATA-MANAGER(Snap-data) to add snapshots to database
Endif
```

Files Used :

Database Tables Used :

Input Variables:
Valid-safety-data-struct

Output Variables:

Valid-safety-data-struct	Data Structure

P6:OUTPUT-CANCEL-REQUEST Return Value :

This module receives and invokes processing for user requested cancellations of previously transmitted low priority requests.

PDL :
Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables :
Cancel-req-struct

Output Variables :

Cancel-req-struct	Data Structure

P6: RECEIVE-EDI-ACKNOWLEDGMENT Return Value :

This module handles incoming ED1 acknowledgments, either interchange or functional acknowledgments. Acknowledgments not reporting errors are used to update the status of transmitted interchanges in the translator generated File-Output-Transaction-Log. If errors were found, an opportunity for correction is made available.

PDL :

```
Initialize Output-retrieve-error flag to FALSE

Open File-EDI-Acknowledgment

If error free acknowledgment Then
  Call OUTPUT-RECONCILE-ACK to reconcile acknowledgment
Elseif acknowledgment with errors Then
  Call OUTPUT-RESOVLE-SAFER-SEND-ERROR to resolve errors
Endif

Delete acknowledgment from ED1 Acknowledgment Log
```

Files Used :

File-EDI-Acknowledgment

Database Tables Used :

Input Variables:

Input-ack-message-struct

Output Variables :

Input-ack-message-struct	Data Structure

P6: SEND-ACCOUNT-ACKNOWLEDGMENT Return Value :

This module receives and invokes processing for responses sent back to the user indicating the status of his request.

PDL :
Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables :
Account-response-msg-struct

Output Variables :

Account-response-msg-struct	Data Structure

P6: SEND-AUTH-SOURCE-REQ Return Value :

This module receives and invokes processing for appropriately formatted requests for an authoritative source. Includes the authoritative source address and request priority.

PDL :
Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables :
Valid-AS-req-struct

Output Variables :
Valid-AS-req-struct

Valid-AS-req-struct	Data Structure

P6: SEND-BILLING-REPORT Return Value :

This module receives and invokes processing for requests made to the VAN at some specified billing interval for billing information for each organization with an account on the system.

PDL :
Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables :
Billing-info-msg-struct

Output Variables :

Billing-info-msg-struct	Data Structure

.

P6: SEND-CARRIER-ACK Return Value :

This module receives and invokes processing for an acknowledgement for an 'Add' type carrier request (includes the new DOT number).

PDL :
Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables :
Carrier-ack-msg-struct

Output Variables :

Carrier-ack-msg-struct	Data Structure

P6: SEND-EDI-ACK Return Value :

This module accepts ED1 acknowledgments and releases them to the output stream.

PDL :
Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables :
Output-ack-message-struct

Output Variables :

Output-ack-message-struct	Data Structure

P6: SEND-ERROR-MESSAGE Return Value :

This module receives SAFER error messages, determines the severity, and if high, immediately notifies the SAFER operator. All messages are placed into an File-Error-Log for review by a SAFER operator.

PDL :
Determine message severity

If high Then
  Notify SAFER operator
Endif

Enter error message into File-Error-Log

Files Used :
File-Error-Log

Database Tables Used :

Input Variables :
Packet-header+
SAFER-date+
SAFER-error-code+
SAFER-time

Output Variables:

Packet-header	Data Structure
SAFER-date	Data Element
SAFER-error-code	Data Element
SAFER-time	Data Element

P6: SEND-IE-BILL Return Value :

This module receives and invokes processing for bills sent to each organization each billing period.

PDL :
Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables :
User-IE-bill-msg-struct

Output Variables :

User_IE_bill_msg_struct	Data Structure

P6: SEND-SAFETY-DATA Return Value :

This module receives and invokes processing for requests to send snapshot, profile or census information to a user.

