Traveler Information Kiosk
Model Deployment Initiative
System Design Document

Version 1.0

SwRI Project No. 10-8684
P.O. No. 7-70030
Req. No. 60115-7-70030

February 19, 1998

Prepared For:

Texas Department of Transportation
TransGuide
3500 NW Loop 410
San Antonio, Texas 78229

Prepared by:

Southwest Research Institute
P.O. Drawer 28510
San Antonio, Texas 78228

Approval Page

Traveler Information Kiosk Project Manager Date

SwRI MDI Project Manager Date

Software Engineering Director Date

Table of Contents

List of Figures

List of Tables

Acronym List

ASCII	American Standard Code for Information Interchange
ATM	Automated Teller Machine
ATMS	Advanced Traffic Management System
BMP	Bit Map Picture
CD-ROM	Compact Disk-Read Only Memory
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DSIF	Data Server Interface
ECR	Engineering Change Request
EMI	Electromagnetic Interference
ESRI	Environmental Systems Research Institute
FCC	Federal Communications Commission
FM	Frequency Modulation
FMSTIC	FM Subcarrier Traffic Information Channel
FU	Field Unit
GB	Gigabyte
GIF	Graphics Interchange Format
GIS	Geographic Information System
GUI	Graphical User Interface
ITS	Intelligent Transportation Systems
IVN	In-Vehicle Navigation
JPEG	Joint Photographic Experts Group
KFU	Kiosk Field Unit
KMC	Kiosk Master Computer
LAN	Local Area Network
LR	Location Reference
MB	Megabyte
MC	Master Computer
MDI	Model Deployment Initiative
MHz	MegaHertz
MO	MapObjects
NavTech	Navigation Technologies
NT	New Technology
RAM	Random Access Memory
RFO	Request For Offer
RISC	Reduced Instruction Set Computer
SCSI	Small Computer Systems Interface
STM	Size Transmission Message
SwRI	Southwest Research Institute
TBD	To Be Determined
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TDD	Telecommunications Device for the Deaf
TIFF	Tagged Image File Format
TIM	Traffic Information Message
TxDOT	Texas Department of Transportation
UL	Underwriters Laboratories

Kiosk Traveler Information System

System Design Document

1. Introduction

The Traveler Information Kiosk Project involves the development and deployment of interactive traveler information Kiosk Field Units throughout the City of San Antonio. The principal function of these field units is to provide multi-modal traffic information to assist users who travel in San Antonio. Individual field units are composed of a computer, touch-screen monitor, and printer, enclosed in a protective housing. A Kiosk Field Unit will be located in areas such as a shopping mall, tourist attraction, or business.

1.1 Purpose of the System

The main purpose of the Kiosk System is to provide the public with readily accessible, useful and timely travel information that has been obtained from a variety of sources. Users of the Kiosk System are able to request area maps, route guidance information, real-time travel conditions, weather updates, VIA transit information, and information relating to the San Antonio International Airport.

1.2 Operational Concept

The operational concept that motivates the development of the Traveler Information Kiosk System is a need to disseminate information from a wide variety of data sources to the traveling public. The Kiosk System provides a focal point for the acquisition of this information and a convenient and easily used medium for its distribution. Users interact with the Kiosk System using touch-screen monitors and are able to request informative computer generated displays as well printed hardcopies of the requested information. Figure 1 shows the basic flow of information from the individual data sources to users of the Kiosk system.

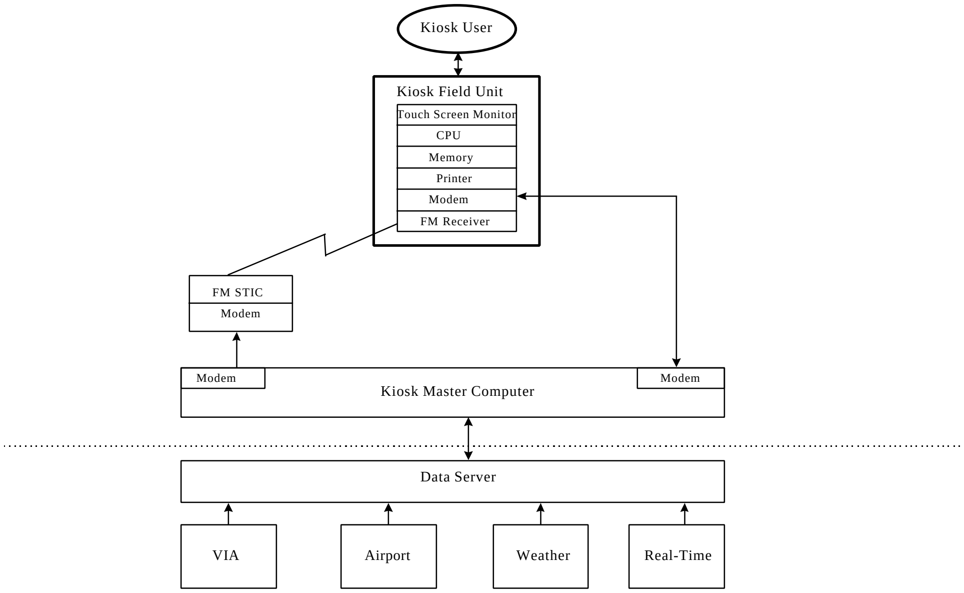

Figure 1 - General Kiosk System Data Flow

1.3 Goals and Objectives

The goal of the Kiosk system is to provide the traveling public with computer generated displays or printed hardcopies of the following types of information:

- San Antonio street map
- route guidance
- real-time traffic conditions
- weather
- VIA
- airport

1.4 Referenced Documents

Southwest Research Institute, *Proposal for the Model Deployment Initiative System Integration*, SwRI Proposal No. 10-20342, November 1996.

Texas Department of Transportation, *Request for Offer (RFO) for the Model Deployment Initiative System Integration, 60115-7-70030*, Specification No. TxDOT 795-SAT-01, October 1996.

Southwest Research Institute, *Traveler Information Kiosk Model Deployment Initiative Preliminary Design Document Version 1.0*, February 14, 1997.

Southwest Research Institute, *In-vehicle Navigation System Model Deployment Initiative Design Document*, January 1998.

Southwest Research Institute, Data Server Model Deployment Initiative Software Design Document Version 1.0, December 1997.

Southwest Research Institute, *Common Code Model Deployment Initiative Design Document Version 1.0*, January 1998.

2. External Interfaces

The Kiosk system has seven External Interfaces as depicted in Figure 2. These interfaces are described below.

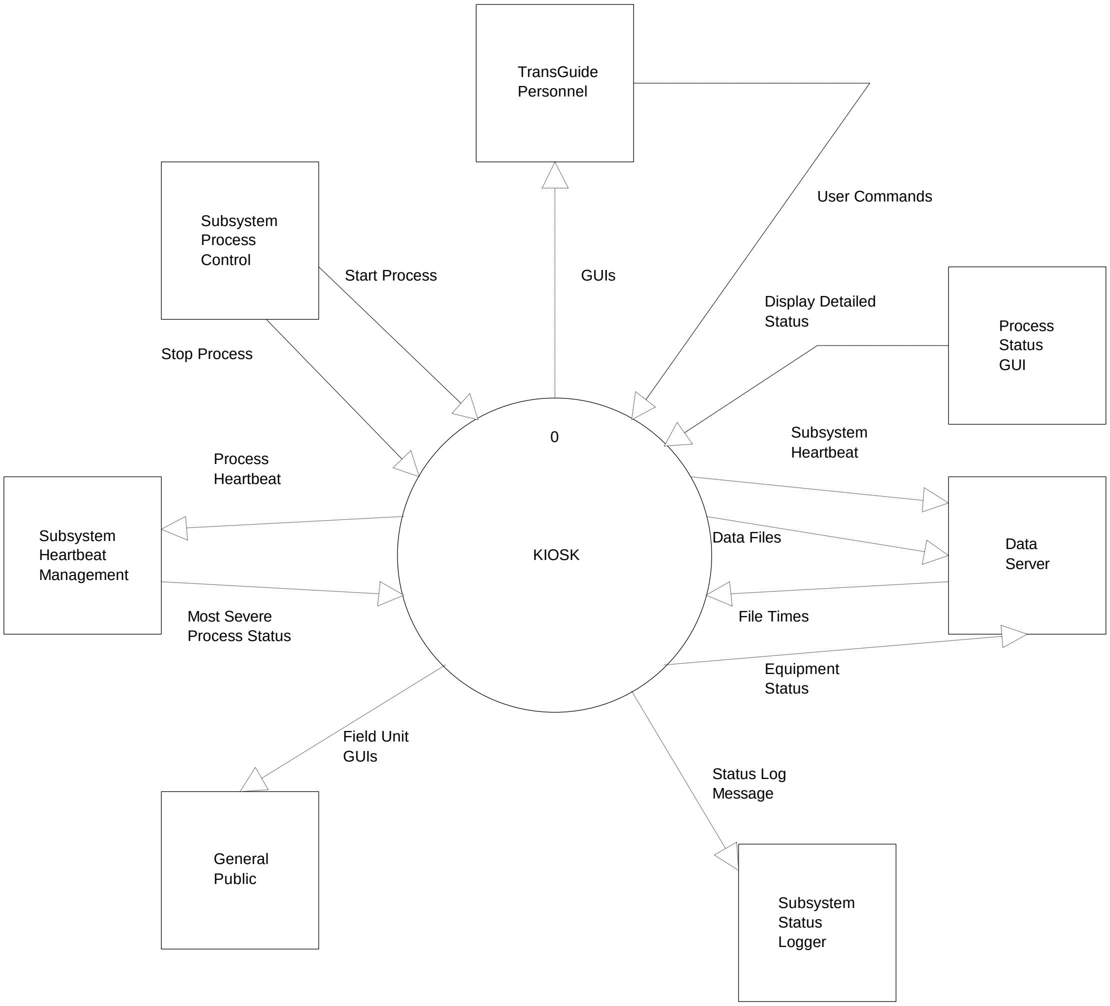

Figure 2 - Kiosk External Interfaces

2.1 TransGuide Personnel

TransGuide Personnel represent the operations and system administration personnel assigned to the TransGuide ATMS. These are the end-users of the Kiosk Master Computer subsystem and will interact with the Kiosk Master Computer subsystem via graphical user interfaces associated with the Detailed Status and System Maintenance GUIs.

2.2 Process Status GUI

Process Status GUI is the graphical user interface providing the visual description of each of the processes within the subsystem. The user has the ability to stop and start processes as configured by the status GUI. The user can also invoke the detailed status GUI of the subsystem from the Process Status GUI. The detailed status GUI can provide information about field equipment associated with the subsystem or other information of importance.

2.3 Data Server

Data Server is the central repository of information generated and maintained by the MDI subsystems. The Kiosk Master Computer subsystem sends weather and transit (VIA) data to the Data Server. The Data Server also receives the subsystem-level heartbeat which includes the overall status of the Kiosk Master Computer subsystem.

2.4 Subsystem Status Logger

Subsystem Status Logger is the process responsible for logging status information to a log file. A log file for each day of the week is maintained. These log files are kept only for the current week.

2.5 General Public

The end users of the Kiosk Field Units.

2.6 Subsystem Heartbeat Management

Subsystem Heartbeat Management receives all the process-level heartbeat messages and maintains the current status information for the subsystem. The most severe process-level status is sent periodically to the Data Server through the subsystem's Data Server Interface.

2.7 Subsystem Process Control

Subsystem Process Control is responsible for starting and automatically restarting the processes associated with the Kiosk Master Computer subsystem.

3. System Requirements

The following sections contain the system requirements for the Traveler Information Kiosk System. The requirements are organized by level and category. The levels that are defined in this document are *general*, *system*, *subsystem*, and *component*. General requirements are non-technical requirements that apply to the program in general. System requirements apply to the system level of the Traveler Information Kiosk System. Subsystem requirements apply to the subsystem design elements that are documented in the Subsystem Level Design section of this document.

The categories of requirements that are defined are general, interface, functional, performance, physical, and miscellaneous. If there are no requirements of a particular category at a particular level, there is no reference to that category at that level. Each of these categories are described below.

- Interface requirements describe the interface between the system and external systems (e.g., the user interface).
- Functional requirements describe the operations which the system must perform (e.g., initialize, acquire data).
- Physical characteristic requirements describe physical constraints of the system (e.g., maximum size, minimum weight).

In addition to the categories described above, there are three types of requirements presented in these sections: MDI RFO requirements, SwRI MDI proposal requirements, and derived requirements. Where a conflict exists, the SwRI MDI Proposal requirements supersede the MDI RFO requirements. In these cases, only the SwRI MDI Proposal requirements are documented. Derived requirements are generated by analysis of the existing requirements.

Several notations are used in the following tables. The requirement number is a three-part number that is used to uniquely identify each requirement. The number consists of the following fields:

<System Mnemonic>-<Requirement Category Mnemonic>-<Requirement Number>

System Mnemonic

The system mnemonic uniquely identifies the Traveler Information Kiosk System to distinguish its requirements from the requirements of the other MDI systems. The system mnemonic for the Traveler Information Kiosk System is *KSK*.

Requirement Category Mnemonic

A mnemonic has been created for each of the requirement categories. They are *GN* - general, *IF* - interface, *FN* - functional, PF - performance, *PY* - physical, and MS - miscellaneous.

Requirement Number

The requirements are numbered sequentially within a given category. The requirements at the system level each have a single requirement number. As requirements are derived at the subsystem and component levels, additional numbers are added to show the relationship between requirements. For example, requirement *KSK-IF-1* at the system level may have two children at the subsystem level, *KSK-IF-1.1* and *KSK-IF-1.2*. With this numbering scheme it is easy to determine a requirement's parent and the level of the requirement.

Each of these requirements are further documented in Section 5 in the traceability matrix. For each requirement, the matrix contains traceability information to show the relationship between the requirement and other requirements, design elements, and the Acceptance Test Plan (ATP).

3.1 System Level Requirements

The requirements listed below are the system level requirements for the MDI Kiosk System.

3.1.1 Physical Requirements

The physical requirements for the Kiosk System are listed below in Table 1.

Number	**Requirement**
KSK-PY-1	The Kiosk Master Computer shall be a Sun Microsystems Ultra SPARCStation with the following configuration: • 167 MHZ SPARC (RISC) CPU, • 4.2 Gigabyte hard disk, • 128 Megabytes RAM, • Floppy Disk, • CD-ROM, • Turbo GX+ Graphics, • 20 Inch color monitor, • 8 port modem server (SCSI) attached, • Dual Ethernet Interface, and • Dual SCSI Channels.
KSK-PY-2	The Indoor and Outdoor Kiosk Field Unit computers shall have, at a minimum, the following configuration: • Windows 95, • 120 MHz processor clock speed, • 32 MB RAM, • 1.6 GB hard disk drive, • 3.5 inch 1.44 MB floppy drive, • 8X CD-ROM drive, • 1 RS-232 asynchronous communication port, • 1 bi-directional parallel port, • 101 key enhanced keyboard, • 2 button mouse, and • an internal modem.

Number	**Requirement**
KSK-PY-4	The Indoor Kiosk shall include the following: • Antenna/receiver assembly, • Processor with keyboard, • Touch-screen monitor, • Speakers, • Printer, • Power strip, • Cooling fan, • UL & FCC certification, • Rated to operate at an ambient temperature range from 60 to 85 degrees Fahrenheit, • Rated to operate at a non-condensing humidity range from 35 to 80 percent relative humidity.
KSK-PY-5	The Outdoor Kiosk shall include the following: • Antenna/receiver assembly, • Processor with keyboard, • Touch-screen monitor, • Speakers, • Printer, • Modem, • Heating/cooling system, • UL & FCC certification, • Rated to operate at an ambient temperature range from -10 to 115 degrees Fahrenheit, • Rated to operate at a non-condensing humidity range from 20 to 100 percent relative humidity.
KSK-PY-6	The Indoor Kiosk enclosure shall be rated at the following environment specifications: • Ambient temperature range of 60 to 85 degrees Fahrenheit. • Non-condensing humidity range from 35 to 80 percent relative humidity.
KSK-PY-7	The Outdoor Kiosk enclosure shall be rated at the following environment specifications: • Ambient temperature range of –10 to 115 degrees Fahrenheit. • Non-condensing humidity range from 20 to 100 percent relative humidity.

Table 1. - Physical Requirements

3.1.2 Interface Requirements

The Interface Requirements for the Kiosk System are listed in Table 2.

Number	**Requirement**
KSK-IF-1	The Kiosk System shall interface with the Data Server.
KSK-IF-2	The Kiosk System shall interface with the In-Vehicle Navigation system data stream being transmitted utilizing the STIC communication system for real-time traffic conditions data.
KSK-IF-3	The Kiosk System shall interface with the weather data source.
KSK-IF-4	The Kiosk System shall interface with the airport data source.
KSK-IF-5	The Kiosk System shall interface with the VIA data source.
KSK-IF-6	The Kiosk System shall interface with screen saver data source(s).
KSK-IF-7	The Kiosk System shall interface with the Kiosk Field Units.
KSK-IF-8	The Kiosk System shall interface with the general public through a touchscreen, using a Graphical User Interface.

Table 2. - Interface Requirements

3.1.3 Functional Requirements

The functional requirements for the Kiosk System are listed in Table 3.

Number	**Requirement**
KSK-FN-1	The Kiosk System shall display the real-time traffic conditions of the highways/roadways monitored by TransGuide.
KSK-FN-2	The Kiosk System shall display weather data.
KSK-FN-3	The Kiosk System shall display airport data.
KSK-FN-4	The Kiosk System shall display VIA data.
KSK-FN-5	The Kiosk System shall display screen saver (advertisements) files when the Kiosk is not being accessed by a user.
KSK-FN-6	The Kiosk System shall provide system diagnostics.
KSK-FN-7	The Kiosk System shall provide route guidance.

Number	Requirement
KSK-FN-8	The Kiosk System shall be capable of printing user selected items.
KSK-FN-9	The Kiosk System shall provide help to assist the user in the operation of the Kiosk application.
KSK-FN-10	Kiosk System Startup.

Table 3. - Functional Requirements

3.2 Subsystem Level Requirements

The following sections describe the subsystem requirements for the Kiosk System. These requirements are divided into two areas, Kiosk Master Computer and Kiosk Field Units.

3.2.1 Kiosk Master Computer

The following sections describe the Kiosk Master Computer interface and functional requirements.

3.2.1.1 Kiosk Master Computer Interface Requirements

The Kiosk Master Computer interface requirements are listed in Table 4.

Number	Requirement
KSK-IF-1.1a	The Kiosk System shall be capable of submitting the San Antonio area weather conditions to the Data Server.
KSK-IF-1.1b	The Kiosk System shall be capable of submitting the San Antonio area weather forecast to the Data Server.
KSK-IF-1.1c	The Kiosk System shall be capable of submitting the current San Antonio area radar map to the Data Server.
KSK-IF-1.1d	The Kiosk System shall be capable of retrieving the San Antonio area weather conditions from the Data Server.
KSK-IF-1.1e	The Kiosk System shall be capable of retrieving the San Antonio area weather forecast from the Data Server.
KSK-IF-1.1f	The Kiosk System shall be capable of retrieving the current San Antonio area radar map from the Data Server.
KSK-IF-1.2a	The Kiosk System shall be capable of submitting airline and airport terminal information to the Data Server.
KSK-IF-1.2b	The Kiosk System shall be capable of submitting airport rental agency information to the Data Server.

Number	**Requirement**
KSK-IF-1.2c	The Kiosk System shall be capable of submitting airport parking lot information to the Data Server.
KSK-IF-1.2d	The Kiosk System shall be capable of retrieving airline and airport terminal information from the Data Server.
KSK-IF-1.2e	The Kiosk System shall be capable of retrieving airport rental agency information from the Data Server.
KSK-IF-1.2f	The Kiosk System shall be capable of retrieving airport parking lot information from the Data Server.
KSK-IF-1.3a	The Kiosk System shall be capable of submitting route schedules to the Data Server.
KSK-IF-1.3b	The Kiosk System shall be capable of submitting standard and special fares to the Data Server.
KSK-IF-1.3c	The Kiosk System shall be capable of submitting park & ride locations to the Data Server.
KSK-IF-1.3d	The Kiosk System shall be capable of submitting special bus events and the associated schedules to the Data Server.
KSK-IF-1.3e	The Kiosk System shall be capable of submitting VIA handicapped bus dispatch (VIATrans) services to the Data Server.
KSK-IF-1.3f	The Kiosk System shall be capable of submitting general VIA information to the Data Server.
KSK-IF-1.3g	The Kiosk System shall be capable of submitting graphical displays of selected bus routes data to the Data Server.
KSK-IF-1.3h	The Kiosk System shall be capable of retrieving route schedules from the Data Server.
KSK-IF-1.3i	The Kiosk System shall be capable of retrieving standard and special fares from the Data Server.
KSK-IF-1.3j	The Kiosk System shall be capable of retrieving park & ride locations from the Data Server.
KSK-IF-1.3k	The Kiosk System shall be capable of retrieving special bus events and the associated schedules from the Data Server.
KSK-IF-1.3l	The Kiosk System shall be capable of retrieving VIA handicapped bus dispatch (VIATrans) services from the Data Server.

Number	**Requirement**
KSK-IF-1.3m	The Kiosk System shall be capable of retrieving general VIA information from the Data Server.
KSK-IF-1.3n	The Kiosk System shall be capable of retrieving displays of selected bus routes data from the Data Server.
KSK-IF-3.1a	The Kiosk System shall be capable of retrieving the San Antonio area weather conditions from the weather data source.
KSK-IF-3.1b	The Kiosk System shall be capable of retrieving the San Antonio area weather forecast from the weather data source.
KSK-IF-3.1c	The Kiosk System shall be capable of retrieving the current San Antonio area radar map data from the weather data source.
KSK-IF-4.1	The Kiosk Master Computer shall be capable of receiving airport terminal, airport rental agency, and airport parking lot data from the airport data source.
KSK-IF-5.1a	The Kiosk Master Computer shall be capable of receiving route schedules from the VIA data source.
KSK-IF-5.1b	The Kiosk Master Computer shall be capable of receiving standard and special fares from the VIA data source.
KSK-IF-5.1c	The Kiosk Master Computer shall be capable of receiving park & ride locations from the VIA data source.
KSK-IF-5.1d	The Kiosk Master Computer shall be capable of receiving special bus events and the associated schedules from the VIA data source.
KSK-IF-5.1e	The Kiosk Master Computer shall be capable of receiving VIA handicapped bus dispatch (VIATrans) services from the VIA data source.
KSK-IF-5.1f	The Kiosk Master Computer shall be capable of receiving general VIA information from the VIA data source.
KSK-IF-5.1g	The Kiosk Master Computer shall be capable of receiving graphical displays of selected bus routes from the VIA data source.
KSK-IF-6.1	The Kiosk Master Computer shall be capable of receiving screen saver files.
KSK-IF-7.1a	The Kiosk Master Computer shall be capable of transmitting the San Antonio area weather conditions to the Kiosk Field Units.
KSK-IF-7.1b	The Kiosk Master Computer shall be capable of transmitting the San Antonio area weather forecast to the Kiosk Field Units.

Number	**Requirement**
KSK-IF-7.1c	The Kiosk Master Computer shall be capable of transmitting the current San Antonio area radar map data to the Kiosk Field Units.
KSK-IF-7.2a	The Kiosk Master Computer shall be capable of transmitting airport terminal data to the Kiosk Field Units.
KSK-IF-7.2b	The Kiosk Master Computer shall be capable of transmitting airport rental agency data to the Kiosk Field Units.
KSK-IF-7.2c	The Kiosk Master Computer shall be capable of transmitting airport parking lot data to the Kiosk Field Units.
KSK-IF-7.3a	The Kiosk Master Computer shall be capable of transmitting route schedules to the Kiosk Field Units.
KSK-IF-7.3b	The Kiosk Master Computer shall be capable of transmitting standard and special fares, park & ride locations to the Kiosk Field Units.
KSK-IF-7.3c	The Kiosk Master Computer shall be capable of transmitting special bus events and the associated schedules to the Kiosk Field Units.
KSK-IF-7.3d	The Kiosk Master Computer shall be capable of transmitting VIA handicapped bus dispatch (VIATrans) services to the Kiosk Field Units.
KSK-IF-7.3e	The Kiosk Master Computer shall be capable of transmitting general VIA information to the Kiosk Field Units.
KSK-IF-7.3f	The Kiosk Master Computer shall be capable of transmitting graphical displays of selected bus routes data to the Kiosk Field Units.
KSK-IF-7.3g	The Kiosk Master Computer shall be capable of transmitting park & ride locations to the Kiosk Field Units.
KSK-IF-7.4	The Kiosk Master Computer shall be capable of transmitting screen saver files to the Kiosk Field Units.

Table 4. - Master Computer Interface Requirements

3.2.1.2 Kiosk Master Computer Functional Requirements

The Kiosk Master Computer functional requirements are listed in Table 5.

Number	**Requirement**
KSK-FN-5.1a	The Kiosk Master Computer shall accept bitmap (.bmp) files for the displaying of graphical displays on the Kiosk Field Unit.
KSK-FN-5.1b	The Kiosk Master Computer shall accept wave (.wav) files for the playing of audio files on the Kiosk Field Unit.
KSK-FN-5.1c	The Kiosk Master Computer shall accept audio video interleaved (.avi) files for playing video clips on the Kiosk Field Unit.
KSK-FN-6.1	A Kiosk Master Computer Diagnostic Status GUI shall be implemented that displays the last known status of the Kiosk Field Units.
KSK-FN-6.2	The Kiosk Master Computer shall automatically interrogate the Kiosk Field Units.
KSK-FN-6.3	The Kiosk Master Computer shall provide the capability to manually interrogate individual Kiosk Field Units.
KSK-FN-6.4	The Kiosk Master Computer shall store the interrogation status results.
KSK-FN-6.6	The Kiosk Master Computer shall have the capability to download data and screen saver files.
KSK-FN-6.7	The Kiosk Master Computer shall upload Kiosk Field Unit usage statistics.
KSK-FN-6.7a	The Kiosk Master Computer shall upload Kiosk Field Unit statistics on the number of times the Kiosk is used.
KSK-FN-6.7b	The Kiosk Master Computer shall upload Kiosk Field Unit statistics on the number of times the San Antonio Map is accessed.
KSK-FN-6.7c	The Kiosk Master Computer shall upload Kiosk Field Unit statistics on the number of times Airport information is accessed.
KSK-FN-6.7d	The Kiosk Master Computer shall upload Kiosk Field Unit statistics on the number of times Weather information is accessed.
KSK-FN-6.7e	The Kiosk Master Computer shall upload Kiosk Field Unit statistics on the number of times VIA Transit information is accessed.
KSK-FN-6.7f	The Kiosk Master Computer shall upload Kiosk Field Unit statistics on the number of times Route Guidance is accessed.

Number	Requirement
KSK-FN-10.2	The Master Computer subsystem shall provide monitoring and restarting of its applications.

Table 5. - Master Computer Functional Requirements

3.2.2 Kiosk Field Unit

The following sections describe the Kiosk Field Unit interface and functional requirements.

3.2.2.1 Kiosk Field Unit Interface Requirements

The Kiosk Field Unit Interface Requirements are listed in Table 6.

Number	Requirement
KSK-IF-2.1	The Kiosk Field Unit shall receive the real-time traffic condition data broadcast from the STIC communication network.
KSK-IF-8.1	The Kiosk Field Unit shall provide touchscreen interaction for users to interface with the Map Display.
KSK-IF-8.2	The Kiosk Field Unit shall provide touchscreen interaction for users to interface with the Transit Display.
KSK-IF-8.3	The Kiosk Field Unit shall provide touchscreen interaction for users to interface with the Airport Display.
KSK-IF-8.4	The Kiosk Field Unit shall provide touchscreen interaction for users to interface with the Weather Display.
KSK-IF-8.5	The Kiosk Field Unit shall provide touchscreen interaction for users to interface with the Route Guidance Display.

Table 6. - Kiosk Field Unit Interface Requirements

3.2.2.2 Kiosk Field Unit Functional Requirements

The Kiosk Field Unit functional requirements are listed in Table 7.

Number	Requirement
KSK-FN-1.2	The Kiosk Field Unit shall be capable of displaying real-time traffic data using a San Antonio Map Display.
KSK-FN-1.3	The Kiosk Field Unit map shall display traffic conditions using color-coding.
KSK-FN-1.4	The Kiosk Field Unit map shall display incidents and lane closures utilizing icons.

Number	Requirement
KSK-FN-1.5	The Kiosk Field Unit shall provide additional information about an incident or lane closure when the respective icon is touched.
KSK-FN-1.7	The Kiosk Field Unit map shall display current airport traffic conditions for instrumented sections of highway around the San Antonio International Airport.
KSK-FN-1.8	The Kiosk Field Unit map shall identify city streets, residential streets, and highways.
KSK-FN-1.9	The Kiosk Field Unit map shall have the capability to zoom in and out of the San Antonio Street Map Display utilizing touch screen input.
KSK-FN-1.10	The Kiosk Field Unit map shall have the capability to pan the San Antonio Street Map Display utilizing touch screen input.
KSK-FN-1.11	The Kiosk Field Unit map shall display icons indicating locations of automated teller machines (ATMs), shopping centers, restaurants, gas stations, tourist attractions, hospitals, schools, parks, airports, and bus stops.
KSK-FN-1.12	The Kiosk Field Unit San Antonio Street Map Display software shall integrate data from the Navigation Technologies San Antonio Region database with real-time data from the Data Server.
KSK-FN-1.13	The Kiosk Field Unit map real-time traffic conditions shall be updated at least every five (5) minutes.
KSK-FN-2.1	The Kiosk Field Unit shall display the current San Antonio weather conditions.
KSK-FN-2.2	The Kiosk Field Unit shall display the local San Antonio forecast.
KSK-FN-2.3	The Kiosk Field Unit shall display a San Antonio area radar map.
KSK-FN-2.4	The Kiosk Field Unit current weather conditions shall be updated when updates are provided by the weather data source.
KSK-FN-2.5	The Kiosk Field Unit San Antonio area radar map shall be updated when updates are provided by the weather data source.
KSK-FN-2.6	The Kiosk Field Unit local San Antonio forecast shall be updated when updates are provided by the weather data source.
KSK-FN-3.1	The Kiosk Field Unit shall display the traffic conditions for the sections of instrumented highway that surround the airport.
KSK-FN-3.2	The Kiosk Field Unit shall display a listing of local airline names, their phone numbers and the terminal in which they are located.

Number	Requirement
KSK-FN-3.3	The Kiosk Field Unit shall display a listing of local rental car agencies and their phone numbers located at the San Antonio International Airport.
KSK-FN-3.4	The Kiosk Field Unit shall display a listing of the location and cost of airport parking lots.
KSK-FN-4.1	The Kiosk Field Unit shall display route schedules and graphical displays of the routes that are available.
KSK-FN-4.2	The Kiosk Field Unit shall provide scheduled times for major bus stops on a selected route.
KSK-FN-4.3	The Kiosk Field Unit shall display a description of standard and special fares.
KSK-FN-4.4	The Kiosk Field Unit shall display a description of park & ride locations.
KSK-FN-4.5	The Kiosk Field Unit shall display a description of special bus events and the associated schedules.
KSK-FN-4.6	The Kiosk Field Unit shall display information about VIA handicapped bus dispatch (VIATrans) services.
KSK-FN-4.7	The Kiosk Field Unit shall display general VIA information.
KSK-FN-5.2	The Kiosk Field Units shall be capable of receiving screen saver files from the Master Computer and updating the existing screen saver.
KSK-FN-5.3	The Kiosk Field Units shall be capable of executing the screen saver.
KSK-FN-6.9	The Kiosk Field Unit shall be capable of reporting status to the Kiosk Master Computer.
KSK-FN-6.10	The Kiosk Field Unit diagnostic software shall accept non-real-time file updates from the Kiosk Master Computer.
KSK-FN-6.12	The Kiosk Field Unit shall keep usage statistics.
KSK-FN-6.12a	The Kiosk Field Unit shall keep statistics on the number of times the Kiosk is used.
KSK-FN-6.12b	The Kiosk Field Unit shall keep statistics on the number of times the San Antonio Map is accessed.
KSK-FN-6.12c	The Kiosk Field Unit shall keep statistics on the number of times Airport information is accessed.

Number	Requirement
KSK-FN-6.12d	The Kiosk Field Unit shall keep statistics on the number of times Weather information is accessed.
KSK-FN-6.12e	The Kiosk Field Unit shall keep statistics on the number of times VIA Transit information is accessed.
KSK-FN-6.12f	The Kiosk Field Unit shall keep statistics on the number of times Route Guidance information is accessed.
KSK-FN-7.1	The Kiosk Field Units shall convert the real-time traffic condition data stream into data that can be interpreted by the Navigation Technologies database and the Route Guidance application.
KSK-FN-7.2	The Kiosk Field Unit shall be capable of displaying route guidance using the Navigation Technologies database.
KSK-FN-7.3	The Kiosk Field Unit shall provide a graphical display of the route from the kiosk's location to the selected destination.
KSK-FN-7.4	The Kiosk Field Unit shall allow the user to ask for a route from the Kiosk Field Unit's location to a selected Point of Interest.
KSK-FN-7.5	The Kiosk Field Unit shall allow the user to select their destination from a list of the points of interest retrieved from the Navigation Technologies database.
KSK-FN-7.6	The Kiosk Field Unit shall allow the user to enter the address of the destination.
KSK-FN-7.7	The Kiosk Field Unit shall utilize a color-coded line segment on the San Antonio Street Map to indicate the calculated route.
KSK-FN-7.8	The Kiosk Field Unit shall utilize real-time speed information to calculate travel time to the selected destination.
KSK-FN-7.9	The Kiosk Field Unit shall display the estimated travel time and speed for the selected route.
KSK-FN-7.10	The Kiosk Field Unit shall display turn-by-turn instructions for a calculated route.
KSK-FN-8.2	The Kiosk Field Unit shall be capable of printing the route map and instructions.
KSK-FN-8.3	The Kiosk Field Unit shall be capable of printing the transit information.
KSK-FN-8.4	The Kiosk Field Unit shall be capable of printing the airport information.
KSK-FN-8.5	The Kiosk Field Unit shall be capable of printing the local weather conditions, the local forecast and the radar map.

Number	**Requirement**
KSK-FN-8.6	The Kiosk Field Unit shall be capable of printing the route instructions and map.
KSK-FN-9.1	The Kiosk Field Unit shall provide Help buttons to provide information on how to use the GUI currently displayed.
KSK-FN-10.3	The Kiosk Field Unit subsystem unattended applications shall automatically startup at boot-up.
KSK-FN-10.4	The Kiosk Field Unit subsystem shall provide monitoring and restarting of its applications.

Table 7. - Kiosk Field Unit Functional Requirements

4. System Design

The Kiosk System design is presented in the sections that follow. The design is presented by describing the Kiosk System Architecture, the System Level Design, and the Kiosk Subsystem Design. The Kiosk design is intended to describe the implementation of the requirements described earlier.

4.1 System Architecture

Conceptually, the Kiosk System Architecture is composed of the Master Computer and Kiosk Field Units. Figure 3 depicts the System Architecture. The basic concept of the Kiosk System is that the Kiosk Master Computer retrieves and distributes the data to the Kiosk Field Units and the Kiosk Field Units utilize the data to display information to users. Physically, the Kiosk Master Computer is comprised of multiple applications residing on multiple systems. These applications provide the necessary functions to retrieve and distribute the data utilized by the Kiosk Field Units. In addition, the IVN Master Computer and the Kiosk Master Computer physically resided on the same computer but are depicted separately to illustrate the Kiosk System Architecture conceptually. The Kiosk Field Units application software is the same on each Kiosk Field Unit. The application software utilizes configuration files located on the Kiosk Field Unit to identify the specific information unique to each Kiosk.

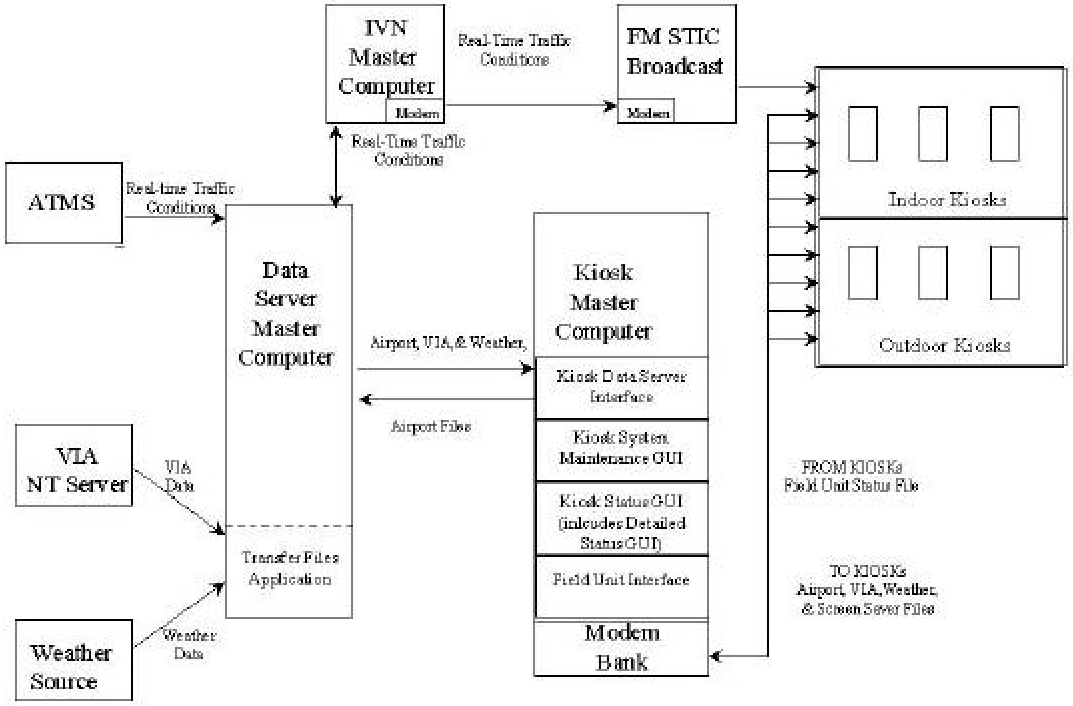

Figure 3 - System Architecture

4.2 System Level Design

The Kiosk System Design is divided into the Kiosk Master Computer (KMC) Subsystem, the Kiosk Field Unit (KFU) Subsystem and the MapMatch Application (builds translation table). The MapMatch Application is an independent application that is used to develop the Translation Table used by the Kiosk Field Units. The application is executed when a new Navigation Technologies data base is received and needs to be distributed to the Kiosk Field Units. Since this application is executed infrequently and provides an input file to the Kiosk system, the design is described in this document, but the application is not considered part of the Kiosk system.

Figure 4 depicts the Kiosk System Design and the data flows between the KMC Subsystem and the KFU Subsystem. Table 8 provides a description of the Kiosk subsystems and their associated data flows.

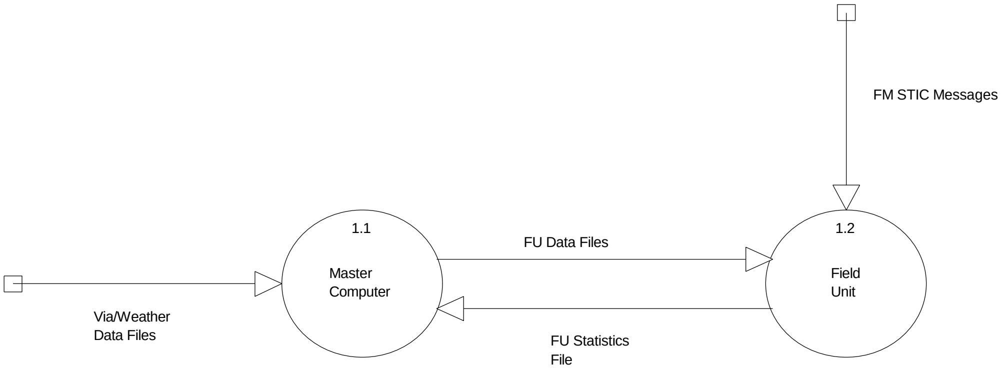

Figure 4 - Kiosk Subsystems and Assoicated Data Flows

Function	Description
Field Unit	The Field Units provides the interface to the general public. The Field Unit provides a San Antonio street map, route guidance to Points of Interest and specific addresses, airport information, weather information, and VIA information. The Field Unit subsystem is composed of the Startup/Error Server Process, the Modem Communications Process, the Real Time Process, and the GUI process.
FM STIC Messages	The FM STIC Messages are broadcast by the In-Vehicle Navigation project and the structure of the messages can be found in the In-Vehicle Navigation Design Document.
FU Data Files	These files are received from the Master Computer and placed into the production directories. These files include Transit files, Weather files, Airport files, and Screen Saver files.
FU Statistics File	Contains the current status of the Field Unit and its usage statistics pertaining to user interaction with the Field Unit.
Master Computer	The Master Computer provides the capability to gather weather, VIA, airport, and screensaver data files, transmit these data files to the Field Units, retrieve status data from the Field Units, interact with the user through a GUI to view status information, and interact with the user through a GUI to modify airport files and the screen saver control file.
Via/Weather Data Files	These files are received from external sources. The VIA files are retrieved from an NT Server that is maintained by VIA. The weather files are retrieved from the TxDOT Web Server. The weather files are placed on the TxDOT Web Server by the weather provider on an hourly basis.

Table 8 - Kiosk Subsystems and Data Flows Description

4.2.1 Kiosk Master Computer Subsystem

The Kiosk Master Computer (KMC) Subsystem is comprised of multiple applications and functions residing on multiple systems. The applications and functions combined together conceptually form the KMC Subsystem. The KMC Subsystem consists of the Status GUI, Detailed Status GUI, System Maintenance GUI, Status Logger, Data Server Interface (DSIF), Transfer Data Files, and Kiosk MC Main. The Status GUI displays the current status of the Kiosk applications executing on the KMC. The Detailed Status GUI, which is invoked from the Status GUI, displays the current state and detailed status information for each Kiosk Field Unit. The System Maintenance GUI provides the capability to modify airport files and the screensaver control file. The Status Logger logs messages sent to it from the other kiosk applications executing on the KMC. The Data Server Interface provides the interface between the Data Server and the kiosk applications executing on the KMC. The Transfer Data Files retrieves weather and VIA data files from their respective sources and submits the files to the Data Server. This application is resident on the Data Server Master Computer. The Kiosk MC Main retrieves data files from the Data Server (through DSIF), transmits the data files to the Field Units, and retrieves status data from the Field Units.

The KMC Subsystem also includes functions in the In-Vehicle Navigation application which retrieve and broadcast the real-time traffic conditions to the Field Units. These functions are being implemented by the In-Vehicle Navigation project. Figure 5 illustrates the conceptual makeup of the Kiosk Master Subsystem and its applications.

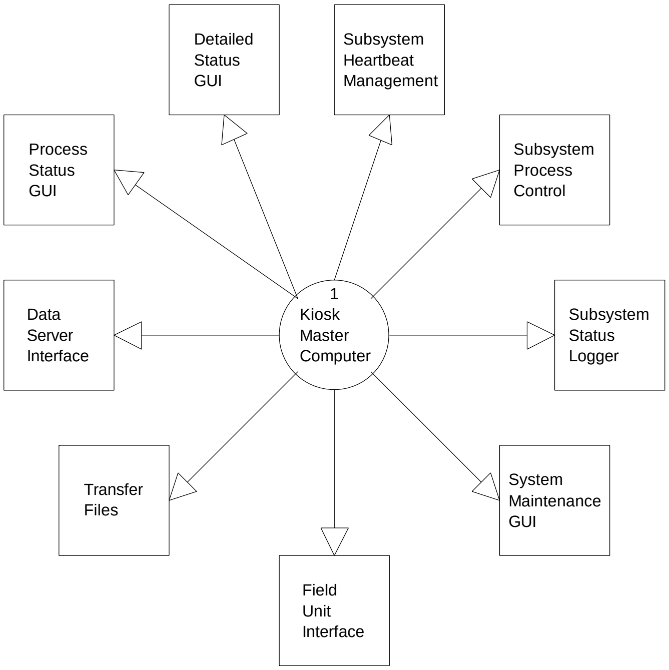

Figure 5 - Kiosk Master Computer Application Identification

Figure 6 illustrates the data flows for the applications residing on the KMC. Table 9 describes the applications and data flows residing on the KMC.

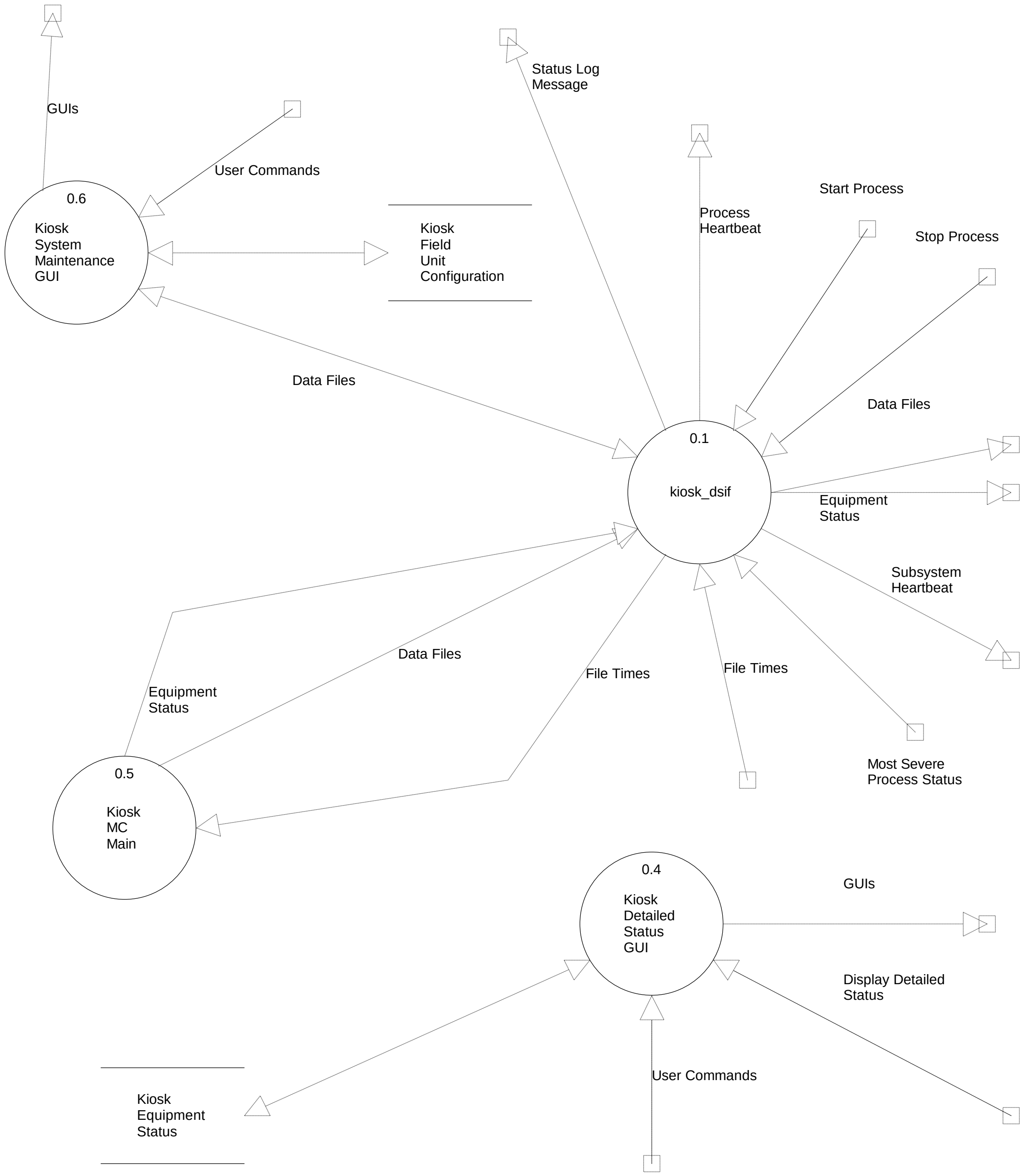

Figure 6 - Kiosk Master Computer Resident Applications and Data Flows

Function	Description
Data Files	Data Files are files that are stored at the Data Server and include files such as weather files, screen saver files, and airport files.
Display Detailed Status	Display Detailed Status is an event used to trigger the display of the subsystem's detailed status GUI.
Equipment Status	Equipment Status is used to define the current state of different equipment. For the Kiosk subsystem the Equipment Status is used to send the Kiosk Field Unit Equipment Status to the Data Server.
File Times	File Times are the last update times associated with the Data Files stored at the Data Server.
GUIs	GUIs are graphical user interfaces. These interfaces are used to communicate information from the subsystem to the user and to allow the user to control certain aspects of the execution of the subsystem.
Kiosk Detailed Status GUI	Kiosk Detailed Status GUI is the graphical user interface providing the TransGuide personnel with the ability to view the current status and data for the Kiosk Field Units being monitored by the Kiosk subsystem. This GUI also allows the TransGuide personnel to modify the state of the Kiosk Field Units and to request downloads to be made to the Kiosk Field Units.
Kiosk Equipment Status	Kiosk Equipment Status defines the status information for each of the Kiosk Field Units. This information includes the current state of the field unit, the number of files that need to be downloaded, the current information about the field unit paper level, disk space, and user accesses as well as the phone number and location of the field unit.
Kiosk Field Unit Configuration	Kiosk Field Unit Configuration contains configuration data for each of the field units being monitored by the Kiosk Subsystem. This file contains the location of the kiosk, the phone number of the kiosk, and the name assigned to the kiosk.
Kiosk MC Main	The Kiosk Master Computer Main Process is responsible for communicating with the Kiosk Field Units.
Kiosk System Maintenance GUI	Kiosk System Maintenance GUI is the graphical user interface providing the TransGuide personnel with the ability to modify the data files associated with the Kiosk Field Units being monitored by the Kiosk subsystem. These data files include the airport information files, the screen saver information files, and the Kiosk Field Unit configuration files. The airport files and screen saver files are stored at the Data Server. The Kiosk Field Unit configuration files are maintained locally on the Kiosk Master Computer.
kiosk_dsif	kiosk_dsif receives messages to be sent to the Data Server and sends these messages on to the Data Server. The results of the interaction with the Data Server are sent back to the originator of the message. This process represents the subsystem's single interface point to the Data Server. This process periodically sends a heartbeat message containing the status of the process to the subsystems heartbeat process.
Most Severe Process Status	Most Severe Process Status is the value of the process status being managed by the Subsystem Heartbeat Management that represents the worst status of all the processes. For example if all processes indicated an ok status except one process indicated a warning status then the Most Severe Process Status would be warning.
Process Heartbeat	Process Heartbeat is the heartbeat pulse sent from each process within the subsystem. The Process Heartbeat contains the status information for the process along with the process identifier.
Start Process	Start Process is an event used to start the execution of a process.

Function	Description
Status Log Message	Status Log Message contains information to be logged to the subsystem log file. Typical Status Log Messages include error messages such as memory allocation errors or data being logged from field equipment associated with the subsystem.
Stop Process	Stop Process is an event used to stop the execution of a process.
Subsystem Heartbeat	Subsystem Heartbeat is the heartbeat message containing the overall status of the KIOSK subsystem. This message is generated by the Subsystem Heartbeat Management process and is passed on to the Data Server by the subsystem's Data Server Interface process.
User Commands	User Commands are the commands selected by the user from the graphical user interfaces. These commands are generated through push buttons, radio buttons, text boxes, and other user interface components.

Table 9 - Kiosk Master Computer Resident Application and Data Flow Descriptions

4.2.1.1 Status GUI

The Status GUI was developed as part of the MDI common code. For design information about the Status GUI consult the Common Code Model Deployment Initiative Design Document.

4.2.1.2 Detailed Status GUI

The Detailed Status GUI provides status information about each of the kiosk field units connected to the Master Computer. Figure 7 depicts the data flows for the Detailed Status GUI. A description of the data flows is provided in Table 10.

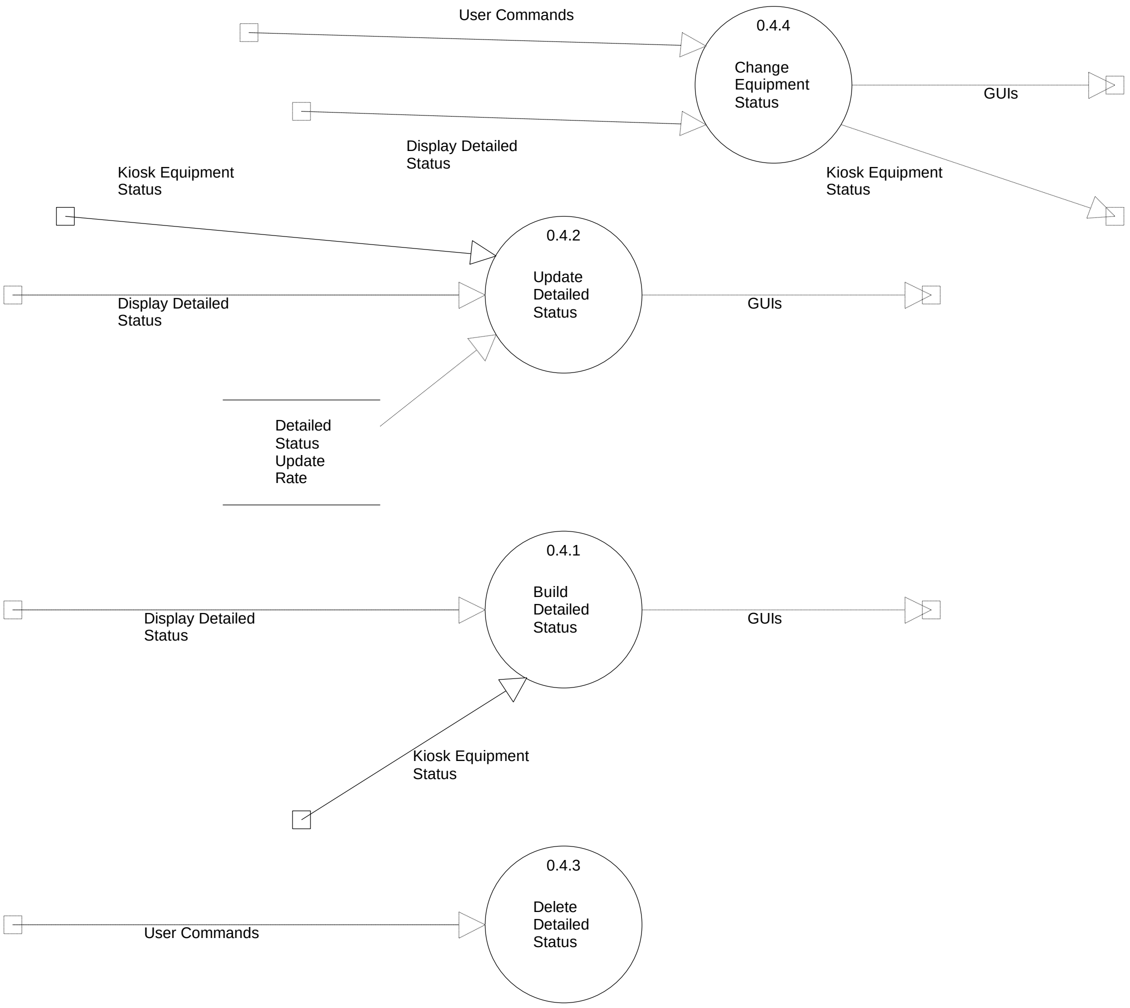

Figure 7 - Detailed Status GUI Data Flow

Function	Description
Build Detailed Status	Build Detailed Status is responsible for generating the initial graphical user interface displaying the KIOSK subsystem detailed status. The detailed status includes the current status of each of the Kiosk Field Units. The Kiosk Equipment Status is used to initially fill in the details of the GUI.
Change Equipment Status	Change Equipment Status allows the user to modify the current state of a selected kiosk field unit. This allows the user to take a field unit offline or to initiate a download of the kiosk field unit.
Delete Detailed Status	Delete Detailed Status deletes the detailed status GUI from the display. This process is invoked when the TransGuide personnel issue the "close" command for the detailed status GUI.
Detailed Status Update Rate	Detailed Status Update Rate is the configuration item that specifies how often the contents of the detailed status GUI are updated. This update rate is specified in seconds.
Display Detailed Status	Display Detailed Status is an event used to trigger the display of the subsystem's detailed status GUI.
GUIs	GUIs are graphical user interfaces. These interfaces are used to communicate information from the subsystem to the user and to allow the user to control certain aspects of the execution of the subsystem.
Kiosk Equipment Status	Kiosk Equipment Status defines the status information for each of the Kiosk Field Units. This information includes the current state of the field unit, the number of files that need to be downloaded, the current information about the field unit paper level, disk space, and user accesses as well as the phone number and location of the field unit.
Update Detailed Status	Update Detailed Status is responsible for periodically updating the status information within the detailed status GUI. The current Kiosk Equipment Status is read and used to display the status within the GUI. The Detailed Status Update Rate is used to cause the periodic update of the GUI.
User Commands	User Commands are the commands selected by the user from the graphical user interfaces. These commands are generated through push buttons, radio buttons, text boxes, and other user interface components.

Table 10 - Detailed Status GUI Data Flow Descriptions

The following subsections provide the detailed design for the Detailed Status GUI. Each subsection contains a description of the routine, a structure chart of the routine and a table containing descriptions of the components defined in the structure chart.

4.2.1.2.1 Kiosk teleuse_Main

This is the main routine of the Kiosk Detailed Status GUI. This routine is supplied by the TeleUSE UIMS tool and is used as the entry point into the process. This routine is responsible for setting up any TeleUSE specific environment and then invoking the application main routine followed by the INITIALLY events in the associated D modules. The structure chart Kiosk teleuse_main is depicted in Figure 8. A description of the routines called by Kiosk teleuse_main is provided in Table 11.

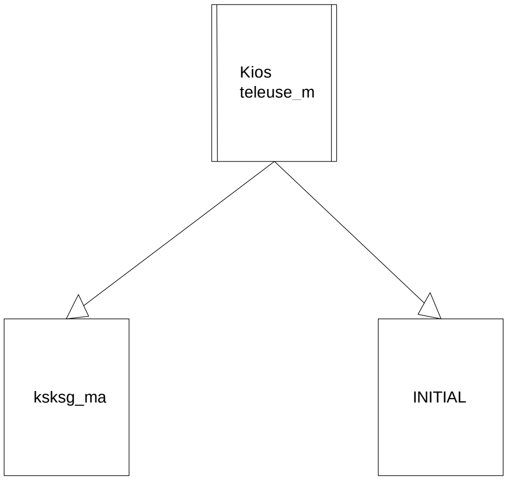

Figure 8 - Kiosk teleuse_main Structure Chart

Function	Description
INITIALLY	This D event is the first event invoked by the TeleUSE runtime environment. This event creates the top-level shell to contain the detailed status information, sets the update rate for the GUI, and then starts the update process by triggering the periodic_update event. Any errors during this event will cause tu_exit to be called to start a graceful shutdown of the process.
ksksg_main	This is the main routine of the Kiosk Detailed Status GUI. This routine is responsible for loading the configuration information, configuring the shared memory manager library, and attaching to the field equipment shared memory segments.

Table 11 - Routines called by Kiosk teleuse_main

4.2.1.2.2 kdsg_main

This is the main routine of the Kiosk Detailed Status GUI. This routine is responsible for loading the configuration information, configuring the shared memory manager library, and attaching to the field equipment shared memory segments. The structure chart for kdsg_main is depicted in Figure 9. A description of the routines called by kdsg_main is provided in Table 12.

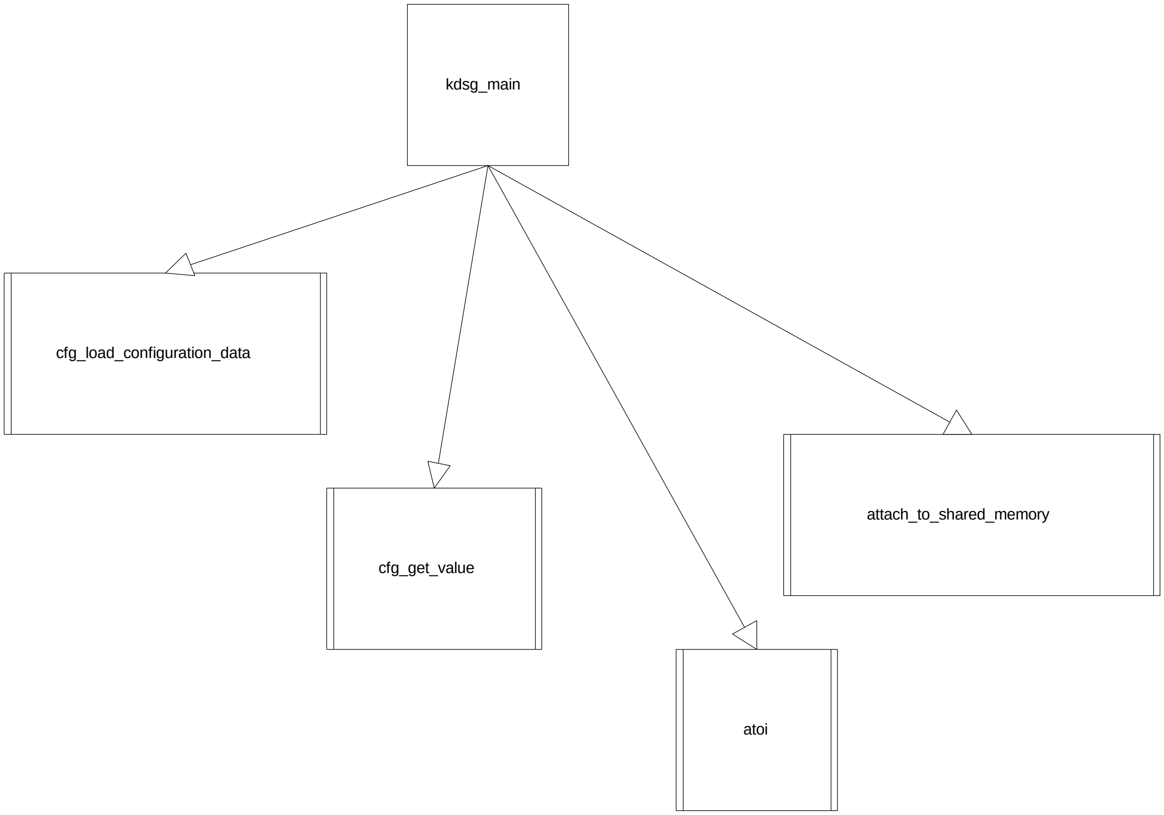

Figure 9 - kdsg_main Structure Chart

Function	Description
atoi	C Library Function to convert an ASCII string to an integer value.
attach_to_shared_memory	An MDI Shared Memory Common Code function used to attach to a specified shared memory segment. The segment id and size are used to attach to the segment.
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
cfg_load_configuration_data	MDI Configuration File routine used to read the configuration name-value pairs from the specified configuration file. These name-value pairs are loaded into memory so they can be accessed on demand by the calling program.

Table 12 - Routines called by kdsg_main

4.2.1.2.3 INITIALLY

This D event is the first event invoked by the TeleUSE runtime environment. This event creates the top-level shell to contain the detailed status information, sets the update rate for the GUI, and then starts the update process by triggering the periodic_update event. Any errors during this event will cause tu_exit to be called to start a graceful shutdown of the process. The structure chart for INITIALLY is depicted in Figure 10. A description of the routines called by INITIALLY is provided in Table 13.

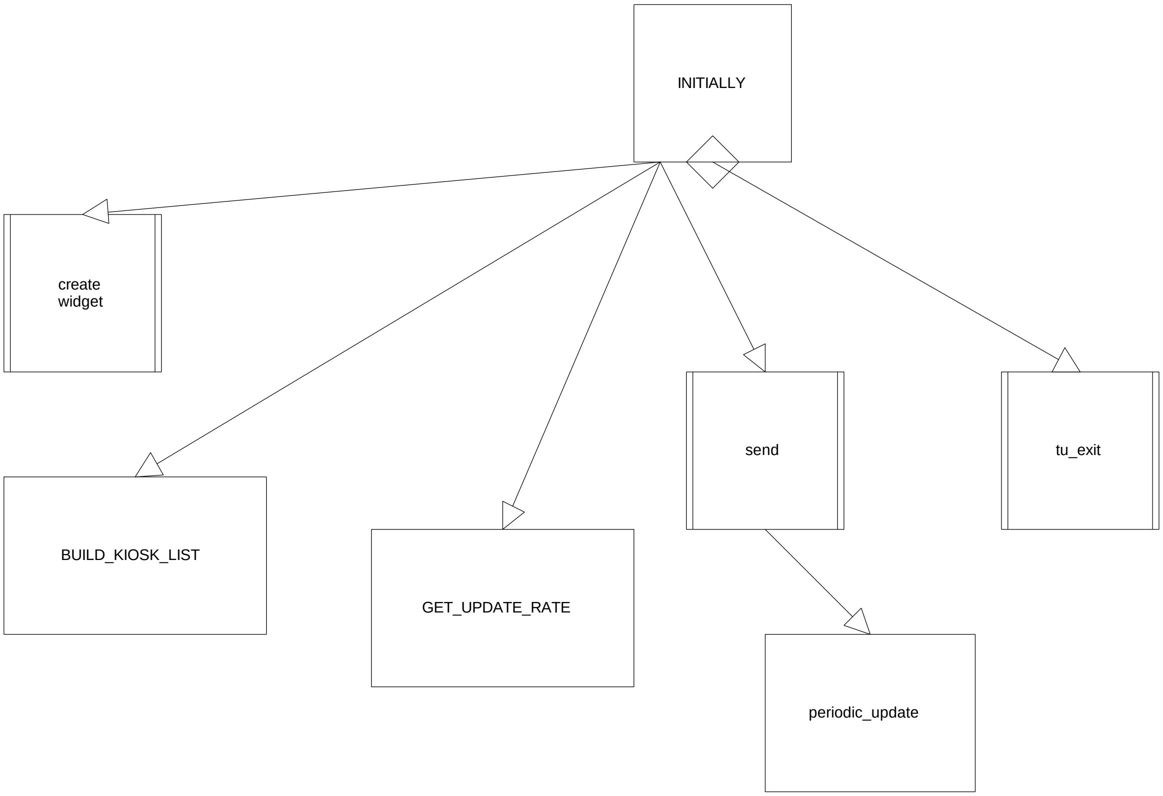

Figure 10 - INITIALLY Structure Chart

Function	**Description**
BUILD_KIOSK_LIST	BUILD_KIOSK_LIST is a bridge layer routine used to invoke the application layer routine which builds the kiosk status indicators.
create widget	Create widget is used to create a widget of a particular TeleUSE template allowing for the specification of a widget name and a parent for the widget.
GET_UPDATE_RATE	A bridge layer routine used to obtain the update rate value from the application layer.
periodic_update	A GUI layer event used to perform the steps necessary to update the details of the GUI on a periodic basis. If any errors occur an error message dialog is created which will cause the application to exit.
PERIODIC_UPDATE	The bridge layer routine that invokes the application layer routine responsible for handling the periodic update requests.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
tu_exit	A TeleUSE library routine used to exit the application.

Table 13 - Routines called by INITIALLY

4.2.1.2.4 BUILD_KIOSK_LIST

BUILD_KIOSK_LIST is a bridge layer routine used to invoke the application layer routine which builds the kiosk status indicators. The structure chart for BUILD_KIOSK_LIST is depicted in Figure 11. A description of the routines called by BUILD_KIOSK_LIST is provided in Table 14.

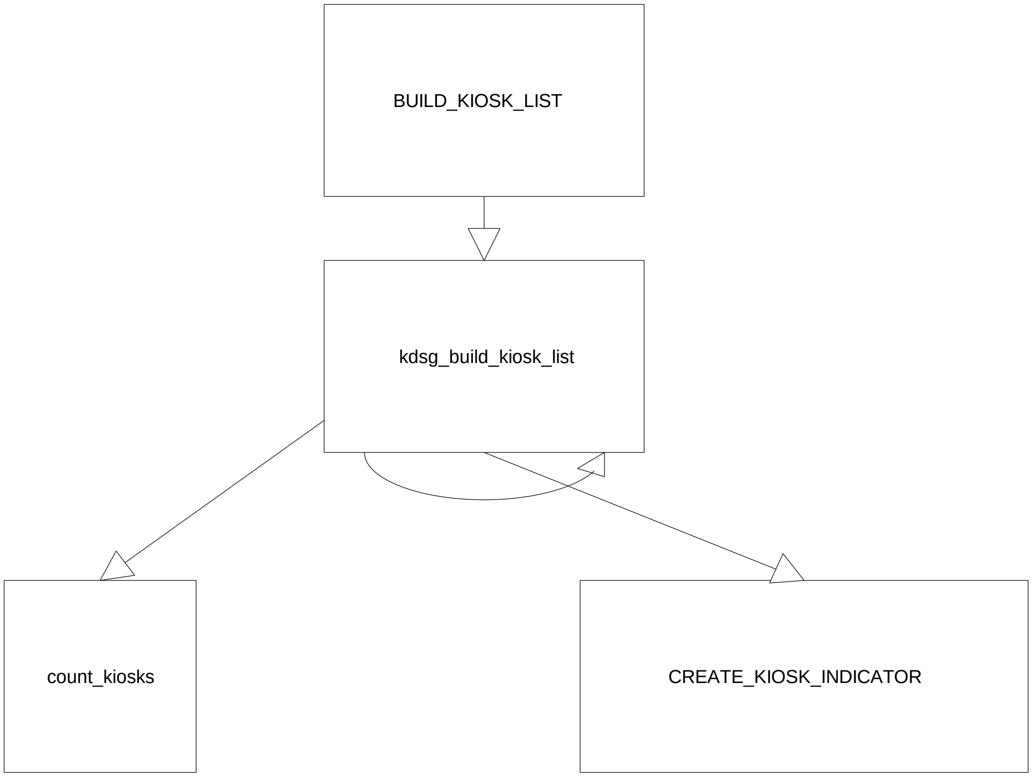

Figure 11 - BUILD_KIOSK_LIST Structure Chart

Function	Description
count_kiosks	count_kiosks read the kiosk field unit configuration file and counts the number of entries. There will be one line of configuration information for each kiosk field unit.
CREATE_KIOSK_INDICATOR	CREATE_KIOSK_INDICATOR is a bridge layer routine that creates and dispatches the GUI layer event responsible for creating the status indicator widget for a kiosk.
kdsg_build_kiosk_list	kdsg_build_kiosk_list loops through the kiosk shared memory and creates a status indicator for each kiosk defined in the shared memory segment.

Table 14 - Routines called by BUILD_KIOSK_LIST

4.2.1.2.5 count_kiosks

The count_kiosks routine reads the kiosk field unit configuration file and counts the number of entries. There will be one line of configuration information for each kiosk field unit. The structure chart for count_kiosks is depicted in Figure 12. A description of the routines called by count_kiosks is provided in Table 15.

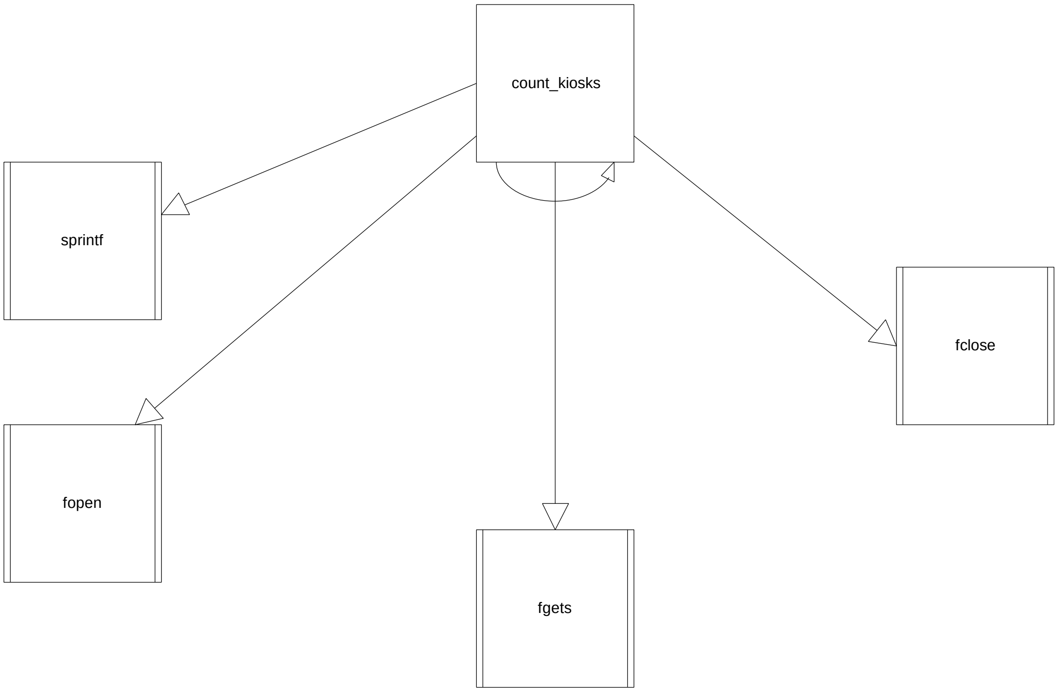

Figure 12 - count_kiosks Structure Chart

Function	Description
fclose	C Library Function used to close an open file.
fgets	C Library Function used to read a line of text from a file.
fopen	C Library Function that opens the specified file using the specified access mode.
sprintf	C Library Function that provides printf capabilities to a character string.

Table 15 - Routines called by count_kiosks

4.2.1.2.6 CREATE_KIOSK_INDICATOR

CREATE_KIOSK_INDICATOR is a bridge layer routine that creates and dispatches the GUI layer event responsible for creating the status indicator widget for a kiosk. The structure chart for CREATE_KIOSK_INDICATOR is depicted in Figure 13. A description of the routines called by CREATE_KIOSK_INDICATOR is provided in Table 16.

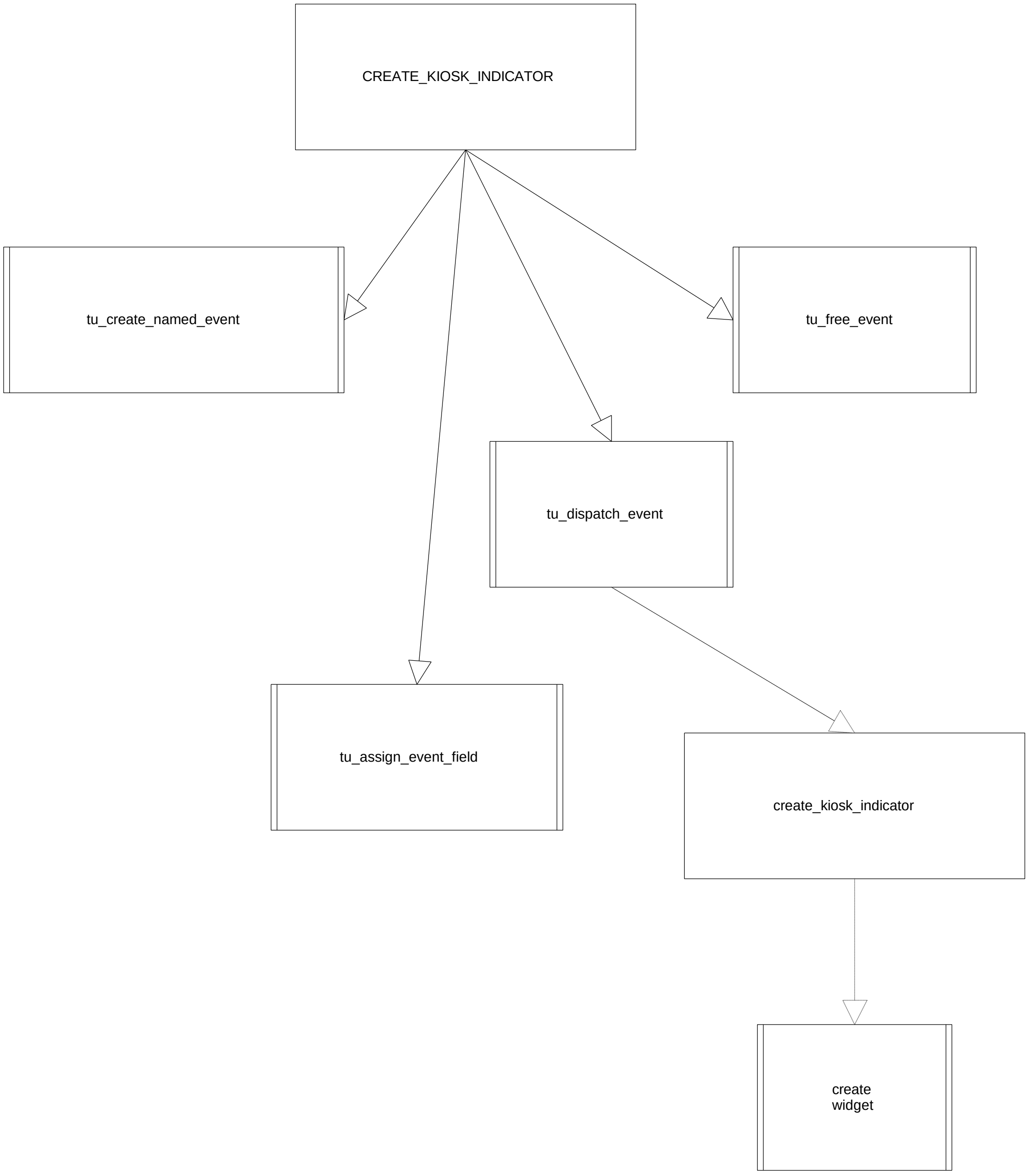

Figure 13 - CREATE_KIOSK_INDICATOR Structure Chart

Function	Description
create widget	create widget is used to create a widget of a particular TeleUSE template allowing for the specification of a widget name and a parent for the widget.
create_kiosk_indicator	create_kiosk_indicator is a GUI layer event that creates a Kiosk Indicator widget and sets the label string and user data to be the name of the kiosk and the index into shared memory for the kiosk.
tu_assign_event_field	TeleUSE Library Function to associate the contents of a C variable with the contents of an event attribute.
tu_create_named_event	TeleUSE Library Function used to create the data structure necessary to interface the C code with the D event code.
tu_dispatch_event	TeleUSE Library Function used to dispatch the created event. This causes the event to be executed.
tu_free_event	TeleUSE Library Function used to free up any memory that was allocated to the event data structure using tu_create_named_event.

Table 16 - Routines called by CREATE_KIOSK_INDICATOR

4.2.1.2.7 GET_UPDATE_RATE

GET_UPDATE_RATE is a routine that obtains the update rate value from the application layer and provides the value to the GUI layer. The structure chart for GET_UPDATE_RATE is depicted in Figure 14. A description of the routines called by GET_UPDATE_RATE is provided in Table 17.

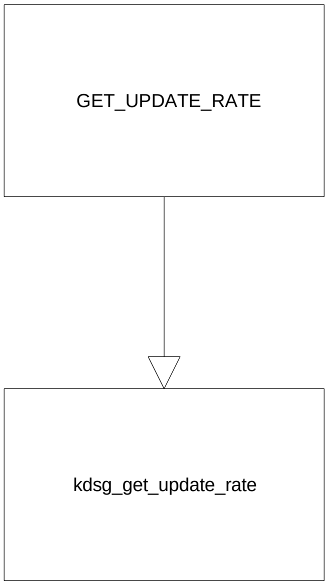

Figure 14 – GET_UPDATE_RATE Structure Chart

Function	Description
kdsg_get_update_rate	The application layer routine responsible for returning the configured update rate for the detailed status GUI.

Table 17 - Routines called by GET_UPDATE_RATE

4.2.1.2.8 periodic_update

periodic_update is a GUI layer event that performs the steps necessary to update the GUI on a periodic basis. If any errors occur an error message dialog is created which will cause the application to exit. The structure chart for periodic_update is depicted in Figure 15. A description of the routines called by periodic_update is provided in Table 18.

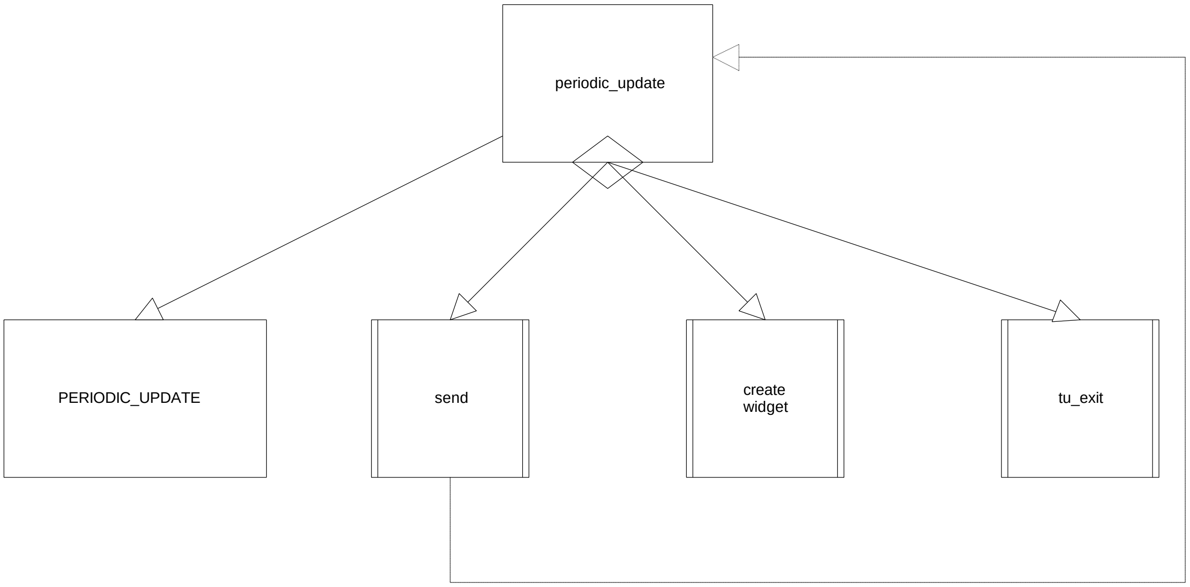

Figure 15 – periodic update Structure Chart

Function	Description
Create widget	create widget is used to create a widget of a particular TeleUSE template allowing for the specification of a widget name and a parent for the widget.
PERIODIC_UPDATE	The bridge layer routine that invokes the application layer routine responsible for handling the periodic update requests.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
tu_exit	A TeleUSE library routine used to exit the application.

Table 18 - Routines called by periodic update

4.2.1.2.9 UPDATE_DETAILS

UPDATE_DETAILS is a bridge layer routine invoked when the application layer wants to modify the kiosk details information on the detailed status GUI. The structure chart for UPDATE_DETAILS is depicted in Figure 16. A description of the routines called by UPDATE_DETAILS is provided in Table 19.

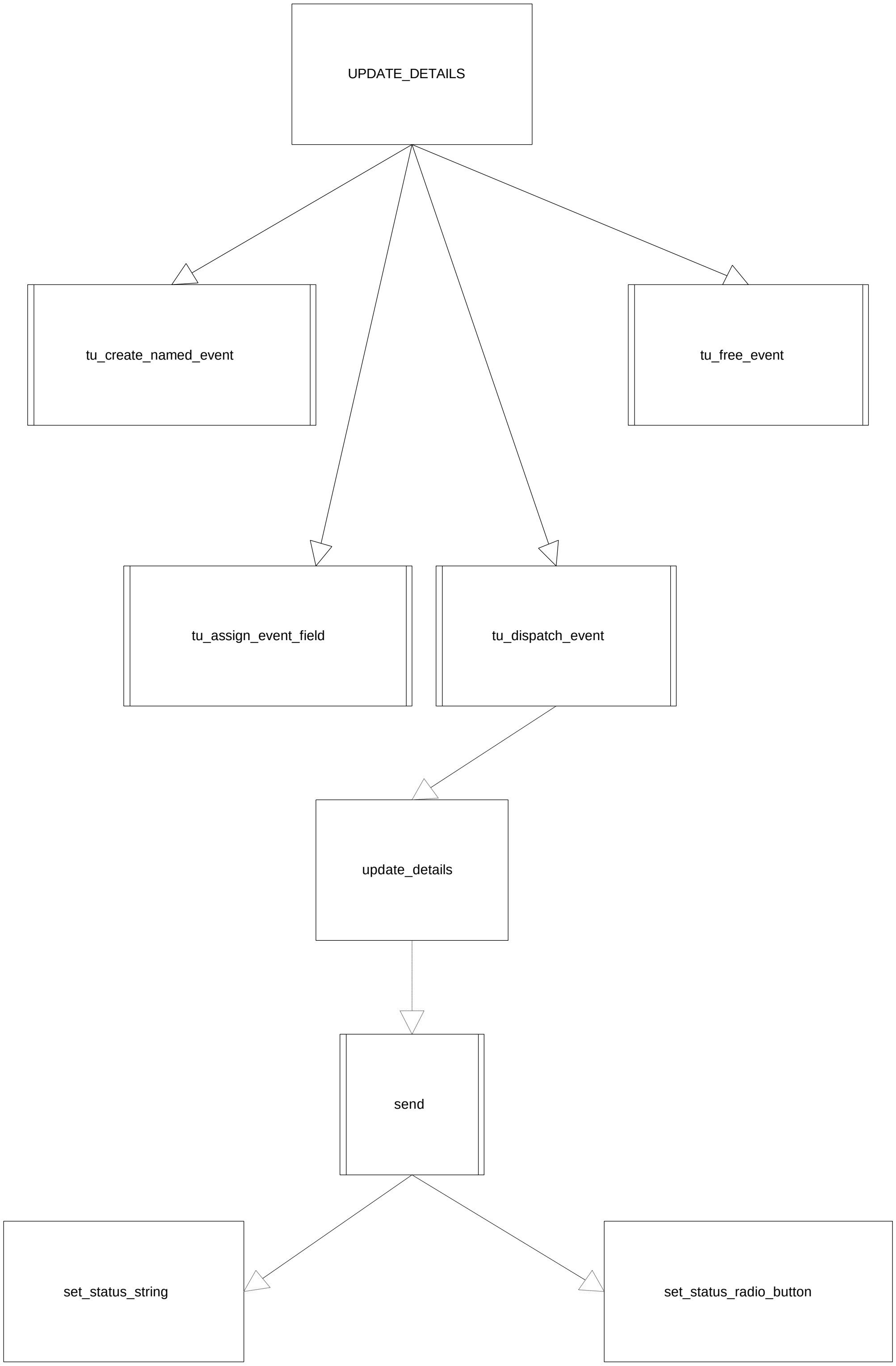

Figure 16 - UPDATE_DETAILS Structure Chart

Function	Description
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
set_status_radio_button	set_status_radio_button is a GUI layer event used to toggle the appropriate status radio button based on the specified status value. These values are used to indicate whether the kiosk is online, has a problem, is being pinged, is dialing, etc.
set_status_string	set_status_string is a GUI layer event used to update the label string of the specified widget using the specified status value. This event takes the integer value and converts it into a text string and associated color for the label.
tu_assign_event_field	TeleUSE Library Function used to associate the contents of a C variable with the contents of an event attribute.
tu_create_named_event	TeleUSE Library Function used to create the data structure necessary to interface the C code with the D event code.
tu_dispatch_event	TeleUSE Library Function used to dispatch the created event. This causes the event to be executed.
tu_free_event	TeleUSE Library Function used to free up any memory that was allocated to the event data structure using tu_create_named_event.
update_details	update_details is a GUI layer event that updates the detail information for the specified kiosk. This information includes the phone number, the address, the current status, last time it was contacted, etc.

Table 19 - Routines called by UPDATE DETAILS

4.2.1.2.10 UPDATE_INDICATOR

UPDATE_INDICATOR is a bridge routine invoked when the application layer wants to modify the status information on the detailed status GUI. The structure chart for UPDATE_INDICATOR is depicted in Figure 17. A description of the routines called by UPDATE_INDICATOR is provided in Table 20.

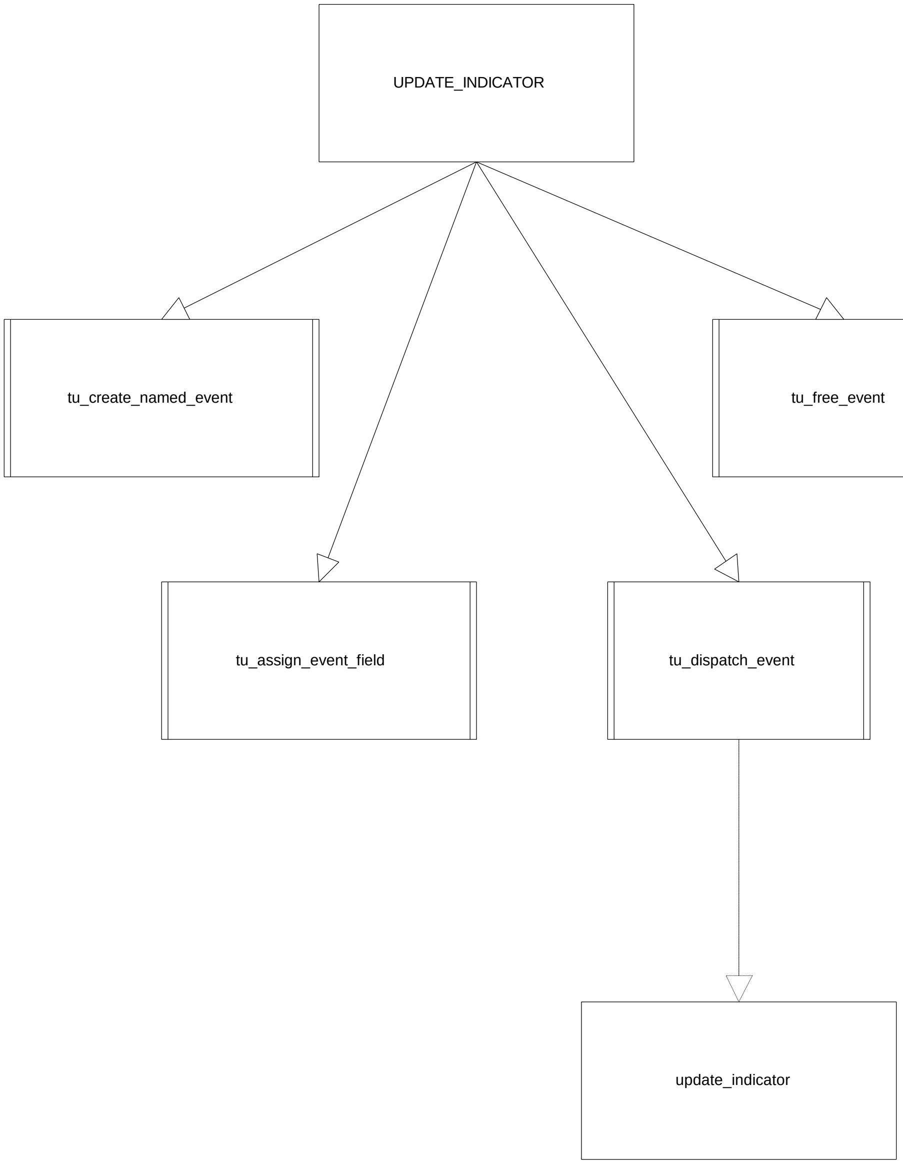

Figure 17 – UPDATE_INDICATOR Structure Chart

Function	Description
tu_assign_event_field	TeleUSE Library Function used to associate the contents of a C variable with the contents of an event attribute.
tu_create_named_event	TeleUSE Library Function used to create the data structure necessary to interface the C code with the D event code.
tu_dispatch_event	TeleUSE Library Function used to dispatch the created event. This causes the event to be executed.
tu_free_event	TeleUSE Library Function used to free up any memory that was allocated to the event data structure using tu_create_named_event.
update_indicator	This GUI layer event is invoked to update an indicator using the status value and the kiosk name. The name of the kiosk is used to create the indicator so the name is used to locate the indicator widget. The Kiosk State button associated with the indicator is modified based on the state value.

Table 20 - Routines called by UPDATE_INDICATOR

4.2.1.2.11 UPDATE_USAGE

UPDATE_USAGE is a bridge layer routine invoked when the application layer wants to modify the kiosk usage information on the detailed status GUI. The structure chart for UPDATE_USAGE is depicted in Figure 18. A description of the routines called by UPDATE_USAGE is provided in Table 21.

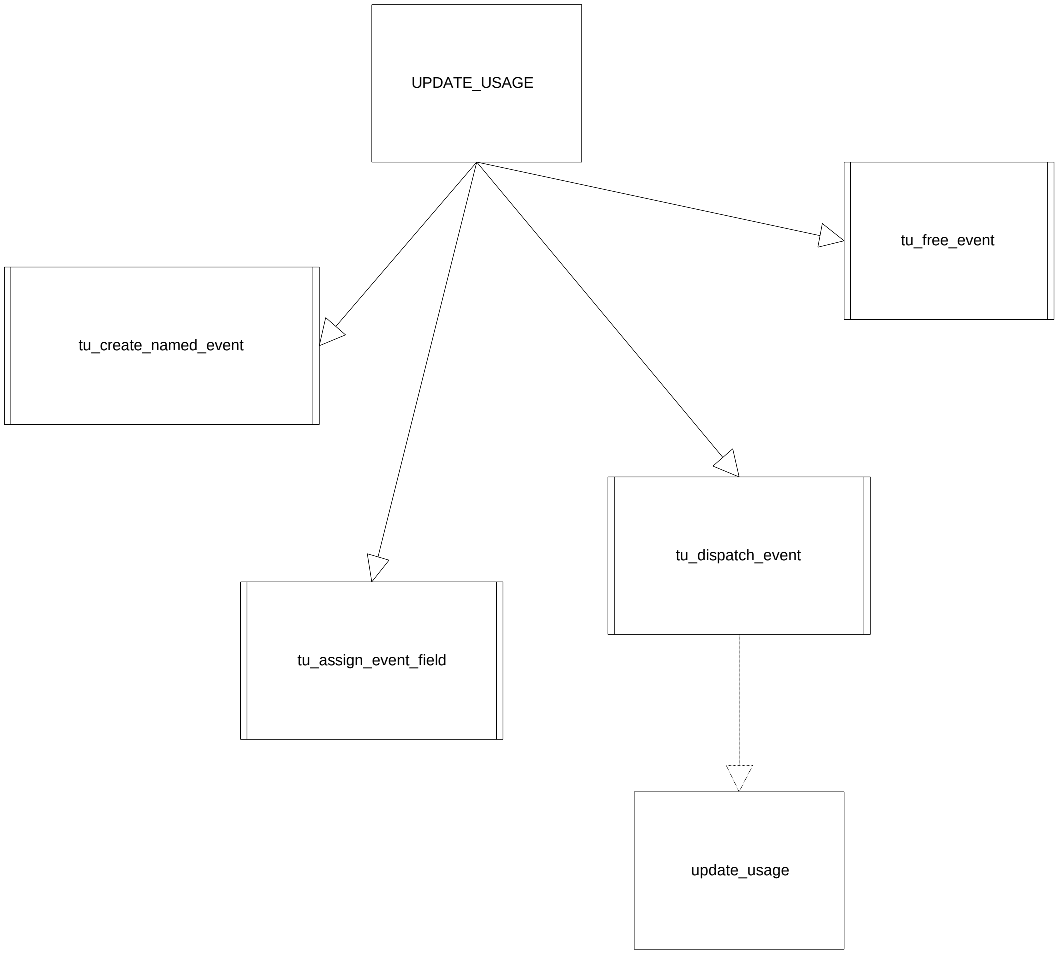

Figure 18 - UPDATE_USAGE Structure Chart

Function	Description
tu_assign_event_field	TeleUSE Library Function to associate the contents of a C variable with the contents of an event attribute.
tu_create_named_event	TeleUSE Library Function used to create the data structure necessary to interface the C code with the D event code.
tu_dispatch_event	TeleUSE Library Function used to dispatch the created event. This causes the event to be executed.
tu_free_event	TeleUSE Library Function used to free up any memory that was allocated to the event data structure using tu_create_named_event.
update_usage	update_usage is a GUI layer event that updates the usage section for the specified kiosk. This section includes the number of restarts, how many times each of the kiosk areas have been accessed, and page and disk usage information.

Table 21 - Routines called by UPDATE_USAGE

4.2.1.2.12 kiosk_download

kiosk_download is the GUI layer event attached to the download radio button for a kiosk field unit. The button is selected when the user desires to force a download to occur for a specific kiosk. The structure chart for kiosk_download is depicted in Figure 19. A description of the routines called by kiosk_download is provided in Table 22.

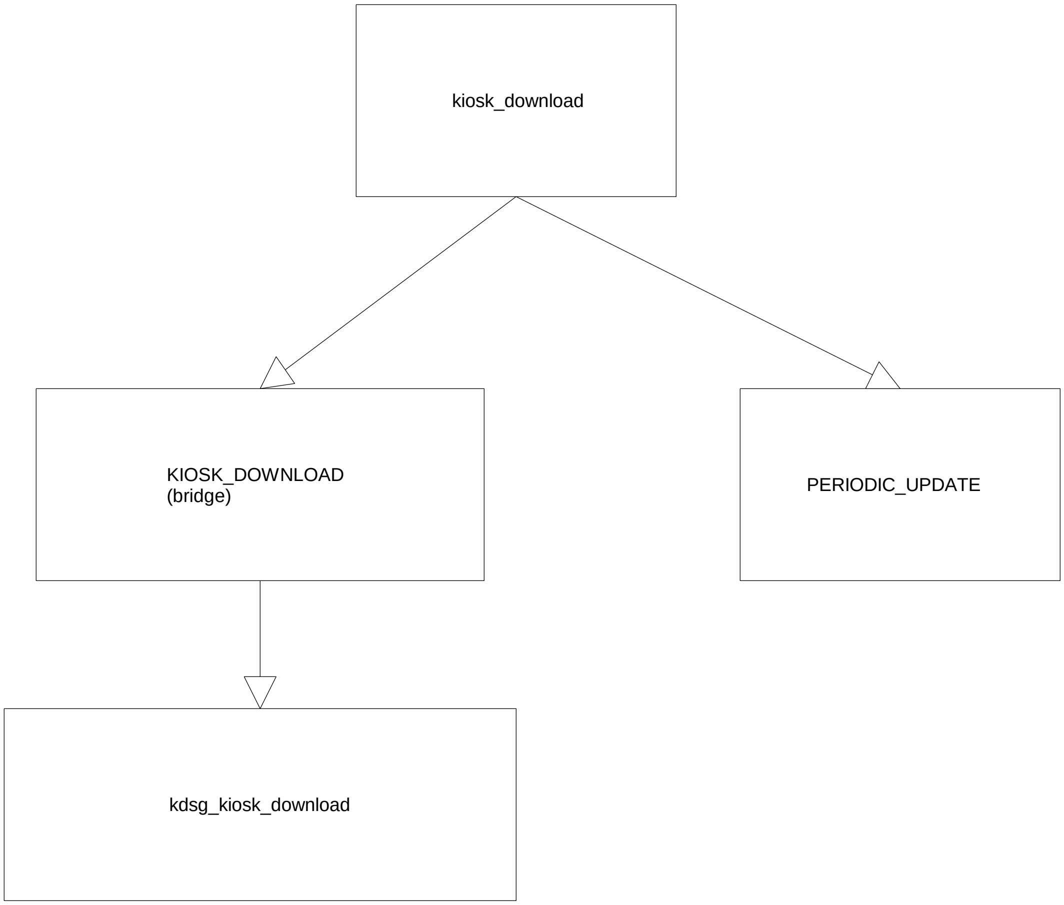

Figure 19 - kiosk_download Structure Chart

Function	Description
kdsg_kiosk_download	kdsg_kiosk_download is the application layer routine responsible for updating the shared memory segment for the specified kiosk to indicate the kiosk should be downloaded.
KIOSK_DOWNLOAD (bridge)	The bridge layer routine that invokes the application layer routine responsible for handling the kiosk download.
PERIODIC_UPDATE	The bridge layer routine that invokes the application layer routine responsible for handling the periodic update requests.

Table 22 - Routines called by kiosk_download

4.2.1.2.13 set_kiosk_in_service

set_kiosk_in_service is the GUI layer event attached to the in service radio button for a kiosk field unit. The button is selected when the user desires to place the associated kiosk field unit in service. The structure chart for set_kiosk_in_service is depicted in Figure 20. A description of the routines called by set_kiosk_in_service is provided in Table 23.

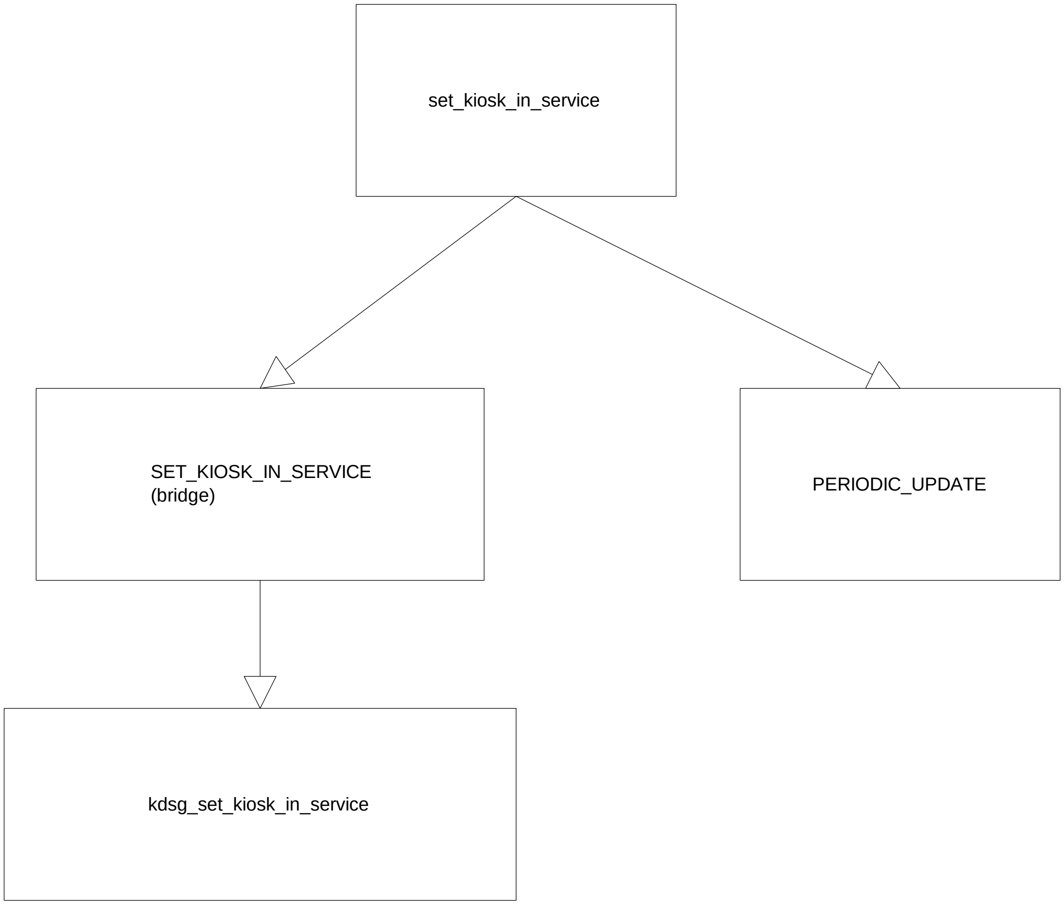

Figure 20 - set_kiosk_in_service Structure Chart

Function	Description
kdsg_set_kiosk_in_service	kdsg_set_kiosk_in_service is the application layer routine responsible for updating the shared memory segment for the specified kiosk to indicate the kiosk is active.
PERIODIC_UPDATE	The bridge layer routine that invokes the application layer routine responsible for handling the periodic update requests.
SET_KIOSK_IN_SERVICE (bridge)	The bridge layer routine that invokes the application layer routine responsible for handling the function of setting the kiosk active.

Table 23 - Routines called by set_kiosk_in_service

4.2.1.2.14 PERIODIC UPDATE

The bridge layer routine that invokes the application layer routine responsible for handling the periodic update requests. The structure chart for PERIODIC UPDATE is depicted in Figure 21. A description of the routines called by PERIODIC UPDATE is provided in Table 24.

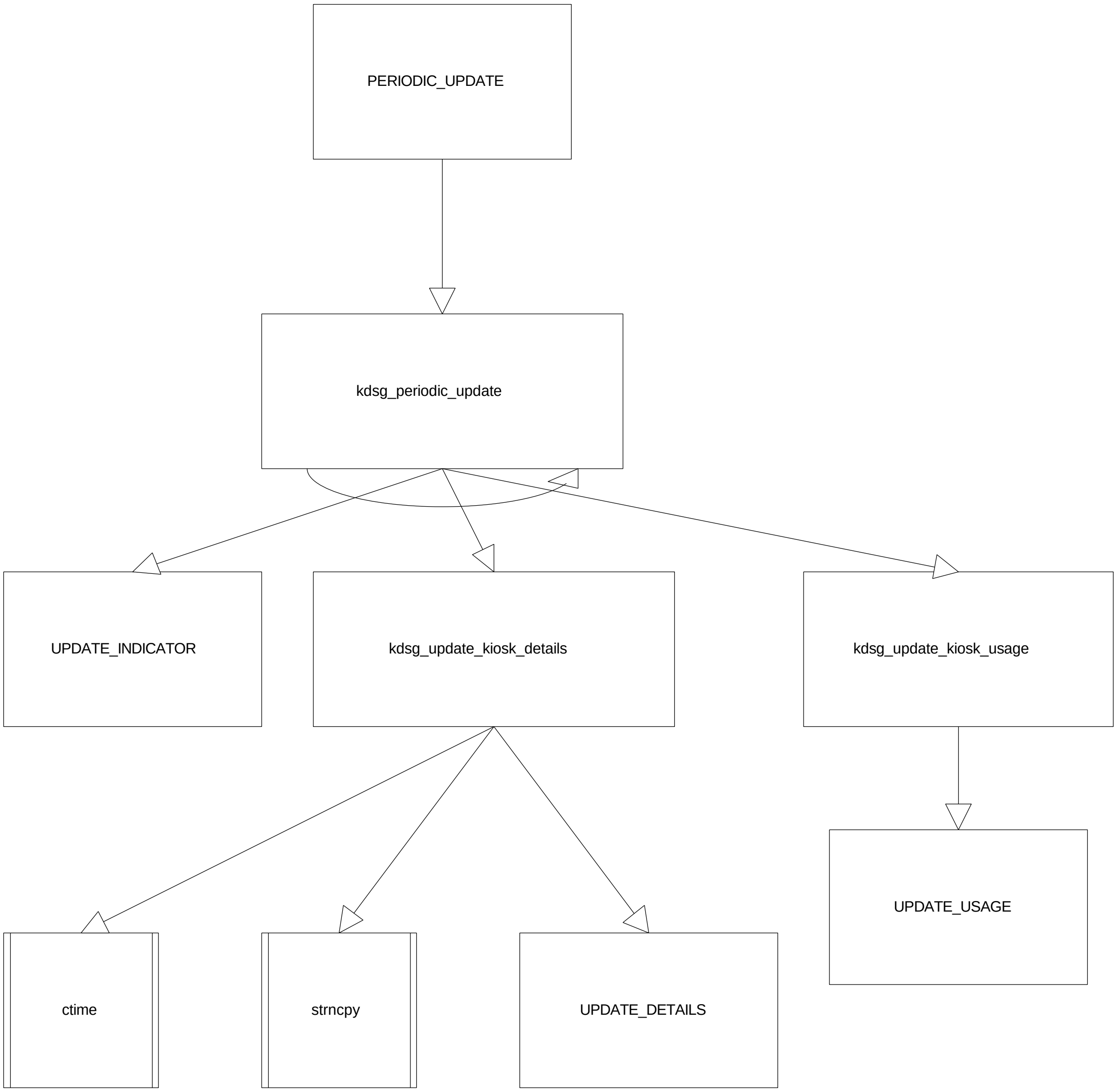

Figure 21 – PERIODIC UPDATE Structure Chart

Function	Description
Ctime	C Library Function used to return the specified time_t value as a character string.
kdsg_periodic_update	The application layer routine responsible for handling the periodic update requests. This routine attaches to the kiosk field units shared memory segments if not attached, reads the data from these segments, and, using the bridge layer, causes the contents of the GUI to be updated based on the contents of the shared memory segments.
kdsg_update_kiosk_details	kdsg_update_kiosk_details is an application layer routine invoked when the details of a single kiosk are needed. This routine is responsible for updating the graphical user interface components using the UPDATE_DETAILS bridge function.
kdsg_update_kiosk_usage	kdsg_update_kiosk_usage is an application layer routine invoked when the usage stats of a single kiosk are needed. This routine is responsible for updating the graphical user interface components using the UPDATE_USAGE bridge function.
Strncpy	C Library Function used to copy a specified number of characters from a source string to a destination string.
UPDATE_DETAILS	UPDATE_DETAILS is a bridge layer routine invoked when the application layer wants to modify the kiosk details information on the detailed status GUI.
UPDATE_INDICATOR	UPDATE_INDICATOR is a bridge routine invoked when the application layer wants to modify the status information on the detailed status GUI.
UPDATE_USAGE	UPDATE_USAGE is a bridge layer routine invoked when the application layer wants to modify the kiosk usage information on the detailed status GUI.

Table 24 - Routines called by PERIODIC UPDATE

4.2.1.2.15 set_kiosk_out_of_service

set_kiosk_out_of_service is the GUI layer event attached to the out of service radio button for a kiosk field unit. The button is selected when the user desires to place the associated kiosk field unit out of service. The structure chart for set_kiosk_out_of_service is depicted in Figure 22. A description of the routines called by set_kiosk_out_of_service is provided in Table 25.

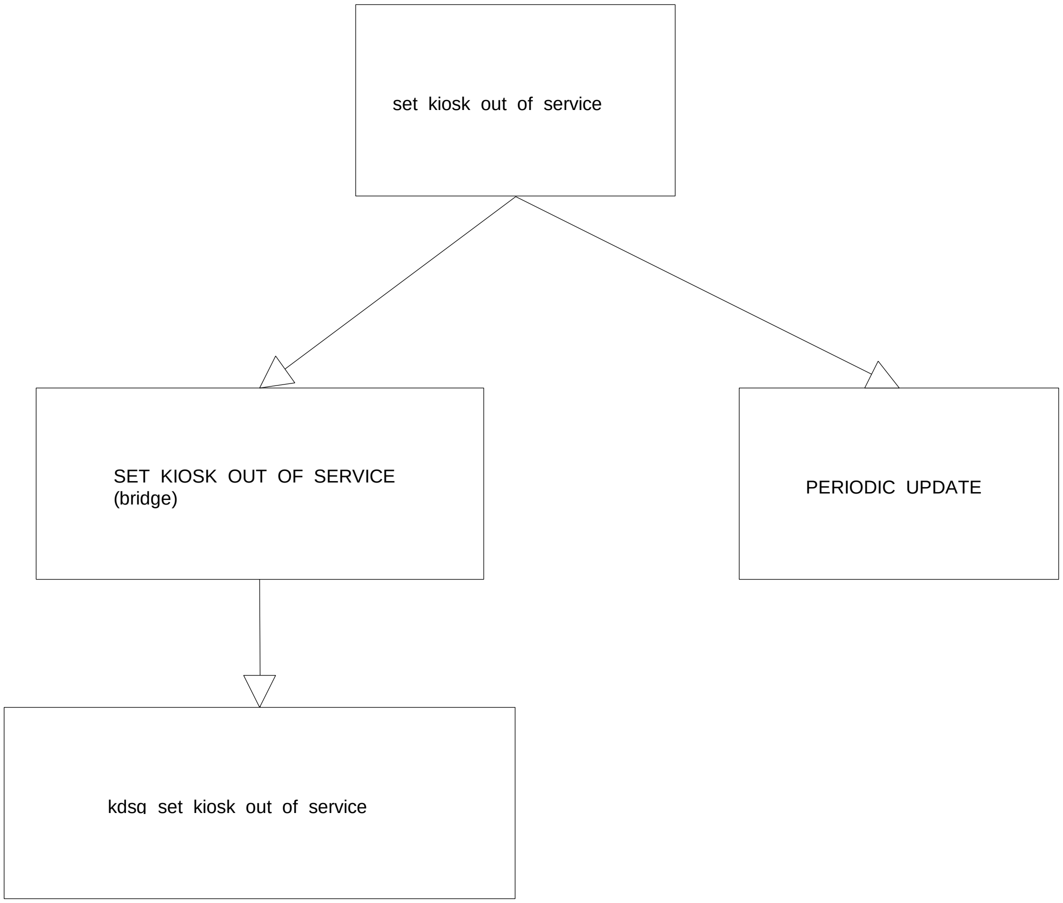
set kiosk out of service
SET KIOSK OUT OF SERVICE (bridge)
PERIODIC UPDATE
kdsg set kiosk out of service

Figure 22 - set_kiosk_out_of_service Structure Chart

Function	Description
kdsg_set_kiosk_out_of_service	kdsg_set_kiosk_out_of_service is the application layer routine responsible for updating the shared memory segment for the specified kiosk to indicate the kiosk is inactive.
PERIODIC_UPDATE	The bridge layer routine that invokes the application layer routine responsible for handling the periodic update requests.
SET_KIOSK_OUT_OF_SERVICE (bridge)	The bridge layer routine that invokes the application layer routine responsible for handling the function of setting the kiosk inactive.

Table 25 - Routines called by set_kiosk_out_of_service

4.2.1.2.16 kiosk_ping

kiosk_ping is the GUI layer event attached to the ping radio button for a kiosk field unit. The button is selected when the user desires to force an update of the status information to occur for a specific kiosk. The structure chart for kiosk_ping is depicted in Figure 23. A description of the routines called by kiosk_ping is provided in Table 26.