PDL :
Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables:
Safety-data-send-struct

Output Variables :

Safety-data-send-struct	Data Structure

P6: SEND-SUBSCRIPTION-REPORT Return Value :

This module receives and invokes processing for a data reports sent to the user on request. It either reflects a user's current subscriber status, and/or system event monitoring status.

PDL :
Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables :
Event-status-msg-struct

Output Variables :

Event-status-msg-struct	Data Structure

P6: SEND-USER-MESSAGE Return Value :

This module receives and invokes processing for messages to be sent to a SAFER user.

PDL :
Receive message

Call OUTPUT-MESSAGE-FORMATTER to process message

Files Used :

Database Tables Used :

Input Variables :
User-message-struct

Output Variables :

User-message-struct	Data Structure

SL-LISTMOV Return Value :

This is a basic routine to get subscription data out of the database and into the correct subscriber processing application format. It is possible that this could not be an actual module, but SQL, data base 'find with a view, etc. It is important, however that it be done in a consistent manner. It is anticipated that SAFER will undergo planned changes, and ensuring that subscription access is always updated correctly and completely is a must.

PDL :

```
For each Subscriber list ID in the process list received,
  Do /*subscribe id*/
  Retrieve a view of subscriber list from subscriber datastore.
  If an error occurs,
    Do /*an err*/
    set all error fields to ON
    Enddo /*an err*/
 Place fields into standard system Subscriber list data structure.

  Enddo /*subscribe id*/
```

Files Used :

Database Tables Used :
Table-Subscriber

Input Variables :
SL-user-seq +
SL-status-flag+
Req-header

Output Variables :
Packet-header +
SL-user-seq +
SL-status-flag+
Req-header+
SL-process-type +
SL-output-type +
SL-process-date +
SL-baseline-data +
[(Carrier-unique-id) I
Query-criteria] +
{Event-name} +
Packet-header +
SL-create-date + Event-update-source +
Event-update-date +
Event-mask

Carrier-unique-id	Data Element
Event-mask	Data Element

Event-name	Data Element
Event-update-date	Data Structure
Event-update-source	Data Element
Packet-header	Data Structure
Query-criteria	Data Structure
Req-header	Data Structure
SL-baseline-data	Data Element
SL-create-date	Data Element
SL-output-type	Data Element
SL-process-date	Data Element
SL-process-type	Data Element
SL-status-flag	Data Element
SL-user-seq	Data Element

TEST-MODE Return Value :

This module is an automated method to get new trading partners on-line and it processes all inbound test transactions. Once the user has successfully completed these tests, he is considered validated'. the User Account User-validated-flag will be updated to TRUE, and the user will be able to send and receive production data.

PDL :

Files Used :

Database Tables Used :

Input Variables:
Input-application-msg-struct

Output Variables :

Input_application_msg_struct	Data Structure

UPDATE-SAFETY-DATA

Return Value :

Profile or snapshot data.

PDL :

Files Used :

Database Tables Used :

Input Variables :
Valid-safety-data-struct

Output Variables :

UPDATE-SUBSCRIBER-LIST

Return Value :

Request for a subscriber list create, update, delete or report.

PDL :

Files Used :

Database Tables Used :

Input Variables :
Subscriber-req-struct

Output Variables :

Credit log

File-EDI -Error
Packet-header+
SAFER-date+
SAFER-error-code +
SAFER-time+
Invalid-transaction+
Translation-error-message

Invalid-transaction	Data Structure
Packet-header	Data Structure
SAFER-date	Data Element
SAFER-error-time	Data Element
SAFER-time	Data Element
Translation-error-message	Data Element

File-EDI-Acknowledgment
Interchange-acknowledgment+
Functional-acknowledgment

Functional-acknowledgment	Data Structure
Interchange-acknowledgment	Data Structure

File-Error-Log
Packet-header +
SAFER-date +
SAFER-error-code +
SAFER-time

Packet-header	Data Structure
SAFER-date	Data Element
SAFER-error-code	Data Element
SAFER-time	Data Element

File-Incoming-Transaction-Log
Functional-group-id+
Input-transaction+
Interchange-id+
Packet-header+
SAFER-date+
S1FER_time+

Transaction-set-header

Functional-group-id	Data Structure
Input-transaction	Data Structure
Interchange-id	Data Structure
Packet-header	Data Structure
SAFER-date	Data Element
SAFER-time	Data Element
Transaction-set-header	Data Structure

File-Outgoin g-Transaction-Log

Interchange-control-header +
Interchange-control-trailer +
Functional-group-header +
Functional-group-trailer +
Interchange-ack-code +
Interchange-note-code +
Output-transaction-struct +
Packet-header +
Output-authorization-code +
Output-operator-id +
Ou tput-terminal-id +
SAFER-date +
SA FER-time

Functional-group-header	Data Structure
Functional-group-trailer	Data Structure
Interchange-ack-code	Data Element
Intcrchange-control-header	Data Structure
Intcrchange-control-trailer	Data Structure
Interchange-note-code	Data Element
Output-authorization-code	Data Element
Output-operator-id	Data Element
Output-terminal-id	Data Element
Output-transaction-struct	Data Structure
Packet-header	Data Structure

SAFER-date	Data Element
SAFER-time	Data Element

File-Receive
Input-interchange-struct

Input-interchange-struct	Data Structure

File-Send
Output-translated-msg-struct

Output-translated-msg-struct	Data Structure

File-Translator-Error
Input-translated-message-struct

Input-translated-message-struct	Data Structure

File-Translntor-Input
Output-auth-msg-struct

Output-auth-msg-struct	Data Structure

File-Translntor-Output
Input-translated-message-struct

Input-translated-message-struct	Data Structure

File-Waiting-to-Send
Packet-header+
Rcq-header+
User-request-struct

Packet-header	Data Structure
Rcq-header	Data Structure
User-request-struct	Data Structure

Invoice log

User Message Queue

Table-Access-Log
Interchange-id +
SAFER-error-code

Interchange-id .	Data Structure
SAFER-error-code	Data Element

Table-Disclosure-Log
Packet-header +
{Personal-data-message) +
SAFER-date +
SAFER-time +
User-id

Packet-header	Data Structure
Personal-data-message	Data Structure
SAFER-date	Data Element
SAFER-time	Data Element
User-id	Data Element

Table-Event
Event-no +
Event-name +
Event-type +
@vent-range-high +
Event-range-low)

Event-name	Data Element
Event-no	Data Element
Even t-range-high	Data Element
Event-range-low	Data Element
Event-type	Data Element

Table-Event-Range
Event-no +
Event-range-no +
Event-range-low +
Event-range-high

Event-no	Data Element
Event-range-high	Data Element

Event-range-low Data EIement

Event-range-no Data Element

Table-Event-Version
E v e n t +
Event-update-date +
Event-update-source

Event-update-date Data Structure

Event-update-source Data Element

Event-vers Data Element

Table-Incoming-Transaction
Packet-header

Packet_header Data Structure

Table-New-Events-Working
SL-ID +
S I-user-seq +
Carrier-unique-id +
Snap-create-date +
S ubscribe-update-time +
Event-mask

Carrier-unique-id Data Element

Event-mask Data Element

SL-ID Data Element

SL-user-seq Data Element

Snap-create-date Data Element

Subscrie-update-time Data Element

Table-Oldsnap
{Snap-data-record-struct}

Snap-data-record-struct Data Structure

Org-name + Suborg-name +