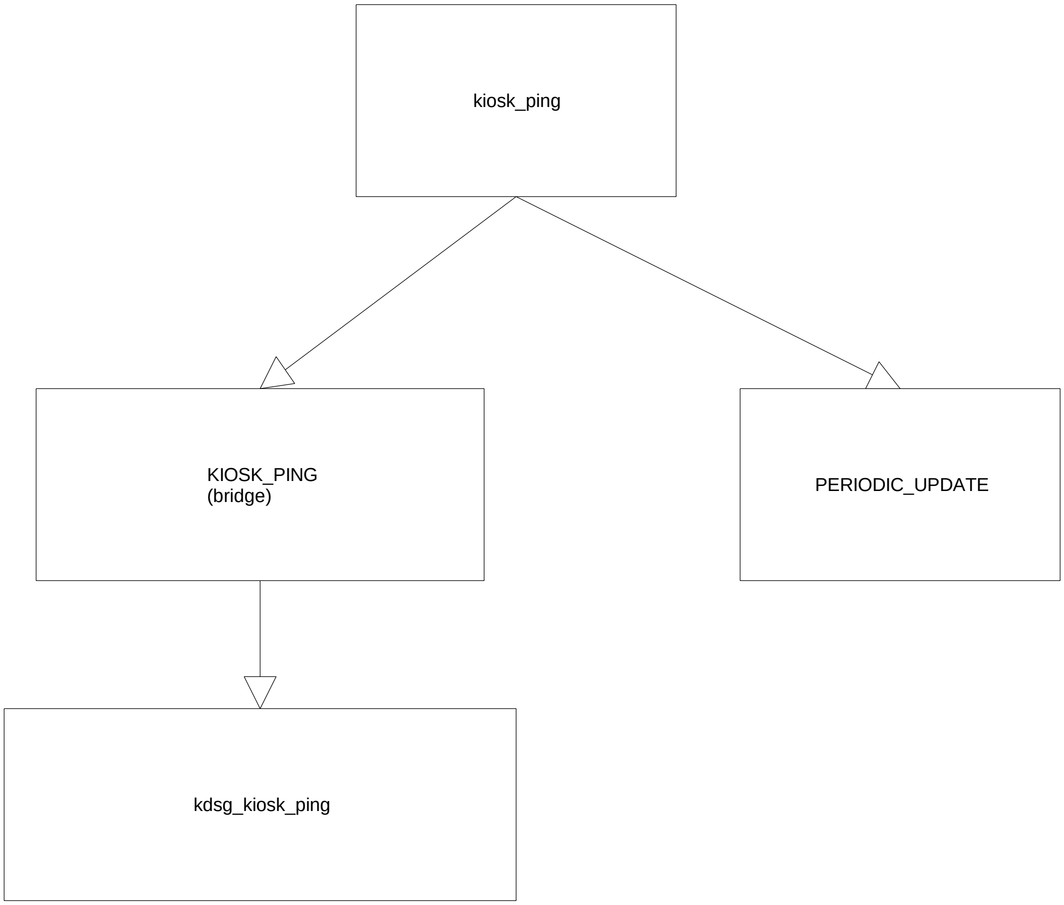

Figure 23 - kiosk_ping Structure Chart

Function	Description
kdsg_kiosk_ping	kdsg_kiosk_ping is the application layer routine responsible for updating the shared memory segment for the specified kiosk to indicate the kiosk should be pinged.
KIOSK_PING (bridge)	The bridge layer routine that invokes the application layer routine responsible for handling the function of setting the kiosk state to ping.
PERIODIC_UPDATE	The bridge layer routine that invokes the application layer routine responsible for handling the periodic update requests.

Table 26 - Routines called by kiosk_ping

4.2.1.2.17 UPDATE_STATUS

This is the bridge layer routine that receives the current status information for a particular Kiosk and then generates the event to update the status information within the detailed status GUI. The structure chart for UPDATE_STATUS is depicted in Figure 24. A description of the routines called by UPDATE_STATUS is provided in Table 27.

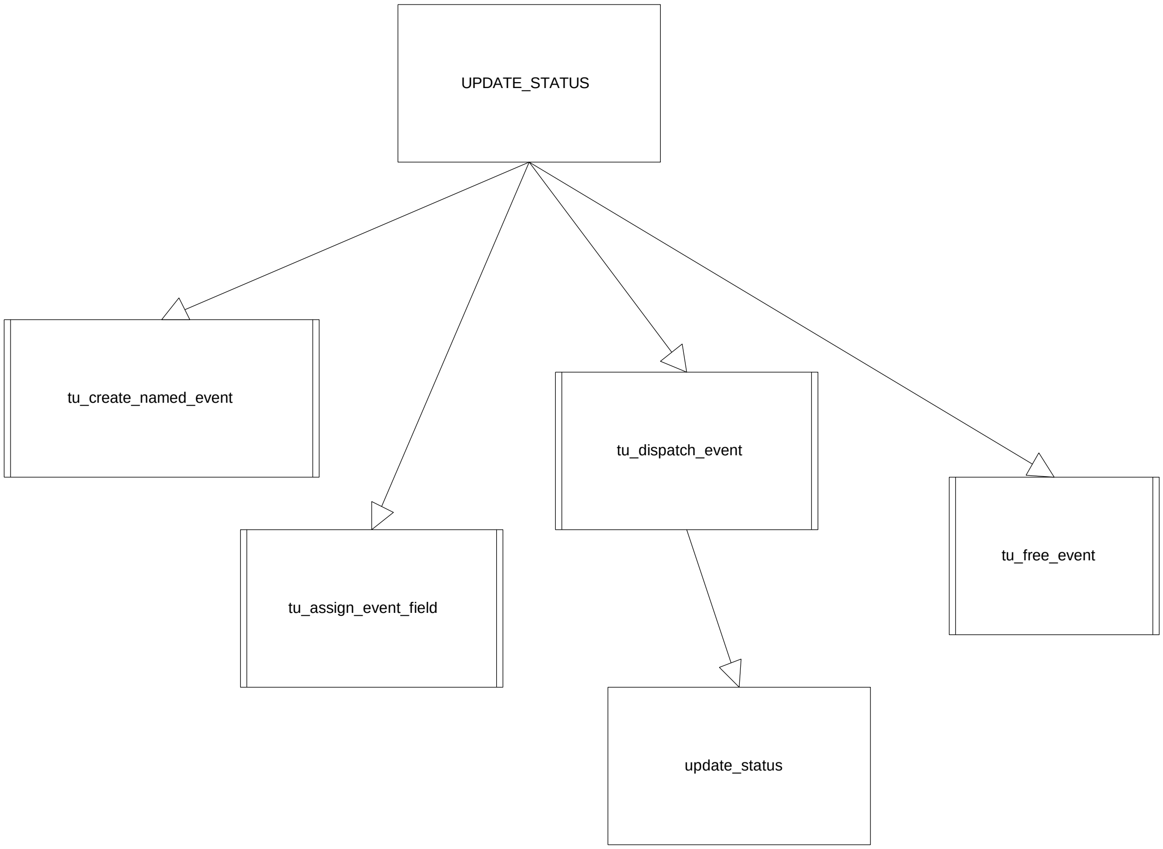

Figure 24 - UPDATE_STATUS Structure Chart

Function	Description
tu_assign_event_field	TeleUSE Library Function to associate the contents of a C variable with the contents of an event attribute.
tu_create_named_event	TeleUSE Library Function used to create the data structure necessary to interface the C code with the D event code.
tu_dispatch_event	TeleUSE Library Function used to dispatch the created event. This causes the event to be executed.
tu_free_event	TeleUSE Library Function used to free up any memory that was allocated to the event data structure using tu_create_named_event.
Update_status	The D event that receives the status information and updates the appropriate GUI components.

Table 27 - Routines called by UPDATE_STATUS

4.2.1.2.18 update_status

The update_status routine receives the status information and updates the appropriate GUI components. The structure chart for update_status is depicted in Figure 25. A description of the routines called by update_status is provided in Table 28.

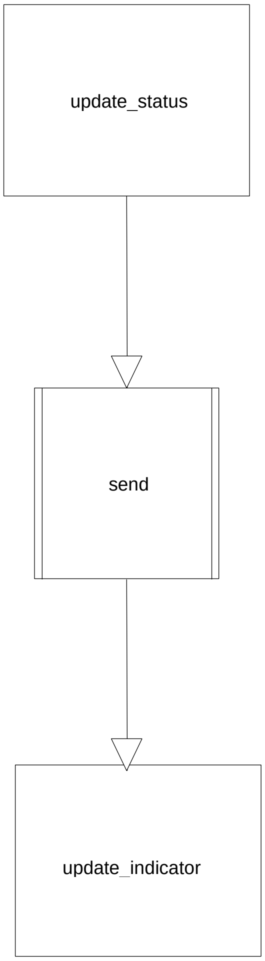

Figure 25 - update_status Structure Chart

Function	Description
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
update_indicator	This GUI layer event is invoked to update an indicator using the status value and the kiosk name. The name of the kiosk is used to create the indicator so the name is used to locate the indicator widget. The Kiosk State button associated with the indicator is modified based on the state value.

Table 28 - Routines called by update_status

4.2.1.3 System Maintenance GUI

The Kiosk System Maintenance GUI provides the capability to create, modify, and delete airline, rental car, airport parking, screensaver, and kiosk configuration files. Figure 26 depicts the data flows for the Detailed Status GUI. A description of the data flows is provided in Table 29.

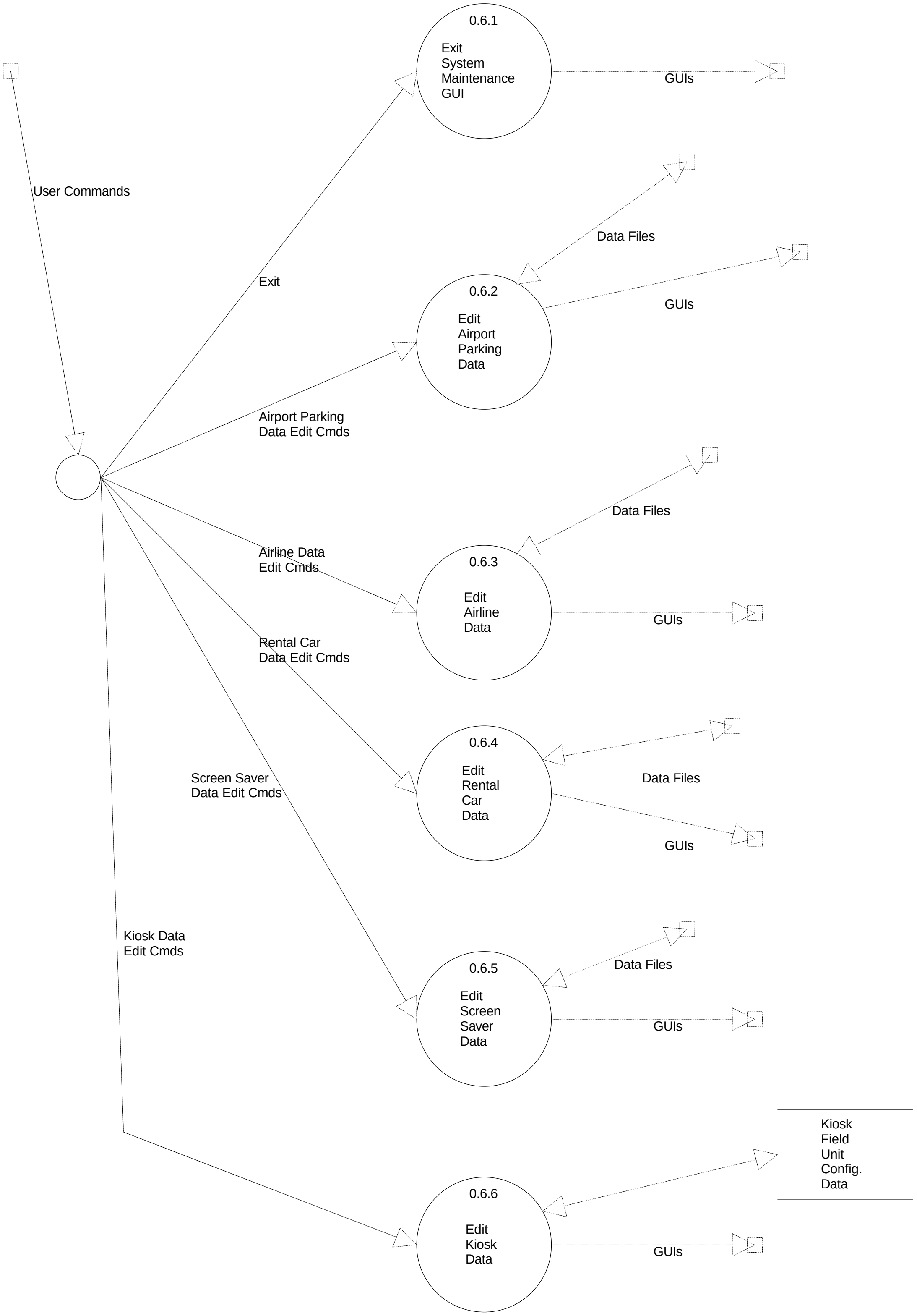

Figure 26 – Kiosk System Maintenance GUI Data Flow

Function	Description
Airline Data Edit Cmds	Airline Data Edit Commands are the Add, Delete, and Modify commands issued by the user when editing the airline data file.
Airport Parking Data Edit Cmds	Airport Parking Data Edit Commands are the Add, Delete, and Modify commands issued by the user when editing the airport parking data file.
Data Files	Data Files are files that are stored at the Data Server and include files such as weather files, screen saver files, and airport files.
Edit Airline Data	Edit Airline Data allows the user to add, delete, and modify entries in the airline data file. This file is maintained in the Data Server and is brought from the Data Server to the Kiosk Master Computer when edits are desired.
Edit Airport Parking Data	Edit Airport Parking Data allows the user to add, delete, and modify entries in the airport parking data file. This file is maintained in the Data Server and is brought from the Data Server to the Kiosk Master Computer when edits are desired.
Edit Kiosk Data	Edit Kiosk Data allows the user to add, delete, and modify entries in the Kiosk data file. This file is maintained at the Kiosk Master Computer and is used during startup of the Kiosk Subsystem.
Edit Rental Car Data	Edit Rental Car Data allows the user to add, delete, and modify entries in the rental car data file. This file is maintained in the Data Server and is brought from the Data Server to the Kiosk Master Computer when edits are desired.
Edit Screen Saver Data	Edit Screen Saver Data allows the user to add, delete, and modify entries in the screen saver data file. This file is maintained in the Data Server and is brought from the Data Server to the Kiosk Master Computer when edits are desired.
Exit	Exit is the command used to exit an application.
Exit System Maintenance GUI	Exit System Maintenance GUI is responsible for the systematic shutdown of the system maintenance GUI components. This includes allowing the user to save any unsaved files and then removing any temporary files that may have been created.
GUIs	GUIs are graphical user interfaces. These interfaces are used to communicate information from the subsystem to the user and to allow the user to control certain aspects of the execution of the subsystem.
Kiosk Data Edit Cmds	Kiosk Data Edit Commands are the Add, Delete, and Modify commands issued by the user when editing the kiosk field unit data file.
Kiosk Field Unit Config. Data	Kiosk Field Unit Configuration Data contains information for each field unit. This includes the name, phone number, and location of the field unit.
Rental Car Data Edit Cmds	Rental Car Data Edit Commands are the Add, Delete, and Modify commands issued by the user when editing the rental car data file.
Screen Saver Data Edit Cmds	Screen Saver Data Edit Commands are the Add, Delete, and Modify commands issued by the user when editing the screen saver data file.
User Commands	User Commands are the commands selected by the user from the graphical user interfaces. These commands are generated through push buttons, radio buttons, text boxes, and other user interface components.

Table 29 – Kiosk System Maintenance GUI Data Flows

4.2.1.3.1 ksmg (teleuse_main)

This is the main routine of the Kiosk System Maintenance GUI. This routine is supplied by the TeleUSE UIMS tool and is used as the entry point into the process. This routine is responsible for setting up any TeleUSE specific environment and then invoking the application main routine followed by the INITIALLY events in the associated D modules. The structure chart for ksmg (teleuse_main) is depicted in Figure 27. A description of the routines called by ksmg (teleuse_main) is provided in Table 30.

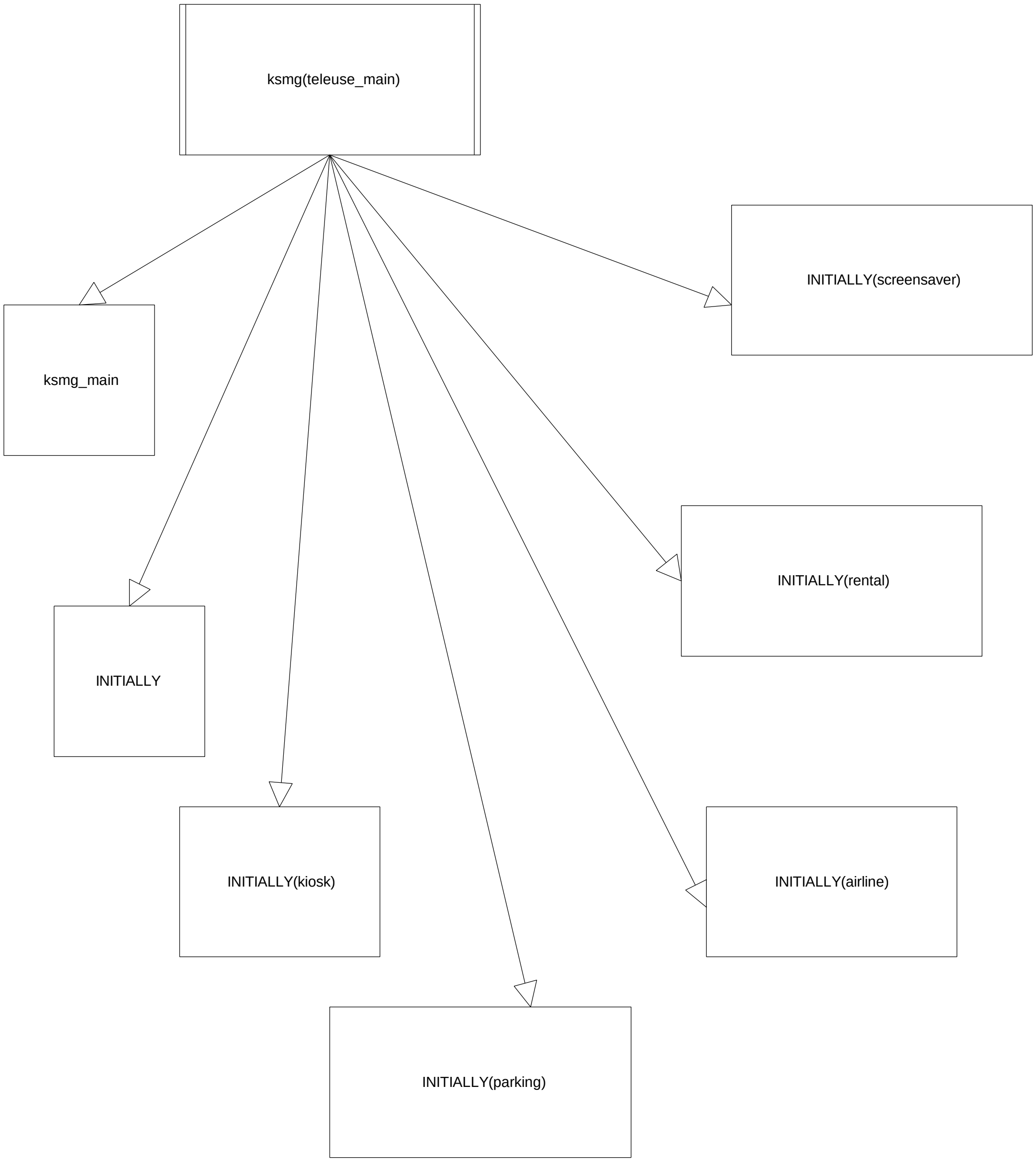

Figure 27 - ksmg (teleuse_main) Structure Chart

Function	Description
INITIALLY	This D event is the first event invoked by the TeleUSE runtime environment. This event creates the top-level shell to contain the table forms and then sends the event to set the sensitivity of the add menu item so no additions are allowed.
INITIALLY(airline)	INITIALLY(airline) is the INITIALLY event for the airline configuration D module. This event is responsible for initializing the D module flags and creating the header string to be used for the configuration item list.
INITIALLY(kiosk)	INITIALLY(kiosk) is the INITIALLY event for the kiosk configuration D module. This event is responsible for initializing the D module flags and creating the header string to be used for the configuration item list.
INITIALLY(parking)	INITIALLY(parking) is the INITIALLY event for the airport parking lot configuration D module. This event is responsible for initializing the D module flags and creating the header string to be used for the configuration item list.
INITIALLY(rental)	INITIALLY(rental) is the INITIALLY event for the rental car agency configuration D module. This event is responsible for initializing the D module flags and creating the header string to be used for the configuration item list.
INITIALLY(screensaver)	INITIALLY(screensaver) is the INITIALLY event for the screen saver configuration D module. This event is responsible for initializing the D module flags and creating the header string to be used for the configuration item list.
ksmg_main	This is the main routine of the KIOSK Detailed Status GUI. This routine is responsible for loading the configuration information, configuring the shared memory manager library, and attaching to the field equipment shared memory segments.

Table 30 - Routines called by ksmg (teleuse_main)

4.2.1.3.2 ksmg_main

This is the main routine of the KIOSK Detailed Status GUI. This routine is responsible for loading the configuration information, configuring the shared memory manager library, and attaching to the field equipment shared memory segments. The structure chart for ksmg_main is depicted in Figure 28. A description of the routines called by ksmg_main is provided in Table 31.

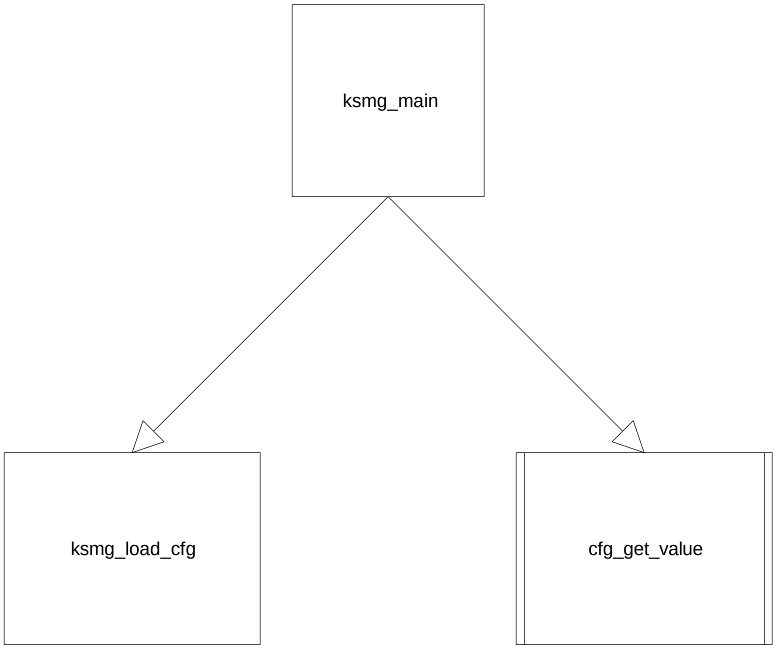

Figure 28 – ksmg_main Structure Chart

Function	Description
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
ksmg_load_cfg	ksmg_load_cfg loads the configuration times for the system maintenance GUI and verifies that all required configuration items are present.

Table 31 - Routines called by ksmg_main

4.2.1.3.3 kmsg_load_cfg

ksmg_load_cfg loads the configuration times for the system maintenance GUI and verifies that all required configuration items are present. The structure chart for kmsg_load_cfg is depicted in Figure 29. A description of the routines called by kmsg_load_cfg is provided in Table 32.

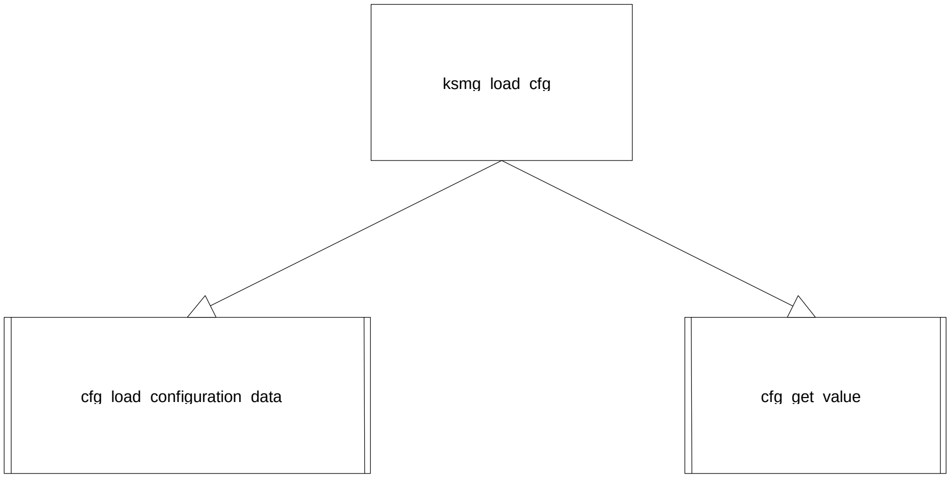

Figure 29 - kmsg_load_cfg Structure Chart

Function	Description
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
cfg_load_configuration_data	MDI Configuration File routine used to read the configuration name-value pairs from the specified configuration file. These name-value pairs are loaded into memory so they can be accessed on demand by the calling program.

Table 32 - Routines called by kmsg_load_cfg

4.2.1.3.4 INITIALLY

This D event is the first event invoked by the TeleUSE runtime environment. This event creates the top-level shell to contain the table forms and then sends the event to set the sensitivity of the add menu item so no additions are allowed. The structure chart for INITIALLY is depicted in Figure 30. A description of the routines called by INITIALLY is provided in Table 33.

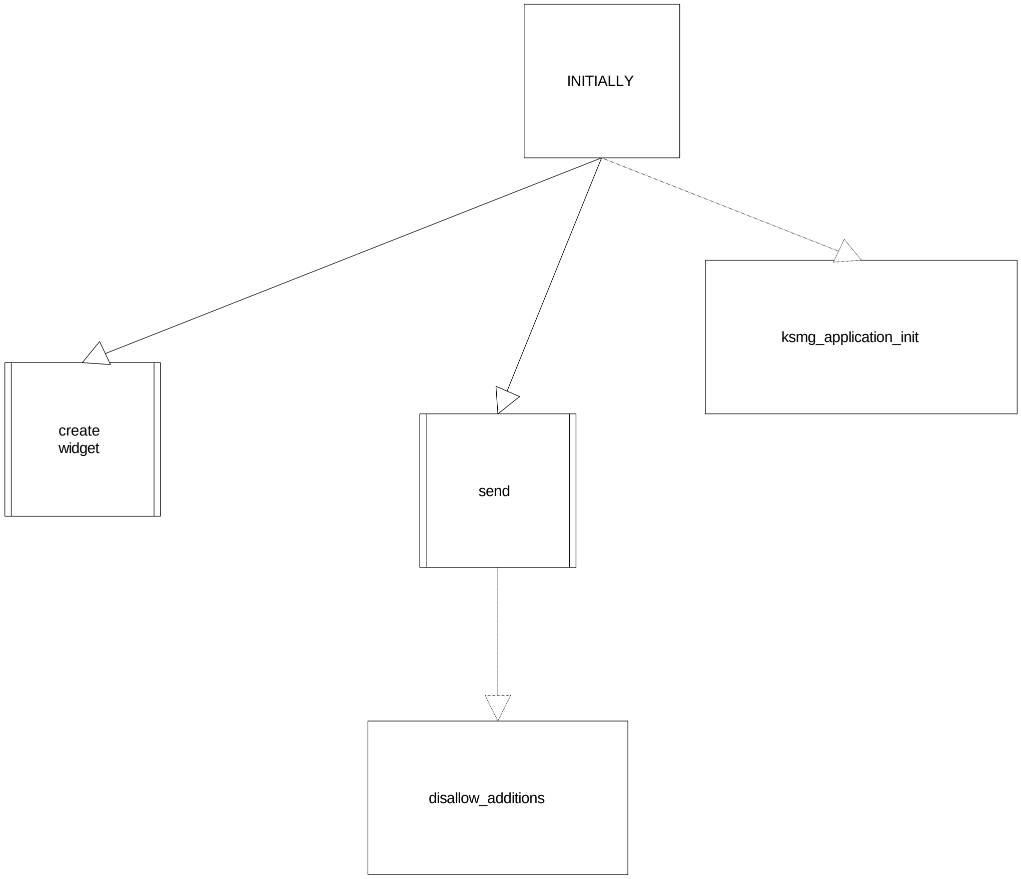

Figure 30 - INITIALLY Structure Chart

Function	Description
create widget	create widget is used to create a widget of a particular TeleUSE template allowing for the specification of a widget name and a parent for the widget.
disallow_additions	disallow_additions is the GUI layer routine that sets the sensitivity of the add menu item so that no additions can be made until a table is selected.
ksmg_application_init	ksmg_application_init is responsible for performing signal setup and connecting to the kiosk_dsif process. An error message is displayed if any errors occur.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.

Table 33 - Routines called by INITIALLY

4.2.1.3.5 ksmg_application_init

ksmg_application_init is responsible for performing signal setup and connecting to the kiosk_dsif process. An error message is displayed if any errors occur. The structure chart for ksmg_application_init is depicted in Figure 31. A description of the routines called by ksmg_application_init is provided in Table 34.

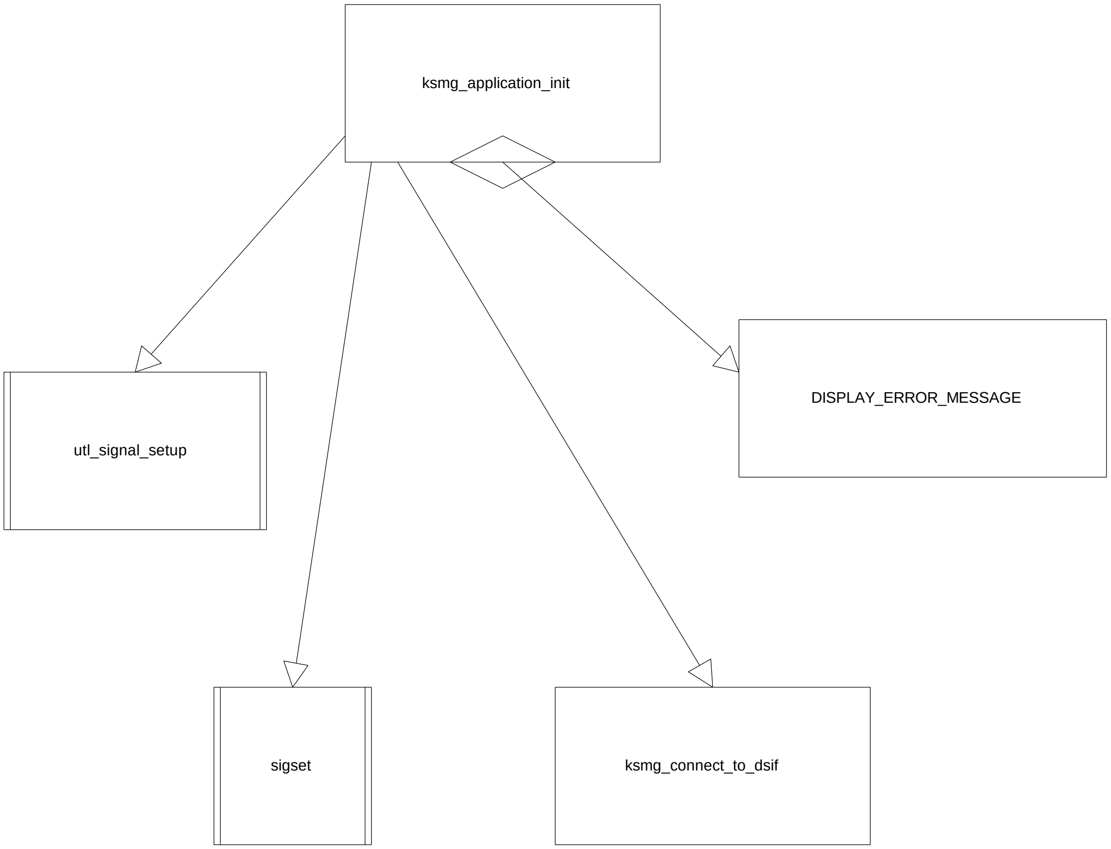

Figure 31 - ksmg_application_init Structure Chart

Function	Description
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
ksmg_connect_to_dsif	ksmg_connect_to_dsif is responsible for connecting to the Kiosk Data Server Interface Process based on the configuration parameters. These parameters specify the host name and service name to be used to connect.
sigset	C Library Function used to modify the disposition of a signal. The signal can be caught, ignored, or returned to the default disposition.
utl_signal_setup	MDI Common Utility Library routine used to set up a default signal handler for all catchable signals.

Table 34 - Routines called by ksmg_application_init

4.2.1.3.6 DISPLAY_ERROR_MESSAGE

DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box. The structure chart for DISPLAY_ERROR_MESSAGE is depicted in Figure 32. A description of the routines called by DISPLAY_ERROR_MESSAGE is provided in Table 35.

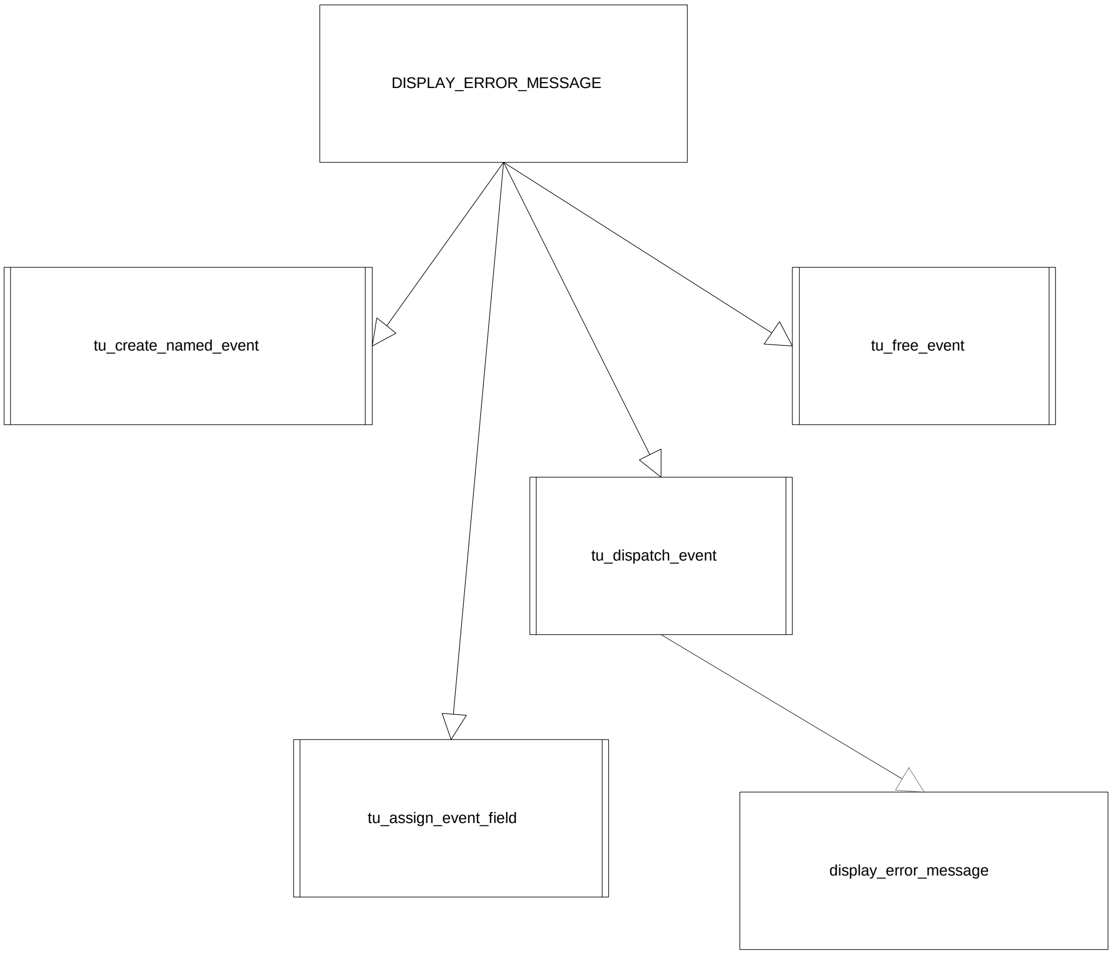

Figure 32 - DISPLAY_ERROR_MESSAGE Structure Chart

Function	Description
display_error_message	display_error_message is a GUI layer event that creates an error dialog box to display the specified error message.
tu_assign_event_field	TeleUSE Library Function to associate the contents of a C variable with the contents of an event attribute.
tu_create_named_event	TeleUSE Library Function used to create the data structure necessary to interface the C code with the D event code.
tu_dispatch_event	TeleUSE Library Function used to dispatch the created event. This causes the event to be executed.
tu_free_event	TeleUSE Library Function used to free up any memory that was allocated to the event data structure using tu_create_named_event.

Table 35 - Routines called by DISPLAY_ERROR_MESSAGE

4.2.1.3.7 display_error_message

display_error_message is a GUI layer event that creates an error dialog box to display the specified error message. The structure chart for display_error_message is depicted in Figure 33. A description of the routines called by display_error_message is provided in Table 36.

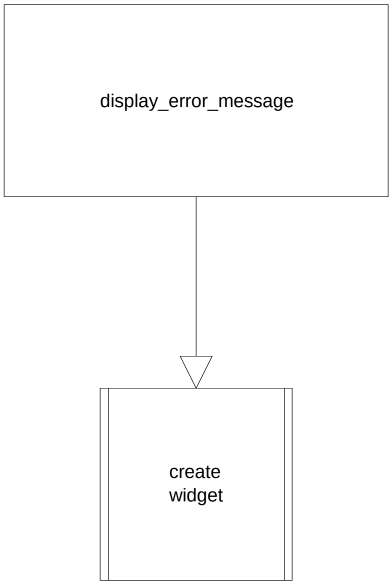

Figure 33 - display_error_message Structure Chart

Function	**Description**
create widget	create widget is used to create a widget of a particular TeleUSE template allowing for the specification of a widget name and a parent for the widget.

Table 36 - Routines called by display_error_message

4.2.1.3.8 ksmg_connect_to_dsif

ksmg_connect_to_dsif is responsible for connect to the Kiosk Data Server Interface Process based on the configuration parameters. These parameters specify the host name and service name to be used to connect. The structure chart for ksmg_connect_to_dsif is depicted in Figure 34. A description of the routines called by ksmg_connect_to_dsif is provided in Table 37.

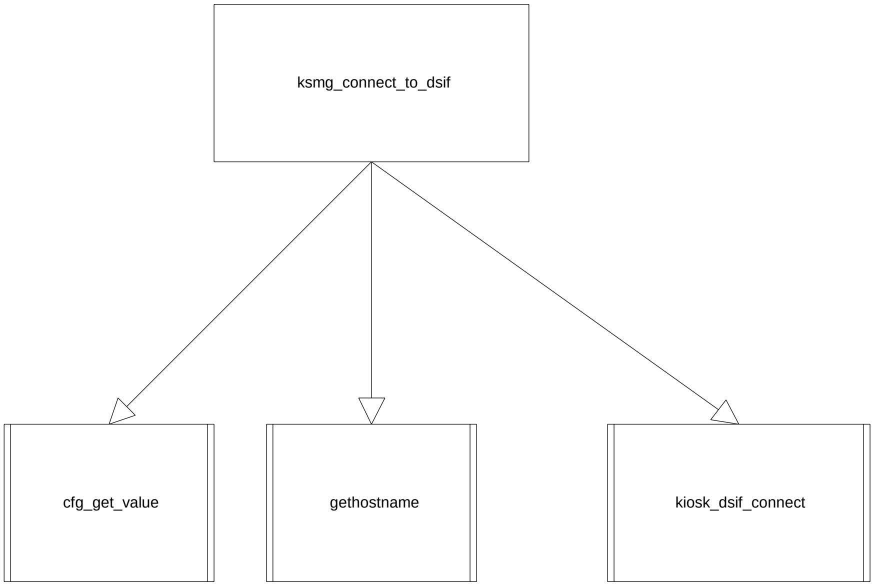

Figure 34 - ksmg_connect_to_dsif Structure Chart

Function	Description
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
gethostname	UNIX system call that returns the hostname of the machine on which the function is executing.
kiosk_dsif_connect	kiosk_dsif_connect is used to connect to the Kiosk Data Server Interface process. The service name is passed to this routine and is used to make the connection to the appropriate port.

Table 37 - Routines called by ksmg_connect_to_dsif

4.2.1.3.9 INITIALLY (Kiosk)

INITIALLY (Kiosk) is the INITIALLY event for the kiosk configuration D module. This event is responsible for initializing the D module flags and creating the header string to be used for the configuration item list. The structure chart for INITIALLY (Kiosk) is depicted in Figure 35. A description of the routines called by INITIALLY (Kiosk) is provided in Table 38.

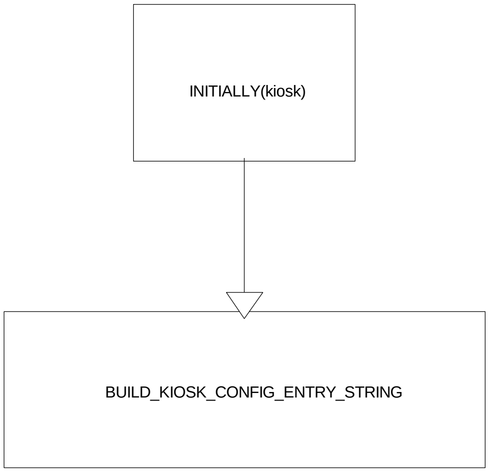

Figure 35 - INITIALLY (Kiosk) Structure Chart

Function	Description
BUILD_KIOSK_CONFIG_ENTRY_STRING	BUILD_KIOSK_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list. All text strings are left justified within the appropriate fields of the entry string.

Table 38 - Routines called by INITIALLY (Kiosk)

4.2.1.3.10 BUILD_KIOSK_CONFIG_ENTRY_STRING

BUILD_KIOSK_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list. All text strings are left justified within the appropriate fields of the entry string. The structure chart for BUILD_KIOSK_CONFIG_ENTRY_STRING is depicted in Figure 36. A description of the routines called by BUILD_KIOSK_CONFIG_ENTRY_STRING is provided in Table 39.

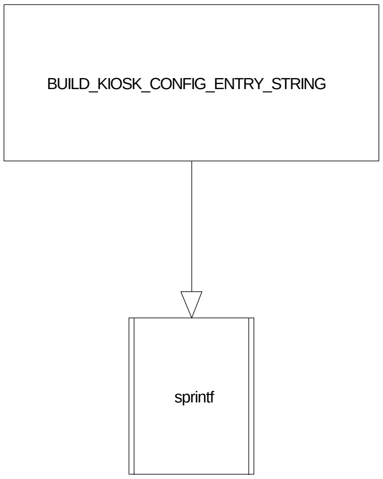

Figure 36 - BUILD_KIOSK_CONFIG_ENTRY_STRING Structure Chart

Function	Description
sprintf	C Library Function that provides printf capabilities to a character string.

Table 39 - Routines called by BUILD_KIOSK_CONFIG_ENTRY_STRING

4.2.1.3.11 INITIALLY (Parking)

INITIALLY (Parking) is the INITIALLY event for the airport parking lot configuration D module. This event is responsible for initializing the D module flags and creating the header string to be used for the configuration item list. The structure chart for INITIALLY (Parking) is depicted in Figure 37. A description of the routines called by INITIALLY (Parking) is provided in Table 40.

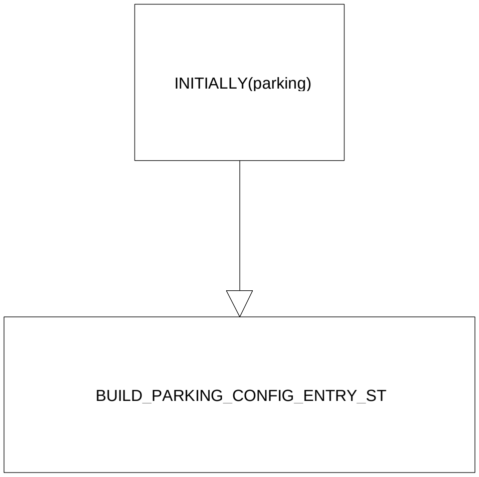

Figure 37 - INITIALLY (Parking) Structure Chart

Function	Description
BUILD_PARKING_CONFIG_ENTRY_STR	BUILD_PARKING_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list.

Table 40 - Routines called by INITIALLY (Parking)

4.2.1.3.12 BUILD_PARKING_CONFIG_ENTRY_STR

BUILD_PARKING_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list. The structure chart for BUILD_PARKING_CONFIG_ENTRY_STR is depicted in Figure 38. A description of the routines called by BUILD_PARKING_CONFIG_ENTRY_STR is provided in Table 41.

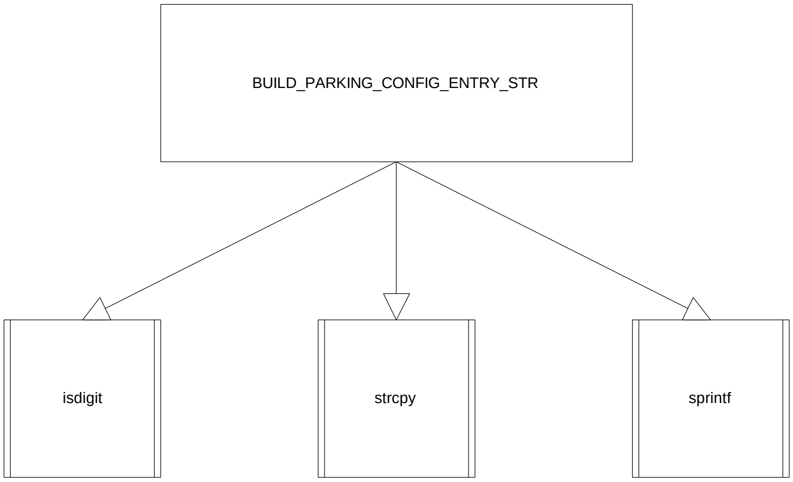

Figure 38 - BUILD_PARKING_CONFIG_ENTRY_STR Structure Chart

Function	Description
isdigit	C Library Function that determines if the given value is a number.
sprintf	C Library Function that provides printf capabilities to a character string.
strcpy	C Library Function used to copy a source string to a destination string.

Table 41 - Routines called by BUILD_PARKING_CONFIG_ENTRY_STR

4.2.1.3.13 INITIALLY (airline)

INITIALLY (airline) is the INITIALLY event for the airline configuration D module. This event is responsible for initializing the D module flags and creating the header string to be used for the configuration item list. The structure chart for INITIALLY (airline) is depicted in Figure 39. A description of the routines called by INITIALLY (airline) is provided in Table 42.

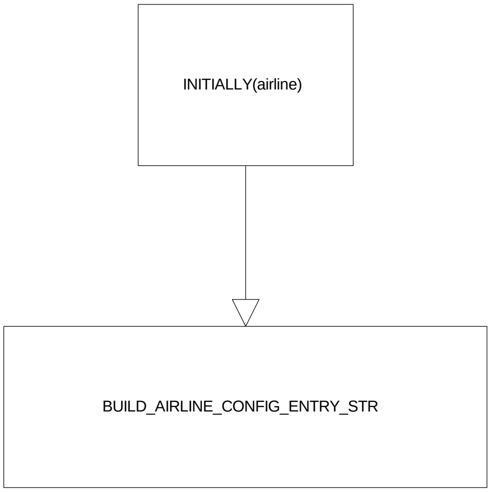

Figure 39 - INITIALLY (airline) Structure Chart

Function	Description
BUILD_AIRLINE_CONFIG_ENTRY_STR	BUILD_AIRLINE_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list. All text strings are left justified within the appropriate fields of the entry string.

Table 42 - Routines called by INITIALLY (airline)

4.2.1.3.14 BUILD_AIRLINE_CONFIG_ENTRY_STR

BUILD_AIRLINE_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list. All text strings are left justified within the appropriate fields of the entry string. The structure chart for BUILD_AIRLINE_CONFIG_ENTRY_STR is depicted in Figure 40. A description of the routines called by BUILD_AIRLINE_CONFIG_ENTRY_STR is provided in Table 43.

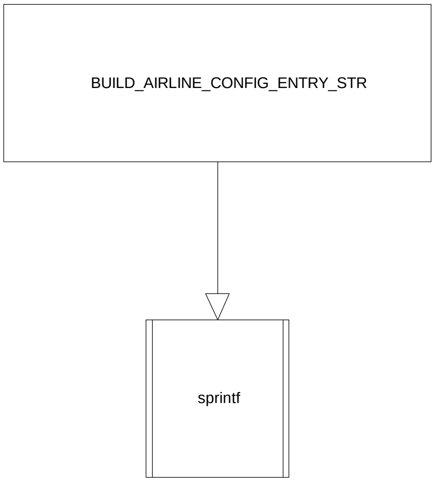

Figure 40 - BUILD_AIRLINE_CONFIG_ENTRY_STR Structure Chart

Function	Description
sprintf	C Library Function that provides printf capabilities to a character string.

Table 43 - Routines called by BUILD_AIRLINE_CONFIG_ENTRY_STR

4.2.1.3.15 INITIALLY (rental)

INITIALLY (rental) is the INITIALLY event for the rental car agency configuration D module. This event is responsible for initializing the D module flags and creating the header string to be used for the configuration item list. The structure chart for INITIALLY (rental) is depicted in Figure 41. A description of the routines called by INITIALLY (rental) is provided in Table 44.

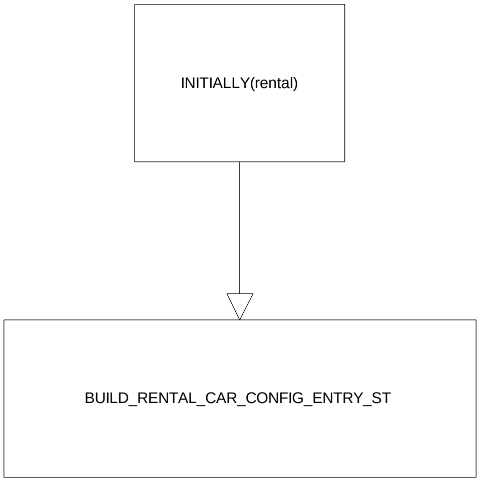

Figure 41 - INITIALLY (rental) Structure Chart

Function	Description
BUILD_RENTAL_CAR_CONFIG_ENTRY_ST	BUILD_RENTAL_CAR_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list. All text strings are left justified within the appropriate fields of the entry string.

Table 44 - Routines called by INITIALLY (rental)

4.2.1.3.16 BUILD_RENTAL_CAR_CONFIG_ENTRY_ST

BUILD_RENTAL_CAR_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list. All text strings are left justified within the appropriate fields of the entry string. The structure chart for BUILD_RENTAL_CAR_CONFIG_ENTRY_ST is depicted in Figure 42. A description of the routines called by BUILD_RENTAL_CAR_CONFIG_ENTRY_ST is provided in Table 45.

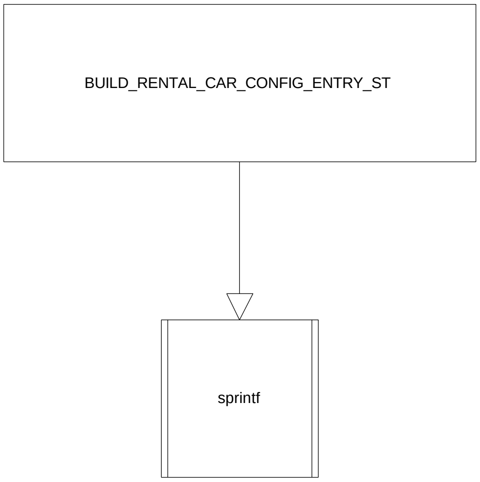

Figure 42 - BUILD_RENTAL_CAR_CONFIG_ENTRY_ST Structure Chart

Function	Description
sprintf	C Library Function that provides printf capabilities to a character string.

Table 45 - Routines called by BUILD_RENTAL_CAR_CONFIG_ENTRY_ST

4.2.1.3.17 INITIALLY (screensaver)

INITIALLY (screensaver) is the INITIALLY event for the screen saver configuration D module. This event is responsible for initializing the D module flags and creating the header string to be used for the configuration item list. The structure chart for INITIALLY (screensaver) is depicted in Figure 43. A description of the routines called by INITIALLY (screensaver) is provided in Table 46.

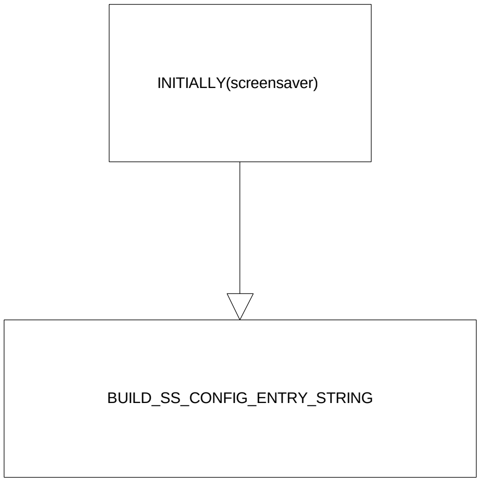

Figure 43 - INITIALLY (screensaver) Structure Chart

Function	Description
BUILD_SS_CONFIG_ENTRY_STRING	BUILD_SS_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list.

Table 46 - Routines called by INITIALLY (screensaver)

4.2.1.3.18 BUILD_SS_CONFIG_ENTRY_STRING

BUILD_SS_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list. The structure chart for BUILD_SS_CONFIG_ENTRY_STRING is depicted in Figure 44. A description of the routines called by BUILD_SS_CONFIG_ENTRY_STRING is provided in Table 47.

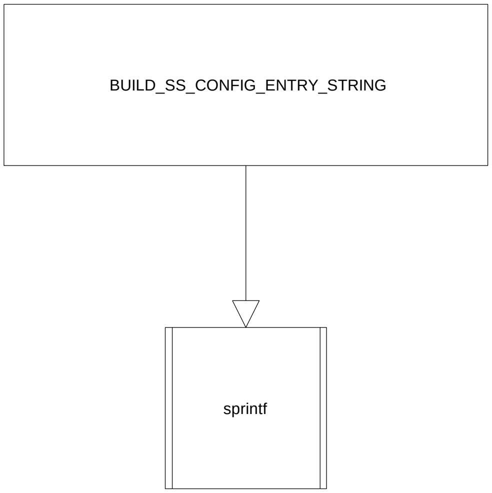

Figure 44 - BUILD_SS_CONFIG_ENTRY_STRING Structure Chart

Function	Description
sprintf	C Library Function that provides printf capabilities to a character string.

Table 47 - Routines called by BUILD_SS_CONFIG_ENTRY_STRING

4.2.1.3.19 configure_airlines

configure_airlines is the GUI layer event attached to the airlines configuration table menu item. The button is selected when the user desires to modify the contents of the airlines configuration file.The structure chart for configure_airlines is depicted in Figure 45. A description of the routines called by configure_airlines is provided in Table 48.

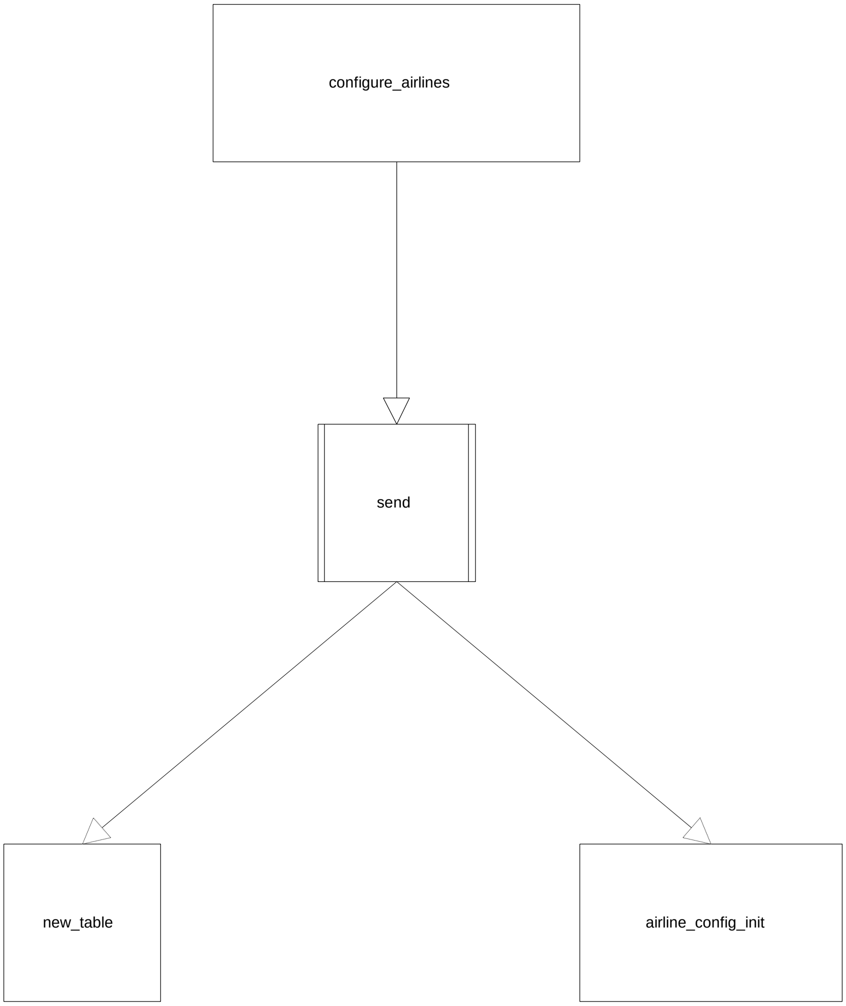

Figure 45 - configure_airlines Structure Chart

Function	Description
airline_config_init	airline_config_init is the GUI layer event used to initialize the user interface when a table (data file) is initially selected for modification. This includes clearing the list of items to be displayed, allowing additions to be made, reading the data file, and creating the data file if it doesn't exist.
new_table	new_table is a GUI layer event triggered when a new table is selected for editing. Each D module contains this event and if the D module is associated with the table currently being edited a check is make to see if the current table should be saved. If so, the user is asked whether or not the table should be saved. The sensitivity of the table menu item of the current table is made sensitive and the form containing the current table is unmanaged.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.

Table 48 - Routines called by configure_airlines

4.2.1.3.20 airline_config_init

airline_config_init is the GUI layer event used to initialize the user interface when a table (data file) is initially selected for modification. This includes clearing the list of items to be displayed, allowing additions to be made, reading the data file, and creating the data file if it doesn't exist. The structure chart for airline_config_init is depicted in Figure 46. A description of the routines called by airline_config_init is provided in Table 49.

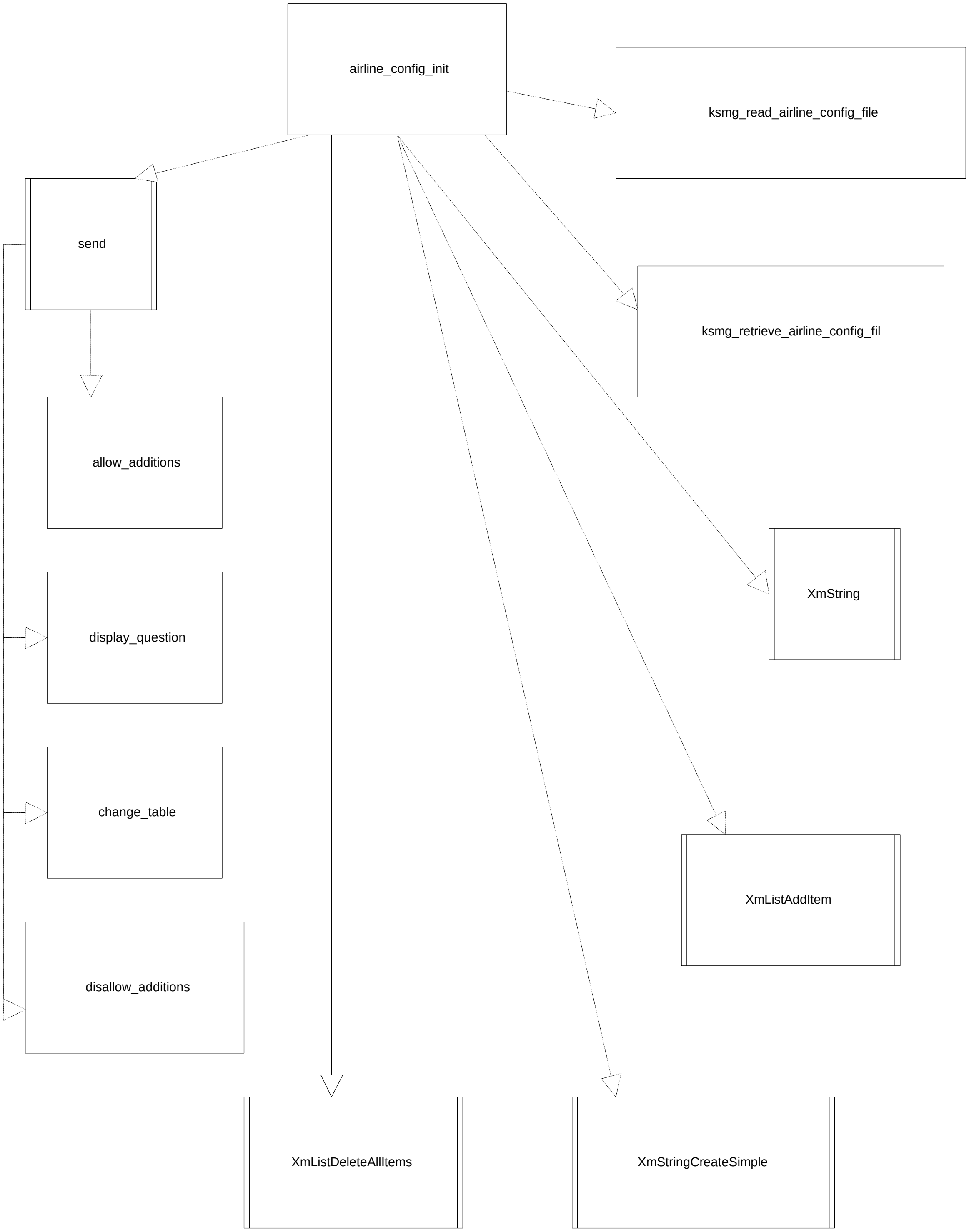

Figure 46 - airline_config_init Structure Chart

Function	Description
Allow_additions	allow_additions is a GUI layer event in charge of setting the sensitivity for the Add button to true and desensitizing the delete and modify buttons.
Change_table	change_table is a GUI layer event responsible for unmapping the current configuration data form and sensitizing the table submenu item associated with the current configuration table.
Disallow_additions	disallow_additions is the GUI layer routine that sets the sensitivity of the add menu item so that no additions can be made until a table is selected.
Display_question	display_question is a GUI layer event that creates a question dialog box to display the specified question to the user.
Ksmg_read_airline_config_file	ksmg_read_airline_config_file is used to read the contents of the airline data file and create the list of airlines to be displayed to the user.
Ksmg_retrieve_airline_config_fil	ksmg_retrieve_airline_config_file is used to retrieve the airline data file from the data server and place it on the local file system.
Send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
XmListAddItem	X Library function that will add a character string to a scroll list.
XmListDeleteAllItems	X Library function that will delete all items from a scroll list.
XmString	X Library function that will create an X string from a character string.
XmStringCreateSimple	X Library function that will create a simple X string from a character string.

Table 49 - Routines called by airline_config_init

4.2.1.3.21 display_question

display_question is a GUI layer event that creates a question dialog box to display the specified question to the user. The structure chart for display_question is depicted in Figure 47. A description of the routines called by display_question is provided in Table 50.

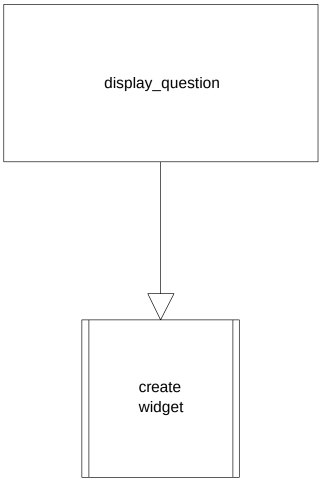

Figure 47 - display_question Structure Chart

Function	Description
create widget	create widget is used to create a widget of a particular TeleUSE template allowing for the specification of a widget name and a parent for the widget.

Table 50 - Routines called by display_question

4.2.1.3.22 ksmg_read_airline_config_file

ksmg_read_airline_config_file is used to read the contents of the airline data file and create the list of airlines to be displayed to the user. The structure chart for ksmg_read_airline_config_file is depicted in Figure 48. A description of the routines called by ksmg_read_airline_config_file is provided in Table 51.

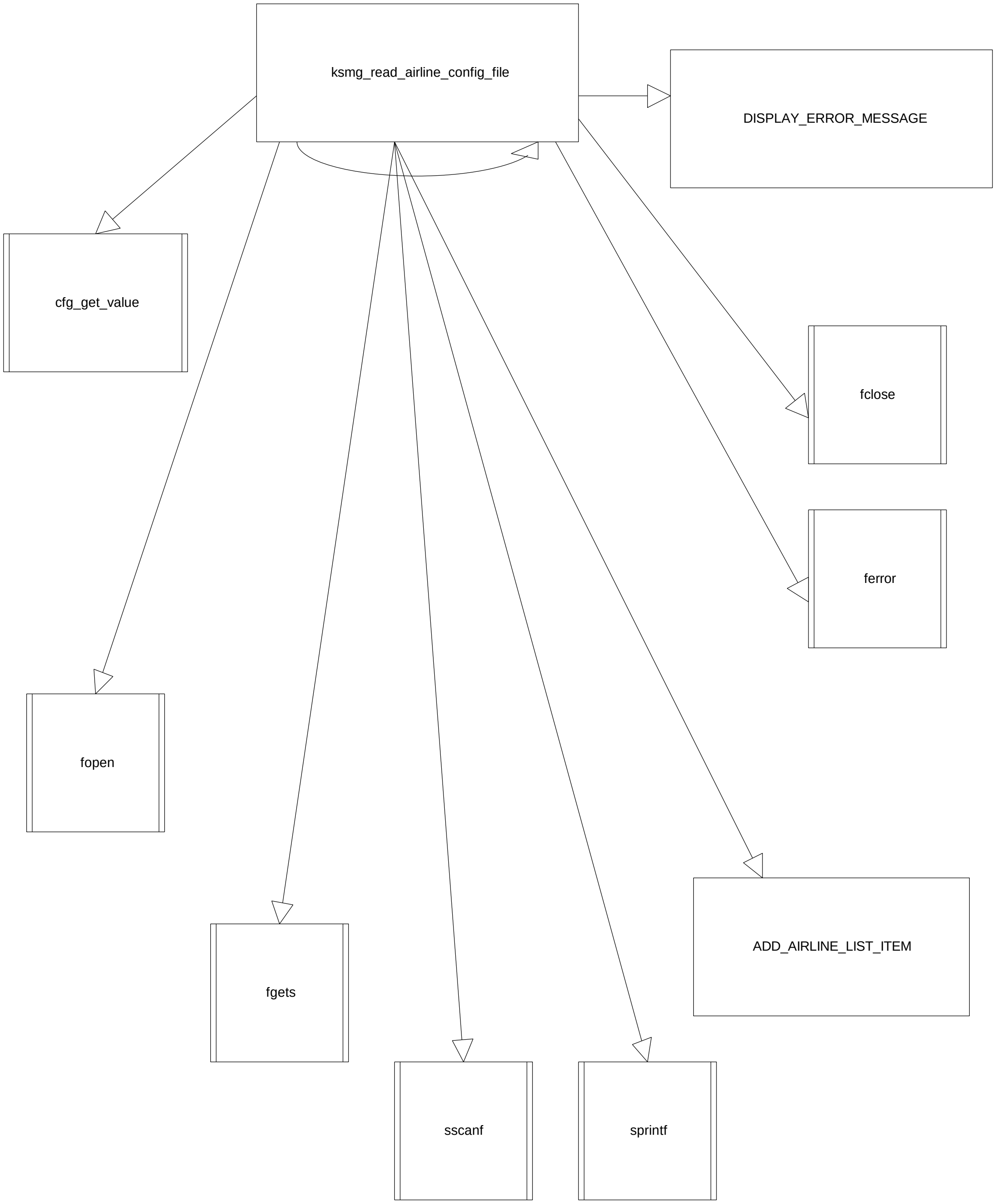

Figure 48 - ksmg_read_airline_config_file Structure Chart

Function	Description
ADD_AIRLINE_LIST_ITEM	ADD_AIRLINE_LIST_ITEM is a bridge layer routine that creates and dispatches the GUI layer event responsible for adding an entry to the airlines list.
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
fclose	C Library Function used to close an open file.
ferror	C Library Function that returns any previous errors on the associated stream.
fgets	C Library Function used to read a line of text from a file.
fopen	C Library Function that opens the specified file using the specified access mode.
sprintf	C Library Function that provides printf capabilities to a character string.
sscanf	UNIX function call that will allow formatted "input" from a NULL terminated character string.

Table 51 - Routines called by ksmg_read_airline_config_file

4.2.1.3.23 ADD_AIRLINE_LIST_ITEM

ADD_AIRLINE_LIST_ITEM is a bridge layer routine that creates and dispatches the GUI layer event responsible for adding an entry to the airlines list. The structure chart for ADD_AIRLINE_LIST_ITEM is depicted in Figure 49. A description of the routines called by ADD_AIRLINE_LIST_ITEM is provided in Table 52.

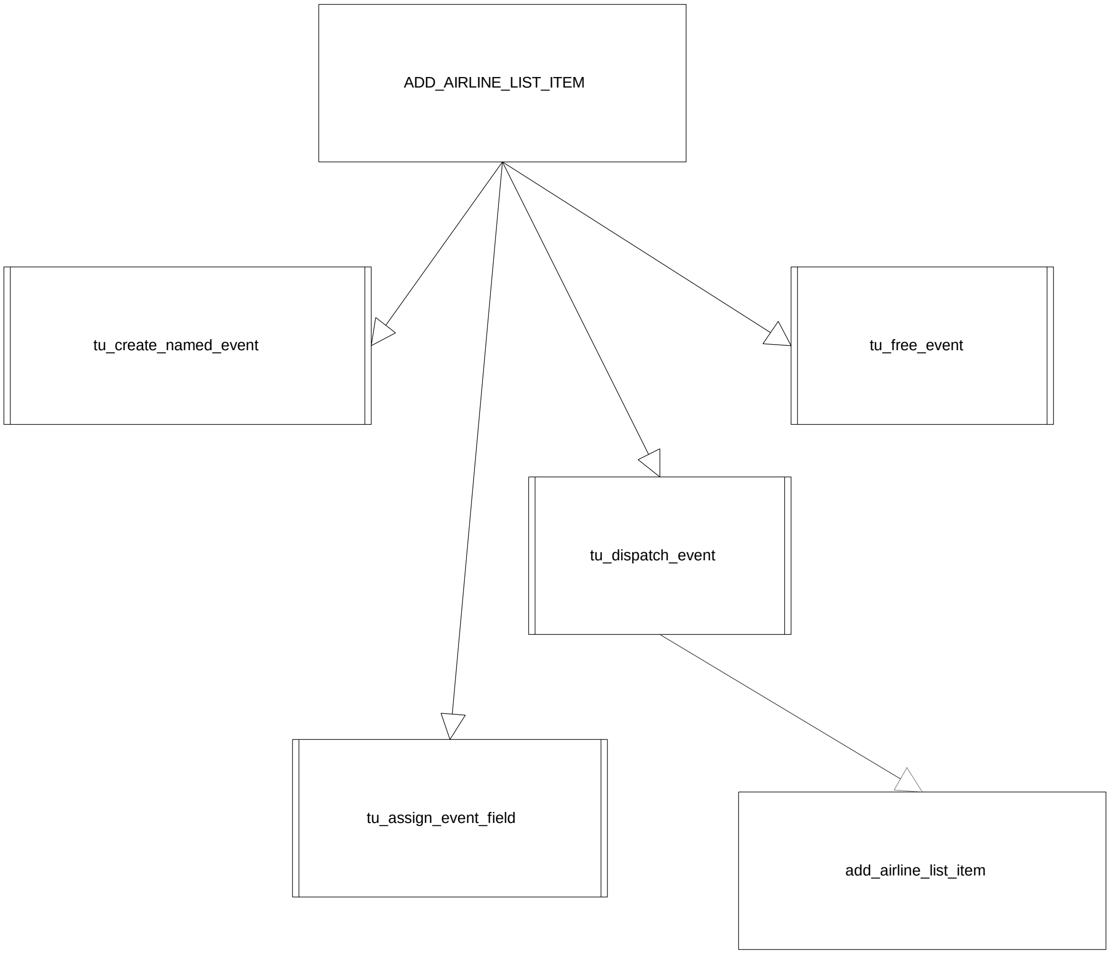

Figure 49 - ADD_AIRLINE_LIST_ITEM Structure Chart

Function	Description
add_airline_list_item	add_airline_list_item is a GUI layer event that creates an item in the airline list using the data specified for the event.
tu_assign_event_field	TeleUSE Library Function to associate the contents of a C variable with the contents of an event attribute.
tu_create_named_event	TeleUSE Library Function used to create the data structure necessary to interface the C code with the D event code.
tu_dispatch_event	TeleUSE Library Function used to dispatch the created event. This causes the event to be executed.
tu_free_event	TeleUSE Library Function used to free up any memory that was allocated to the event data structure using tu_create_named_event.

Table 52 - Routines called by ADD_AIRLINE_LIST_ITEM

4.2.1.3.24 ksmg_retrieve_airline_config_file

ksmg_retrieve_airline_config_file is used to retrieve the airline data file from the data server and place it on the local file system. The structure chart for ksmg_retrieve_airline_config_file is depicted in Figure 50. A description of the routines called by ksmg_retrieve_airline_config_file is provided in Table 53.

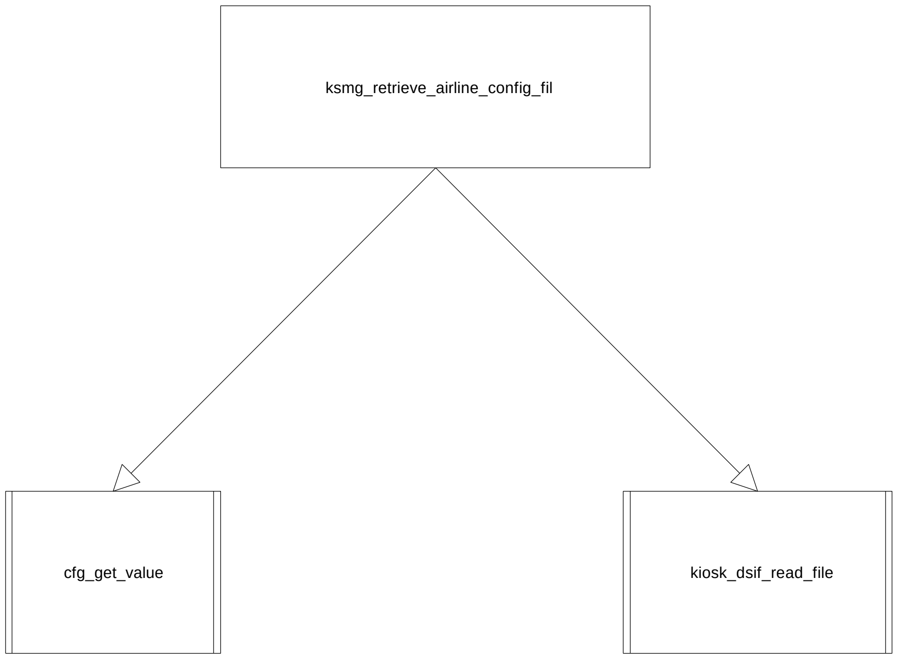

Figure 50 - ksmg_retrieve_airline_config_file Structure Chart

Function	Description
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
kiosk_dsif_read_file	kiosk_dsif_read_file packages the read file request into the appropriate kiosk_dsif message and sends it to the kiosk_dsif process.

Table 53 - Routines called by ksmg_retrieve_airline_config_file

4.2.1.3.25 config_add_ok (airline)

config_add_ok (airline) is a GUI layer event triggered by the user selecting the ok button on the Airline Configuration Add Entry Dialog. The structure chart for config_add_ok (airline) is depicted in Figure 51. A description of the routines called by config_add_ok (airline) is provided in Table 54.

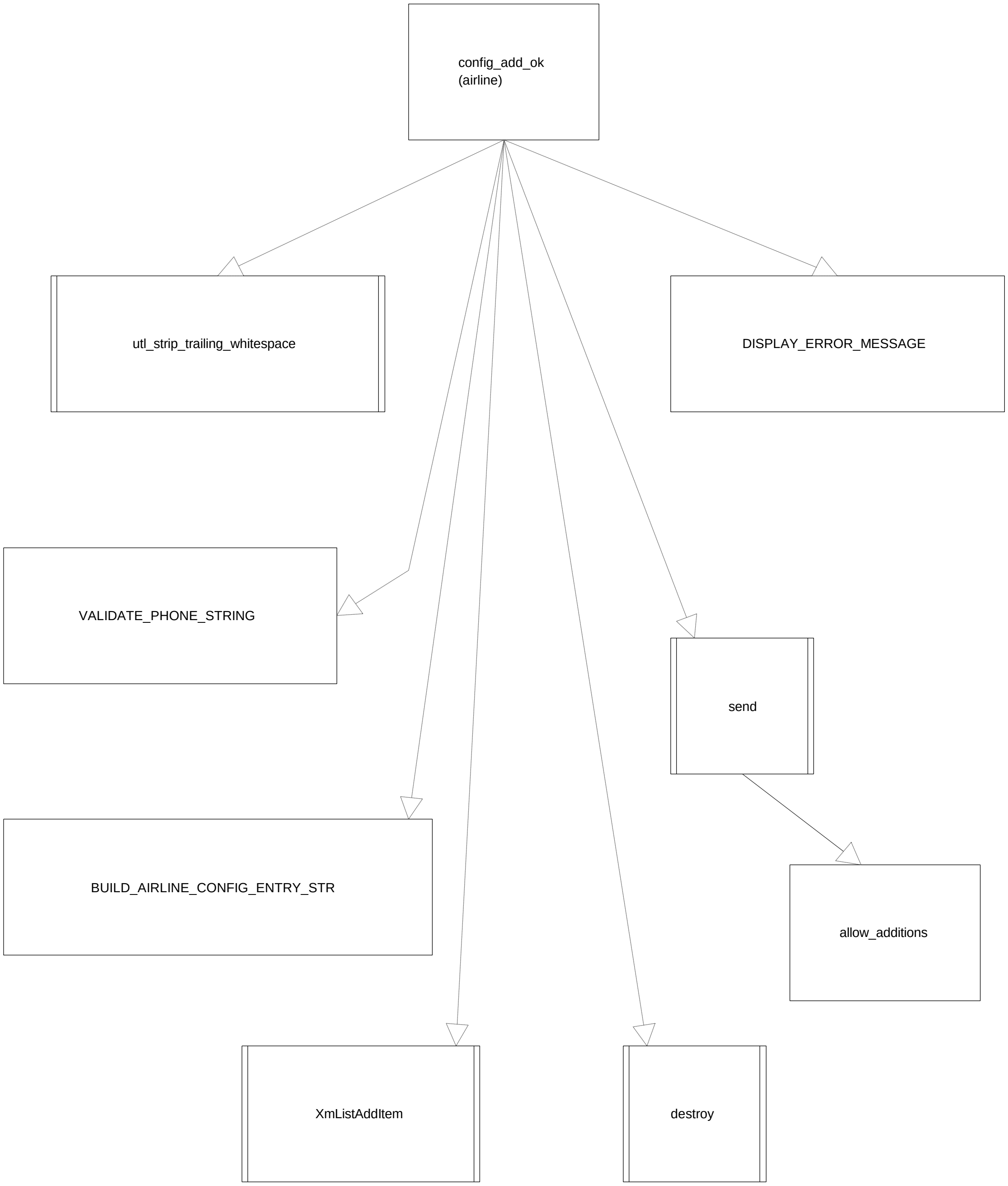

Figure 51 - conifg_add_ok (airline) Structure Chart

Function	Description
allow_additions	allow_additions is a GUI layer event in charge of setting the sensitivity for the Add button to true and desensitizing the delete and modify buttons.
BUILD_AIRLINE_CONFIG_ENTRY_STR	BUILD_AIRLINE_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list. All text strings are left justified within the appropriate fields of the entry string.
destroy	Function which will return allocated X resources back to the system.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
utl_strip_trailing_whitespace	MDI Common Utility Library routine used to remove trailing blanks from a stirng.
VALIDATE_PHONE_STRING	VALIDATE_PHONE_STRING is a bridge layer routine used to determine the validity of a telephone number entered by the user. Accepted strings are in the form of xxx-xxxx or x-xxx-xxx-xxxx.
XmListAddItem	X Library function that will add a character string to a scroll list.

Table 54 - Routines called by config_add_ok (airline)

4.2.1.3.26 VALIDATE_PHONE_STRING

VALIDATE_PHONE_STRING is a bridge layer routine used to determine the validity of a telephone number entered by the user. Accepted strings are in the form of xxx-xxxx or x-xxx-xxx-xxxx. The structure chart for VALIDATE_PHONE_STRING is depicted in Figure 52. A description of the routines called by VALIDATE_PHONE_STRING is provided in Table 55.

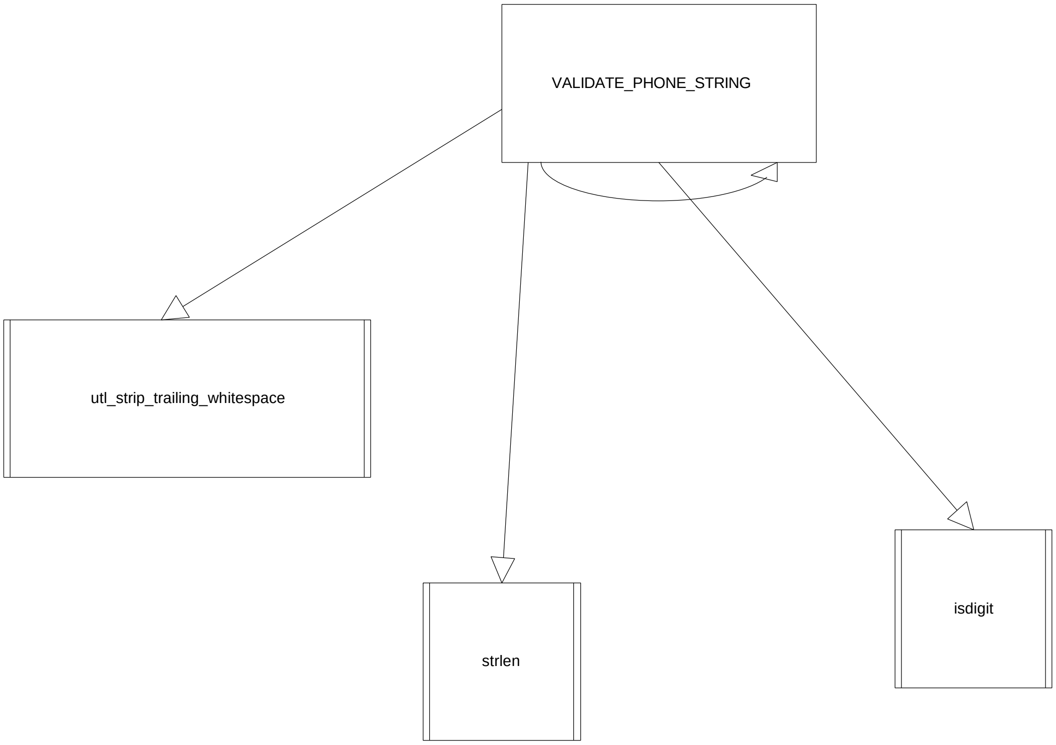

Figure 52 - VALIDATE_PHONE_STRING Structure Chart

Function	**Description**
isdigit	C Library Function that determines if the given value is a number.
strlen	UNIX system call that computes the number of characters in a NULL terminated string.
utl_strip_trailing_whitespace	MDI Common Utility Library routine used to remove trailing blanks from a stirng.

Table 55 - Routines called by VALIDATE_PHONE_STRING

4.2.1.3.27 ksmg_create_airline_config_file

ksmg_create_airline_config_file is used to create a new airline data file and write the blank file to the data server. The structure chart for ksmg_create_airline_config_file is depicted in Figure 53. A description of the routines called by ksmg_create_airline_config_file is provided in Table 56.

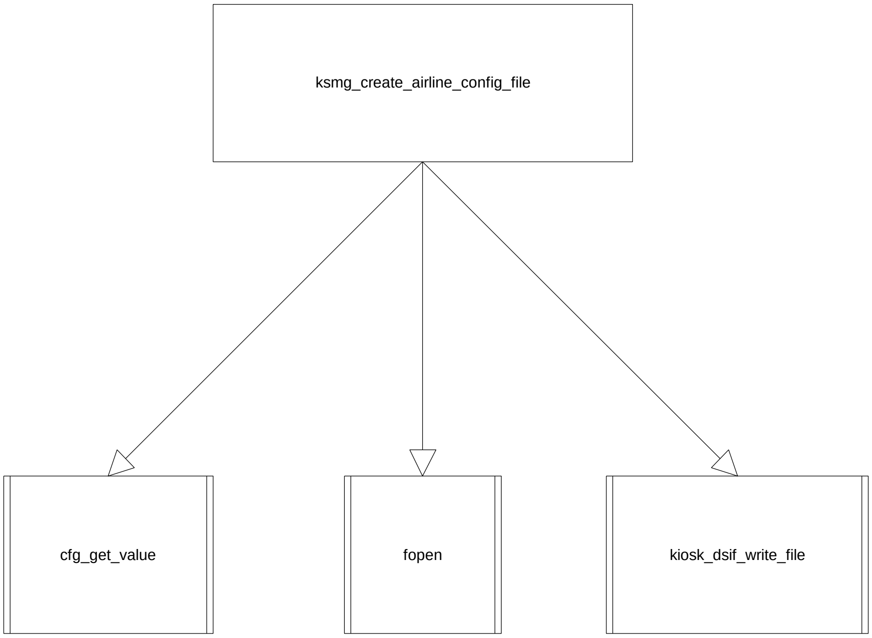

Figure 53 - ksmg_create_airline_config_file Structure Chart

Function	Description
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
fopen	C Library Function that opens the specified file using the specified access mode.
kiosk_dsif_write_file	kiosk_dsif_write_file packages the write file data into the appropriate kiosk_dsif message and sends it to the kiosk_dsif process.

Table 56 - Routines called by ksmg_create_airline_config_file

4.2.1.3.28 config_modify_ok (airline)

config_modify_ok (airline) is a GUI layer event triggered by the user selecting the ok button on the Airline Configuration Modify Entry Dialog. The structure chart for config_modify_ok (airline) is depicted in Figure 54. A description of the routines called by config_modify_ok (airline) is provided in Table 57.

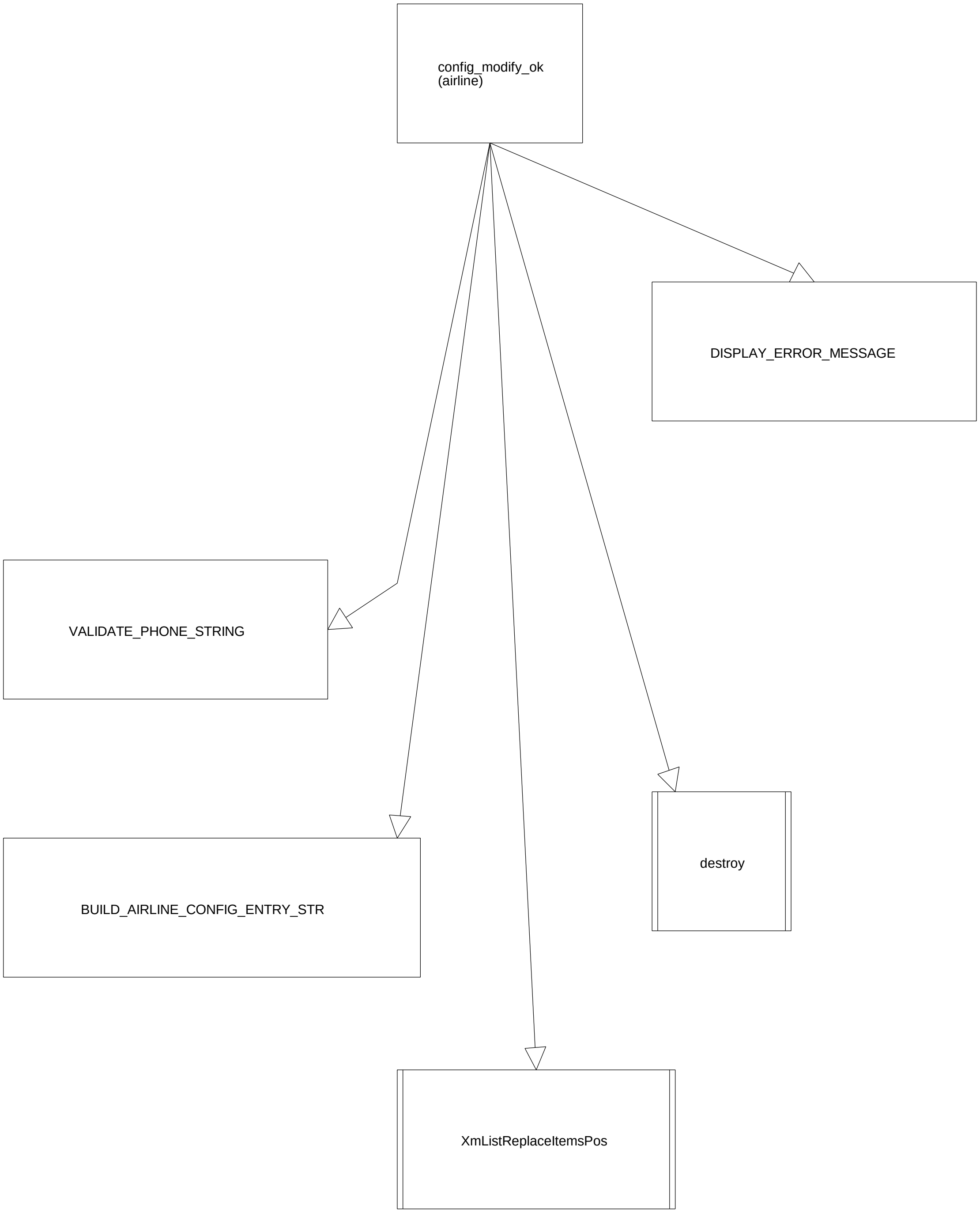

Figure 54 - config_modify_ok (airline) Structure Chart

Function	Description
BUILD_AIRLINE_CONFIG_ENTRY_STR	BUILD_AIRLINE_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list. All text strings are left justified within the appropriate fields of the entry string.
destroy	Function which will return allocated X resources back to the system.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
VALIDATE_PHONE_STRING	VALIDATE_PHONE_STRING is a bridge layer routine used to determine the validity of a telephone number entered by the user. Accepted strings are in the form of xxx-xxxx or x-xxx-xxx-xxxx.
XmListReplaceItemsPos	X Library function that will replace the instance of one X string with another instance in a Scroll List.

Table 57 - Routines called by config_modify_ok (airline)

4.2.1.3.29 save_table (airline)

save_table is responsibile for creating the string list object that is used to write the configuration data to the file. The structure chart for save_table (airline) is depicted in Figure 55. A description of the routines called by save_table (airline) is provided in Table 58.

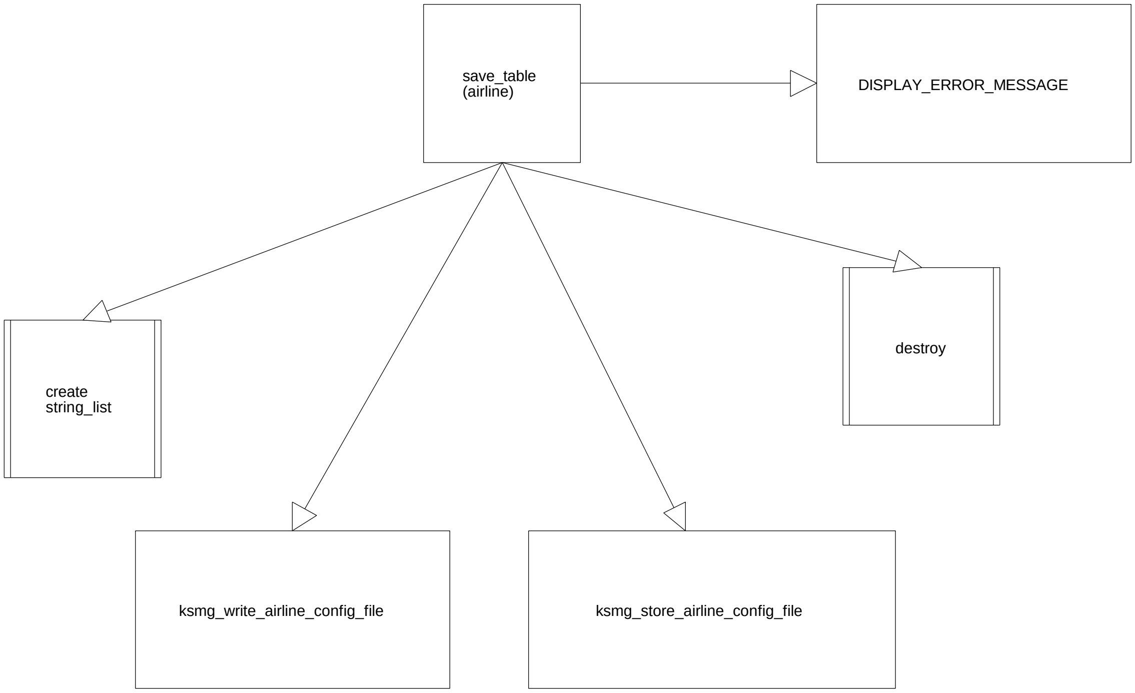

Figure 55 - save_table (airline) Structure Chart

Function	Description
create string_list	Function that creates a C character string from a series of X strings.
destroy	Function which will return allocated X resources back to the system.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
ksmg_store_airline_config_file	ksmg_store_airline_config_file is used to store the airline data file from the local file system to the data server.
ksmg_write_airline_config_file	ksmg_write_airline_config_file is used to write the list of airline data displayed to the user to the airline data file.

Table 58 - Routines called by save_table (airline)

4.2.1.3.30 ksmg_write_airline_config_file

ksmg_write_airline_config_file is used to write the list of airline data displayed to the user to the airline data file. The structure chart for ksmg_write_airline_config_file is depicted in Figure 56. A description of the routines called by ksmg_write_airline_config_file is provided in Table 59.

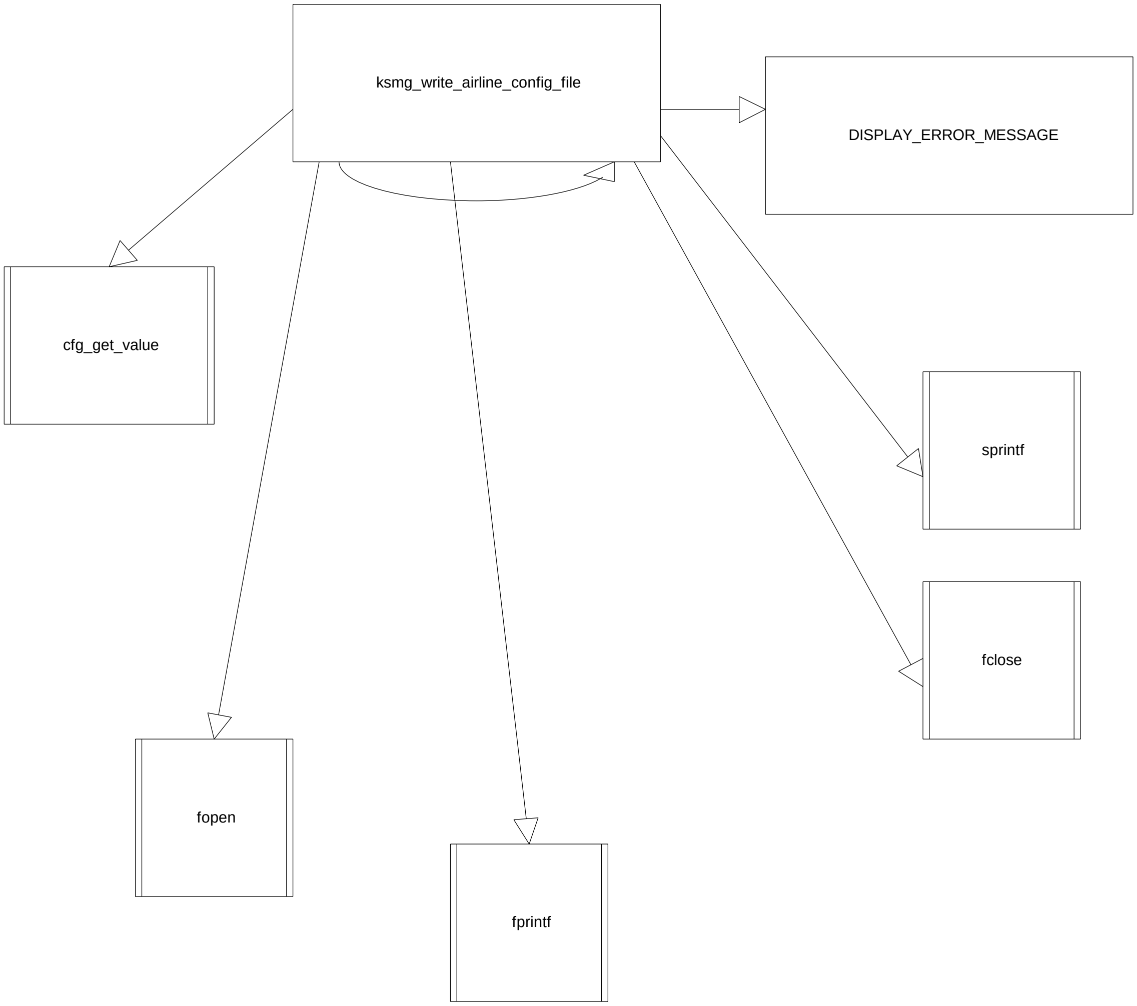

Figure 56 - ksmg_write_airline_config_file Structure Chart

Function	Description
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
fclose	C Library Function used to close an open file.
fopen	C Library Function that opens the specified file using the specified access mode.
fprintf	UNIX system call to print formatted data to a file stream.
sprintf	C Library Function that provides printf capabilities to a character string.

Table 59 - Routines called by ksmg_write_airline_config_file

4.2.1.3.31 ksmg_store_airline_config_file

ksmg_store_airline_config_file is used to store the airline data file from the local file system to the data server. The structure chart for ksmg_store_airline_config_file is depicted in Figure 57. A description of the routines called by ksmg_store_airline_config_file is provided in Table 60.

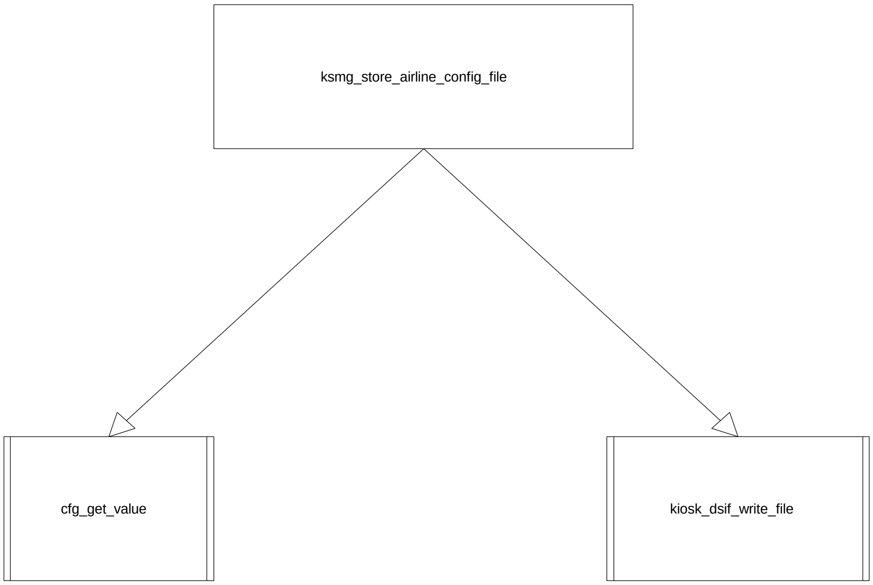

Figure 57 - ksmg_store_airline_config_file Structure Chart

Function	Description
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
kiosk_dsif_write_file	kiosk_dsif_write_file packages the write file data into the appropriate kiosk_dsif message and sends it to the kiosk_dsif process.

Table 60 - Routines called by ksmg_store_airline_config_file

4.2.1.3.32 configure_airport_parking

configure_airport_parking is the GUI layer event attached to the airport parking configuration table menu item. The button is selected when the user desires to modify the contents of the airport parking configuration file. The structure chart for configure_airport_parking is depicted in Figure 58. A description of the routines called by configure_airport_parking is provided in Table 61.

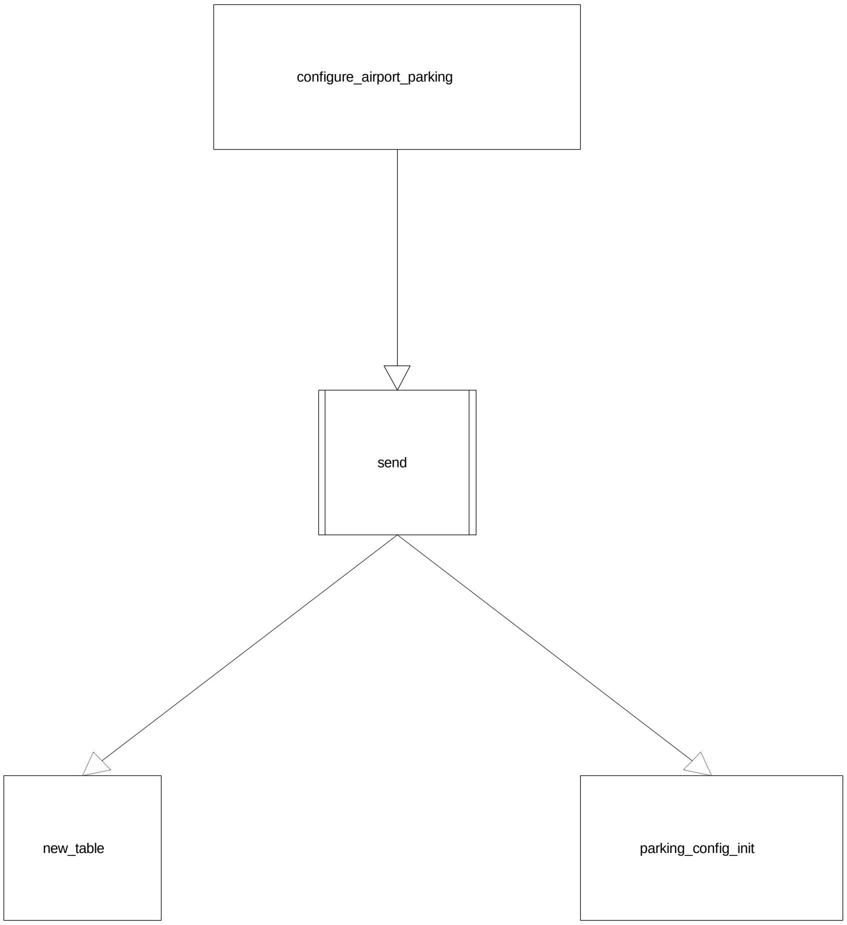

Figure 58 - configure_airport_parking Structure Chart

Function	Description
new_table	new_table is a GUI layer event triggered when a new table is selected for editing. Each D module contains this event and if the D module is associated with the table currently being edited a check is make to see if the current table should be saved. If so, the user is asked whether or not the table should be saved. The sensitivity of the table menu item of the current table is made sensitive and the form containing the current table is unmanaged.
parking_config_init	parking_config_init is the GUI layer event used to initialize the user interface when a table (data file) is initially selected for modification. This includes clearing the list of items to be displayed, allowing additions to be made, reading the data file, and creating the data file if it doesn't exist.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.

Table 61 - Routines called by configure_airport_parking

4.2.1.3.33 parking_config_init

parking_config_init is the GUI layer event used to initialize the user interface when a table (data file) is initially selected for modification. This includes clearing the list of items to be displayed, allowing additions to be made, reading the data file, and creating the data file if it doesn't exist. The structure chart for parking_config_init is depicted in Figure 59. A description of the routines called by parking_config_init is provided in Table 62.

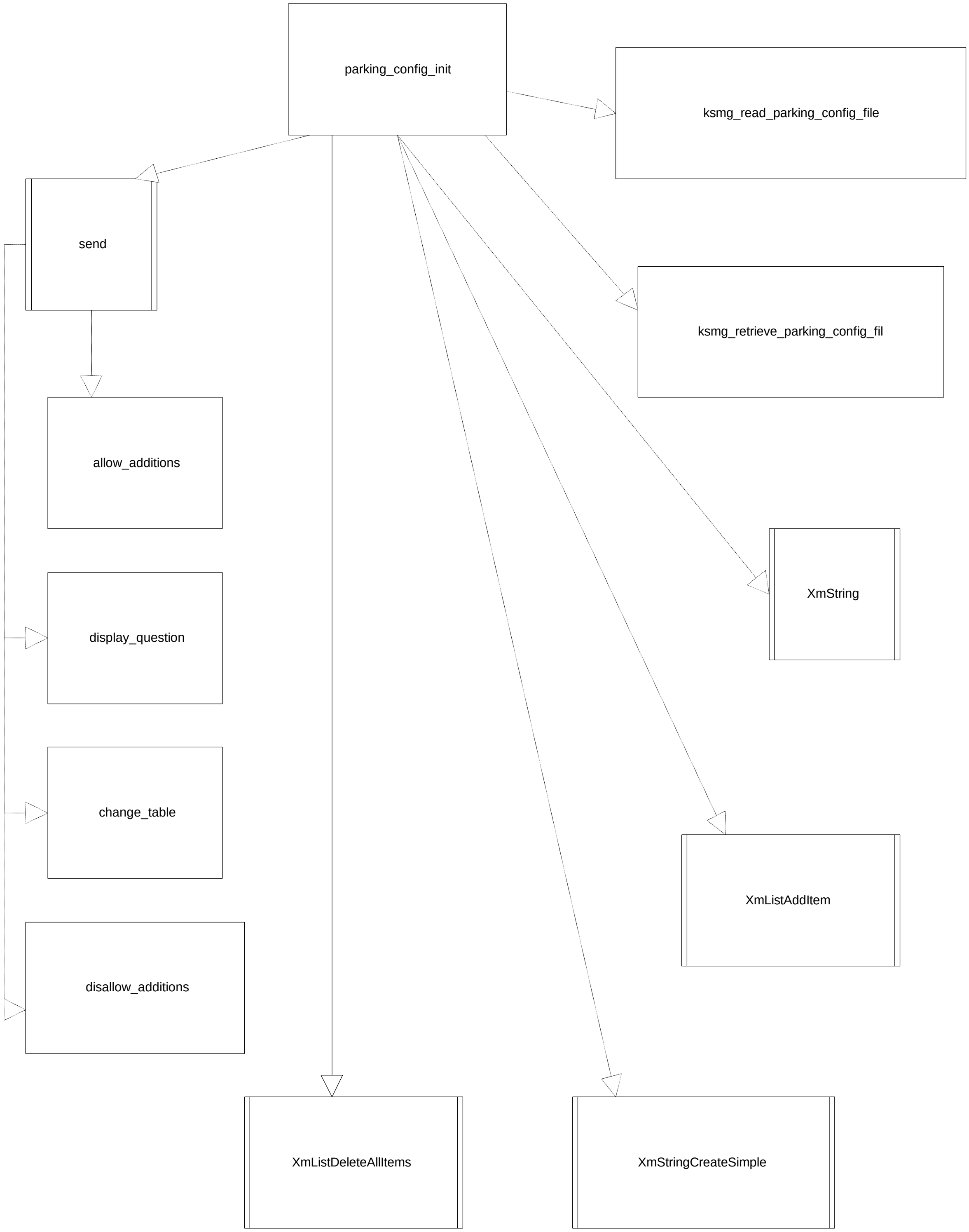

Figure 59 - parking_config_init Structure Chart

Function	Description
allow_additions	allow_additions is a GUI layer event in charge of setting the sensitivity for the Add button to true and desensitizing the delete and modify buttons.
change_table	change_table is a GUI layer event responsible for unmapping the current configuration data form and sensitizing the table submenu item associated with the current configuration table.
disallow_additions	disallow_additions is the GUI layer routine that sets the sensitivity of the add menu item so that no additions can be made until a table is selected.
display_question	display_question is a GUI layer event that creates a question dialog box to display the specified question to the user.
ksmg_read_parking_config_file	ksmg_read_parking_config_file is used to read the contents of the airport parking data file and create the list of airport parking lots to be displayed to the user.
ksmg_retrieve_parking_config_fil	ksmg_retrieve_parking_config_file is used to retrieve the airport parking data file from the data server and place it on the local file system.
Send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
XmListAddItem	X Library function that will add a character string to a scroll list.
XmListDeleteAllItems	X Library function that will delete all items from a scroll list.
XmString	X Library function that will create an X string from a character string.
XmStringCreateSimple	X Library function that will create a simple X string from a character string.

Table 62 - Routines called by parking_config_init

4.2.1.3.34 ksmg_read_parking_config_file

ksmg_read_parking_config_file is used to read the contents of the airport parking data file and create the list of airport parking lots to be displayed to the user. The structure chart for ksmg_read_parking_config_file is depicted in Figure 60. A description of the routines called by ksmg_read_parking_config_file is provided in Table 63.

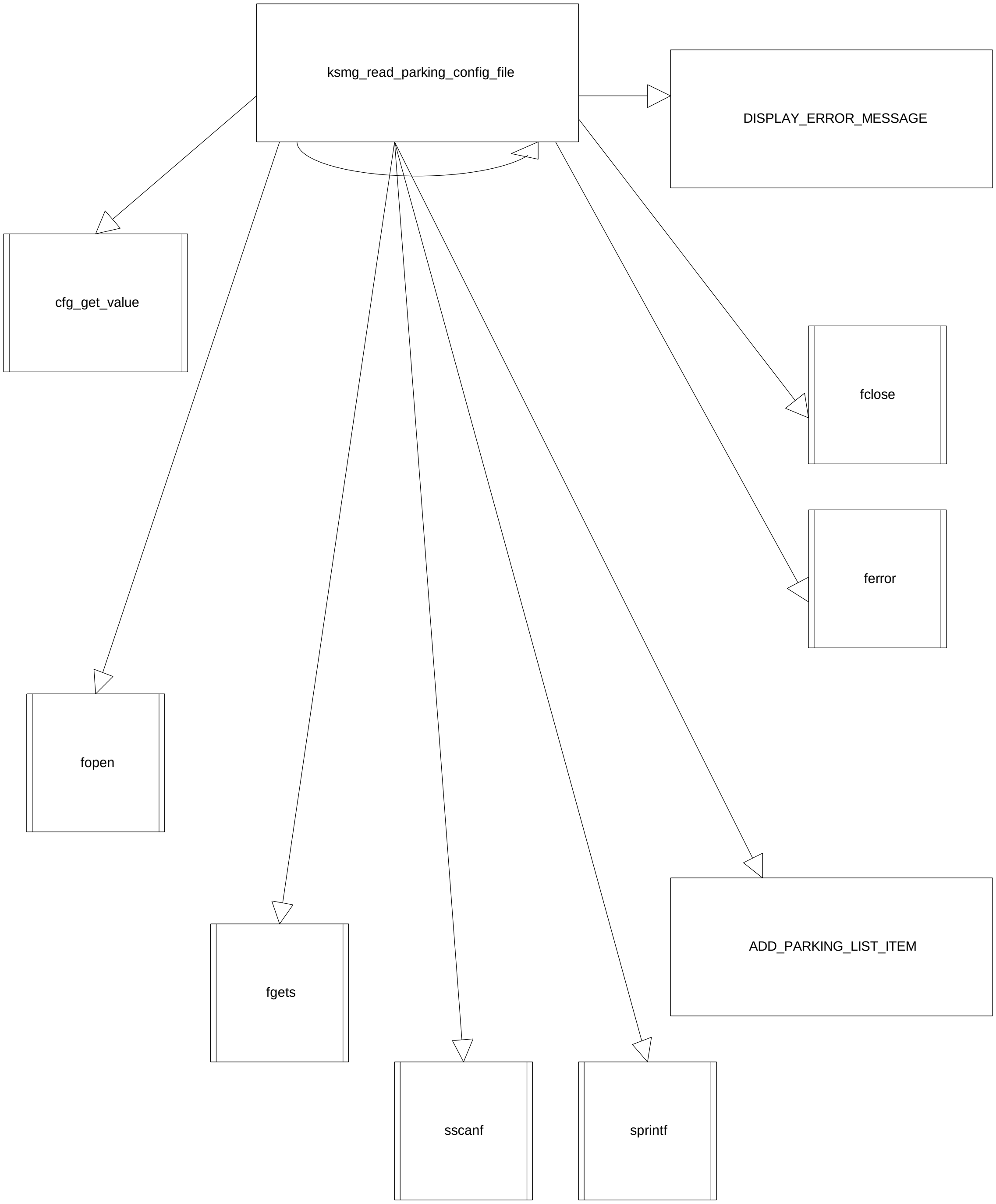

Figure 60 - ksmg_read_parking_config_file Structure Chart

Function	Description
ADD_PARKING_LIST_ITEM	ADD_PARKING_LIST_ITEM is a bridge layer routine that creates and dispatches the GUI layer event responsible for adding an entry to the airport parking lot list.
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
fclose	C Library Function used to close an open file.
ferror	C Library Function that returns any previous errors on the associated stream.
fgets	C Library Function used to read a line of text from a file.
fopen	C Library Function that opens the specified file using the specified access mode.
sprintf	C Library Function that provides printf capabilities to a character string.
sscanf	UNIX function call that will allow formatted "input" from a NULL terminated character string.

Table 63 - Routines called by ksmg_read_parking_config_file

4.2.1.3.35 ADD_PARKING_LIST_ITEM

ADD_PARKING_LIST_ITEM is a bridge layer routine that creates and dispatches the GUI layer event responsible for adding an entry to the airport parking lot list. The structure chart for ADD_PARKING_LIST_ITEM is depicted in Figure 61. A description of the routines called by ADD_PARKING_LIST_ITEM is provided in Table 64.