Org-shortname + Org-mailing-address + Org-phone + Org-fax + Org-email + Org-contact + Org-billing-address + Org-billing-contact-name + Org-billing-contact-phon Org-billing-contact.-fax + Org-billing-contact-email +
Org-netwoekaddress + Org-type +

+ Account-id + Org-bank-info + Account-$-value + Last-bill-balance + Last-bill-duedate

Account-$-value	Data Element
Account_id	Data Element
Last-bill-balance	Data Element
Last-bill-duedate	Data Element
Org-bank-info	Data Structure
Org-billing-address	Data Structure
Org-billing-contact-email	Data Element
Org-billing-contact-fax	Data Element
Org-billing-contact-name	Data Element
Org-billing-contact-phone	Data Element
Org-contact	Data Element
Org-emanil	Data Element
Org-fax	Data Element
Org-keywords	Data Element
Org-mailing-adress	Data Structure
Org-name	Data Element
Org-network-address	Data Element
Org-phone	Data Element
Org-shortname	Data Element
Org-type	Data Element
Su borg-name	Data Element

Table-Org-Account-Activity

Org-name +
Account-id +
Acoount-create-date +
Account-delete-date +
Account-update-date+
Changes-to-account

Account-create-date	Data Element

Account-delete-date	Data Element
Account-id	Data Element
Account-update-date	Data Element
Changes-to-account	Data Element
Org-name	Data Element

The-Org-User-Accounts-Xref
{Acount-id + User-id + User-privilege + User-password}

Account-id	Data Element
User-id	Data Element
User-password	Data Element
User-privilege	Data Element

Table-Pend ing
Time-tag +
Send-req +
[AS-profile-req + (Profile-user-list) I Snap-refresh-req]

AS-profile-req	Data Structure
Profile-user-list	Data Structure
Send-req	Data Element
Sanp-refresh-req	Data Structure
Time-tag	Data Element

Table-Profile
{Profile-date-record-struct}

Profile_data_record_struct	Data Structure

Table-Sendlist-Pending
SL-ID +
S L-user-seq +
Rcq-user-id +
SL-process-date +
Carrier-unique-id

Carrier-unique-id	Data Element
Req-user-id	Data Element

SL-ID	Data Element
SL-process-date	Data Element
SL-user-seq	Data Element

Table-Snap
(Snap-data-record-struct}

Snap-data-record-struct	Data Structure

Table-Subscriber
SL-ID +
Account-id +
User-id +
SL-user-seq +
SL-process-type +
SL-output-type +
Sl-process-date +
Q uery-no +
Event-mask +
Ranges-mask +
Veiw-name

Account-id	Data Element
Event-mask	Data Element
Query-no	Data Element
Ranges-mask	Data Element
SL-ID	Data Element
SL-output-type	Data Element
SL-process-date	Data Element
S L-process-type	Data Element
SL-user-seq	Data Element
User-id	Data Element
View-name	Data Element

Table-Subscriber-Parms
SL-ID +
Q uery arm-no +
Q very-parm-value

. Q uery-parm-no	Data Element

Query-parm-value	Data Element
SL-ID	Data Element

Table-User--Account

{Account-id + User-id + User-acct-status + (Reason-for-status-change) + User-access-violation-count + User-address + User-datetime + User-donor + User-exfile + User-privacy + User-standard + User-transactions + User-type + User-subscribe + User-last-name + User-first-name + User-MI + User-phone + User-fax + Card-no + Expiration-date + Van-info)

Account-id	Data Element
Card-no	Data Element
Expiration-date	Data Element
Reason-for-status-change	Data Element
User-MI	Data Element
User-access-violation-count	Data Element
User-acct-status	Data Element
User-address	Data Element
User-datetime	Data Element
User-donor	Data EIement
User-exfile	Data Element
User-fax	Data Element
User-first-name	Data Element
User-id	Data Element
User-last-name	Data Element
User-phone	Data Element
User-privacy	Data Element
User-standard	Data Element
User-subscribe	Data Element
User-transactions	Data Element
User-type	Data Element
Van-info	Data Structure

Table-User-Account-Activity

User-id +
User-first-name +
User-last-name +
Account-id +
Account-create-date +
Account-delete-date +
Changes-to-account +
Account-update-date

Account-create-date	Data Element
Account-delete-date	Data Element
Account-id	Data Element
Account-update-date	Data Element
Changes -to-account	Data Element
User-first-name	Data Element
User-id	Data Element
User-last-name	Data Element

Table-View

View-name +
{Event-name}

Event-name	Data Element
View_name	Data Element