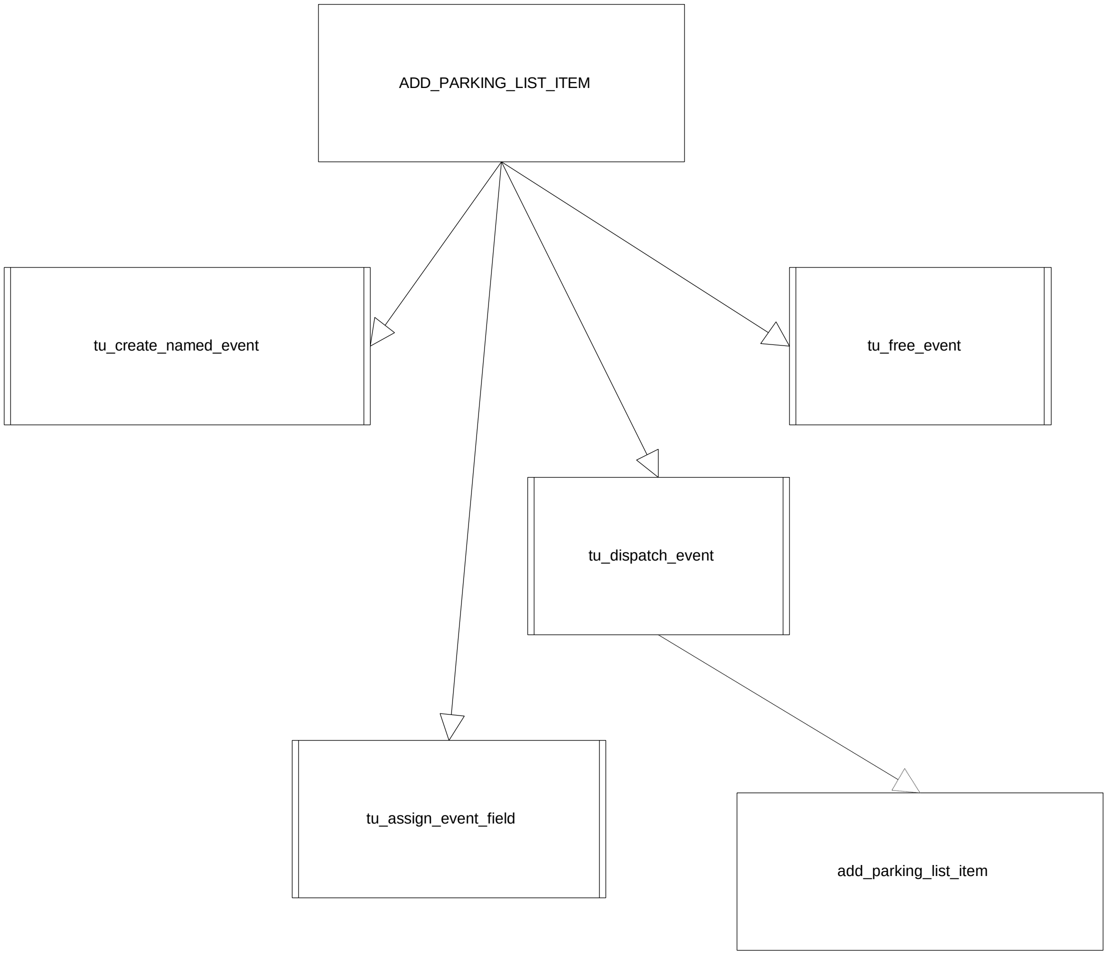

Figure 61 - ADD_PARKING_LIST_ITEM Structure Chart

Function	Description
add_parking_list_item	add_parking_list_item is a GUI layer event that creates an item in the airport parking lot list using the data specified for the event.
tu_assign_event_field	TeleUSE Library Function used to associate the contents of a C variable with the contents of an event attribute.
tu_create_named_event	TeleUSE Library Function used to create the data structure necessary to interface the C code with the D event code.
tu_dispatch_event	TeleUSE Library Function used to dispatch the created event. This causes the event to be executed.
tu_free_event	TeleUSE Library Function used to free up any memory that was allocated to the event data structure using tu_create_named_event.

Table 64 - Routines called by ADD_PARKING_LIST_ITEM

4.2.1.3.36 ksmg_retrieve_parking_config_file

ksmg_retrieve_parking_config_file is used to retrieve the airport parking data file from the data server and place it on the local file system. The structure chart for ksmg_retrieve_parking_config_file is depicted in Figure 62. A description of the routines called by ksmg_retrieve_parking_config_file is provided in Table 65.

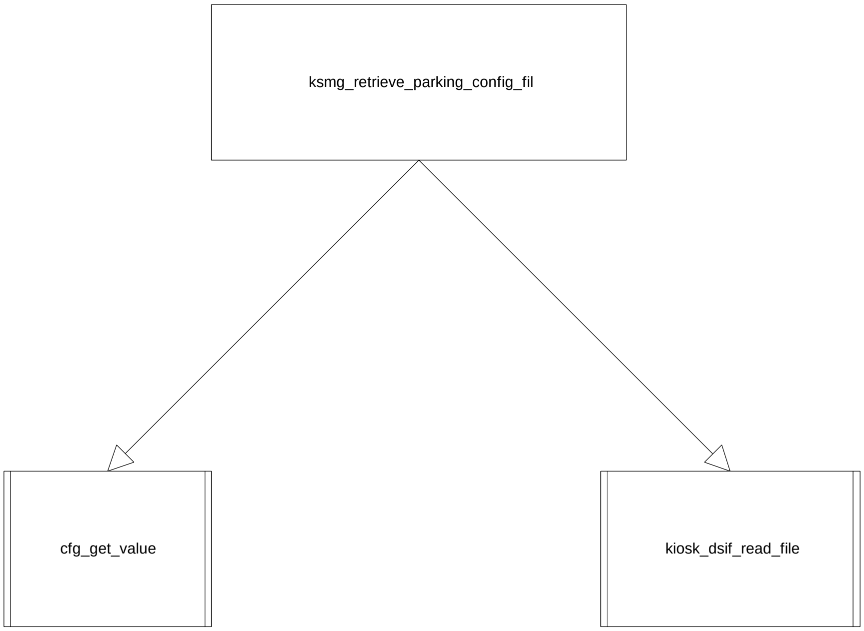

Figure 62 - ksmg_retrieve_parking_config_file Structure Chart

Function	Description
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
kiosk_dsif_read_file	kiosk_dsif_read_file packages the read file request into the appropriate kiosk_dsif message and sends it to the kiosk_dsif process.

Table 65 - Routines called by ksmg_retrieve_parking_config_file

4.2.1.3.37 config_add_ok (parking)

config_add_ok (parking) is a GUI layer event triggered by the user selecting the ok button on the Parking Configuration Add Entry Dialog. The structure chart for config_add_ok (parking) is depicted in Figure 63. A description of the routines called by config_add_ok (parking) is provided in Table 66.

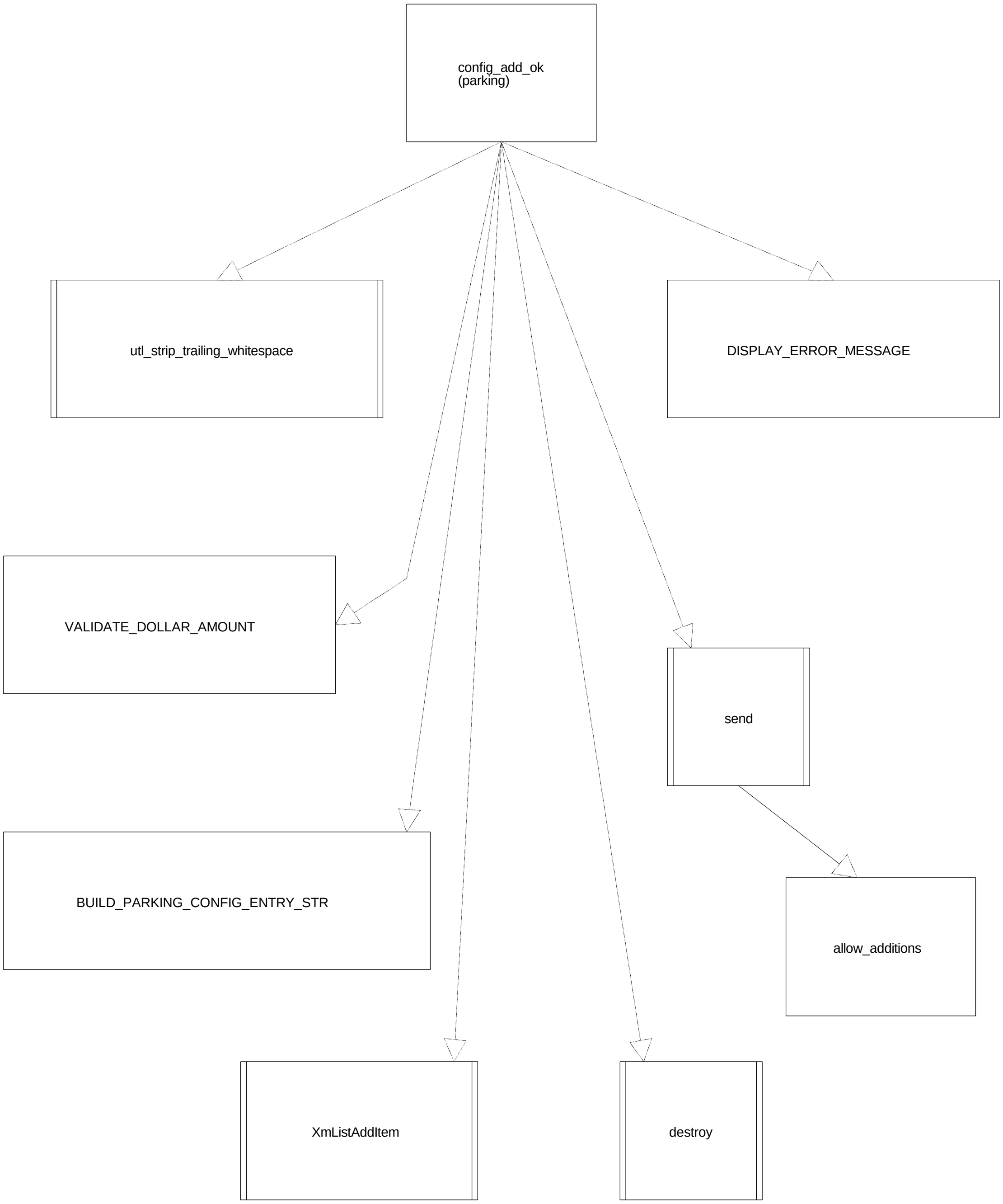

Figure 63 - config_add_ok (parking) Structure Chart

Function	Description
allow_additions	allow_additions is a GUI layer event in charge of setting the sensitivity for the Add button to true and desensitizing the delete and modify buttons.
BUILD_PARKING_CONFIG_ENTRY_STR	BUILD_PARKING_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list.
destroy	Function which will return allocated X resources back to the system.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
utl_strip_trailing_whitespace	MDI Common Utility Library routine used to remove trailing blanks from a stirng.
VALIDATE_DOLLAR_AMOUNT	VALIDATE_DOLLAR_AMOUNT is a bridge layer routine used to determine the validity of a dollar amount entered by the user.
XmListAddItem	X Library function that will add a character string to a scroll list.

Table 66 - Routines called by config_add_ok (parking)

4.2.1.3.38 VALIDATE_DOLLAR_AMOUNT

VALIDATE_DOLLAR_AMOUNT is a bridge layer routine used to determine the validity of a dollar amount entered by the user. The structure chart for VALIDATE_DOLLAR_AMOUNT is depicted in Figure 64. A description of the routines called by VALIDATE_DOLLAR_AMOUNT is provided in Table 67.

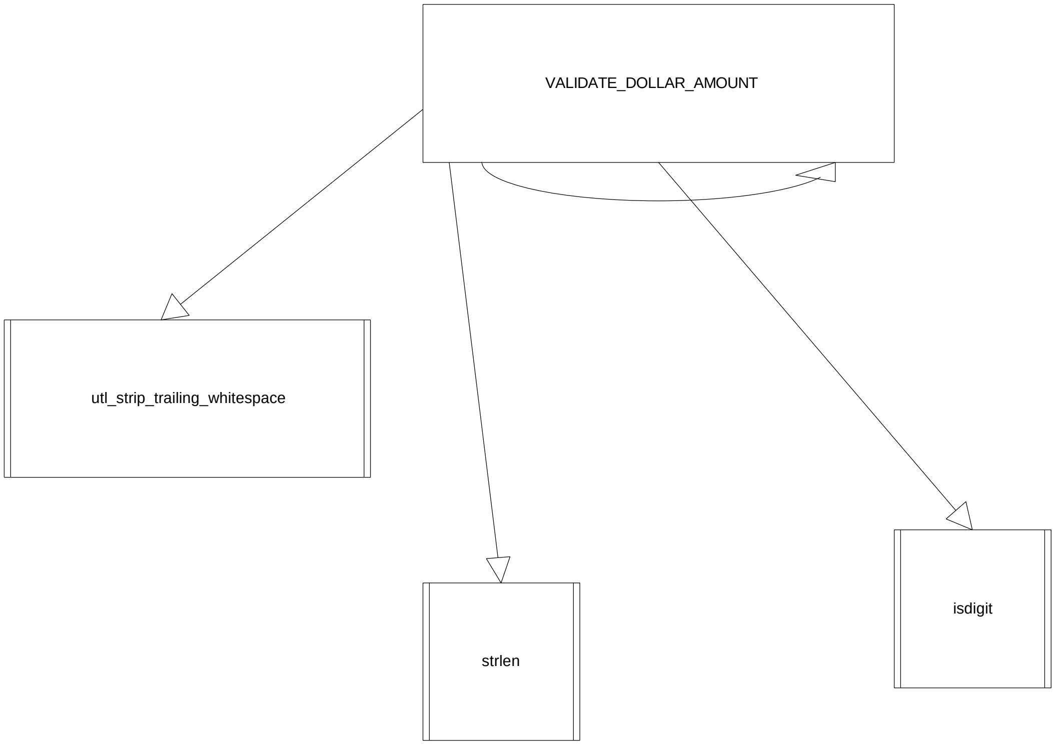

Figure 64 - VALIDATE_DOLLAR_AMOUNT Structure Chart

Function	Description
Isdigit	C Library Function that determines if the given value is a number.
Strlen	UNIX system call that computes the number of characters in a NULL terminated string.
utl_strip_trailing_whitespace	MDI Common Utility Library routine used to remove trailing blanks from a stirng.

Table 67 - Routines called by VALIDATE_DOLLAR_AMOUNT

4.2.1.3.39 ksmg_create_parking_config_file

ksmg_create_parking_config_file is used to create a new airport parking data file and write the blank file to the data server. The structure chart for ksmg_create_parking_config_file is depicted in Figure 65. A description of the routines called by ksmg_create_parking_config_file is provided in Table 68.

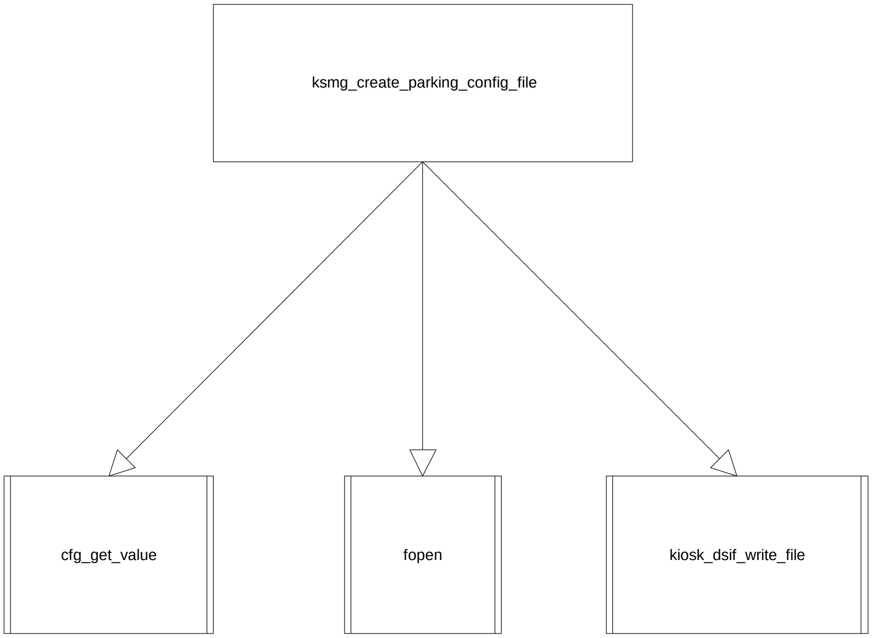

Figure 65 - ksmg_create_parking_config_file Structure Chart

Function	Description
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
fopen	C Library Function that opens the specified file using the specified access mode.
kiosk_dsif_write_file	kiosk_dsif_write_file packages the write file data into the appropriate kiosk_dsif message and sends it to the kiosk_dsif process.

Table 68 - Routines called by ksmg_create_parking_config_file

4.2.1.3.40 config_modify_ok (parking)

config_modify_ok (parking) is a GUI layer event triggered by the user selecting the ok button on the Parking Configuration Modify Entry Dialog. The structure chart for config_modify_ok (parking) is depicted in Figure 66. A description of the routines called by config_modify_ok (parking) is provided in Table 69.

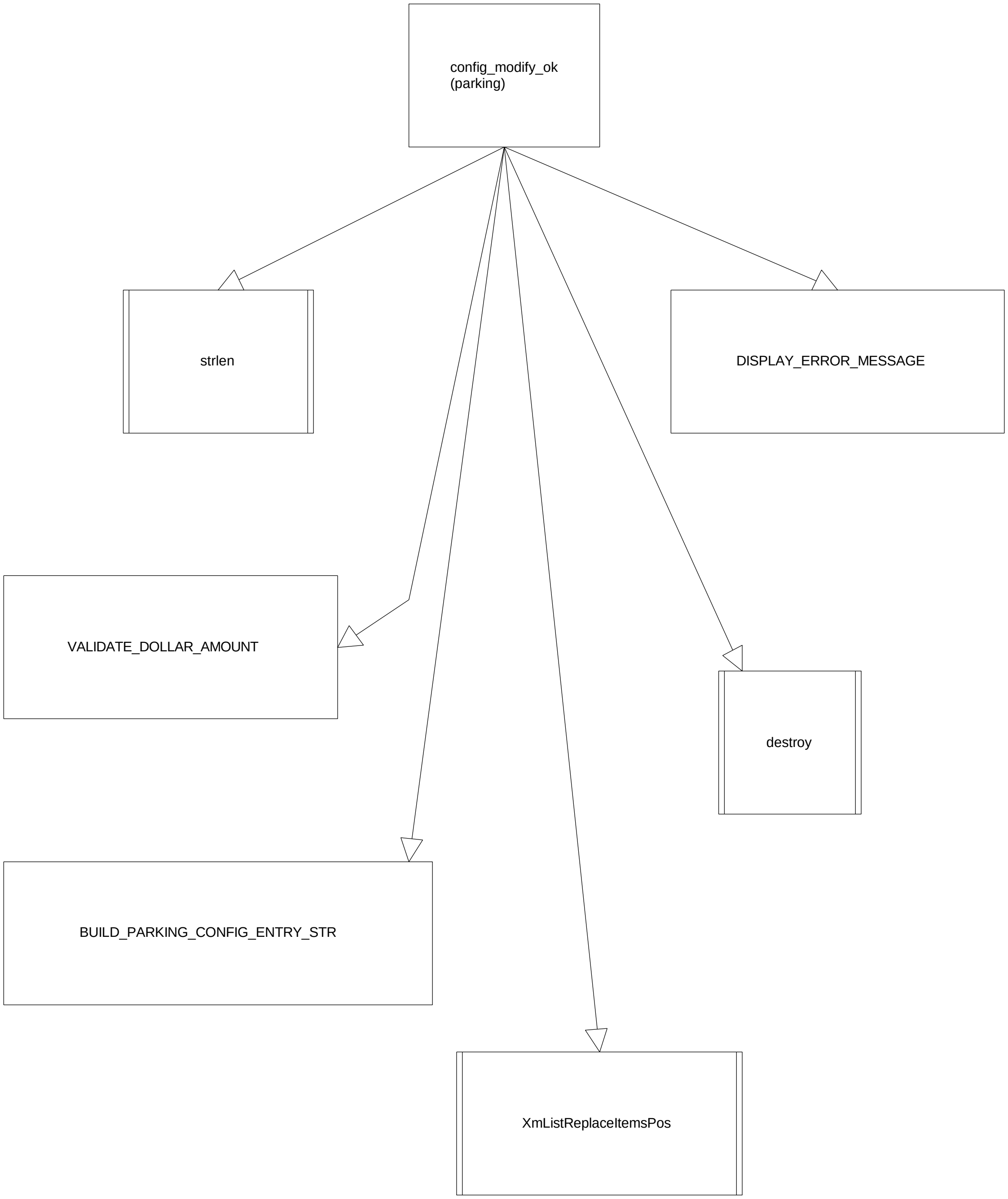

Figure 66 - config_modify_ok (parking) Structure Chart

Function	Description
BUILD_PARKING_CONFIG_ENTRY_STR	BUILD_PARKING_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list.
destroy	Function which will return allocated X resources back to the system.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
strlen	UNIX system call that computes the number of characters in a NULL terminated string.
VALIDATE_DOLLAR_AMOUNT	VALIDATE_DOLLAR_AMOUNT is a bridge layer routine used to determine the validity of a dollar amount entered by the user.
XmListReplaceItemsPos	X Library function that will replace the instance of one X string with another instance in a Scroll List.

Table 69 - Routines called by config_modify_ok (parking)

4.2.1.3.41 save_table (parking)

save_table is responsibile for creating the string list object that is used to write the configuration data to the file. The structure chart for save_table (parking) is depicted in Figure 67. A description of the routines called by save_table (parking) is provided in Table 70.

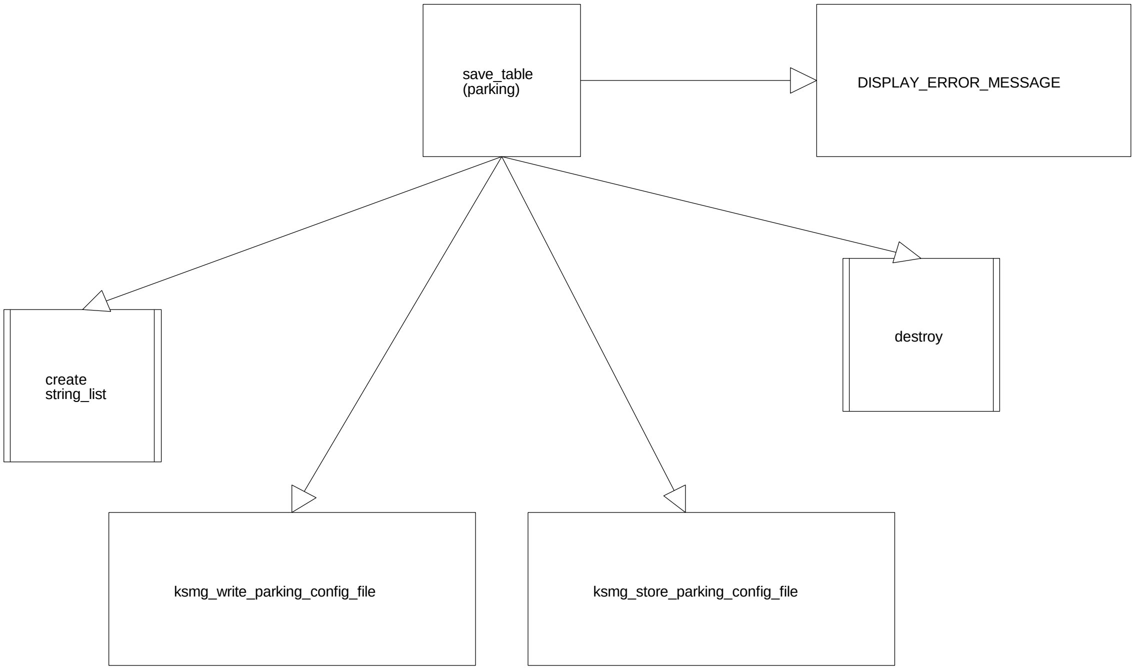

Figure 67 - save_table (parking) Structure Chart

Function	Description
create string_list	Function that creates a C character string from a series of X strings.
destroy	Function which will return allocated X resources back to the system.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
ksmg_store_parking_config_file	ksmg_store_parking_config_file is used to store the airport parking data file from the local file system to the data server.
ksmg_write_parking_config_file	ksmg_write_parking_config_file is used to write the list of airport parking lots displayed to the user to the airport parking lot data file.

Table 70 - Routines called by save_table (parking)

4.2.1.3.42 ksmg_write_parking_config_file

ksmg_write_parking_config_file is used to write the list of airport parking lots displayed to the user to the airport parking lot data file. The structure chart for ksmg_write_parking_config_file is depicted in Figure 68. A description of the routines called by ksmg_write_parking_config_file is provided in Table 71.

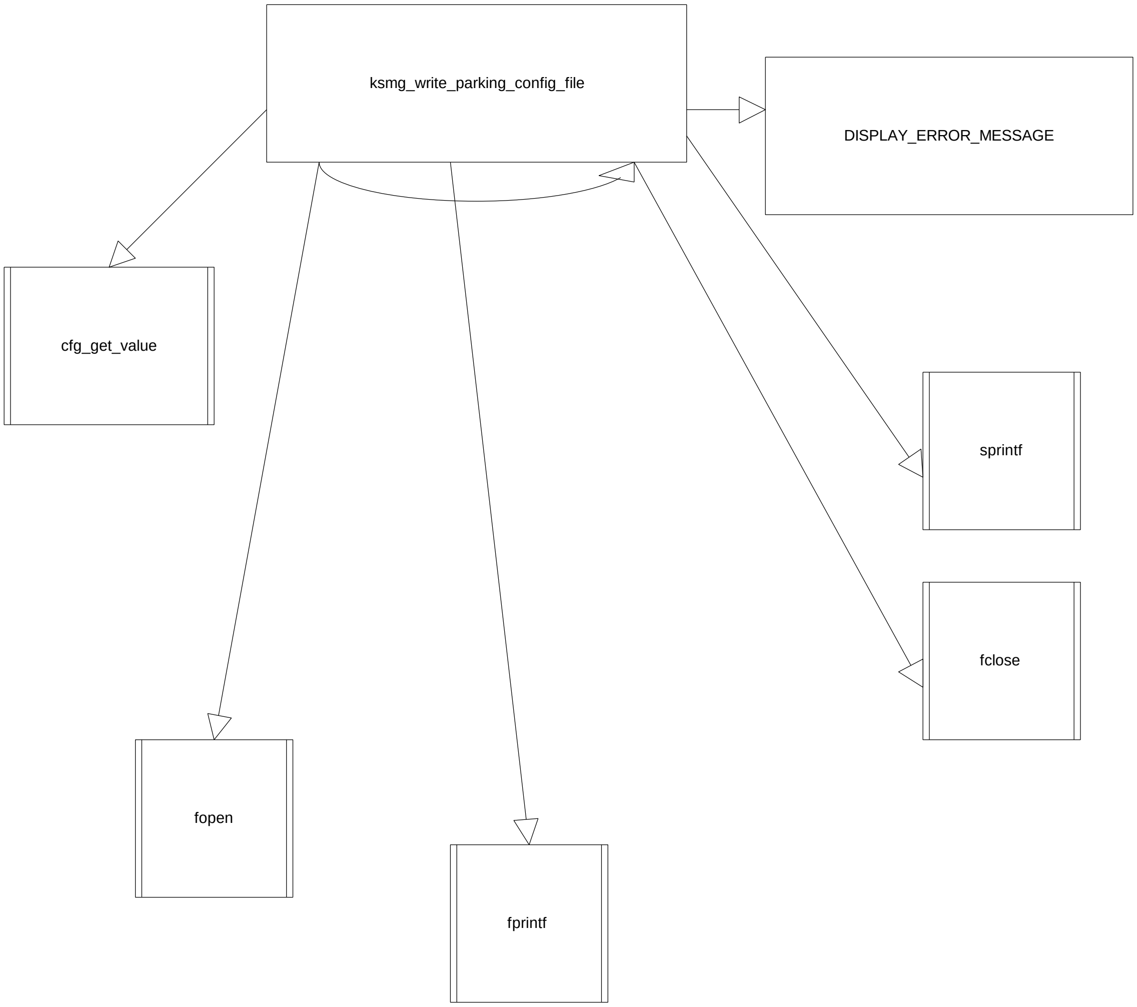

Figure 68 - ksmg_write_parking_config_file Structure Chart

Function	Description
cfg_get_vaue	MDI Configuration File routine used to return the value of the specified configuration name.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
fclose	C Library Function used to close an open file.
fopen	C Library Function that opens the specified file using the specified access mode.
fprintf	UNIX system call to print formatted data to a file stream.
sprintf	C Library Function that provides printf capabilities to a character string.

Table 71 - Routines called by ksmg_write_parking_config_file

4.2.1.3.43 ksmg_store_parking_config_file

ksmg_store_parking_config_file is used to store the airport parking data file from the local file system to the data server. The structure chart for ksmg_store_parking_config_file is depicted in Figure 69. A description of the routines called by ksmg_store_parking_config_file is provided in Table 72.

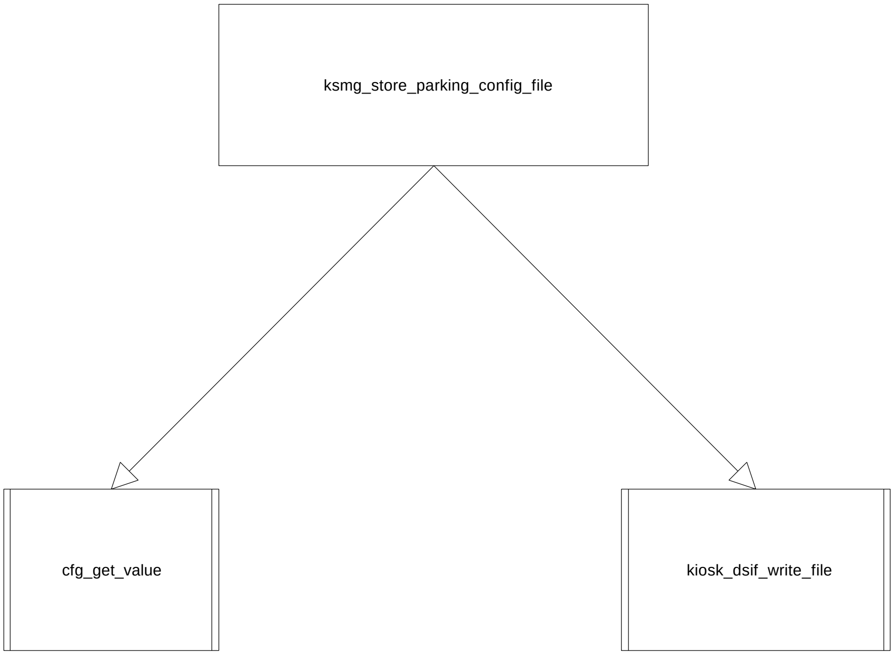

Figure 69 - ksmg_store_parking_config_file Structure Chart

Function	Description
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
kiosk_dsif_write_file	kiosk_dsif_write_file packages the write file data into the appropriate kiosk_dsif message and sends it to the kiosk_dsif process.

Table 72 - Routines called by ksmg_store_parking_config_file

4.2.1.3.44 configure_rental_car

configure_rental_car is the GUI layer event attached to the rental car configuration table menu item. The button is selected when the user desires to modify the contents of the rental car agency configuration file. The structure chart for configure_rental_car is depicted in Figure 70. A description of the routines called by configure_rental_car is provided in Table 73.

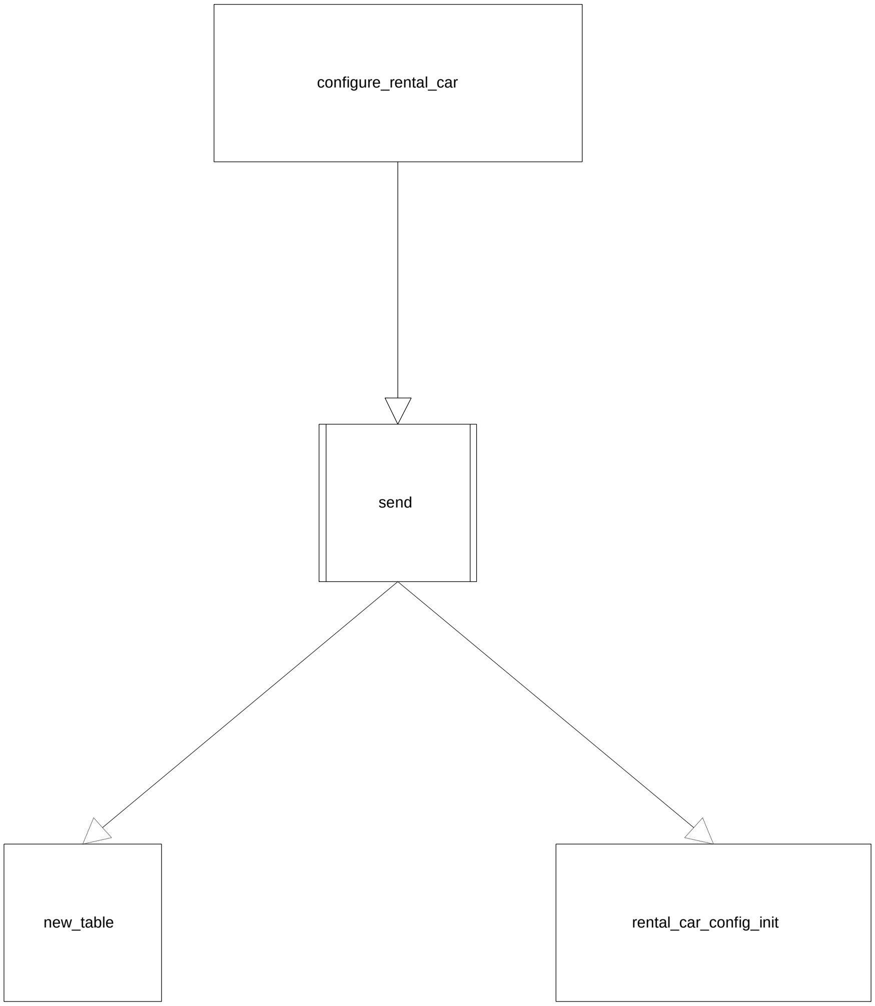

Figure 70 - configure_rental_car Structure Chart

Function	Description
new_table	new_table is a GUI layer event triggered when a new table is selected for editing. Each D module contains this event and if the D module is associated with the table currently being edited a check is make to see if the current table should be saved. If so, the user is asked whether or not the table should be saved. The sensitivity of the table menu item of the current table is made sensitive and the form containing the current table is unmanaged.
rental_car_config_init	rental_car_config_init is the GUI layer event used to initialize the user interface when a table (data file) is initially selected for modification. This includes clearing the list of items to be displayed, allowing additions to be made, reading the data file, and creating the data file if it doesn't exist.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.

Table 73 - Routines called by configure_rental_car

4.2.1.3.45 rental_car_config_init

rental_car_config_init is the GUI layer event used to initialize the user interface when a table (data file) is initially selected for modification. This includes clearing the list of items to be displayed, allowing additions to be made, reading the data file, and creating the data file if it doesn't exist. The structure chart for rental_car_config_init is depicted in Figure 71. A description of the routines called by rental_car_config_init is provided in Table 74.

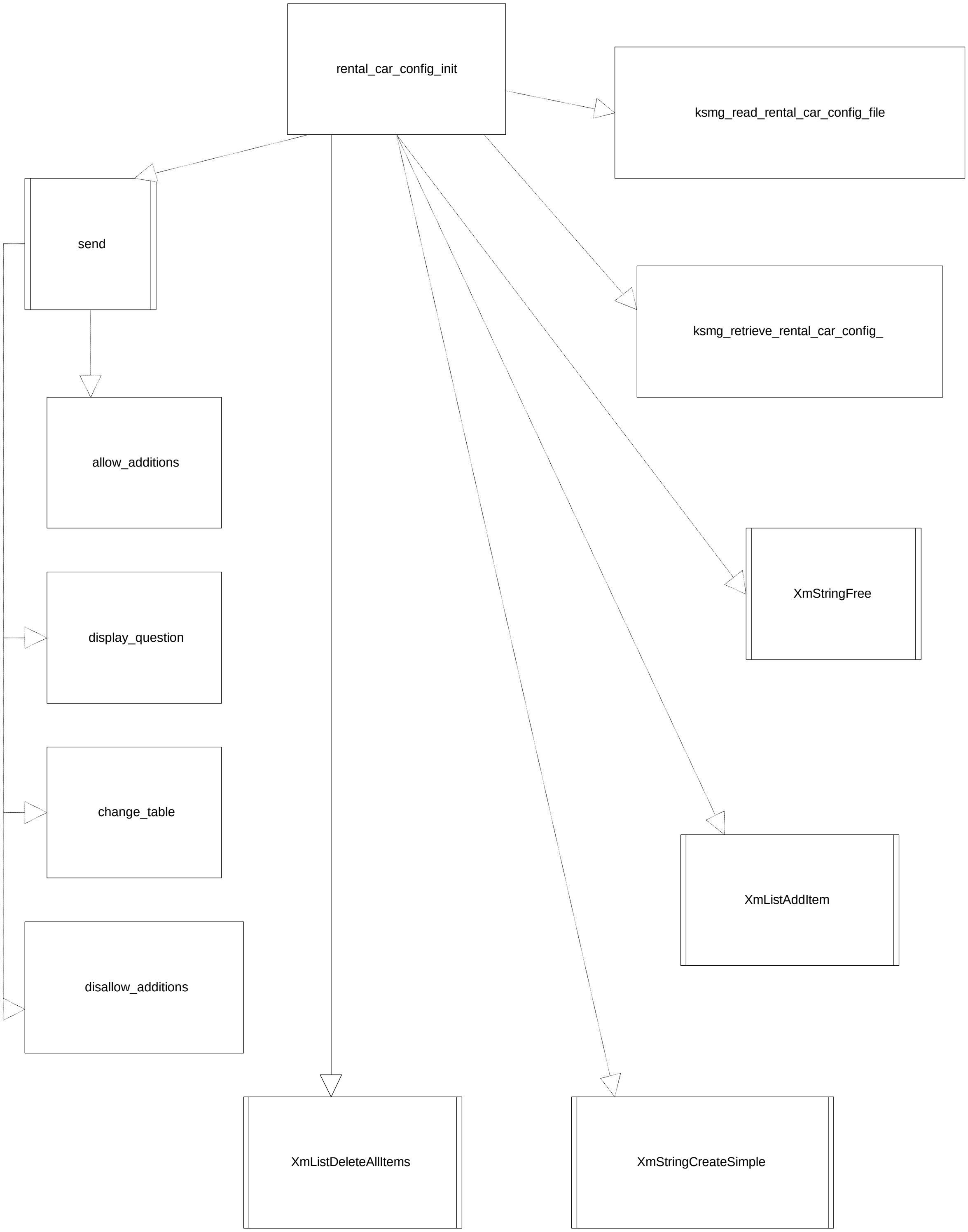

Figure 71 - rental_car_config_init Structure Chart

Function	Description
allow_additions	allow_additions is a GUI layer event in charge of setting the sensitivity for the Add button to true and desensitizing the delete and modify buttons.
change_table	change_table is a GUI layer event responsible for unmapping the current configuration data form and sensitizing the table submenu item associated with the current configuration table.
disallow_additions	disallow_additions is the GUI layer routine that sets the sensitivity of the add menu item so that no additions can be made until a table is selected.
display_question	display_question is a GUI layer event that creates a question dialog box to display the specified question to the user.
ksmg_read_rental_car_config_file	ksmg_read_rental_car_config_file is used to read the contents of the rental car agency data file and create the list of rental car agencies to be displayed to the user.
ksmg_retrieve_rental_car_config_	ksmg_retrieve_rental_car_config_file is used to retrieve the rental car agency data file from the data server and place it on the local file system.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
XmListAddItem	X Library function that will add a character string to a scroll list.
XmListDeleteAllItems	X Library function that will delete all items from a scroll list.
XmStringCreateSimple	X Library function that will create a simple X string from a character string.
XmStringFree	X Library function that frees a previously allocated X string.

Table 74 - Routines called by rental_car_config_init

4.2.1.3.46 ksmg_read_rental_car_config_file

ksmg_read_rental_car_config_file is used to read the contents of the rental car agency data file and create the list of rental car agencies to be displayed to the user. The structure chart for ksmg_read_rental_car_config_file is depicted in Figure 72. A description of the routines called by ksmg_read_rental_car_config_file is provided in Table 75.

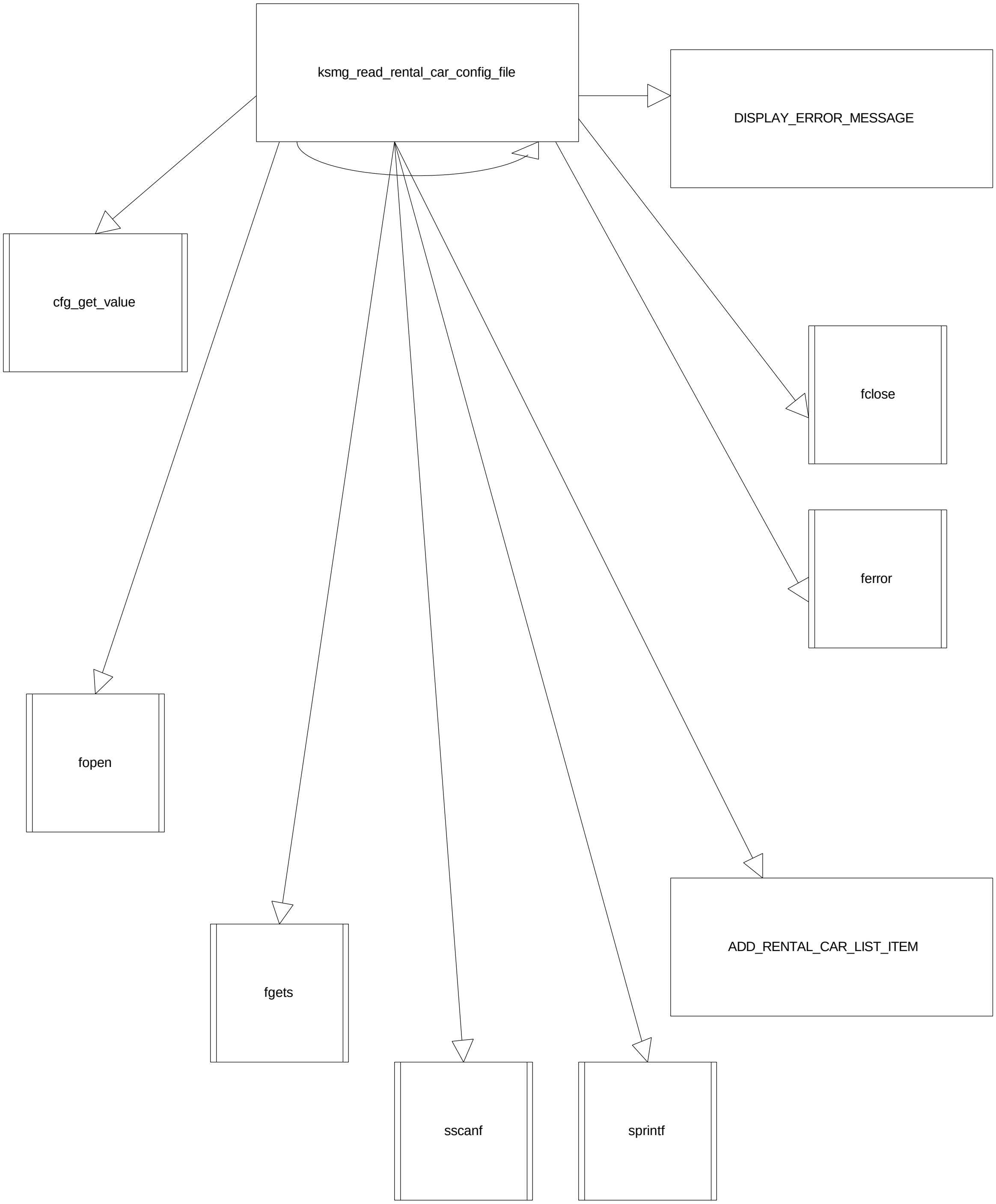

Figure 72 - ksmg_read_rental_car_config_file Structure Chart

Function	Description
ADD_RENTAL_CAR_LIST_ITEM	ADD_RENTAL_CAR_LIST_ITEM is a bridge layer routine that creates and dispatches the GUI layer event responsible for adding an entry to the rental car agency list.
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
fclose	C Library Function used to close an open file.
ferror	C Library Function that returns any previous errors on the associated stream.
fgets	C Library Function used to read a line of text from a file.
fopen	C Library Function that opens the specified file using the specified access mode.
sprintf	C Library Function that provides printf capabilities to a character string.
sscanf	UNIX function call that will allow formatted "input" from a NULL terminated character string.

Table 75 - Routines called by ksmg_read_rental_car_config_file

4.2.1.3.47 ADD_RENTAL_CAR_LIST_ITEM

ADD_RENTAL_CAR_LIST_ITEM is a bridge layer routine that creates and dispatches the GUI layer event responsible for adding an entry to the rental car agency list. The structure chart for ADD_RENTAL_CAR_LIST_ITEM is depicted in Figure 73. A description of the routines called by ADD_RENTAL_CAR_LIST_ITEM is provided in Table 76.

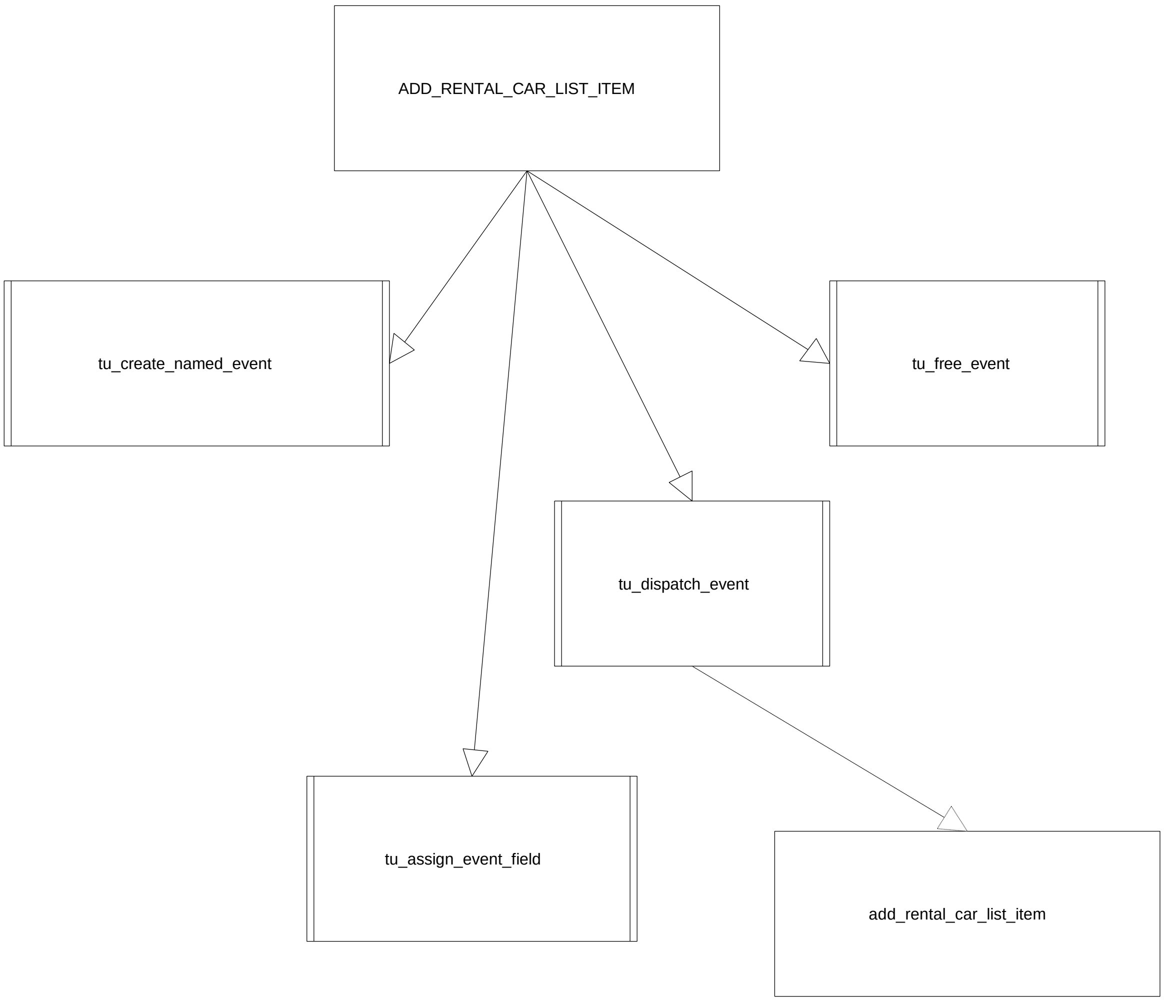

Figure 73 - ADD_RENTAL_CAR_LIST_ITEM Structure Chart

Function	Description
add_rental_car_list_item	add_rental_car_list_item is a GUI layer event that creates an item in the rental car agency list using the data specified for the event.
tu_assign_event_field	TeleUSE Library Function used to associate the contents of a C variable with the contents of an event attribute.
tu_create_named_event	TeleUSE Library Function used to create the data structure necessary to interface the C code with the D event code.
tu_dispatch_event	TeleUSE Library Function used to dispatch the created event. This causes the event to be executed.
tu_free_event	TeleUSE Library Function used to free up any memory that was allocated to the event data structure using tu_create_named_event.

Table 76 - Routines called by ADD_RENTAL_CAR_LIST_ITEM

4.2.1.3.48 ksmg_retrieve_rental_car_config

ksmg_retrieve_rental_car_config_file is used to retrieve the rental car agency data file from the data server and place it on the local file system. The structure chart for ksmg_retrieve_rental_car_config is depicted in Figure 74. A description of the routines called by ksmg_retrieve_rental_car_config is provided in Table 77.

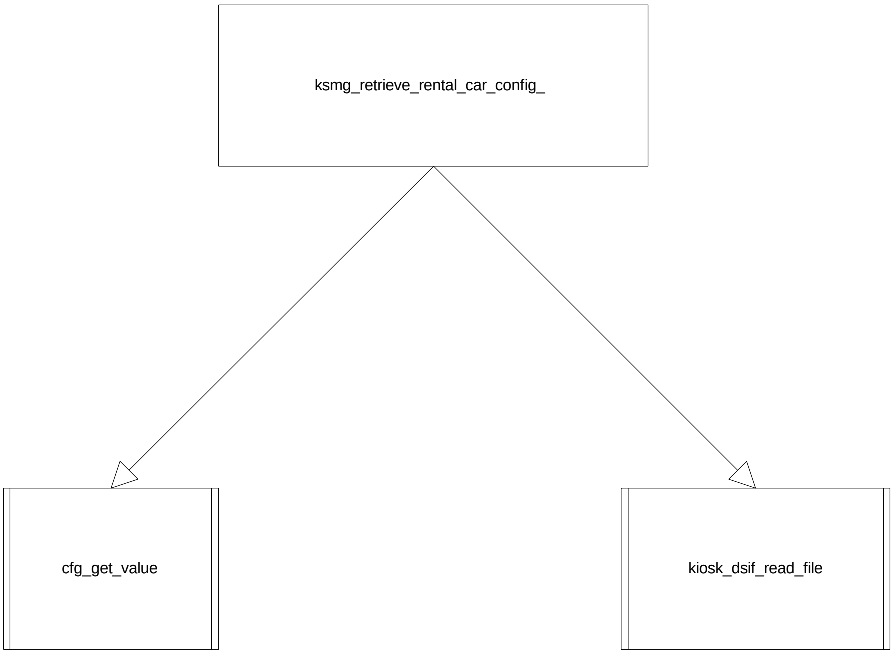

Figure 74 - ksmg_retrieve_rental_car_config Structure Chart

Function	Description
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
kiosk_dsif_read_file	kiosk_dsif_read_file packages the read file request into the appropriate kiosk_dsif message and sends it to the kiosk_dsif process.

Table 77 - Routines called by ksmg_retrieve_rental_car_config

4.2.1.3.49 config_add_ok (rental)

config_add_ok (rental) is a GUI layer event triggered by the user selecting the ok button on the Rental Car Configuration Add Entry Dialog. The structure chart for config_add_ok (rental) is depicted in Figure 75. A description of the routines called by config_add_ok (rental) is provided in Table 78.

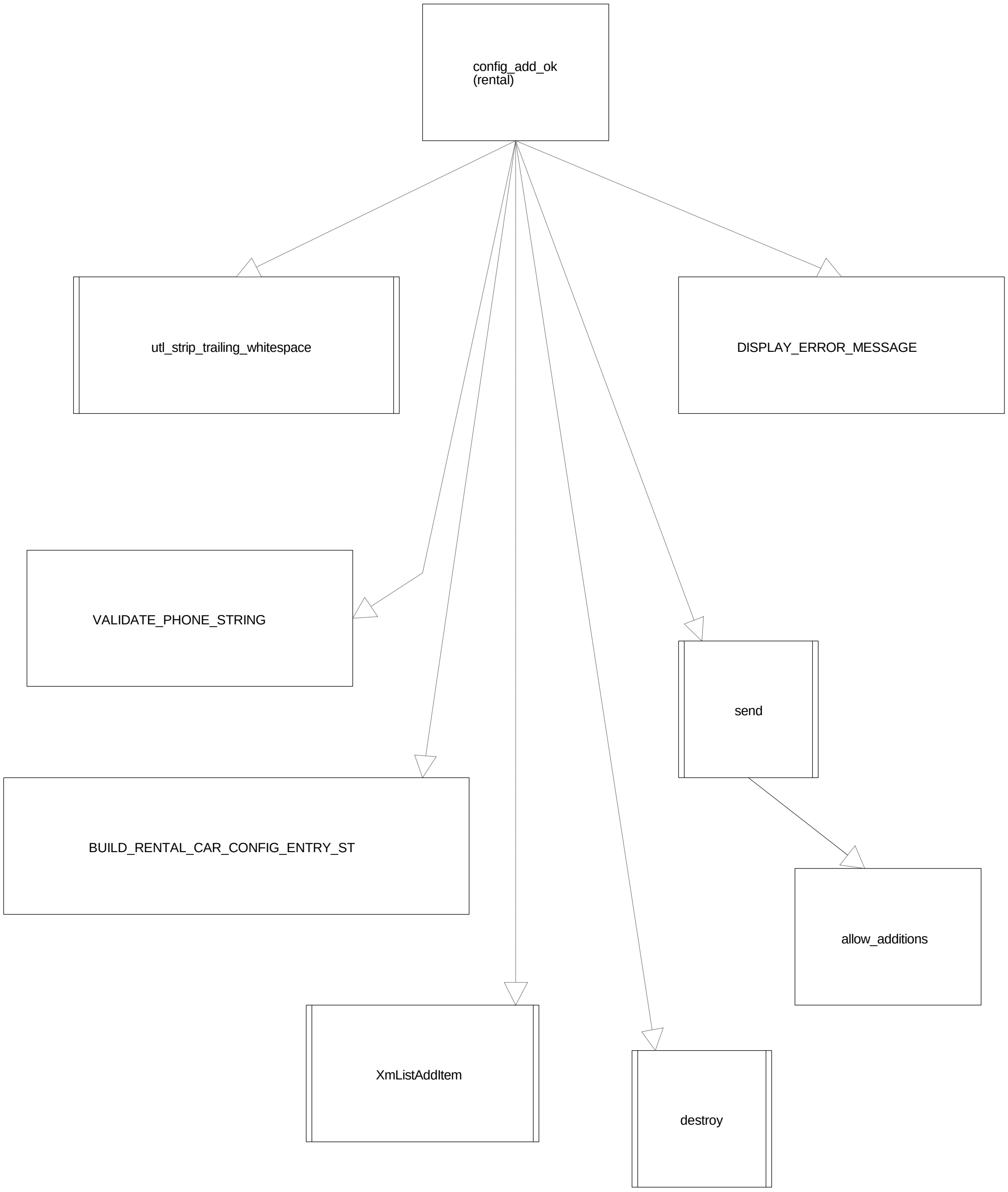

Figure 75 - config_add_ok (rental) Structure Chart

Function	Description
allow_additions	allow_additions is a GUI layer event in charge of setting the sensitivity for the Add button to true and desensitizing the delete and modify buttons.
BUILD_RENTAL_CAR_CONFIG_ENTRY_ST	BUILD_RENTAL_CAR_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list. All text strings are left justified within the appropriate fields of the entry string.
destroy	Function which will return allocated X resources back to the system.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
utl_strip_trailing_whitespace	MDI Common Utility Library routine used to remove trailing blanks from a string.
VALIDATE_PHONE_STRING	VALIDATE_PHONE_STRING is a bridge layer routine used to determine the validity of a telephone number entered by the user. Accepted strings are in the form of xxx-xxxx or x-xxx-xxx-xxxx.
XmListAddItem	X Library function that will add a character string to a scroll list.

Table 78 - Routines called by config_add_ok (rental)

4.2.1.3.50 ksmg_create_rental_car_config_file

ksmg_create_rental_car_config_file is used to create a new rental car agency data file and write the blank file to the data server. The structure chart for ksmg_create_rental_car_config_file is depicted in Figure 76. A description of the routines called by ksmg_create_rental_car_config_file is provided in Table 79.

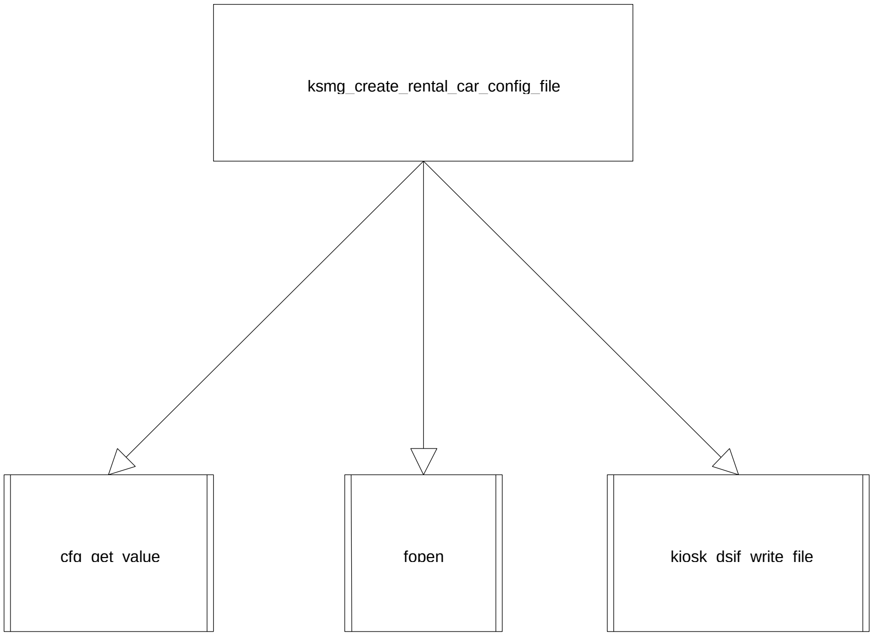

Figure 76 - ksmg_create_rental_car_config_file Structure Chart

Function	Description
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
fopen	C Library Function that opens the specified file using the specified access mode.
kiosk_dsif_write_file	kiosk_dsif_write_file packages the write file data into the appropriate kiosk_dsif message and sends it to the kiosk_dsif process.

Table 79 - Routines called by ksmg_create_rental_car_config_file

4.2.1.3.51 config_modify_ok (rental)

config_modify_ok (rental) is a GUI layer event triggered by the user selecting the ok button on the Rental Car Configuration Modify Entry Dialog. The structure chart for config_modify_ok (rental) is depicted in Figure 77. A description of the routines called by config_modify_ok (rental) is provided in Table 80.

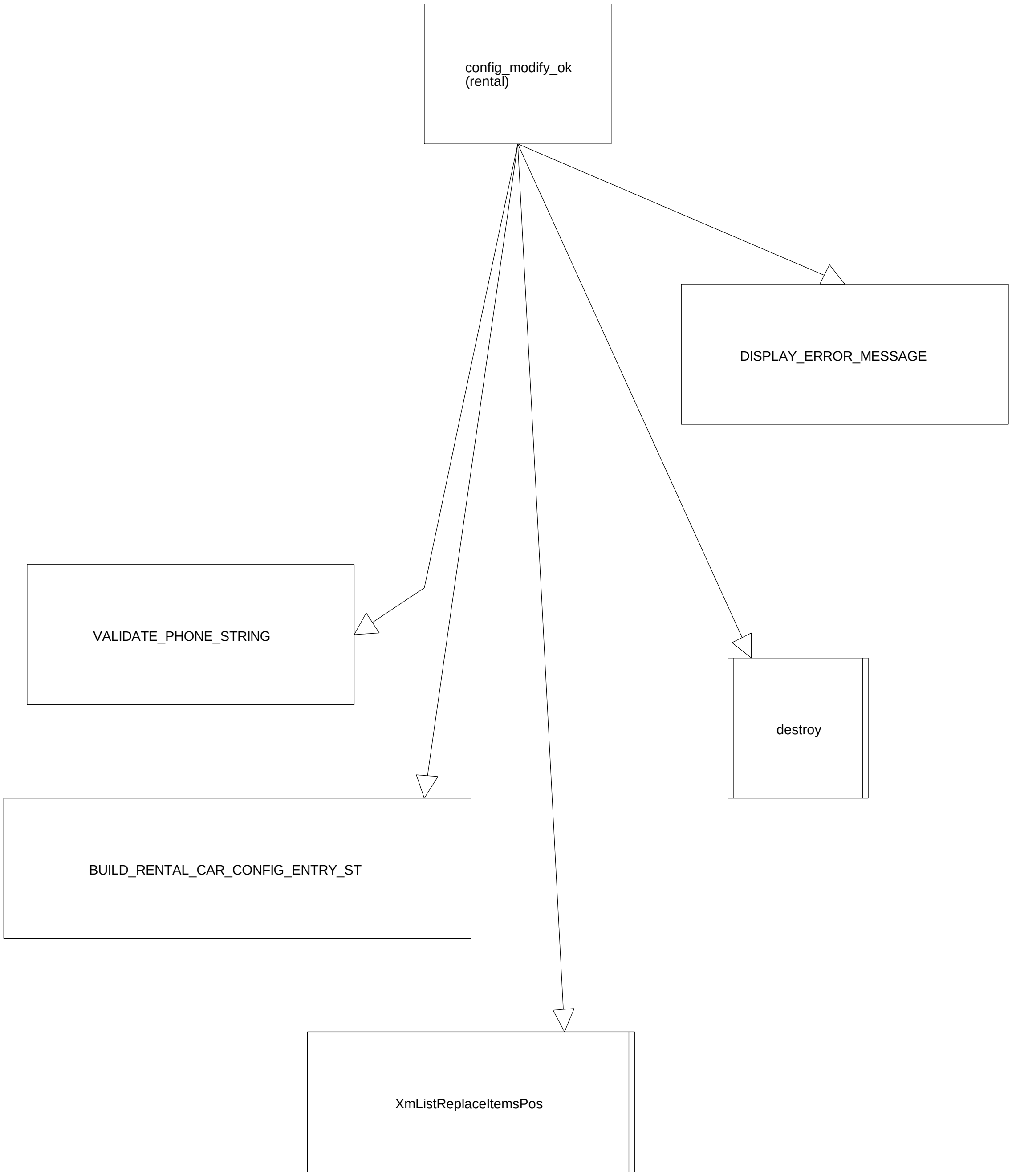

Figure 77 - config_modify_ok (rental) Structure Chart

Function	Description
BUILD_RENTAL_CAR_CONFIG_ENTRY_ST	BUILD_RENTAL_CAR_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list. All text strings are left justified within the appropriate fields of the entry string.
destroy	Function which will return allocated X resources back to the system.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
VALIDATE_PHONE_STRING	VALIDATE_PHONE_STRING is a bridge layer routine used to determine the validity of a telephone number entered by the user. Accepted strings are in the form of xxx-xxxx or x-xxx-xxx-xxxx.
XmListReplaceItemsPos	X Library function that will replace the instance of one X string with another instance in a Scroll List.

Table 80 - Routines called by config_modify_ok (rental)

4.2.1.3.52 save_table (rental)

save_table is responsibile for creating the string list object that is used to write the configuration data to the file. The structure chart for save_table (rental) is depicted in Figure 78. A description of the routines called by save_table (rental) is provided in Table 81.

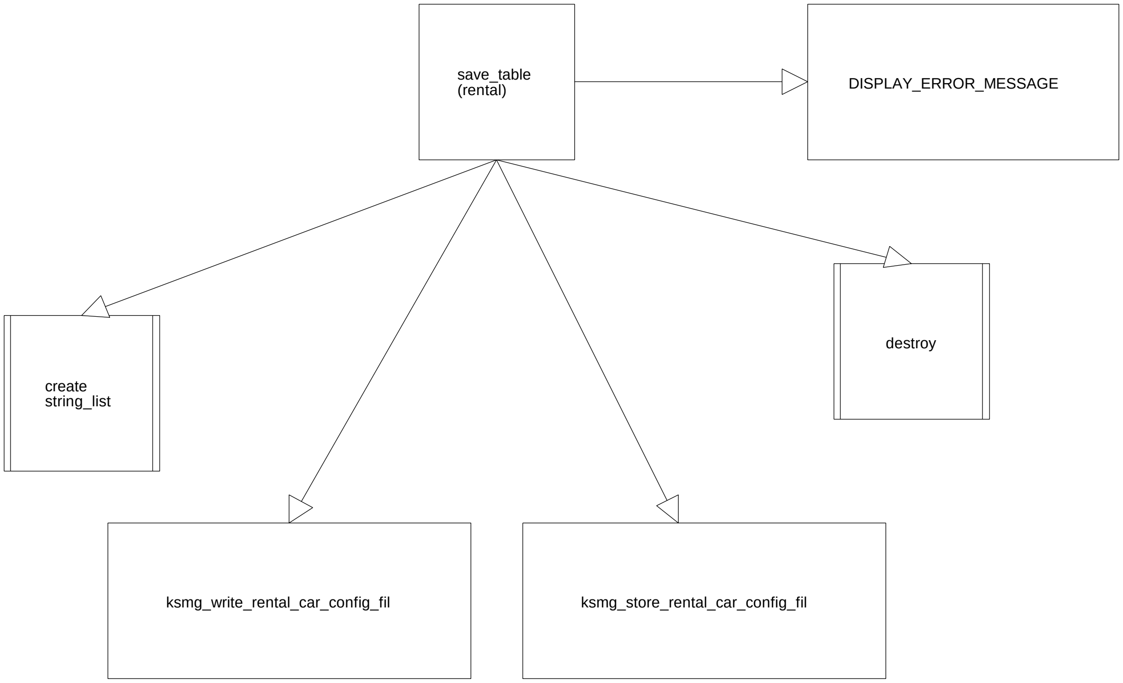
save_table
(rental)
DISPLAY_ERROR_MESSAGE
create
string_list
destroy
ksmg_write_rental_car_config_fil
ksmg_store_rental_car_config_fil

Figure 78 - save_table (rental) Structure Chart

Function	Description
create string_list	Function that creates a C character string from a series of X strings.
destroy	Function which will return allocated X resources back to the system.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
ksmg_store_rental_car_config_fil	ksmg_store_rental_car_config_file is used to store the rental car agency data file from the local file system to the data server.
ksmg_write_rental_car_config_fil	ksmg_write_rental_car_config_file is used to write the list of rental car agencies displayed to the user to the rental car agency data file.

Table 81 - Routines called by save_table (rental)

4.2.1.3.53 ksmg_write_rental_car_config_fil

ksmg_write_rental_car_config_file is used to write the list of rental car agencies displayed to the user to the rental car agency data file. The structure chart for ksmg_write_rental_car_config_fil is depicted in Figure 79. A description of the routines called by ksmg_write_rental_car_config_fil is provided in Table 82.

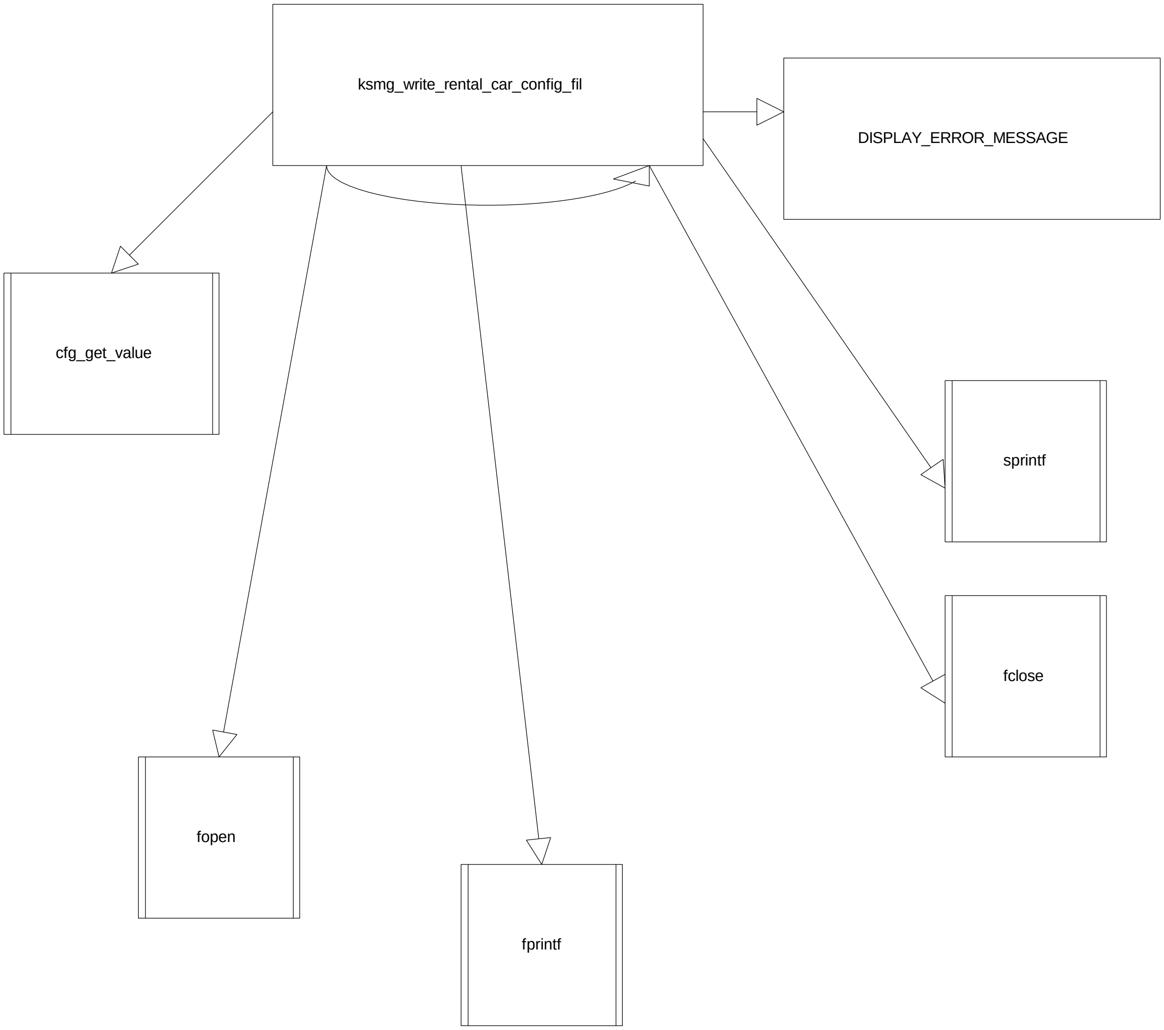

Figure 79 - ksmg_write_rental_car_config_fil Structure Chart

Function	Description
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
fclose	C Library Function used to close an open file.
fopen	C Library Function that opens the specified file using the specified access mode.
fprintf	UNIX system call to print formatted data to a file stream.
sprintf	C Library Function that provides printf capabilities to a character string.

Table 82 - Routines called by ksmg_write_rental_car_config_fil

4.2.1.3.54 ksmg_store_rental_car_config_fil

ksmg_store_rental_car_config_file is used to store the rental car agency data file from the local file system to the data server. The structure chart for ksmg_store_rental_car_config_fil is depicted in Figure 80. A description of the routines called by ksmg_store_rental_car_config_fil is provided in Table 83.

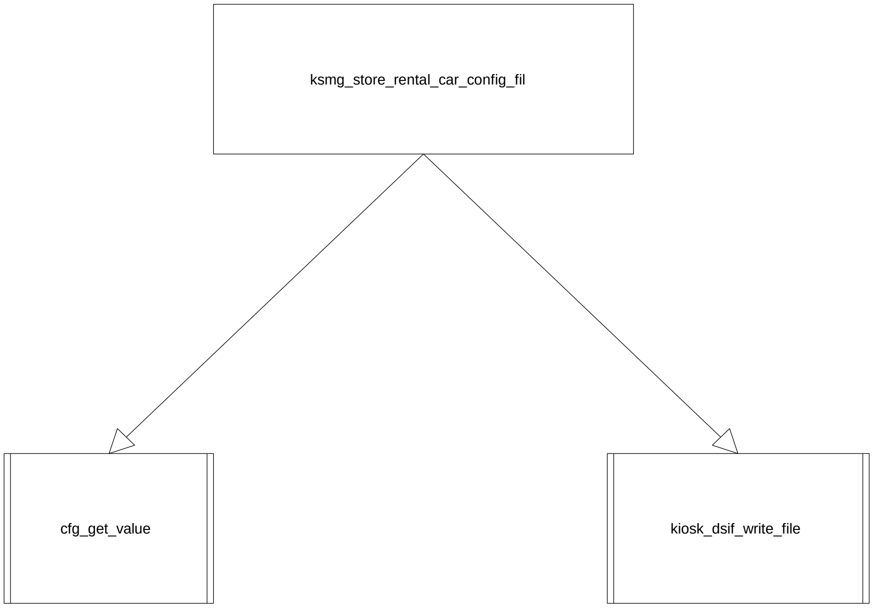

Figure 80 - ksmg_store_rental_car_config_fil Structure Chart

Function	Description
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
kiosk_dsif_write_file	kiosk_dsif_write_file packages the write file data into the appropriate kiosk_dsif message and sends it to the kiosk_dsif process.

Table 83 - Routines called by ksmg_store_rental_car_config_fil

4.2.1.3.55 configure_screen_saver

configure_screen_saver is the GUI layer event attached to the screen saver configuration table menu item. The button is selected when the user desires to modify the contents of the screen saver configuration file. The structure chart for configure_screen_saver is depicted in Figure 81. A description of the routines called by configure_screen_saver is provided in Table 84.

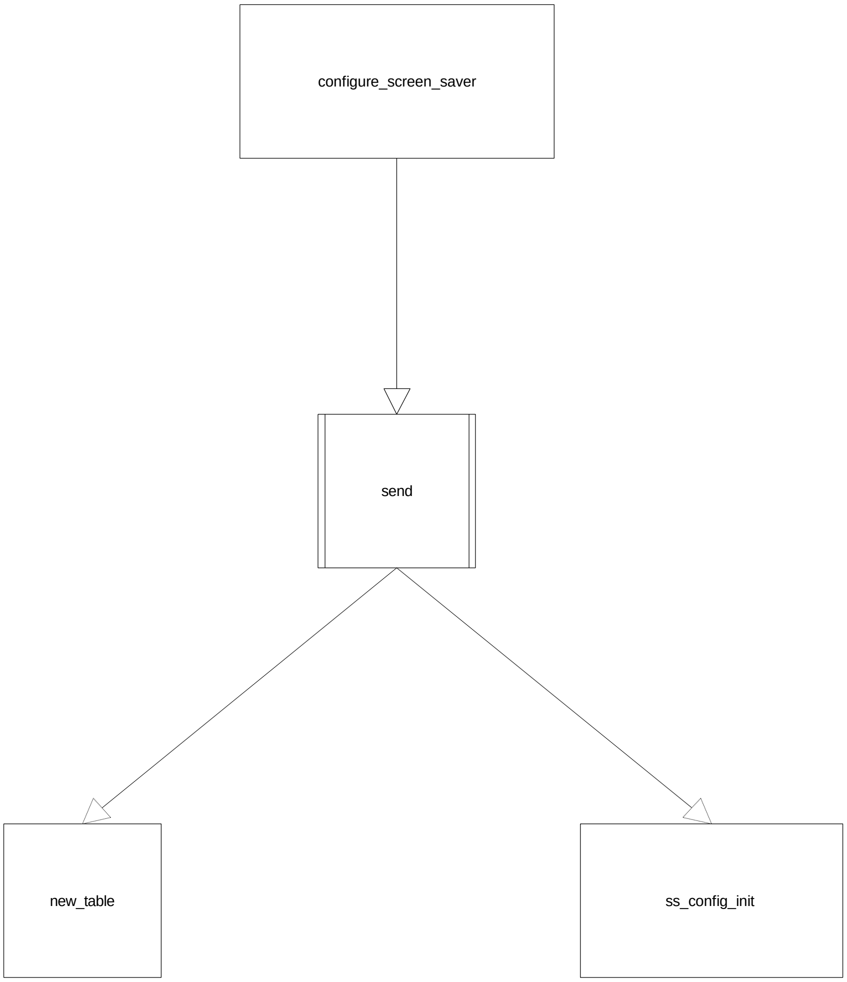

Figure 81 - configure_screen_saver Structure Chart

Function	Description
new_table	new_table is a GUI layer event triggered when a new table is selected for editing. Each D module contains this event and if the D module is associated with the table currently being edited a check is make to see if the current table should be saved. If so, the user is asked whether or not the table should be saved. The sensitivity of the table menu item of the current table is made sensitive and the form containing the current table is unmanaged.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
ss_config_init	ss_config_init is the GUI layer event used to initialize the user interface when a table (data file) is initially selected for modification. This includes clearing the list of items to be displayed, allowing additions to be made, reading the data file, and creating the data file if it doesn't exist.

Table 84 - Routines called by configure_screen_saver

4.2.1.3.56 ss_config_init

ss_config_init is the GUI layer event used to initialize the user interface when a table (data file) is initially selected for modification. This includes clearing the list of items to be displayed, allowing additions to be made, reading the data file, and creating the data file if it doesn't exist. The structure chart for ss_config_init is depicted in Figure 82. A description of the routines called by ss_config_init is provided in Table 85.

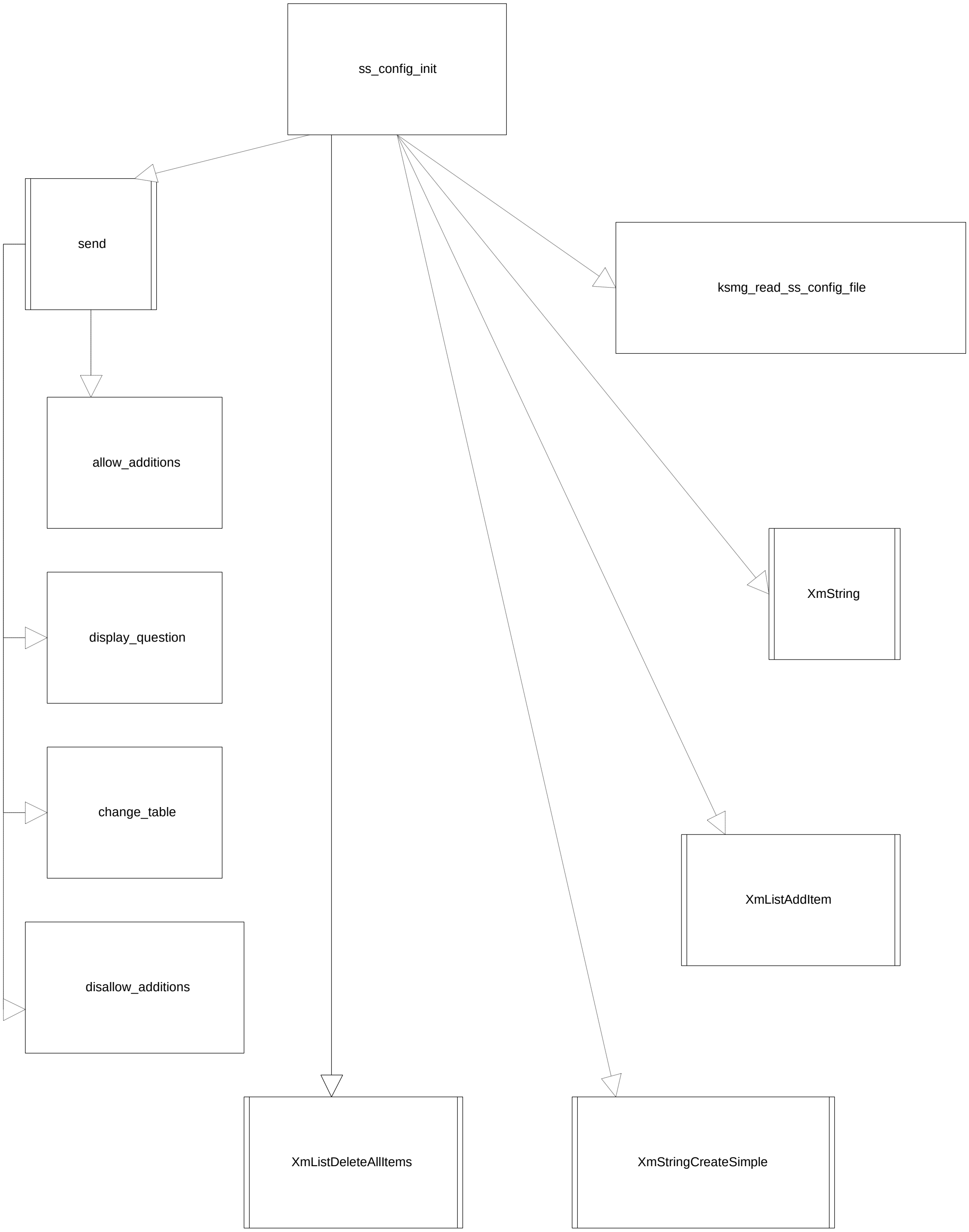

Figure 82 - ss_config_init Structure Chart

Function	Description
allow_additions	allow_additions is a GUI layer event in charge of setting the sensitivity for the Add button to true and desensitizing the delete and modify buttons.
change_table	change_table is a GUI layer event responsible for unmapping the current configuration data form and sensitizing the table submenu item associated with the current configuration table.
disallow_additions	disallow_additions is the GUI layer routine that sets the sensitivity of the add menu item so that no additions can be made until a table is selected.
display_question	display_question is a GUI layer event that creates a question dialog box to display the specified question to the user.
ksmg_read_ss_config_file	ksmg_read_ss_config_file is used to read the contents of the screen saver data file and create the list of the screen saver files to be displayed to the user.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
XmListAddItem	X Library function that will add a character string to a scroll list.
XmListDeleteAllItems	X Library function that will delete all items from a scroll list.
XmString	X Library function that will create an X string from a character string.
XmStringCreateSimple	X Library function that will create a simple X string from a character string.

Table 85 - Routines called by ss_config_init

4.2.1.3.57 ksmg_read_ss_config_file

ksmg_read_ss_config_file is used to read the contents of the screen saver data file and create the list of the screen saver files to be displayed to the user. The structure chart for ksmg_read_ss_config_file is depicted in Figure 83. A description of the routines called by ksmg_read_ss_config_file is provided in Table 86.

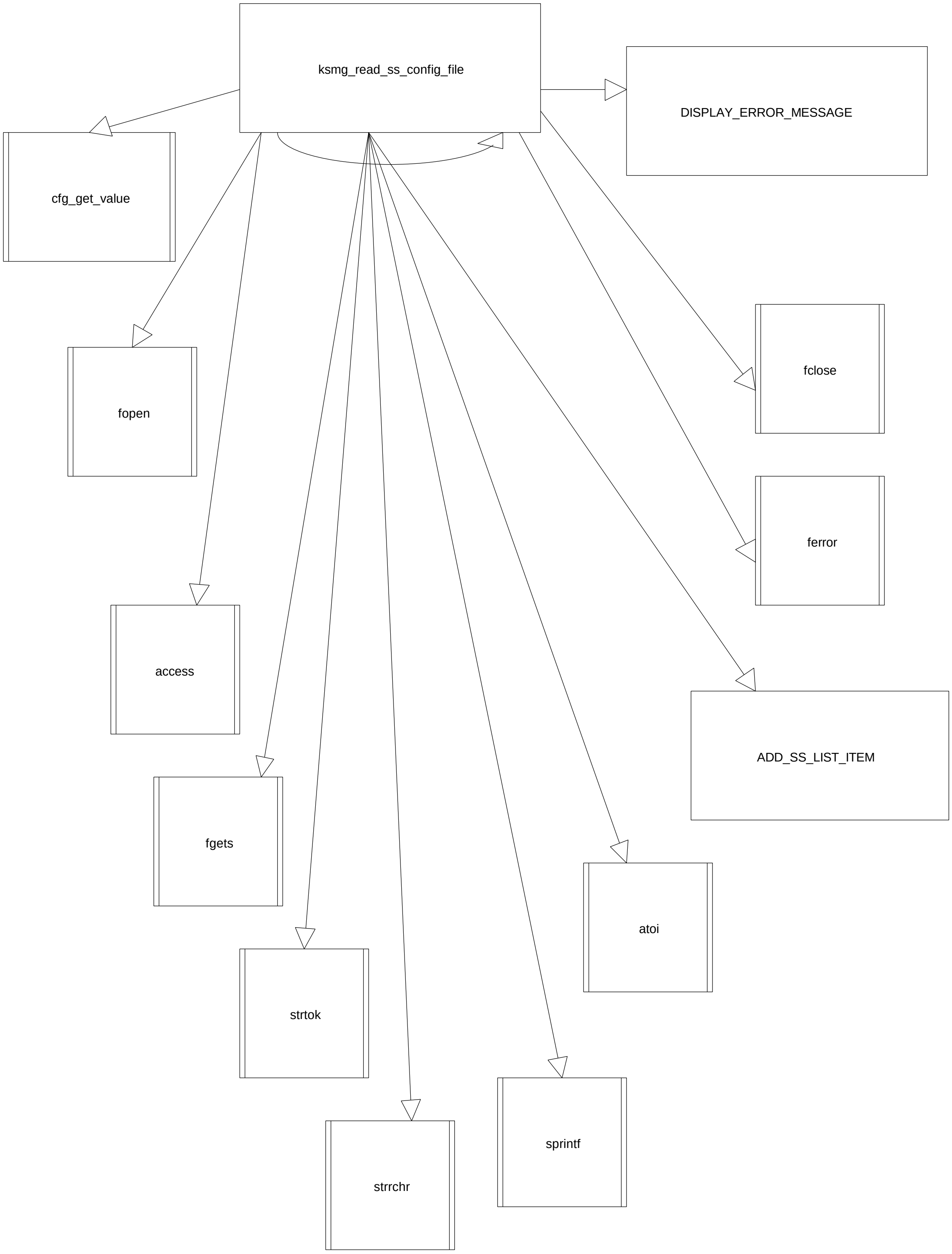

Figure 83 - ksmg_read_ss_config_file Structure Chart

Function	Description
access	UNIX Library function that tests the UNIX access rights for a specific file.
ADD_SS_LIST_ITEM	ADD_SS_LIST_ITEM is a bridge layer routine that creates and dispatches the GUI layer event responsible for adding an entry to the screen saver list.
atoi	C Library Function to convert an ASCII string to an integer value.
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
fclose	C Library Function used to close an open file.
ferror	C Library Function that returns any previous errors on the associated stream.
fgets	C Library Function used to read a line of text from a file.
fopen	C Library Function that opens the specified file using the specified access mode.
sprintf	C Library Function that provides printf capabilities to a character string.
strrchr	C Library Function that provides the capability to find the first occurence of the specified character in the given string.
strtok	C Library Function used to break the specified string into a sequence of tokens.

Table 86 - Routines called by ksmg_read_ss_config_file

4.2.1.3.58 ADD_SS_LIST_ITEM

ADD_SS_LIST_ITEM is a bridge layer routine that creates and dispatches the GUI layer event responsible for adding an entry to the screen saver list. The structure chart for ADD_SS_LIST_ITEM is depicted in Figure 84. A description of the routines called by ADD_SS_LIST_ITEM is provided in Table 87.

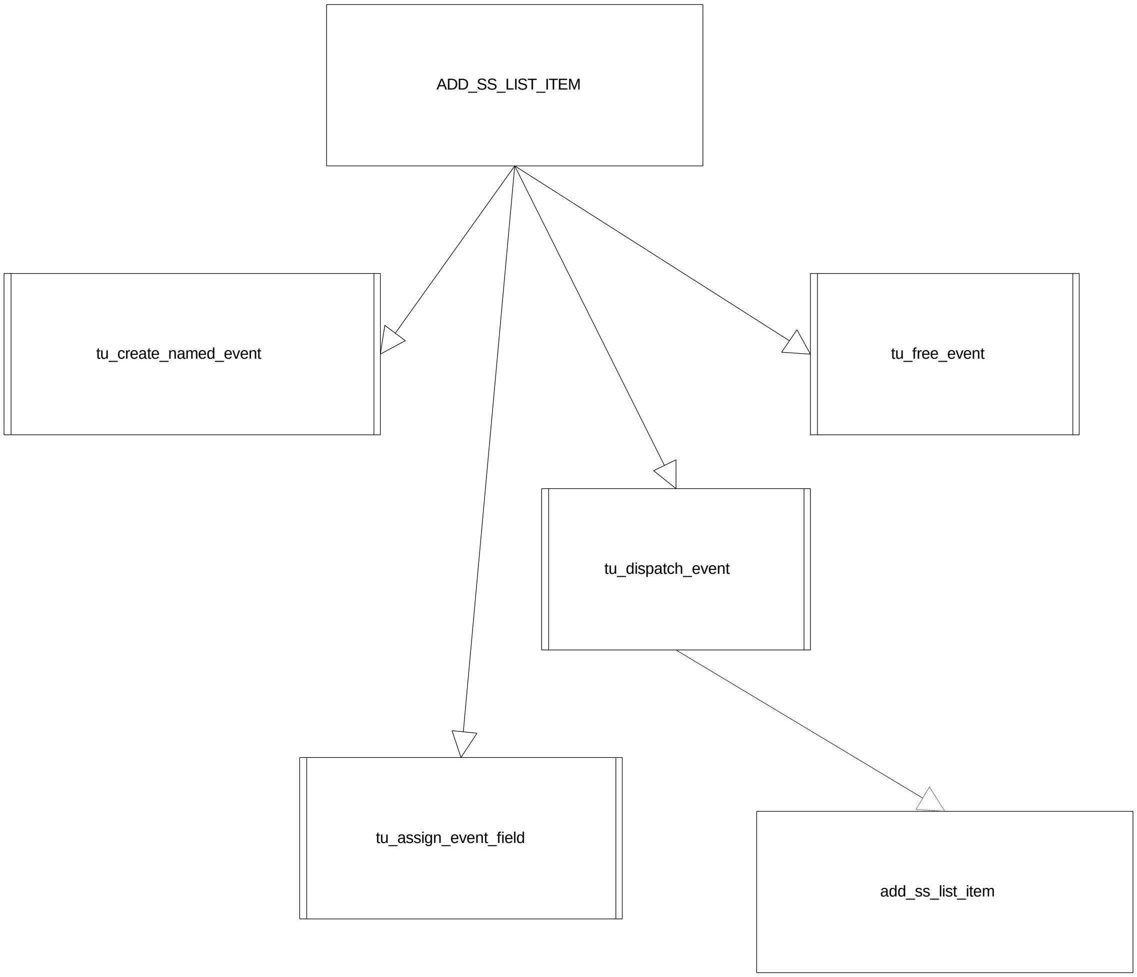

Figure 84 - ADD_SS_LIST_ITEM Structure Chart

Function	Description
add_ss_list_item	add_ss_list_item is a GUI layer event that creates an item in the screen saver list using the data specified for the event.
tu_assign_event_field	TeleUSE Library Function used to associate the contents of a C variable with the contents of an event attribute.
tu_create_named_event	TeleUSE Library Function used to create the data structure necessary to interface the C code with the D event code.
tu_dispatch_event	TeleUSE Library Function used to dispatch the created event. This causes the event to be executed.
tu_free_event	TeleUSE Library Function used to free up any memory that was allocated to the event data structure using tu_create_named_event.

Table 87 - Routines called by ADD_SS_LIST_ITEM

4.2.1.3.59 config_add_ok (ss)

config_add_ok (ss) is a GUI layer event triggered by the user selecting the ok button on the Screen Saver Configuration Add Entry Dialog. The structure chart for config_add_ok (ss) is depicted in Figure 85. A description of the routines called by config_add_ok (ss) is provided in Table 88.

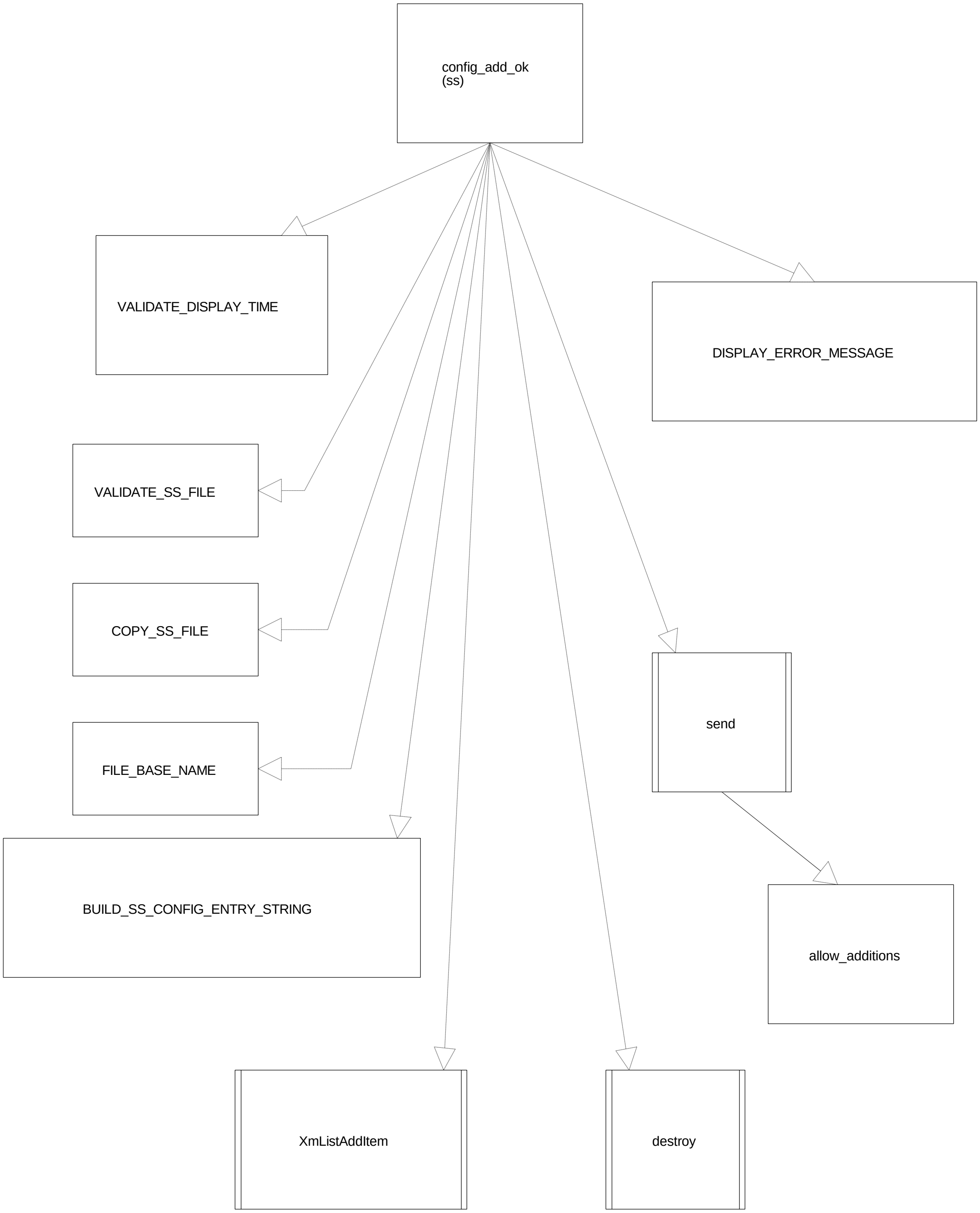

Figure 85 - config_add_ok (ss) Structure Chart

Function	Description
allow_additions	allow_additions is a GUI layer event in charge of setting the sensitivity for the Add button to true and desensitizing the delete and modify buttons.
BUILD_SS_CONFIG_ENTRY_STRING	BUILD_SS_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list.
COPY_SS_FILE	COPY_SS_FILES is a bridge layer routine used to copy the screen saver file from the directory containing the screen saver file to the kiosk data directory where screen saver files are stored for download to the kiosk.
destroy	Function which will return allocated X resources back to the system.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
FILE_BASE_NAME	FILE_BASE_NAME is a bridge layer routine used to obtain the base name from a file name that may contain the full pathname of the file.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
VALIDATE_DISPLAY_TIME	VALIDATE_DISPLAY_TIME is a bridge layer routine used to determine the validity of the screen saver display time entered by the user. The display time must contain only digits and cannot be less than or equal to 0.
VALIDATE_SS_FILE	VALIDATE_SS_FILE is a bridge layer routine used to determine the validity of the screen saver file name and type entered by the user. Using the file type (AVI, BMP, BMP-WAV), this routine checks for the existence of the necessary files.
XmListAddItem	X Library function that will add a character string to a scroll list.

Table 88 - Routines called by config_add_ok (ss)

4.2.1.3.60 VALIDATE_DISPLAY_TIME

VALIDATE_DISPLAY_TIME is a bridge layer routine used to determine the validity of the screen saver display time entered by the user. The display time must contain only digits and cannot be less than or equal to 0. The structure chart for VALIDATE_DISPLAY_TIME is depicted in Figure 86. A description of the routines called by VALIDATE_DISPLAY_TIME is provided in Table 89.

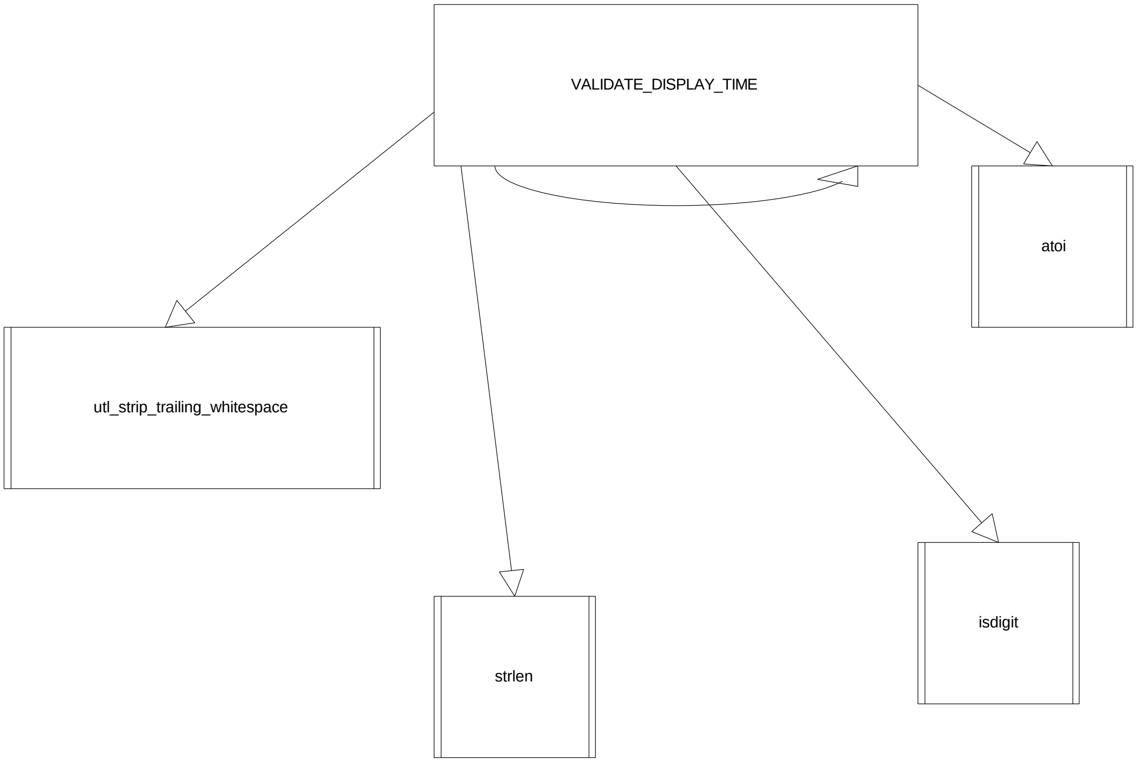

Figure 86 - VALIDATE_DISPLAY_TIME Structure Chart

Function	Description
atoi	C Library Function to convert an ASCII string to an integer value.
isdigit	C Library Function that determines if the given value is a number.
strlen	UNIX system call that computes the number of characters in a NULL terminated string.
utl_strip_trailing_whitespace	MDI Common Utility Library routine used to remove trailing blanks from a string.

Table 89 - Routines called by VALIDATE_DISPLAY_TIME

4.2.1.3.61 VALIDATE_SS_FILE

VALIDATE_SS_FILE is a bridge layer routine used to determine the validity of the screen saver file name and type entered by the user. Using the file type (AVI, BMP, BMP-WAV), this routine checks for the existence of the necessary files. The structure chart for VALIDATE_SS_FILE is depicted in Figure 87. A description of the routines called by VALIDATE_SS_FILE is provided in Table 90.

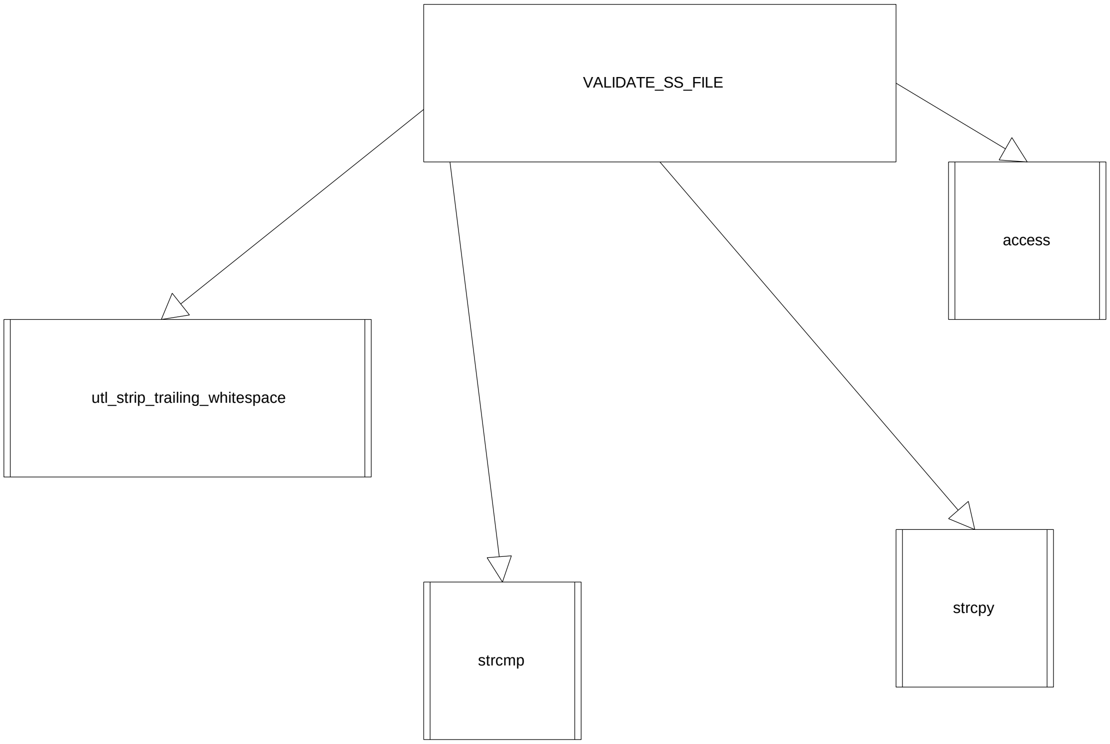

Figure 87 - VALIDATE_SS_FILE Structure Chart

Function	Description
access	UNIX Library function that tests the UNIX access rights for a specific file.
strcmp	UNIX function that will compare the contents of two NULL terminated character strings.
strcpy	C Library Function used to copy a source string to a destination string.
utl_strip_trailing_whitespace	MDI Common Utility Library routine used to remove trailing blanks from a stirng.

Table 90 - Routines called by VALIDATE_SS_FILE

4.2.1.3.62 COPY_SS_FILES

COPY_SS_FILES is a bridge layer routine used to copy the screen saver file from the directory containing the screen saver file to the kiosk data directory where screen saver files are stored for download to the kiosk. The structure chart for COPY_SS_FILES is depicted in Figure 88. A description of the routines called by COPY_SS_FILES is provided in Table 91.

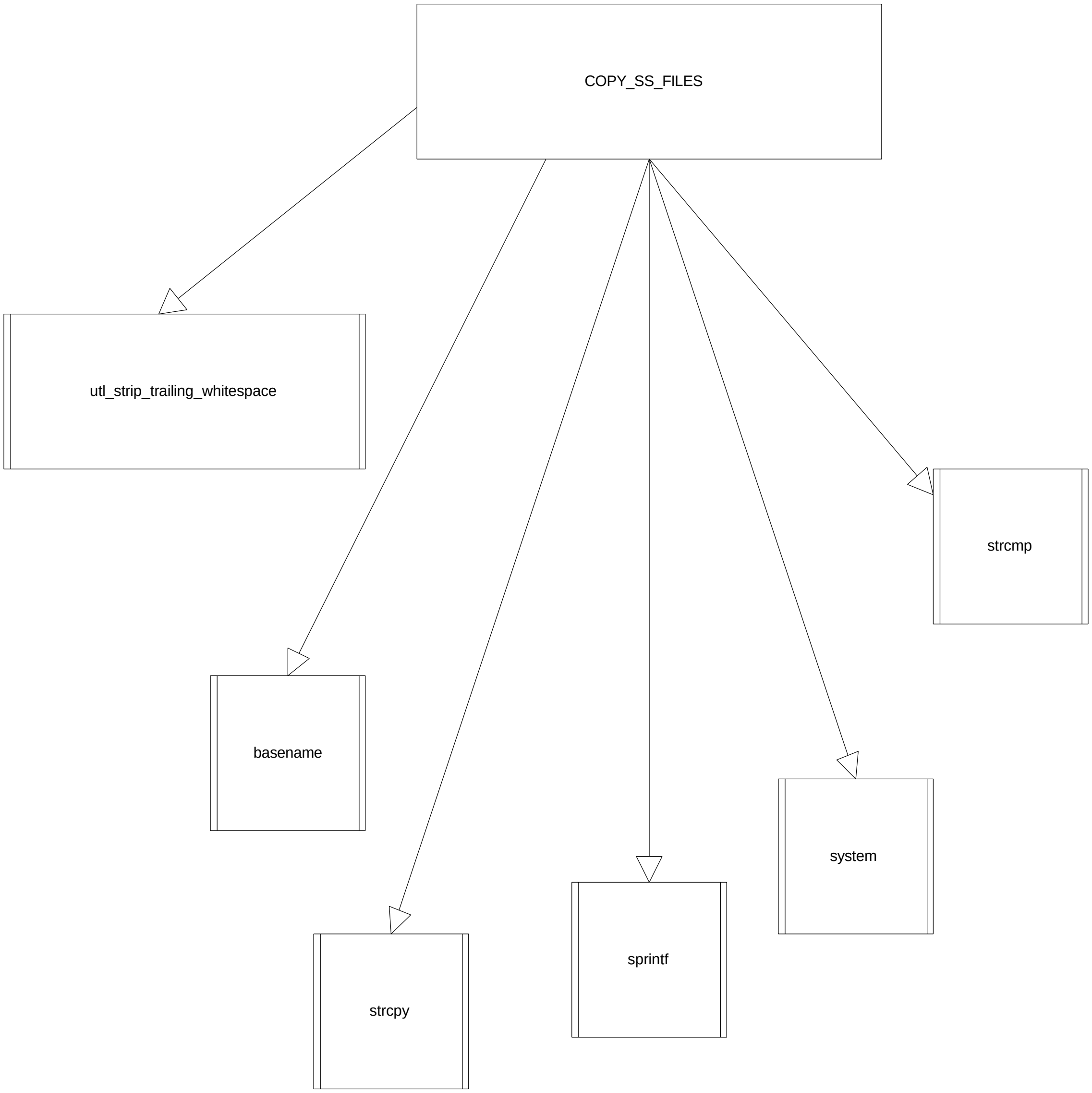

Figure 88 - COPY_SS_FILES Structure Chart

Function	Description
basename	UNIX function that will strip of the directory information from a filename (i.e., it will return just the base filename).
sprintf	C Library Function that provides printf capabilities to a character string.
strcmp	UNIX function that will compare the contents of two NULL terminated character strings.
strcpy	C Library Function used to copy a source string to a destination string.
system	UNIX function call that will execute the specified UNIX command in the shell.
utl_strip_trailing_whitespace	MDI Common Utility Library routine used to remove trailing blanks from a string.

Table 91 - Routines called by COPY_SS_FILES

4.2.1.3.63 FILE_BASE_NAME

FILE_BASE_NAME is a bridge layer routine used to obtain the base name from a file name that may contain the full pathname of the file. The structure chart for FILE_BASE_NAME is depicted in Figure 89. A description of the routines called by FILE_BASE_NAME is provided in Table 92.

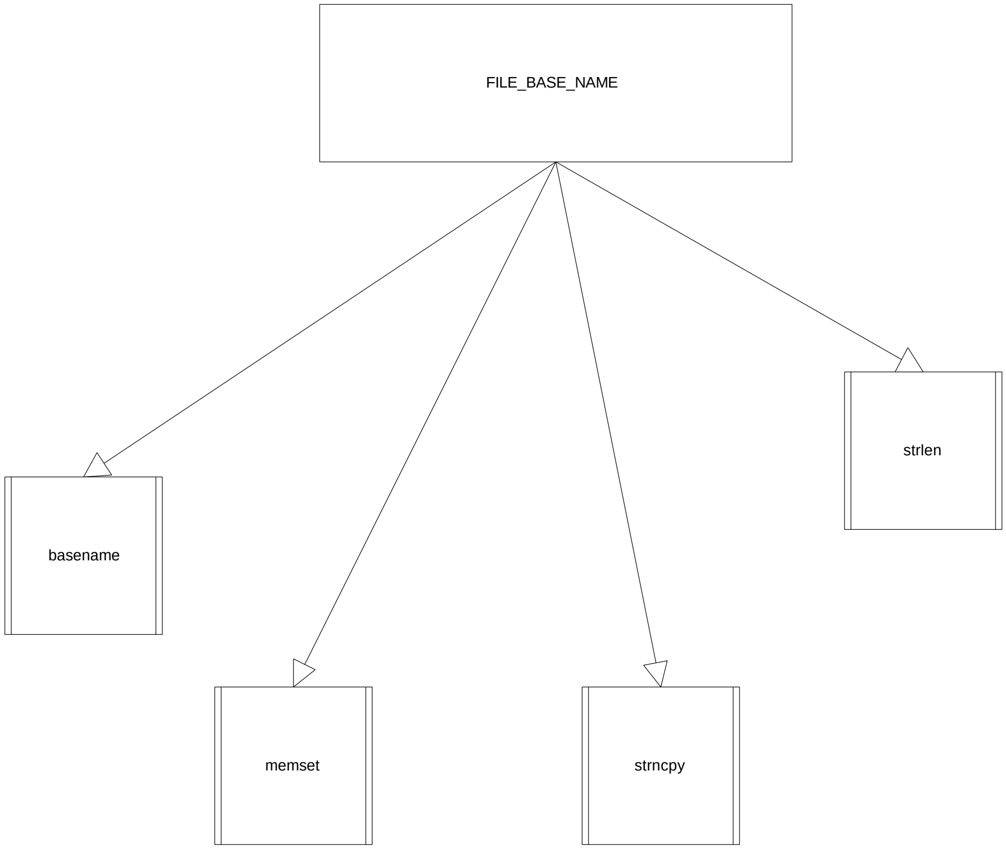

Figure 89 - FILE_BASE_NAME Structure Chart

Function	Description
basename	UNIX function that will strip the directory information from a filename (i.e., it will return just the base filename).
memset	C Library Function used to set an area of memory to a specified value.
strlen	UNIX system call that computes the number of characters in a NULL terminated string.
strncpy	C Library Function used to copy a specified number of characters from a source string to a destination string.

Table 92 - Routines called by FILE_BASE_NAME

4.2.1.3.64 ksmg_create_ss_config_file

ksmg_create_ss_config creates an empty screen saver configuration data file. This routine is used when the screen saver file does not exist. The structure chart for ksmg_create_ss_config_file is depicted in Figure 90. A description of the routines called by ksmg_create_ss_config_file is provided in Table 93.

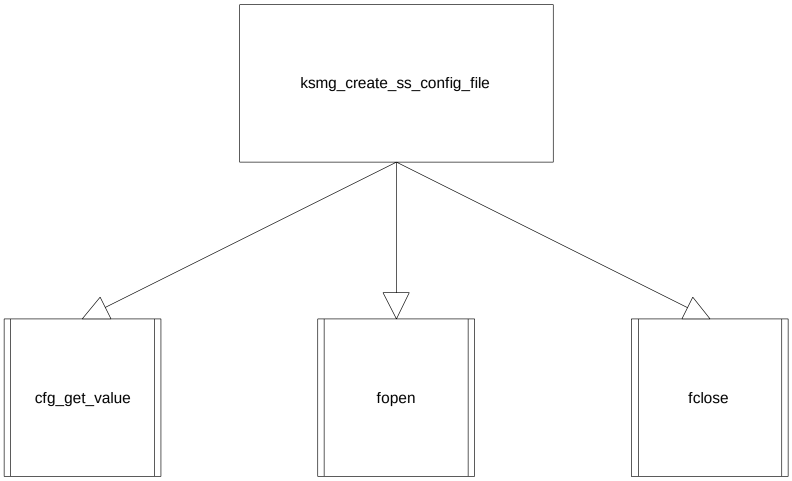

Figure 90 - ksmg_create_ss_config_file Structure Chart

Function	Description
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
fclose	C Library Function used to close an open file.
fopen	C Library Function that opens the specified file using the specified access mode.

Table 93 - Routines called by ksmg_create_ss_config_file

4.2.1.3.65 config_modify_cancel (ss)

config_modify_cancel (ss) is a GUI layer event triggered when the user selects the cancel button on the Modify Screen Saver Entry Dialog. Modifications are not allowed so this is a safety net which displays an error message to the user. The structure chart for config_modify_cancel (ss) is depicted in Figure 91. A description of the routines called by config_modify_cancel (ss) is provided in Table 94.

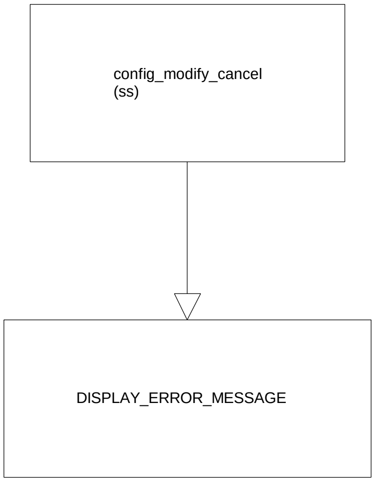

Figure 91 - config_modify_cancel (ss) Structure Chart

Function	Description
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.

Table 94 - Routines called by config_modify_cancel (ss)

4.2.1.3.66 config_modify_ok (ss)

config_modify_ok (ss) is a GUI layer event triggered when the user selects the ok button on the Modify Screen Saver Entry Dialog. Modifications are not allowed so this is a safety net which displays an error message to the user. The structure chart for config_modify_ok (ss) is depicted in Figure 92. A description of the routines called by config_modify_ok (ss) is provided in Table 95.

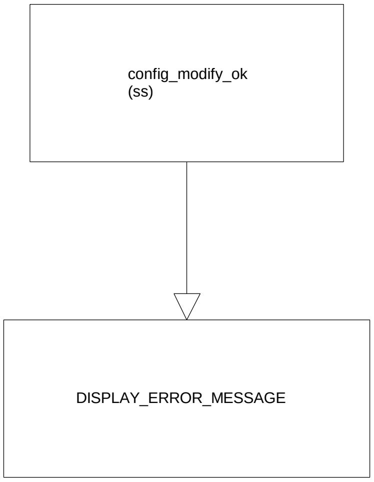

Figure 92 - config_modify_ok (ss) Structure Chart

Function	Description
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.

Table 95 - Routines called by config_modify_ok (ss)

4.2.1.3.67 modify_entry (ss)

modify_entry (ss) is the GUI layer event triggered when the user attempts to modify a screen saver list entry. Modification of a screen saver list entry is not allowed and an error dialog is displayed with a message stating this fact. The structure chart for modify_entry (ss) is depicted in Figure 93. A description of the routines called by modify_entry (ss) is provided in Table 96.

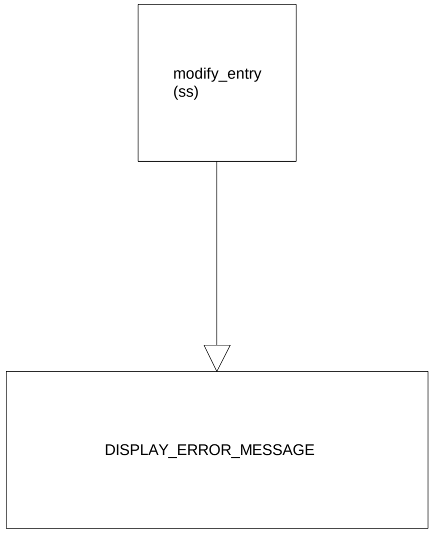

Figure 93 - modify_entry (ss) Structure Chart

Function	Description
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.

Table 96 - Routines called by modify_entry (ss)

4.2.1.3.68 delete_entry (ss)

delete_entry is a GUI level event defined in each of the D modules associated with a particular configuration table type. This event is responsible for deleting all the items selected in the configuration data list. The callbacks for the Ok and Cancel button are added by this routine so the events local to the D module are used in response to these button selections. The structure chart for delete_entry (ss) is depicted in Figure 94. A description of the routines called by delete_entry (ss) is provided in Table 97.

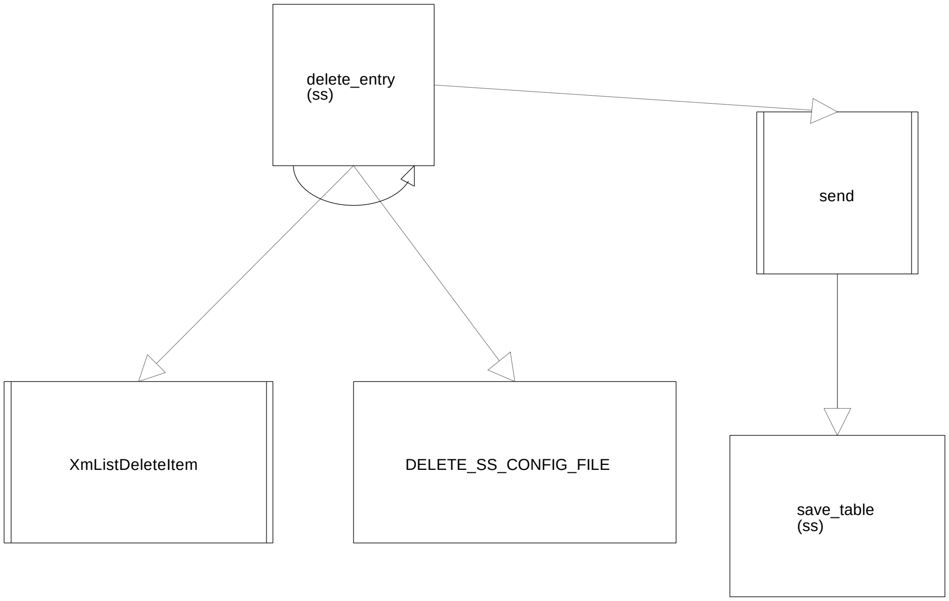

Figure 94 - delete_entry (ss) Structure Chart

Function	Description
DELETE_SS_CONFIG_FILE	DELETE_SS_CONFIG_FILE is a bridge layer routine that takes the name of the screen saver file and breaks it into the file name and the file type (AVI, BMP, WAV). These two items are then used to delete the screen saver file.
save_table (ss)	save_table is responsibile for creating the string list object that is used to write the configuration data to the file.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
XmListDeleteItem	X Function call that will delete an item from a Scroll List.

Table 97 - Routines called by delete_entry (ss)

4.2.1.3.69 DELETE_SS_CONFIG_FILE

DELETE_SS_CONFIG_FILE is a bridge layer routine that takes the name of the screen saver file and breaks it into the file name and the file type (AVI, BMP, WAV). These two items are then used to delete the screen saver file. The structure chart for DELETE_SS_CONFIG_FILE is depicted in Figure 95. A description of the routines called by DELETE_SS_CONFIG_FILE is provided in Table 98.

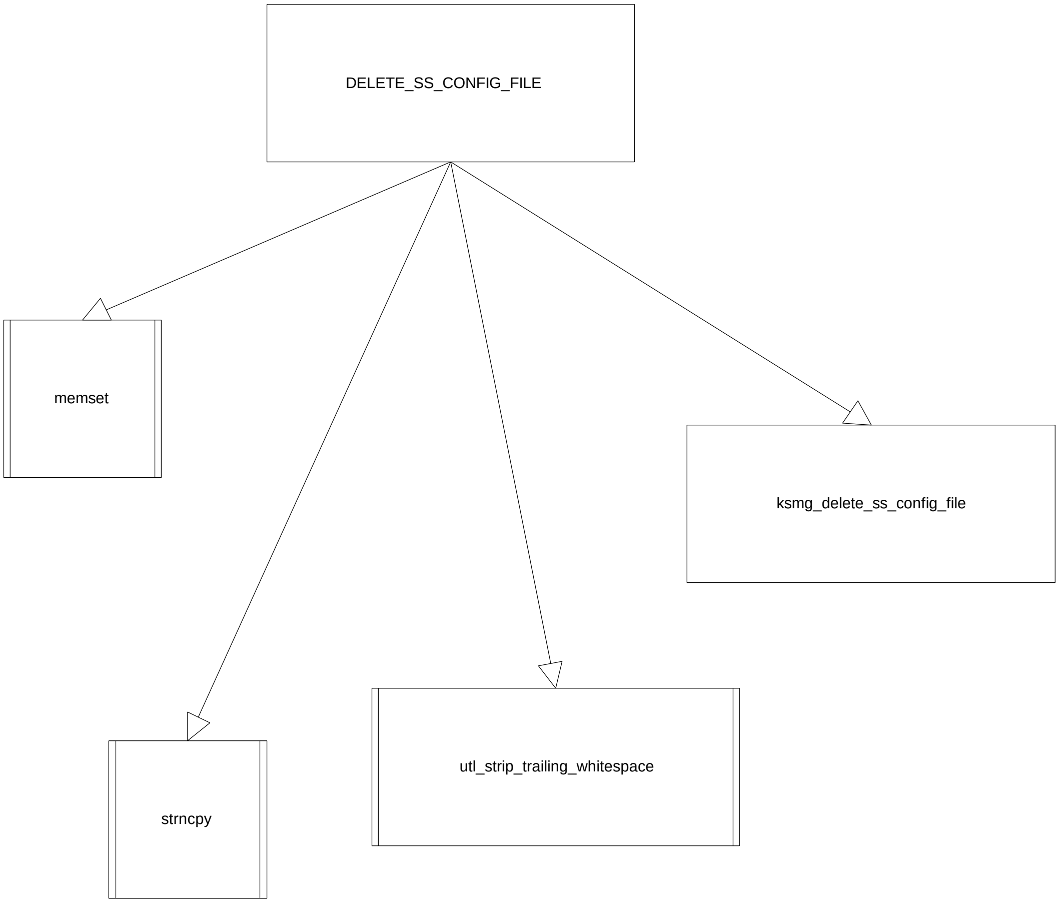

Figure 95 - DELETE_SS_CONFIG_FILE Structure Chart

Function	Description
ksmg_delete_ss_config_file	ksmg_delete_ss_config_file is used to delete screen saver files based on the advertisement type and the file name.
memset	C Library Function used to set an area of memory to a specified value.
strncpy	C Library Function used to copy a specified number of characters from a source string to a destination string.
utl_strip_trailing_whitespace	MDI Common Utility Library routine used to remove trailing blanks from a stirng.

Table 98 - Routines called by DELETE_SS_CONFIG_FILE

4.2.1.3.70 ksmg_delete_ss_config_file

ksmg_delete_ss_config_file is used to delete screen saver files based on the advertisement type and the file name. The structure chart for ksmg_delete_ss_config_file is depicted in Figure 96. A description of the routines called by ksmg_delete_ss_config_file is provided in Table 99.

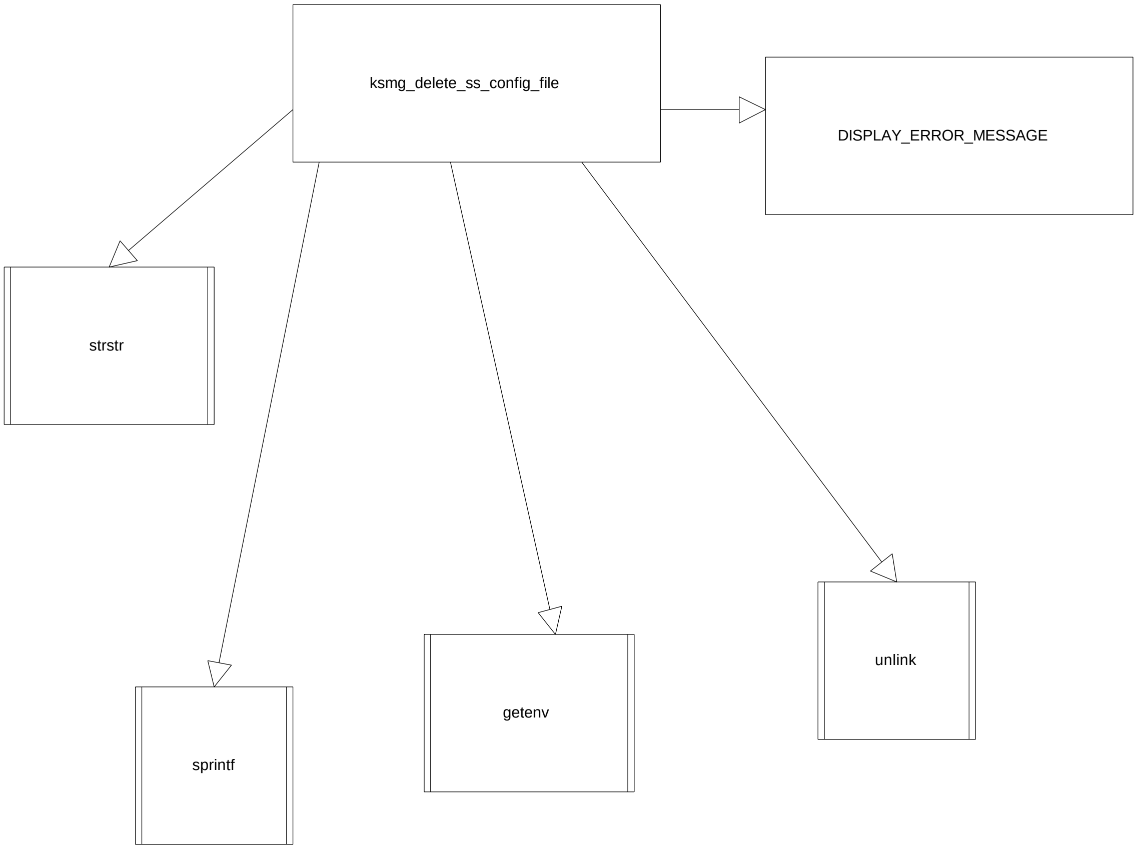

Figure 96 - ksmg_delete_ss_config_file Structure Chart

Function	Description
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
getenv	UNIX function call that will search the environment for the specified variable.
sprintf	C Library Function that provides printf capabilities to a character string.
strstr	C Library Function that will return a pointer to the first occurrence of a string within a string.
unlink	UNIX system call that will remove the specified file from the file system.

Table 99 - Routines called by ksmg_delete_ss_config_file

4.2.1.3.71 save_table (ss)

save_table is responsibile for creating the string list object that is used to write the configuration data to the file. The structure chart for save_table (ss) is depicted in Figure 97. A description of the routines called by save_table (ss) is provided in Table 100.

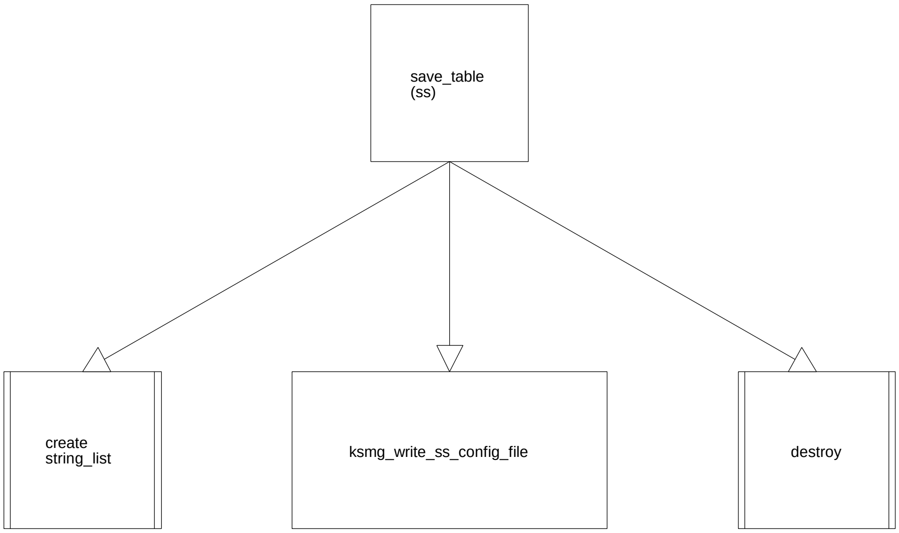

Figure 97 - save_table (ss) Structure Chart

Function	Description
create string_list	Function that creates a C character string from a series of X strings.
destroy	Function which will return allocated X resources back to the system.
ksmg_write_ss_config_file	ksmg_write_ss_config_file is used to write the list of screen saver data displayed to the user to the screen saver data file.

Table 100 - Routines called by save_table (ss)

4.2.1.3.72 ksmg_write_ss_config_file

ksmg_write_ss_config_file is used to write the list of screen saver data displayed to the user to the screen saver data file. The structure chart for ksmg_write_ss_config_file is depicted in Figure 98. A description of the routines called by ksmg_write_ss_config_file is provided in Table 101.

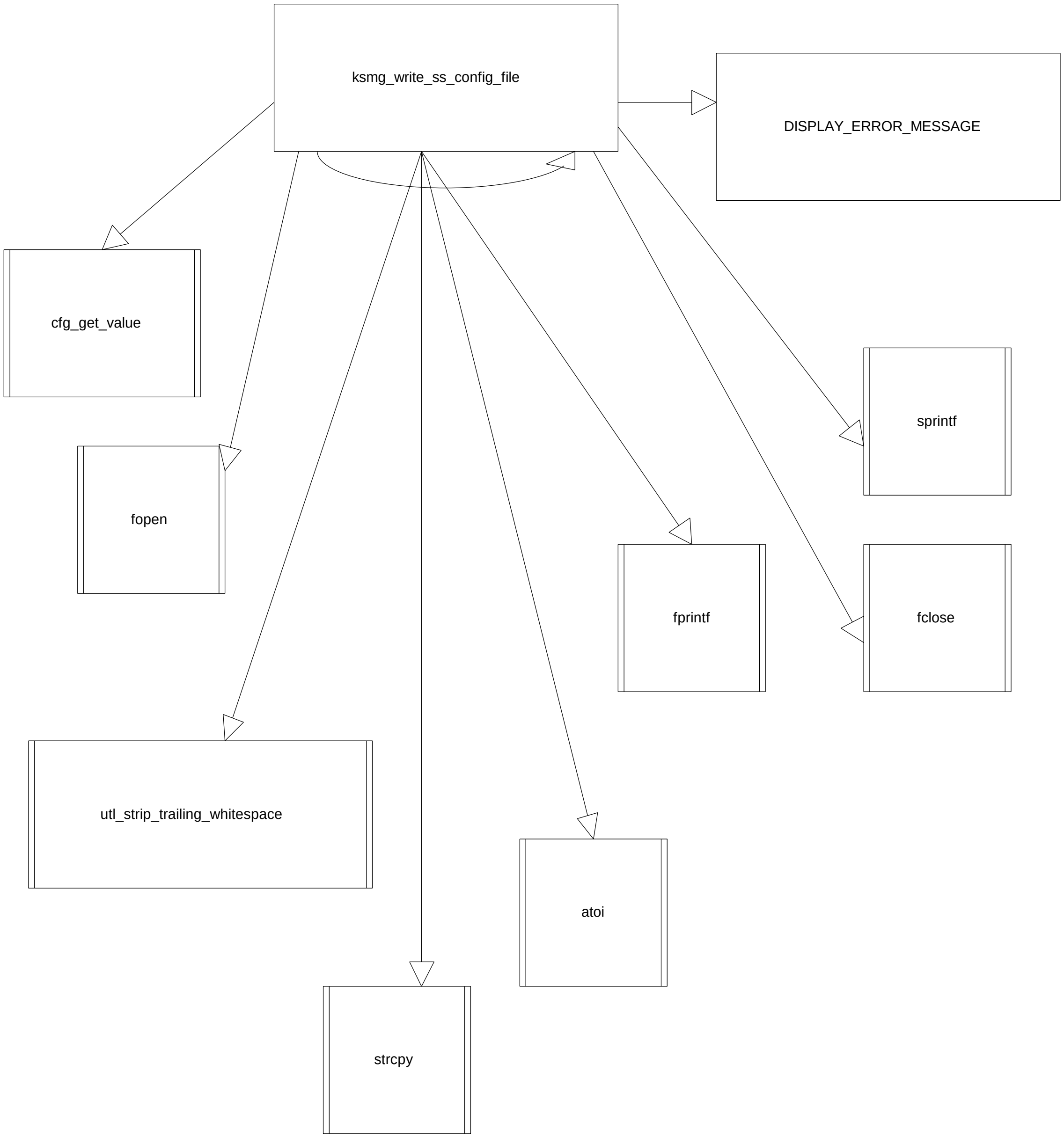

Figure 98 - ksmg_write_ss_config_file Structure Chart

Function	Description
atoi	C Library Function to convert an ASCII string to an integer value.
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
fclose	C Library Function used to close an open file.
fopen	C Library Function that opens the specified file using the specified access mode.
fprintf	UNIX system call to print formatted data to a file stream.
sprintf	C Library Function that provides print capabilities to a character string.
strcpy	C Library Function used to copy a source string to a destination string.
utl_strip_trailing_whitespace	MDI Common Utility Library routine used to remove trailing blanks from a string.

Table 101 - Routines called by ksmg_write_ss_config_file

4.2.1.3.73 configure_kiosk

configure_kiosk is the GUI layer event attached to the kiosk configuration table menu item. The button is selected when the user desires to modify the contents of the kiosk configuration file. The structure chart for configure_kiosk is depicted in Figure 99. A description of the routines called by configure_kiosk is provided in Table 102.

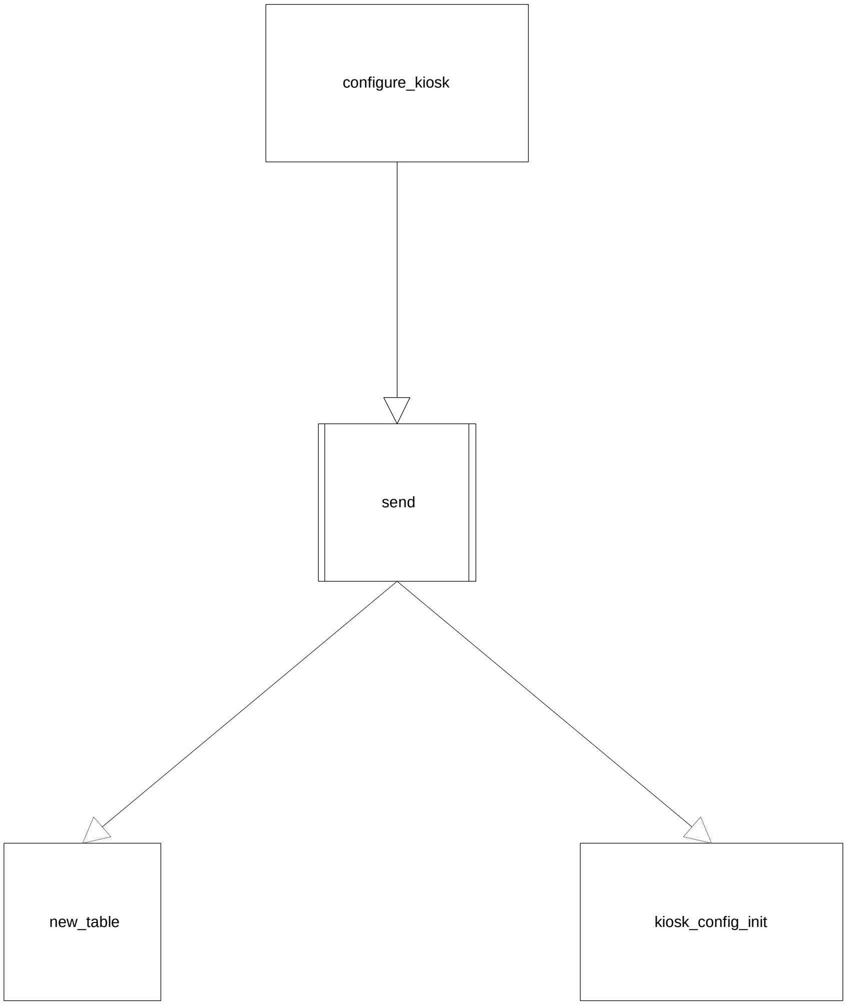

Figure 99 - configure_kiosk Structure Chart

Function	Description
kiosk_config_init	kiosk_config_init is the GUI layer event used to initialize the user interface when a table (data file) is initially selected for modification. This includes clearing the list of items to be displayed, allowing additions to be made, reading the data file, and creating the data file if it doesn't exist.
new_table	new_table is a GUI layer event triggered when a new table is selected for editing. Each D module contains this event and if the D module is associated with the table currently being edited a check is make to see if the current table should be saved. If so, the user is asked whether or not the table should be saved. The sensitivity of the table menu item of the current table is made sensitive and the form containing the current table is unmanaged.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.

Table 102 - Routines called by configure_kiosk

4.2.1.3.74 kiosk_config_init

kiosk_config_init is the GUI layer event used to initialize the user interface when a table (data file) is initially selected for modification. This includes clearing the list of items to be displayed, allowing additions to be made, reading the data file, and creating the data file if it doesn't exist. The structure chart for kiosk_config_init is depicted in Figure 100. A description of the routines called by kiosk_config_init is provided in Table 103.

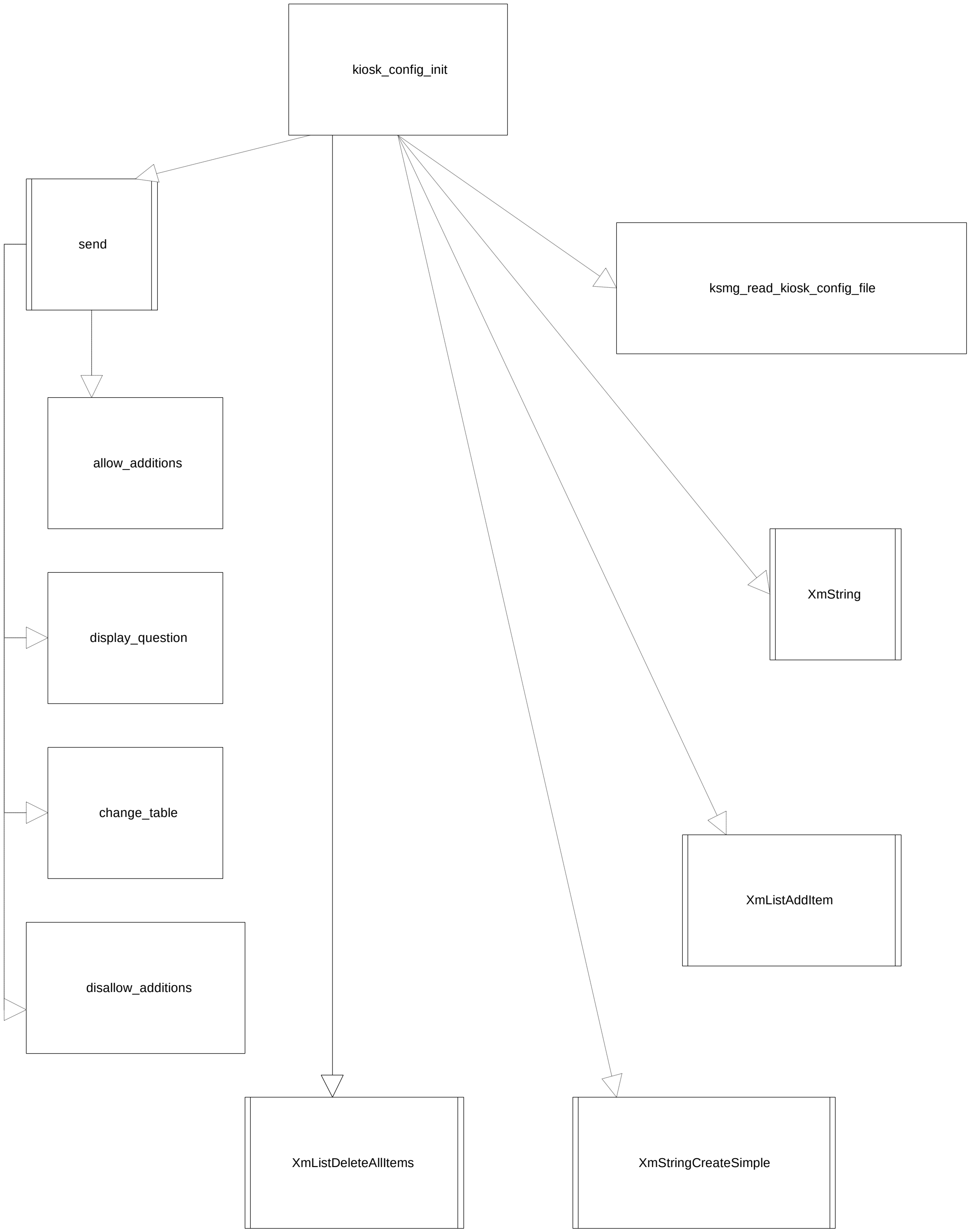

Figure 100 - kiosk_config_init Structure Chart

Function	Description
allow_additions	allow_additions is a GUI layer event in charge of setting the sensitivity for the Add button to true and desensitizing the delete and modify buttons.
change_table	change_table is a GUI layer event responsible for unmapping the current configuration data form and sensitizing the table submenu item associated with the current configuration table.
disallow_additions	disallow_additions is the GUI layer routine that sets the sensitivity of the add menu item so that no additions can be made until a table is selected.
display_question	display_question is a GUI layer event that creates a question dialog box to display the specified question to the user.
ksmg_read_kiosk_config_file	ksmg_read_kiosk_config_file is used to read the contents of the kiosk data file and create the list of kiosks to be displayed to the user.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
XmListAddItem	X Library function that will add a character string to a scroll list.
XmListDeleteAllItems	X Library function that will delete all items from a scroll list.
XmString	X Library function that will create an X string from a character string.
XmStringCreateSimple	X Library function that will create a simple X string from a character string.

Table 103 - Routines called by kiosk_config_init

4.2.1.3.75 ksmg_read_kiosk_config_file

ksmg_read_kiosk_config_file is used to read the contents of the kiosk data file and create the list of kiosks to be displayed to the user. The structure chart for ksmg_read_kiosk_config_file is depicted in Figure 101. A description of the routines called by ksmg_read_kiosk_config_file is provided in Table 104.

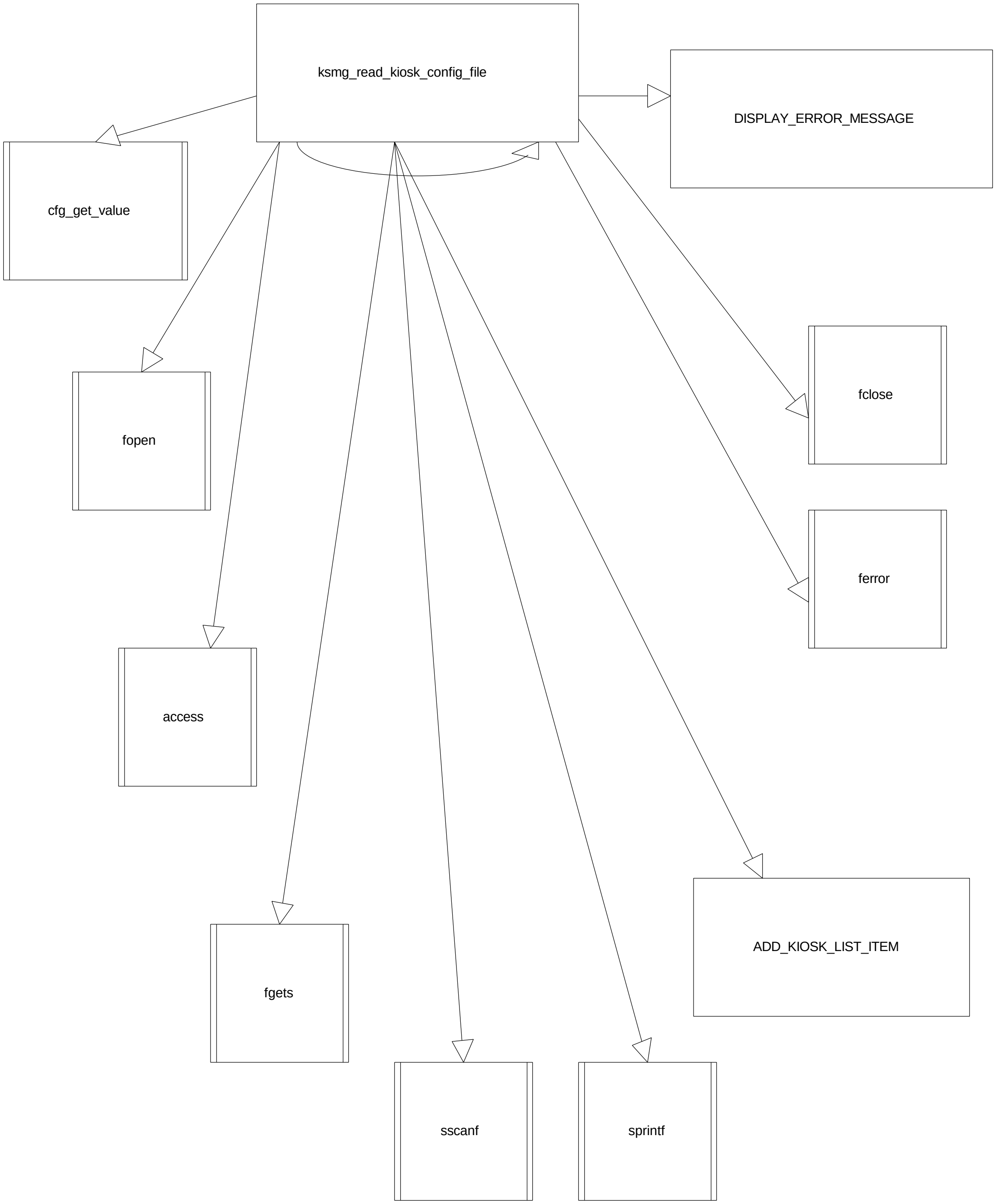

Figure 101 - ksmg_read_kiosk_config_file Structure Chart

Function	Description
access	UNIX Library function that tests the UNIX access rights for a specific file.
ADD_KIOSK_LIST_ITEM	ADD_KIOSK_LIST_ITEM is a bridge layer routine that creates and dispatches the GUI layer event responsible for adding an entry to the kiosk list.
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
fclose	C Library Function used to close an open file.
ferror	C Library Function that returns any previous errors on the associated stream.
fgets	C Library Function used to read a line of text from a file.
fopen	C Library Function that opens the specified file using the specified access mode.
sprintf	C Library Function that provides printf capabilities to a character string.
sscanf	UNIX function call that will allow formatted "input" from a NULL terminated character string.

Table 104 - Routines called by ksmg_read_kiosk_config_file

4.2.1.3.76 ADD_KIOSK_LIST_ITEM

ADD_KIOSK_LIST_ITEM is a bridge layer routine that creates and dispatches the GUI layer event responsible for adding an entry to the kiosk list. The structure chart for ADD_KIOSK_LIST_ITEM is depicted in Figure 102. A description of the routines called by ADD_KIOSK_LIST_ITEM is provided in Table 105.

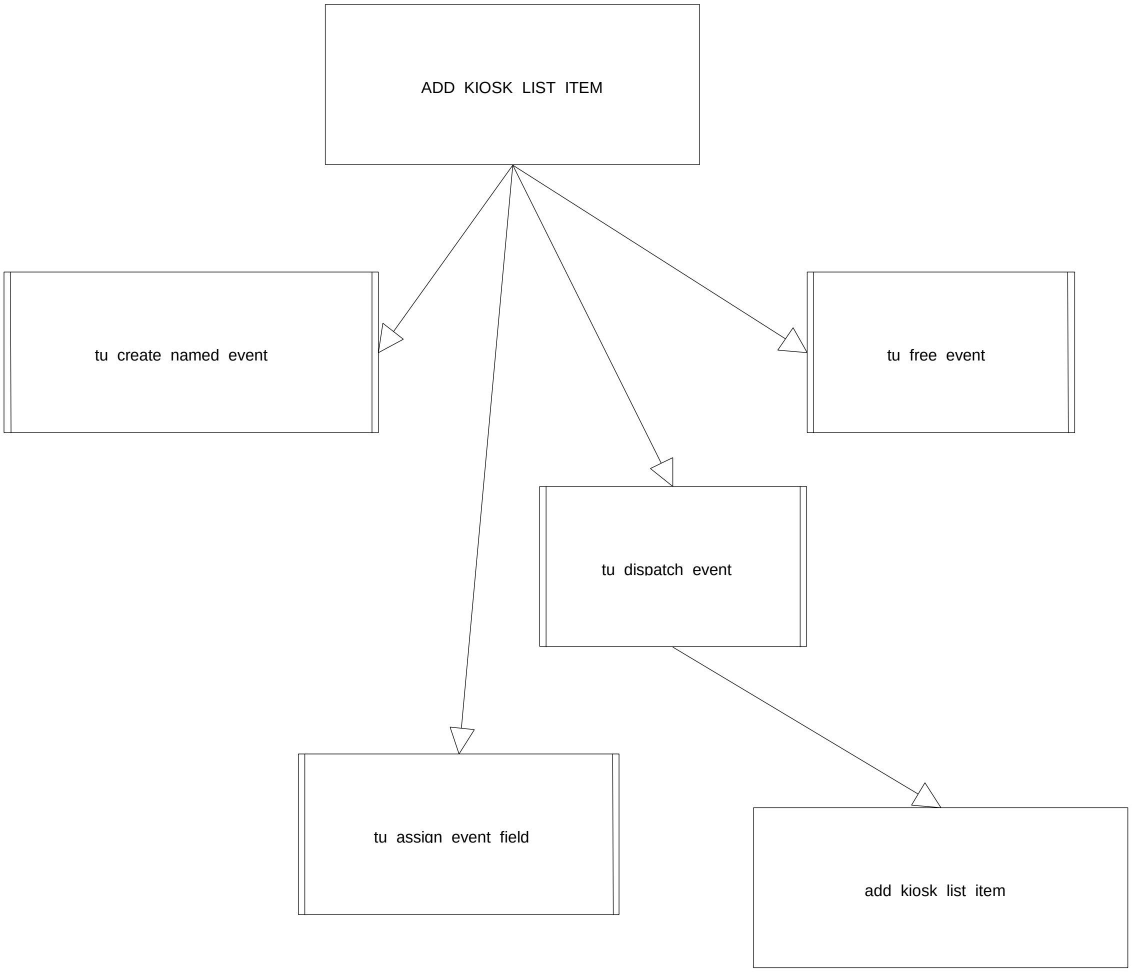

Figure 102 - ADD_KIOSK_LIST_ITEM Structure Chart

Function	Description
add_kiosk_list_item	add_kiosk_list_item is a GUI layer event that creates an item in the kiosk list using the data specified for the event.
tu_assign_event_field	TeleUSE Library Function used to associate the contents of a C variable with the contents of an event attribute.
tu_create_named_event	TeleUSE Library Function used to create the data structure necessary to interface the C code with the D event code.
tu_dispatch_event	TeleUSE Library Function used to dispatch the created event. This causes the event to be executed.
tu_free_event	TeleUSE Library Function used to free up any memory that was allocated to the event data structure using tu_create_named_event.

Table 105 - Routines called by ADD_KIOSK_LIST_ITEM

4.2.1.3.77 config_add_ok (Kiosk)

config_add_ok (kiosk) is a GUI layer event triggered by the user selecting the ok button on the Kiosk Configuration Add Entry Dialog. The structure chart for config_add_ok (Kiosk) is depicted in Figure 103. A description of the routines called by config_add_ok (Kiosk) is provided in Table 106.

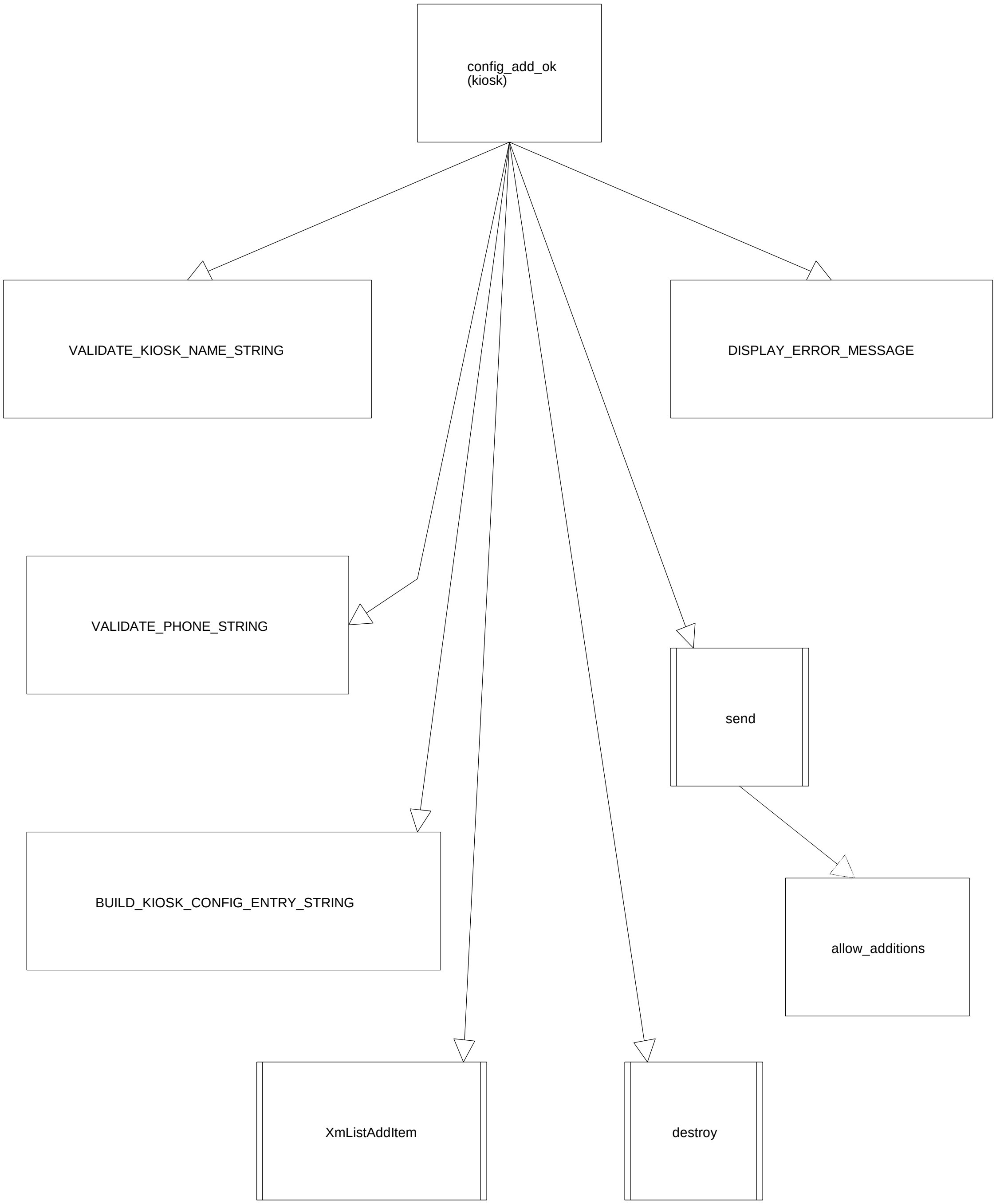

Figure 103 - config_add_ok (Kiosk) Structure Chart

Function	Description
allow_additions	allow_additions is a GUI layer event in charge of setting the sensitivity for the Add button to true and desensitizing the delete and modify buttons.
BUILD_KIOSK_CONFIG_ENTRY_STRING	BUILD_KIOSK_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list. All text strings are left justified within the appropriate fields of the entry string.
destroy	Function which will return allocated X resources back to the system.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
Send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
VALIDATE_KIOSK_NAME_STRING	VALIDATE_KIOSK_NAME_STRING is a bridge layer routine used to determine the validity of a kiosk name entered by the user. Accepted strings do not contain any whitespace.
VALIDATE_PHONE_STRING	VALIDATE_PHONE_STRING is a bridge layer routine used to determine the validity of a telephone number entered by the user. Accepted strings are in the form of xxx-xxxx or x-xxx-xxx-xxxx.
XmListAddItem	X Library function that will add a character string to a scroll list.

Table 106 - Routines called by config_add_ok (Kiosk)

4.2.1.3.78 VALIDATE_KIOSK_NAME_STRING

VALIDATE_KIOSK_NAME_STRING is a bridge layer routine used to determine the validity of a kiosk name entered by the user. Accepted strings do not contain any whitespace. The structure chart for VALIDATE_KIOSK_NAME_STRING is depicted in Figure 104. A description of the routines called by VALIDATE_KIOSK_NAME_STRING is provided in Table 107.

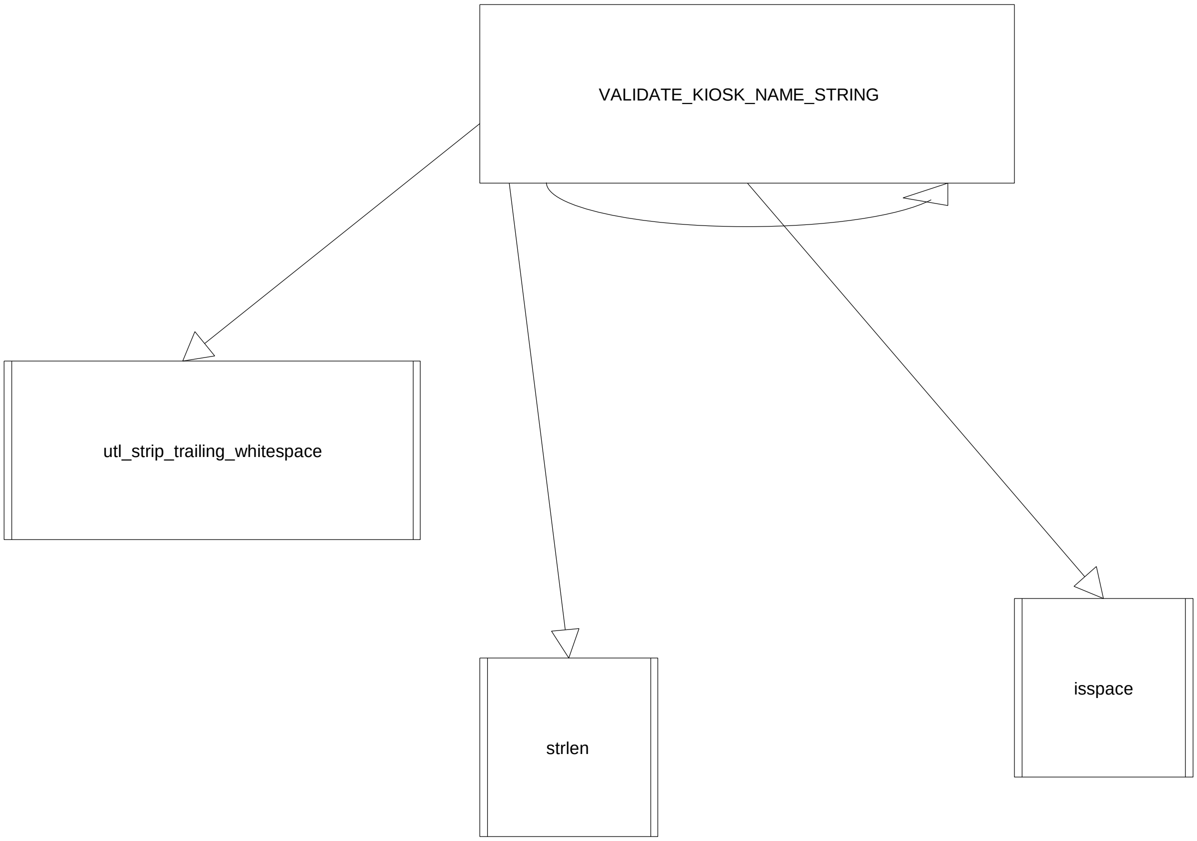

Figure 104 - VALIDATE_KIOSK_NAME_STRING Structure Chart

Function	Description
isspace	C Library Function that determines if the given value is a blank space.
strlen	UNIX system call that computes the number of characters in a NULL terminated string.
utl_strip_trailing_whitespace	MDI Common Utility Library routine used to remove trailing blanks from a stirng.

Table 107 - Routines called by VALIDATE_KIOSK_NAME_STRING

4.2.1.3.79 ksmg_create_kiosk_config_file

ksmg_create_kiosk_config creates an empty kiosk configuration data file. This routine is used when the kiosk file does not exist. The structure chart for ksmg_create_kiosk_config_file is depicted in Figure 105. A description of the routines called by ksmg_create_kiosk_config_file is provided in Table 108.

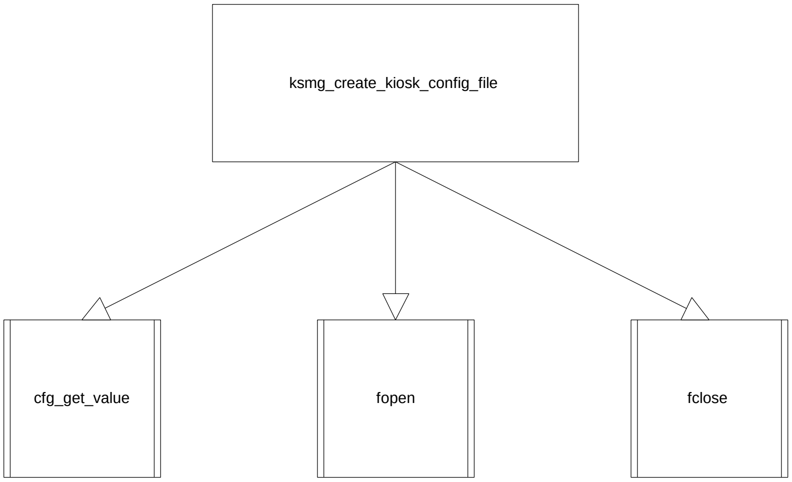

Figure 105 ksmg_create_kiosk_config_file Structure Chart

Function	Description
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
fclose	C Library Function used to close an open file.
fopen	C Library Function that opens the specified file using the specified access mode.

Table 108 - Routines called by ksmg_create_kiosk_config_file

4.2.1.3.80 config_modify_ok (Kiosk)

config_modify_ok (kiosk) is a GUI layer event triggered by the user selecting the ok button on the Kiosk Configuration Modify Entry Dialog. The structure chart for config_modify_ok (Kiosk) is depicted in Figure 106. A description of the routines called by config_modify_ok (Kiosk) is provided in Table 109.

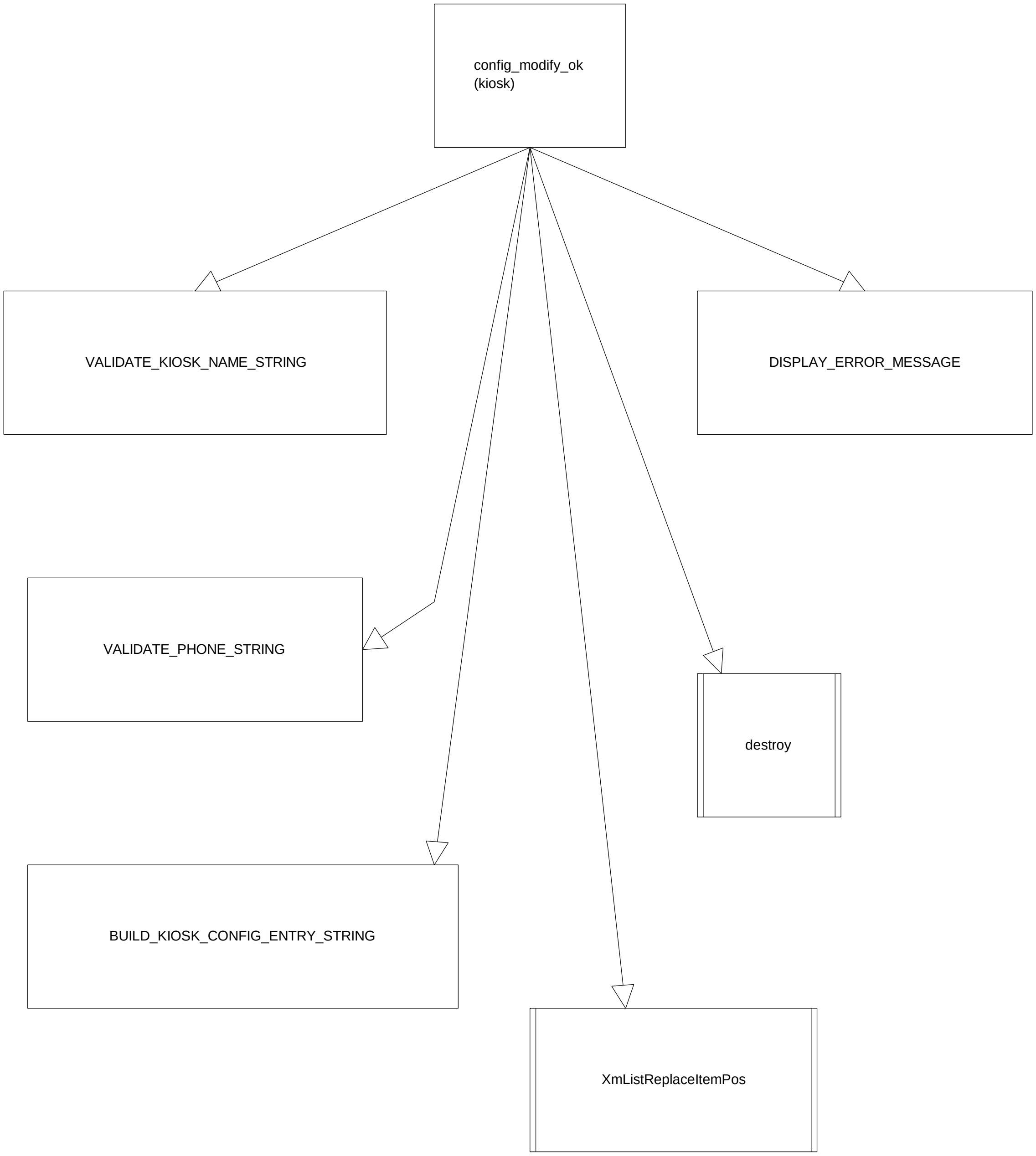

Figure 106 - config_modify_ok (Kiosk) Structure Chart

Function	Description
BUILD_KIOSK_CONFIG_ENTRY_STRING	BUILD_KIOSK_CONFIG_ENTRY_STRING is used to piece together the character string to be used as the text for an item in the list. All text strings are left justified within the appropriate fields of the entry string.
destroy	Function which will return allocated X resources back to the system.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
VALIDATE_KIOSK_NAME_STRING	VALIDATE_KIOSK_NAME_STRING is a bridge layer routine used to determine the validity of a kiosk name entered by the user. Accepted strings do not contain any whitespace.
VALIDATE_PHONE_STRING	VALIDATE_PHONE_STRING is a bridge layer routine used to determine the validity of a telephone number entered by the user. Accepted strings are in the form of xxx-xxxx or x-xxx-xxx-xxxx.
XmListReplaceItemPos	X Library function that will replace the instance of one X string with another instance in a Scroll List.

Table 109 - Routines called by config_modify_ok (Kiosk)

4.2.1.3.81 save_table (Kiosk)

save_table is responsibile for creating the string list object that is used to write the configuration data to the file. The structure chart for save_table (Kiosk) is depicted in Figure 107. A description of the routines called by save_table (Kiosk) is provided in Table 110.

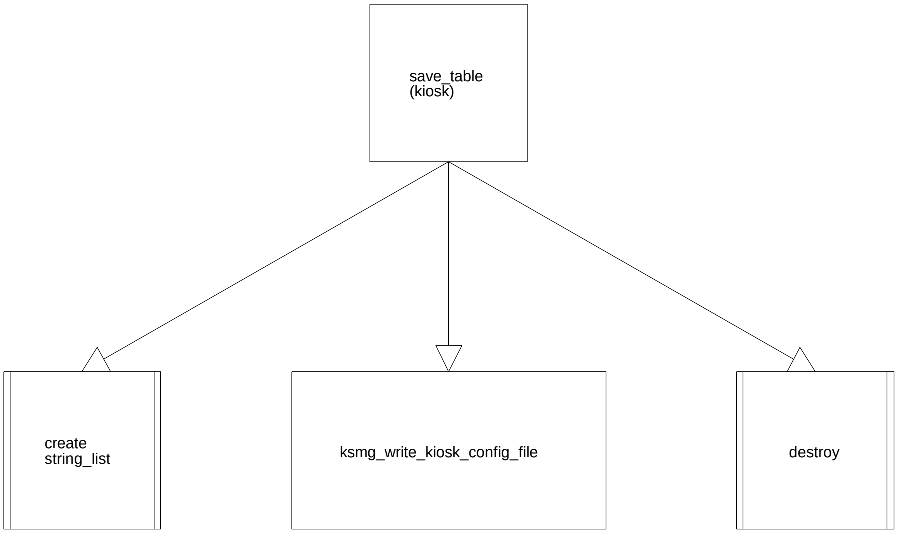

Figure 107 - save_table (Kiosk) Structure Chart

Function	Description
create string_list	Function that will creates a C character string from a series of X strings.
destroy	Function which will return allocated X resources back to the system.
ksmg_write_kiosk_config_file	ksmg_write_kiosk_config_file is used to write the list of kiosk data displayed to the user to the kiosk data file.

Table 110 - Routines called by save_table (Kiosk)

4.2.1.3.82 ksmg_write_kiosk_config_file

ksmg_write_kiosk_config_file is used to write the list of kiosk data displayed to the user to the kiosk data file. The structure chart for ksmg_write_kiosk_config_file is depicted in Figure 108. A description of the routines called by ksmg_write_kiosk_config_file is provided in Table 111.

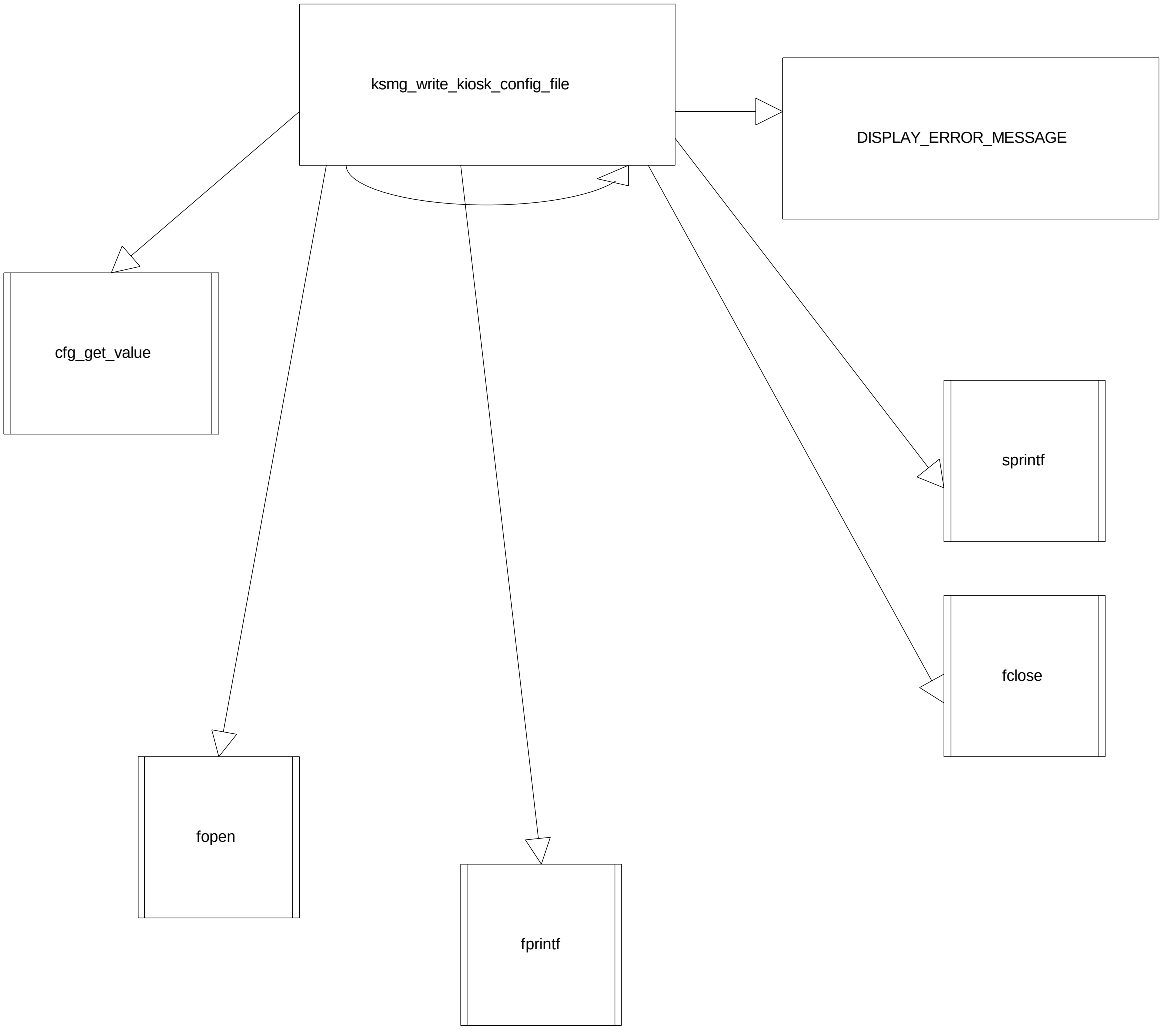

Figure 108 - ksmg_write_kiosk_config_file Structure Chart

Function	Description
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
DISPLAY_ERROR_MESSAGE	DISPLAY_ERROR_MESSAGE is a bridge layer routine that creates and dispatches the GUI layer event responsible for displaying the error dialog box.
fclose	C Library Function used to close an open file.
fopen	C Library Function that opens the specified file using the specified access mode.
fprintf	UNIX system call to print formatted data to a file stream.
sprintf	C Library Function that provides printf capabilities to a character string.

Table 111 - Routines called by ksmg_write_kiosk_config_file

4.2.1.3.83 add_entry

add_entry is a GUI level event defined in each of the D modules associated with a particular configuration table type. This event is responsible for displaying the dialog box containing the input fields for the configuration data. The callbacks for the Ok and Cancel button are added by this routine so the events local to the D module are used in response to these button selections. The structure chart for add_entry is depicted in Figure 109. A description of the routines called by add_entry is provided in Table 112.

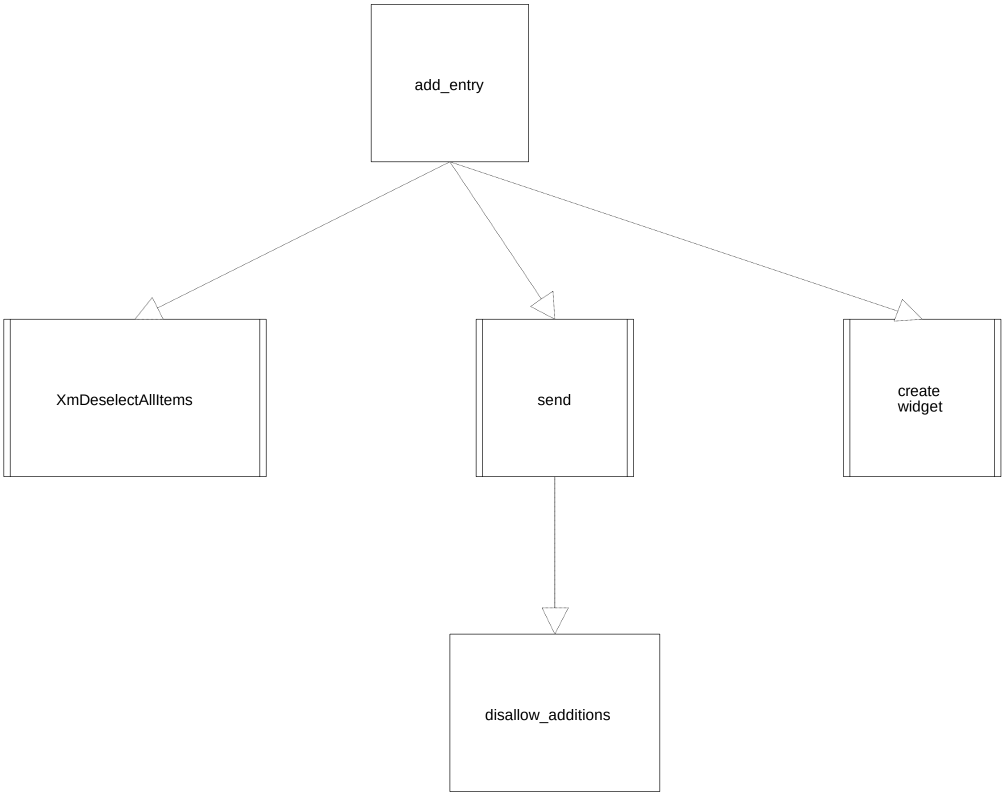

Figure 109 - add_entry Structure Chart

Function	Description
create widget	create widget is used to create a widget of a particular TeleUSE template allowing for the specification of a widget name and a parent for the widget.
disallow_additions	disallow_additions is the GUI layer routine that sets the sensitivity of the add menu item so that no additions can be made until a table is selected.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
XmDeselectAllItems	X Function call that will force all items in a Scroll List to be un-selected.

Table 112 - Routines called by add_entry

4.2.1.3.84 config_add_cancel

config_add_cancel is a GUI layer event triggered when the user selects the cancel button on the Add Entry Dialog. The structure chart for config_add_cancel is depicted in Figure 110. A description of the routines called by config_add_cancel is provided in Table 113.

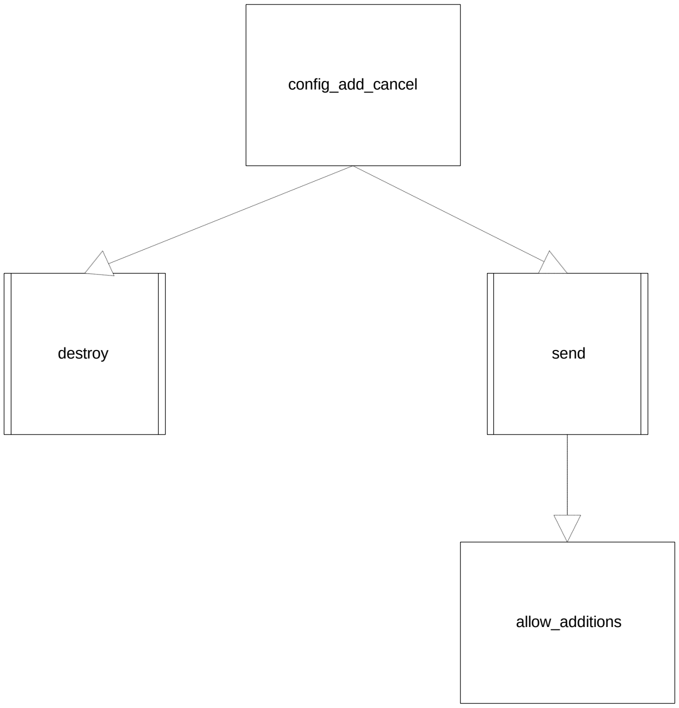

Figure 110 - config_add_cancel Structure Chart

Function	Description
allow_additions	allow_additions is a GUI layer event in charge of setting the sensitivity for the Add button to true and desensitizing the delete and modify buttons.
destroy	Function which will return allocated X resources back to the system.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.

Table 113 - Routines called by config_add_cancel

4.2.1.3.85 modify_entry

modify_entry is a GUI layer event triggered when the user selects the modify menu item. The data from the entry selected from the list is used to fill in the data entry fields of the data entry dialog box. This dialog box is created and displayed to the user allowing for Ok and Cancel of the modify task. The structure chart for modify_entry is depicted in Figure 111. A description of the routines called by modify_entry is provided in Table 114.

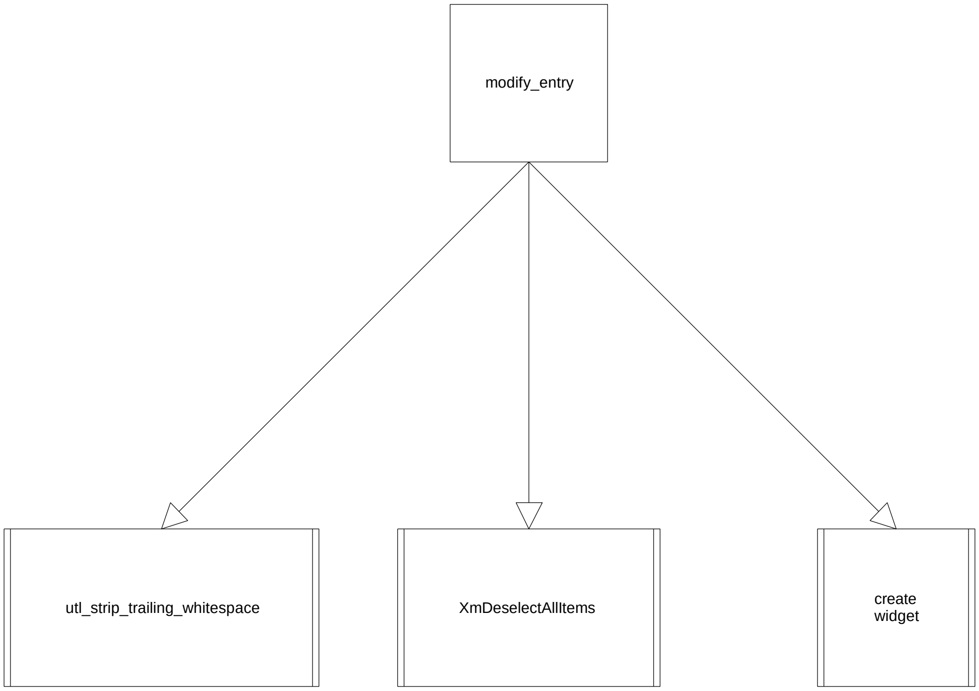

Figure 111 - modify_entry Structure Chart

Function	**Description**
create widget	create widget is used to create a widget of a particular TeleUSE template allowing for the specification of a widget name and a parent for the widget.
utl_strip_trailing_whitespace	MDI Common Utility Library routine used to remove trailing blanks from a string.
XmDeselectAllItems	X Function call that will force all items in a Scroll List to be un-selected.

Table 114 - Routines called by modify_entry

4.2.1.3.86 config_modify_cancel

config_modify_cancel is a GUI layer event triggered when the user selects the cancel button on the Modify Entry Dialog. The structure chart for config_modify_cancel is depicted in Figure 112. A description of the routines called by config_modify_cancel is provided in Table 115.

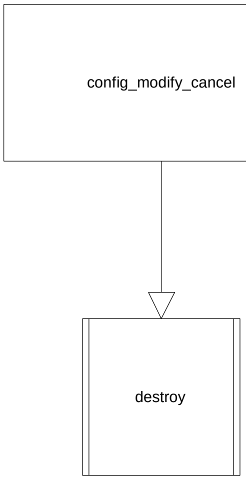

Figure 112 - config_modify_cancel Structure Chart

Function	Description
destroy	Function which will return allocated X resources back to the system.

Table 115 - Routines called by config_modify_cancel

4.2.1.3.87 delete_entry

delete_entry is a GUI level event defined in each of the D modules associated with a particular configuration table type. This event is responsible for deleting all the items selected in the configuration data list. The callbacks for the Ok and Cancel button are added by this routine so the events local to the D module are used in response to these button selections. The structure chart for delete_entry is depicted in Figure 113. A description of the routines called by delete_entry is provided in Table 116.

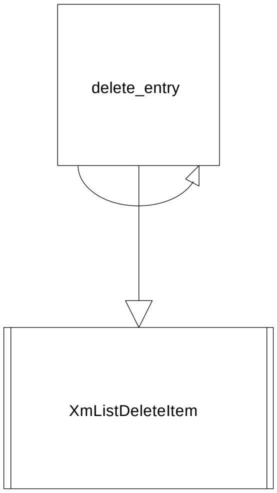

Figure 113 - delete_entry Structure Chart

Function	Description
XmListDeleteItem	X Function call that will delete an item from a Scroll List.

Table 116 - Routines called by delete_entry

4.2.1.3.88 ksmg_delete_widget

ksmg_delete_widget is the GUI layer event attached to all widgets that need to be deleted based on some form of user interaction. The structure chart for ksmg_delete_widget is depicted in Figure 114. A description of the routines called by ksmg_delete_widget is provided in Table 117.

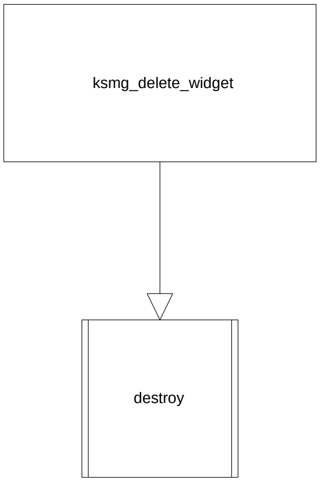

Figure 114 - ksmg_delete_widget Structure Chart

Function	Description
destroy	Function which will return allocated X resources back to the system.

Table 117 - Routines called by ksmg_delete_widget

4.2.1.3.89 ksmg_disconnect_from_dsif

ksmg_disconnect_from_dsif manages the task of disconnecting from the Kiosk Data Server Interface Process. The structure chart for ksmg_disconnect_from_dsif is depicted in Figure 115. A description of the routines called by ksmg_disconnect_from_dsif is provided in Table 118.

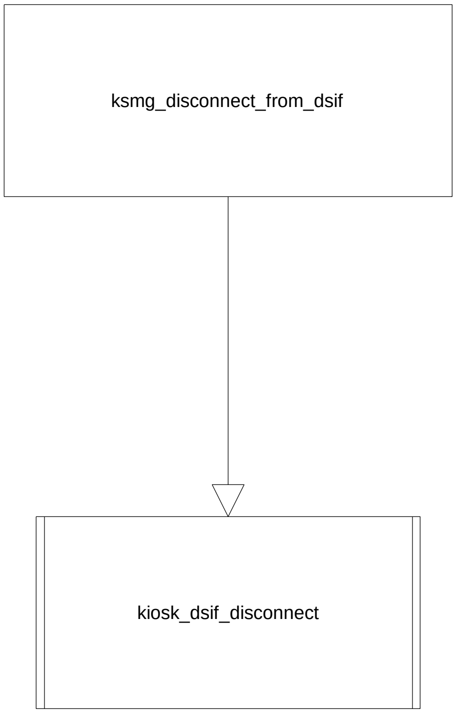

Figure 115 - ksmg_disconnect_from_dsif Structure Chart

Function	Description
kiosk_dsif_disconnect	kiosk_dsif_disconnect is used to disconnect a process from the kiosk_dsif process.

Table 118 - Routines called by ksmg_disconnect_from_dsif

4.2.1.3.90 exit_application

exit_application is the GUI layer event responsible for performing housekeeping functions prior to termination of the application. The structure chart for exit_application is depicted in Figure 116. A description of the routines called by exit_application is provided in Table 119.

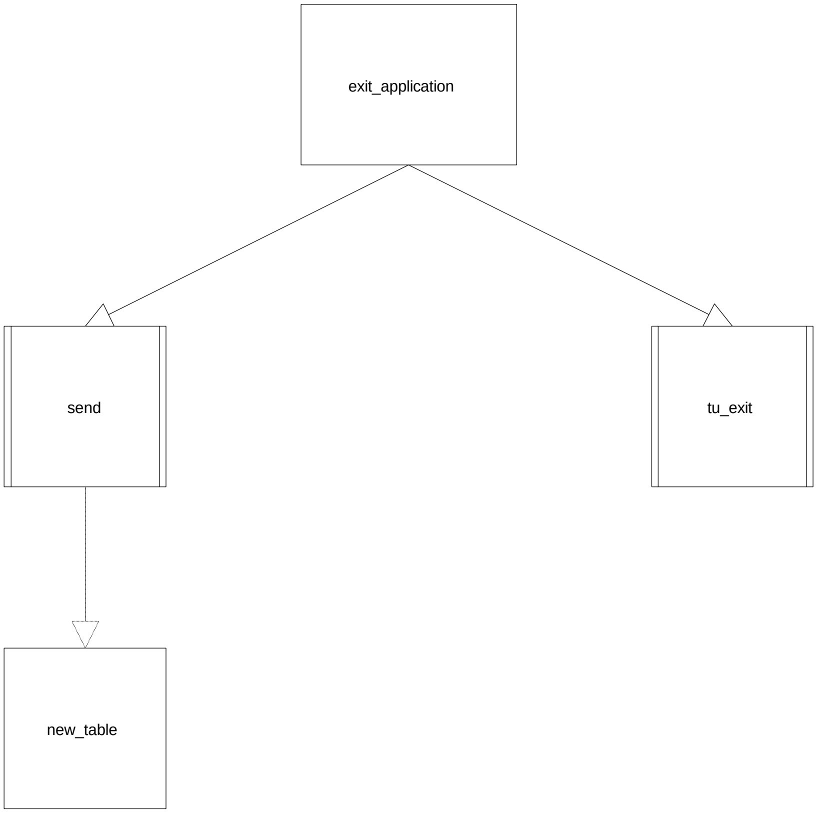

Figure 116 - exit_application Structure Chart

Function	Description
new_table	new_table is a GUI layer event triggered when a new table is selected for editing. Each D module contains this event and if the D module is associated with the table currently being edited a check is make to see if the current table should be saved. If so, the user is asked whether or not the table should be saved. The sensitivity of the table menu item of the current table is made sensitive and the form containing the current table is unmanaged.
send	A TeleUSE statement used to trigger events immediately or queue events for later dispatch.
tu_exit	A TeleUSE library routine used to exit the application.

Table 119 - Routines called by exit_application

4.2.1.4 Status Logger

The Status Logger was developed as part of the MDI common code. For design information about the Status Logger consult the Common Code Model Deployment Initiative Design Document.

4.2.1.5 Data Server Interface

The Data Server Interface (DSIF) provides the communication path between the Data Server and the KMC applications residing on the KMC. The DSIF accepts requests from the KMC applications and returns the requested data or an error. Figure 117 depicts the DSIF data flows and Table 120 provides descriptions for the DSIF data flows.

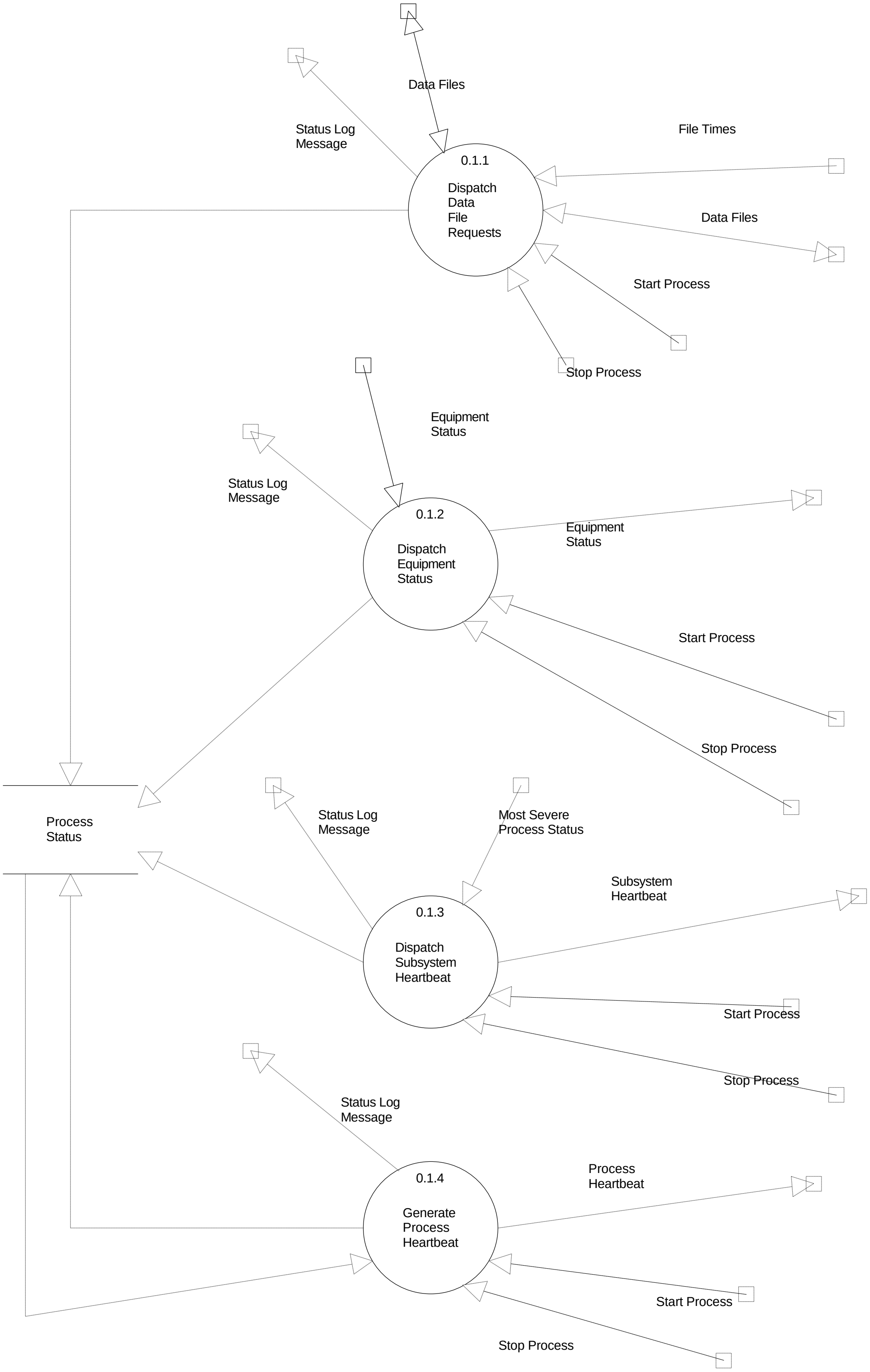

Figure 117 - DSIF Data Flows

Function	Description
Data Files	Data Files are files that are stored at the Data Server and include files such as weather files, screen saver files, and airport files.
Dispatch Data File Requests	Dispatch Data File Requests is responsible for receiving the Data Files to write to the Data Server or to use to receive File Times from the Data Server. Errors that occur reading or writing files or obtaining file times from the Data Server are logged using Status Log Messages.
Dispatch Equipment Status	Dispatch Equipment Status is responsible for receiving the Equipment Status from the Kiosk Field Unit data process and sending the Equipment Status to the Data Server. Errors that occur sending the Equipment Status to the Data Server are logged using Status Log Messages.
Dispatch Subsystem Heartbeat	Dispatch Subsystem Heartbeat is responsible for receiving the Most Severe Process Status from the Subsystem Heartbeat Management data process and sending the Subsystem Heartbeat to the Data Server. Errors that occur sending the Subsystem Heartbeat to the Data Server are logged using Status Log Messages.
Equipment Status	Equipment Status is used to define the current state of different equipment. For the Kiosk subsystem the Equipment Status is used to send the Kiosk Field Unit Equipment Status to the Data Server.
File Times	File Times are the last update times associated with the Data Files stored at the Data Server.
Generate Process Heartbeat	Generate Process Heartbeat periodically sends the Process Heartbeat to the Subsystem Heartbeat Management process. The current Process Status is read and sent as part of the Process Heartbeat. The time interval for sending the Process Heartbeat is specified by the Heartbeat Interval configuration item. Errors and other status information is logged using the Status Log Message.
Most Severe Process Status	Most Severe Process Status is the value of the process status being managed by the Subsystem Heartbeat Management that represents the worst status of all the processes. For example if all processes indicated an ok status except one process indicated a warning status then the Most Severe Process Status would be warning.
Process Status	Process Status contains the current value associated with the execution status of the process. This status can indicate an OK condition, a warning condition, or an error condition.
Start Process	Start Process is an event used to start the execution of a process.
Status Log Message	Status Log Message contains information to be logged to the subsystem log file. Typical Status Log Messages include error messages such as memory allocation errors or data being logged from field equipment associated with the subsystem.
Stop Process	Stop Process is an event used to stop the execution of a process.
Subsystem Heartbeat	Subsystem Heartbeat is the heartbeat message containing the overall status of the KIOSK subsystem. This message is generated by the Subsystem Heartbeat Management process and is passed on to the Data Server by the subsystem's Data Server Interface process.

Table 120 – DSIF Data Flow Descriptions

The following subsections provide the detailed design for the DSIF. Each subsection contains a description of the routine, a structure chart of the routine and a table containing descriptions of the components defined in the structure chart.

4.2.1.5.1 kiosk_dsif_main

The kiosk_dsif main routine is responsible for setting up configuration information, opening the socket used for communication, and connecting to the status logger. This routine enters a loop waiting for data server messages and periodically sending heartbeat messages to the subsystem heartbeat process. The structure chart for kiosk_dsif_main is depicted in Figure 118. A description of the routines called by kiosk_dsif_main is provided in Table 121.

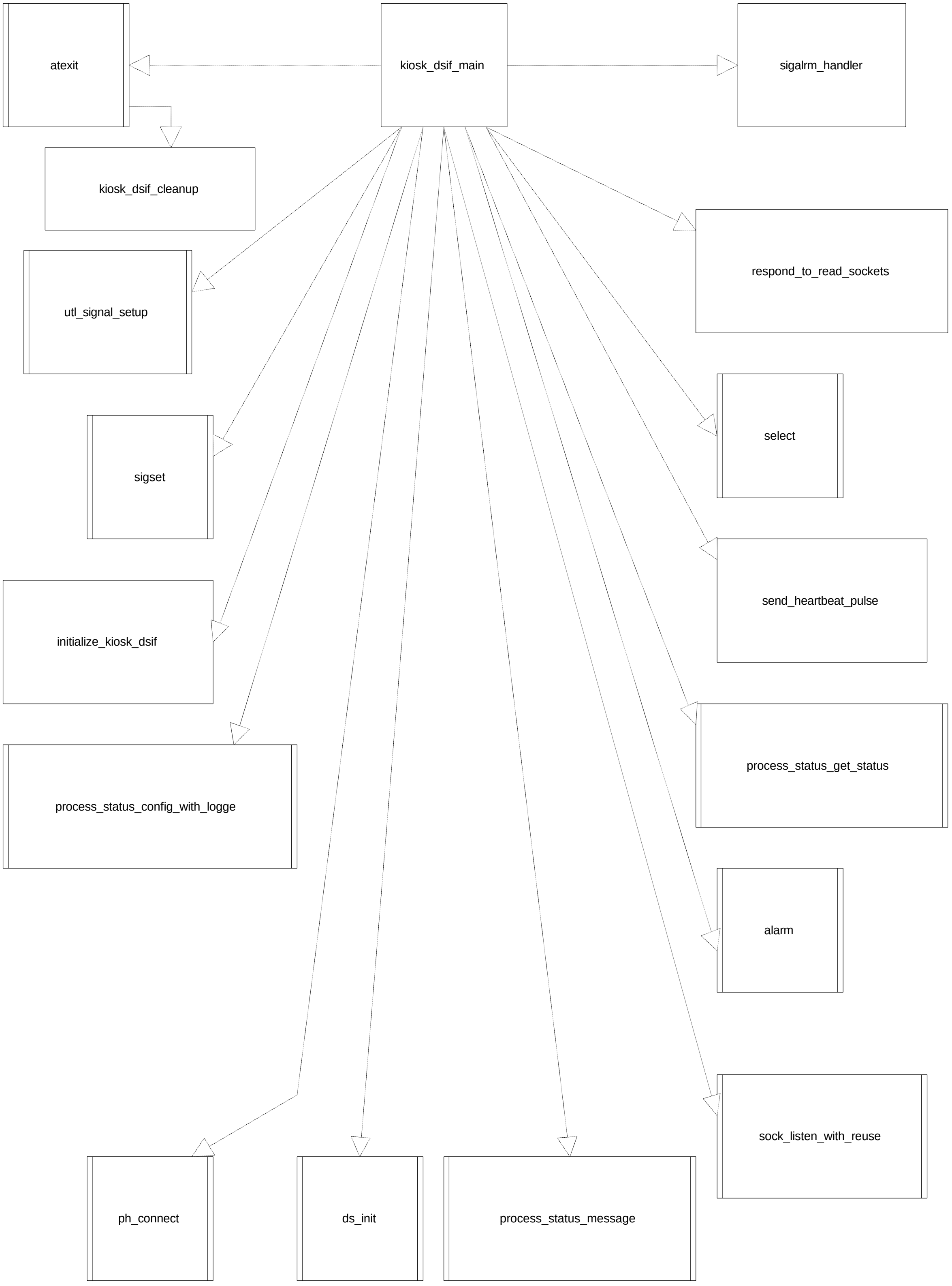

Figure 118 - kiosk_dsif_main Structure Chart

Function	Description
alarm	System Call used to set the alarm clock of the calling process to send a SIGALRM signal after the specified number of seconds have elapsed.
atexit	C Library Function used to register routines to be called on normal termination of a program.
ds_init	MDI Data Server library routine used to initialize the connection to the Data Server.
initialize_kiosk_dsif	The kiosk_dsif configuration file specified on the command line is read to obtain the values of the configurable items of the kiosk_dsif process.
kiosk_dsif_cleanup	Called when kiosk_dsif exits. This routine is responsible for performing the housekeeping necessary for a graceful shutdown. This includes sending a last heartbeat, disconnecting from the process-level heartbeat service, disconnecting from the Data Server, and closing any sockets that are open for communicating with the kiosk_dsif process.
ph_connect	MDI Process Heartbeat routine used to connect to the specified process-level heartbeat service. The host name and service name are used to make the connection.
process_status_config_with_logge	process_status_config_with_logger is an MDI Process Status Common routine used to configure the process status handling for the process. This routine is used to set up the connection to the status logger used by the calling program.
process_status_get_status	MDI Process Status routine used to obtain the most severe process-level status. This is an aggregation of the status for each of the status types defined for the process.
process_status_message	MDI Process Status routine used to log a status message for the specified status type. If the process status library was configured to use a status logger then the message is forwarded to the status logger. Otherwise the message is written to the configured status log file.
respond_to_read_sockets	Loops through the list of socket descriptors ready for reading and either accepts connections, if the socket descriptor is for the listen socket, or receives messages containing information to be sent to the Data Server.
select	C Library Function used to multiplex synchronous I/O. The list of file descriptors for reading, writing, and receiving exceptions are examined and any file descriptors that are ready for reading, writing, or have an exceptional condition pending are identified.
send_heartbeat_pulse	Sends the process-level heartbeat to the Subsystem Heartbeat process.
sigalrm_handler	The signal handler for the SIGALRM signal. This signal is used to indicate when the process-level heartbeat should be sent to the Kiosk subsystem heartbeat process. The alarm is reinitialized as part of this routine.
sigset	C Library Function used to modify the disposition of a signal. The signal can be caught, ignored, or returned to the default disposition.
sock_listen_with_reuse	MDI Common Socket routine used to set up a socket to listen for connections and to make the socket address reusable.
utl_signal_setup	MDI Common Utility Library routine used to set up a default signal handler for all catchable signals.

Table 121 - Routines called by kiosk_dsif_main

4.2.1.5.2 kiosk_dsif_cleanup

kiosk_dsif_cleanup is called when kiosk_dsif exits. This routine is responsible for performing the housekeeping necessary for a graceful shutdown. This includes sending a last heartbeat, disconnecting from the process-level heartbeat service, disconnecting from the Data Server, and closing any sockets that are open for communicating with the kiosk_dsif process. The structure chart for kiosk_dsif_cleanup is depicted in Figure 119. A description of the routines called by kiosk_dsif_cleanup is provided in Table 122.

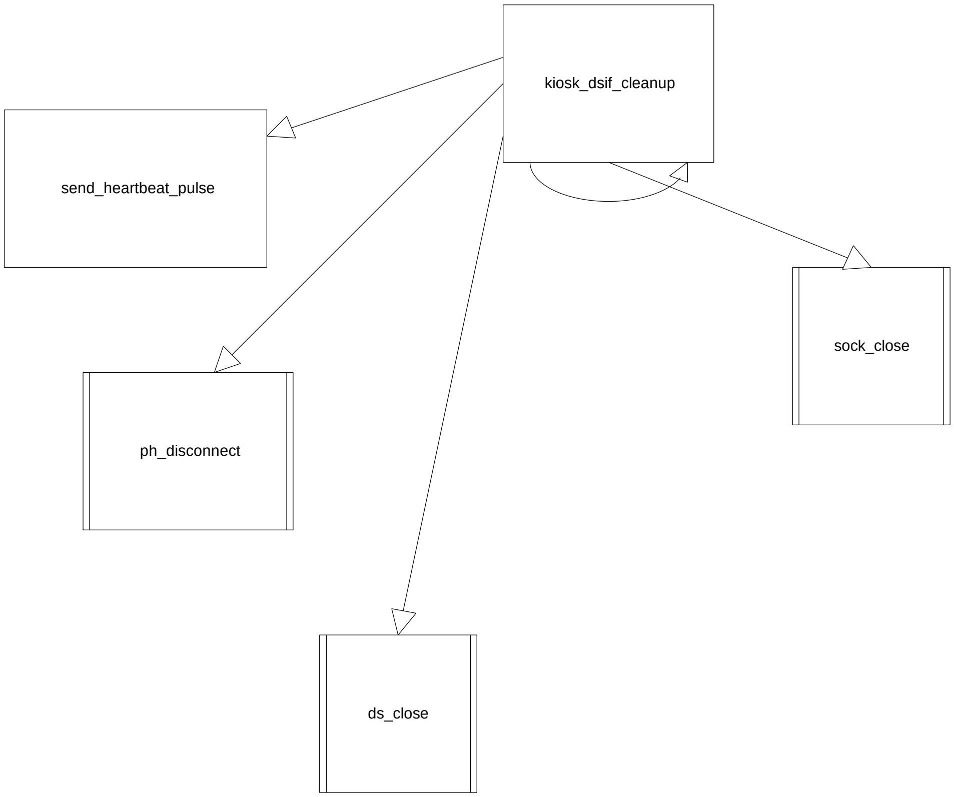

Figure 119 - kiosk_dsif_cleanup Structure Chart

Function	Description
ds_close	MDI Data Server routine used to close the connection to the Data Server.
ph_disconnect	MDI Process Heartbeat routine used to disconnect from the process-level heartbeat service.
send_heartbeat_pulse	Sends the process-level heartbeat to the Subsystem Heartbeat process.
sock_close	MDI Socket routine used to close the specified socket connection.

Table 122 - Routines called by kiosk_dsif_cleanup

4.2.1.5.3 send_heartbeat_pulse

send_heartbeat_pulse sends the process-level heartbeat to the Subsystem Heartbeat process. The structure chart for send_heartbeat_pulse is depicted in Figure 120. A description of the routines called by send_heartbeat_pulse is provided in Table 123.

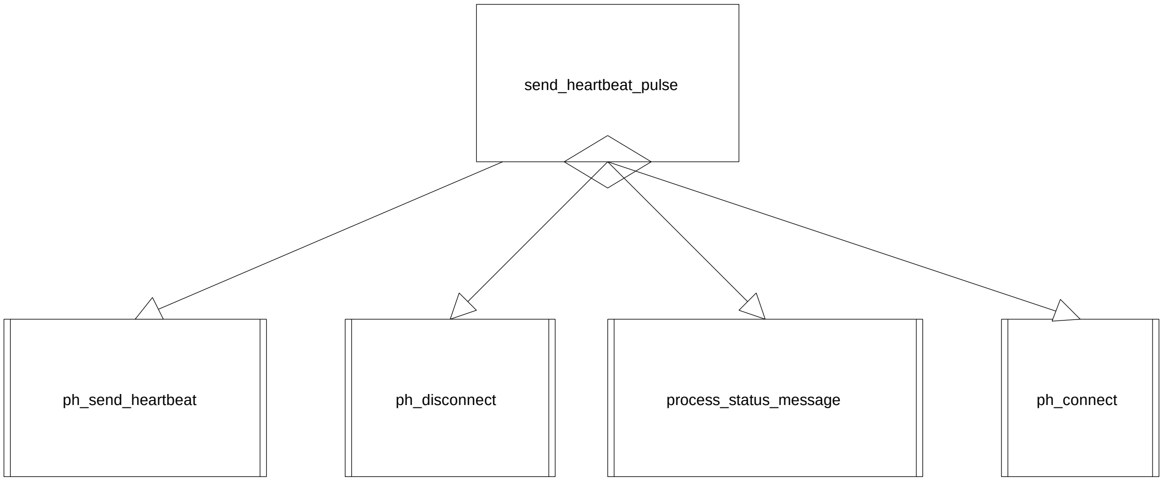

Figure 120 - send_heartbeat_pulse Structure Chart

Function	Description
ph_connect	MDI Process Heartbeat routine used to connect to the specified process-level heartbeat service. The host name and service name are used to make the connection.
ph_disconnect	MDI Process Heartbeat routine used to disconnect from the process-level heartbeat service.
ph_send_heartbeat	MDI Process Heartbeat routine used to send the specified status value to the heartbeat service configured by the ph_connect call.
process_status_message	MDI Process Status routine used to log a status message for the specified status type. If the process status library was configured to use a status logger then the message is forwarded to the status logger. Otherwise the message is written to the configured status log file.

Table 123 - Routines called by send_heartbeat_pulse

4.2.1.5.4 initialize_kiosk_dsif

initialize_kiosk_dsif reads the kiosk_dsif configuration file specified on the command line to obtain the values of the configurable items of the kiosk_dsif process. The structure chart for initialize_kiosk_dsif is depicted in Figure 121. A description of the routines called by initialize_kiosk_dsif is provided in Table 124.

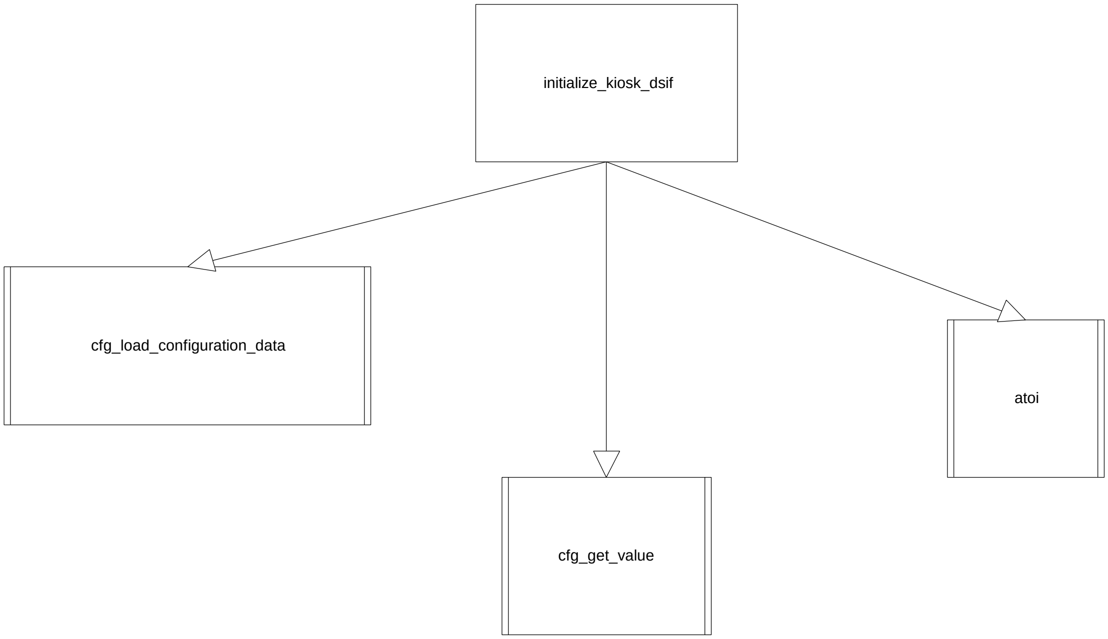

Figure 121 - initialize_kiosk_dsif Structure Chart

Function	Description
atoi	C Library Function to convert an ASCII string to an integer value.
cfg_get_value	MDI Configuration File routine used to return the value of the specified configuration name.
cfg_load_configuration_data	MDI Configuration File routine used to read the configuration name-value pairs from the specified configuration file. These name-value pairs are loaded into memory so they can be accessed on demand by the calling program.

Table 124 - Routines called by initialize_kiosk_dsif

4.2.1.5.5 sigalm_handler

sigalm_handler is the signal handler for the SIGALRM signal. This signal is used to indicate when the process-level heartbeat should be sent to the Kiosk subsystem heartbeat process. The alarm is reinitialized as part of this routine. The structure chart for sigalm_handler is depicted in Figure 122. A description of the routines called by sigalm_handler is provided in Table 125.

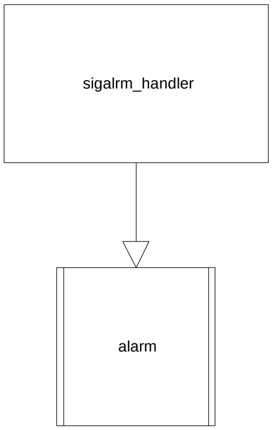

Figure 122 - sigalm_handler Structure Chart

Function	Description
alarm	System Call used to set the alarm clock of the calling process to send a SIGALRM signal after the specified number of seconds have elapsed.

Table 125 - Routines called by sigalm_handler

4.2.1.5.6 respond_to_read_sockets

The respond_to_read_sockets routine loops through the list of socket descriptors ready for reading and either accepts connections, if the socket descriptor is for the listen socket, or receives messages containing information to be sent to the Data Server. The structure chart for respond_to_read_sockets is depicted in Figure 123. A description of the routines called by respond_to_read_sockets is provided in Table 126.

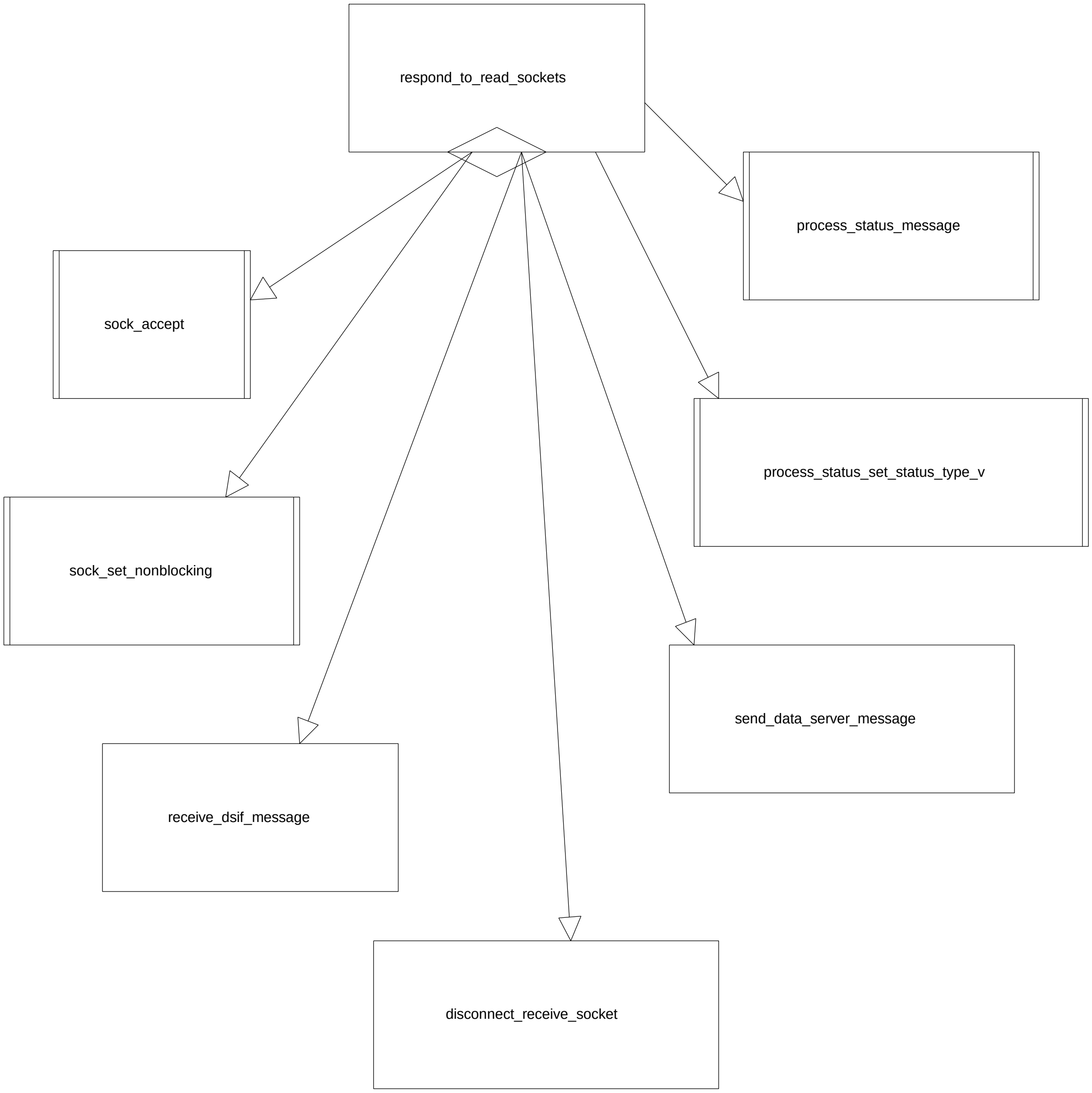

Figure 123 - respond_to_read_sockets Structure Chart

Function	Description
disconnect_receive_socket	Removes the specified socket descriptor from the specified file descriptor set and shuts down and closes the associated socket.
process_status_message	MDI Process Status routine used to log a status message for the specified status type. If the process status library was configured to use a status logger then the message is forwarded to the status logger. Otherwise the message is written to the configured status log file.
process_status_set_status_type_value	process_status_set_status_type_value is used to set the value associated with the specified process status type.
receive_dsif_message	Reads a message from the specified socket. There is no attempt to clear the socket data or try to resynchronize the message data if any errors occur during reading.
send_data_server_message	Extracts the contents of the message and sends the contents on to the Data Server. The message sent to the Data Server could have a response that needs to be read from the Data Server.
sock_accept	MDI Socket routine that accepts connections on the specified listen socket.
sock_set_nonblocking	MDI Socket routine that sets the specified socket to be a non-blocking socket.

Table 126 - Routines called by respond_to_read_sockets

4.2.1.5.7 receive_dsif_message

receive_dsif_message reads a message from the specified socket. There is no attempt to clear the socket data or try to resynchronize the message data if any errors occur during reading. The structure chart for receive_dsif_message is depicted in Figure 124. A description of the routines called by receive_dsif_message is provided in Table 127.

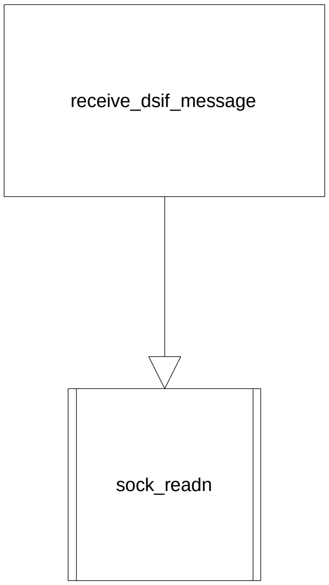

Figure 124 - receive_dsif_message Structure Chart

Function	Description
sock_readn	MDI Socket routine that reads a specified number of bytes from the specified socket.

Table 127 - Routines called by receive_dsif_message

4.2.1.5.8 disconnect_receive_socket

disconnect_receive_socket removes the specified socket descriptor from the specified file descriptor set and shuts down and closes the associated socket. The structure chart for disconnect_receive_socket is depicted in Figure 125. A description of the routines called by disconnect_receive_socket is provided in Table 128.

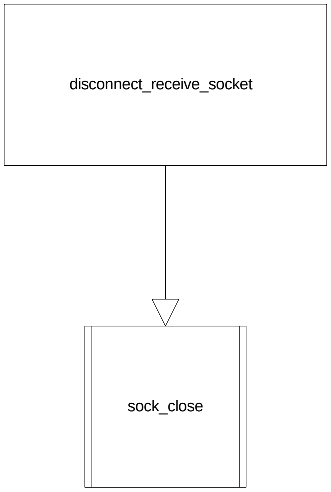

Figure 125 - disconnect_receive_socket Structure Chart

Function	Description
sock_close	MDI Socket routine used to close the specified socket connection.

Table 128 - Routines called by disconnect_receive_socket

4.2.1.5.9 send_data_server_message

The send_data_server_message routine extracts the contents of the message and sends the contents on to the Data Server. The message sent to the Data Server could have a response that needs to be read from the Data Server. The structure chart for send_data_server_message is depicted in Figure 126. A description of the routines called by send_data_server_message is provided in Table 129.

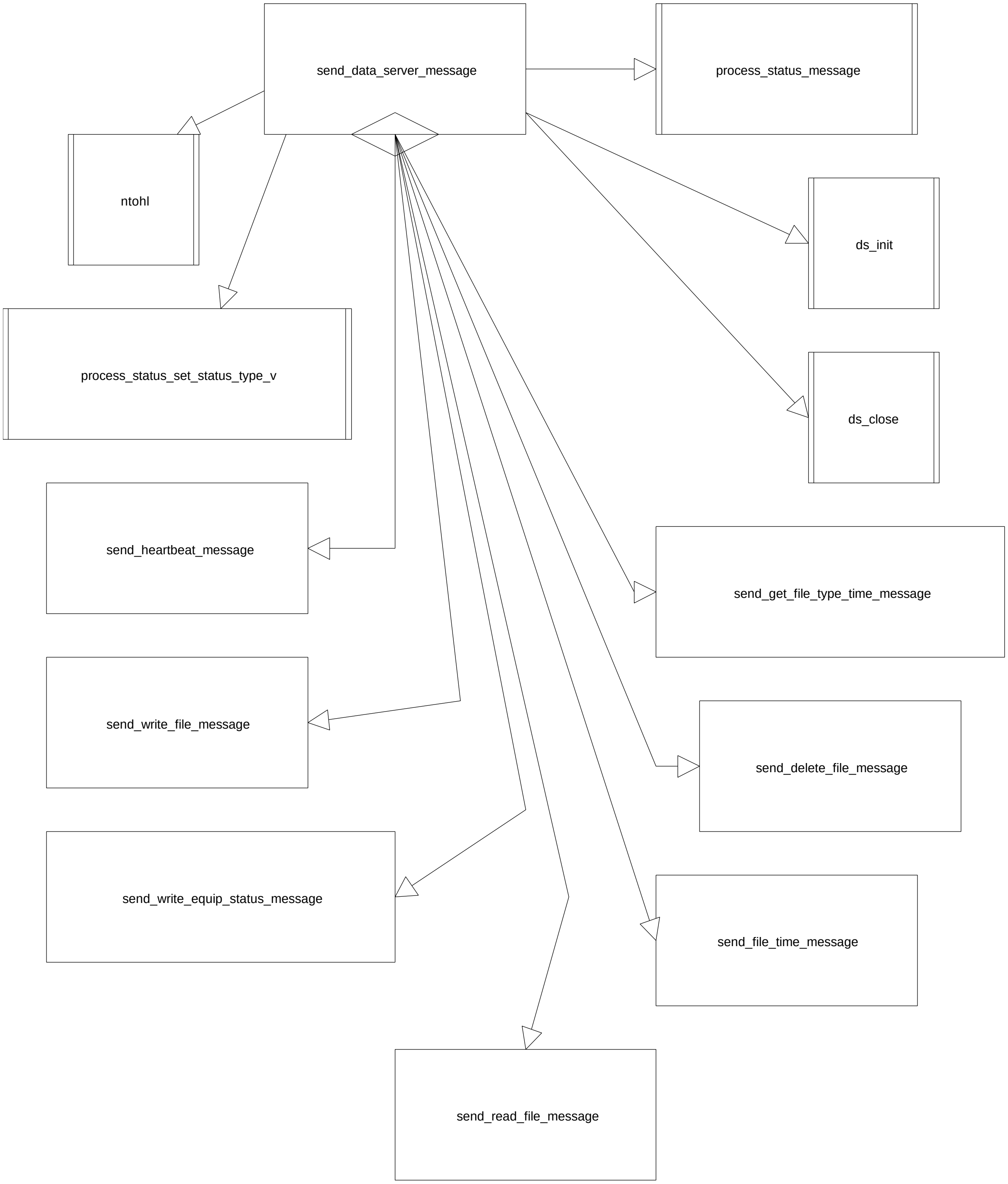

Figure 126 - send_data_server_message Structure Chart

Function	Description
ds_close	MDI Data Server routine used to close the connection to the Data Server.
ds_init	MDI Data Server library routine used to initialize the connection to the Data Server.
ntohl	Network Function used to convert between network and host byte order.
process_status_message	MDI Process Status routine used to log a status message for the specified status type. If the process status library was configured to use a status logger then the message is forwarded to the status logger. Otherwise the message is written to the configured status log file.
process_status_set_status_type_value	process_status_set_status_type_value is used to set the value associated with the specified process status type.
send_delete_file_message	send_delete_file_message extracts the specific information from the received message and calls the data server delete file routine.
send_file_time_message	send_file_time_message extracts the specific information from the received message and calls the data server file time routine.
send_get_file_type_time_message	send_get_file_type_time_message extracts the specific information from the received message and calls the data server get file type time routine.
send_heartbeat_message	send_heartbeat_message extracts the specific information from the received message calls the data server heartbeat routine.
send_read_file_message	send_read_file_message extracts the specific information from the received message and calls the data server read file routine.
send_write_equip_status_message	send_write_equip_status_message extracts the specific information from the received message and calls the data server write equipment status routine.
send_write_file_message	send_write_file_message extracts the specific information from the received message and calls the data server write file routine.

Table 129 - Routines called by send_data_server_message

4.2.1.5.10 send_heartbeat_message

send_heartbeat_message extracts the specific information from the received message and calls the data server heartbeat routine. The structure chart for send_heartbeat_message is depicted in Figure 127. A description of the routines called by send_heartbeat_message is provided in Table 130.

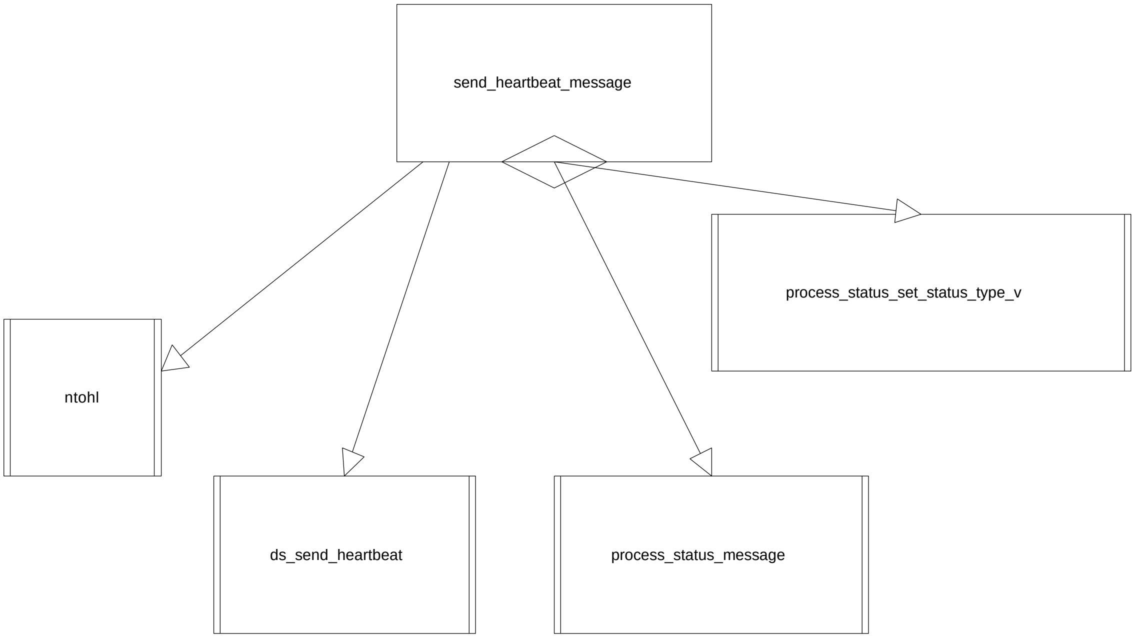

Figure 127 - send_heartbeat_message Structure Chart

Function	Description
ds_send_heartbeat	MDI Data Server routine used to send the subsystem-level heartbeat message to the Data Server. The heartbeat status is the overall status for the subsystem.
ntohl	Network Function used to convert between network and host byte order.
process_status_message	MDI Process Status routine used to log a status message for the specified status type. If the process status library was configured to use a status logger then the message is forwarded to the status logger. Otherwise the message is written to the configured status log file.
process_status_set_status_type_value	process_status_set_status_type_value is used to set the value associated with the specified process status type.

Table 130 - Routines called by send_heartbeat_message

4.2.1.5.11 send_write_file_message

send_write_file_message extracts the specific information from the received message and calls the data server write file routine. The structure chart for send_write_file_message is depicted in Figure 128. A description of the routines called by send_write_file_message is provided in Table 131.

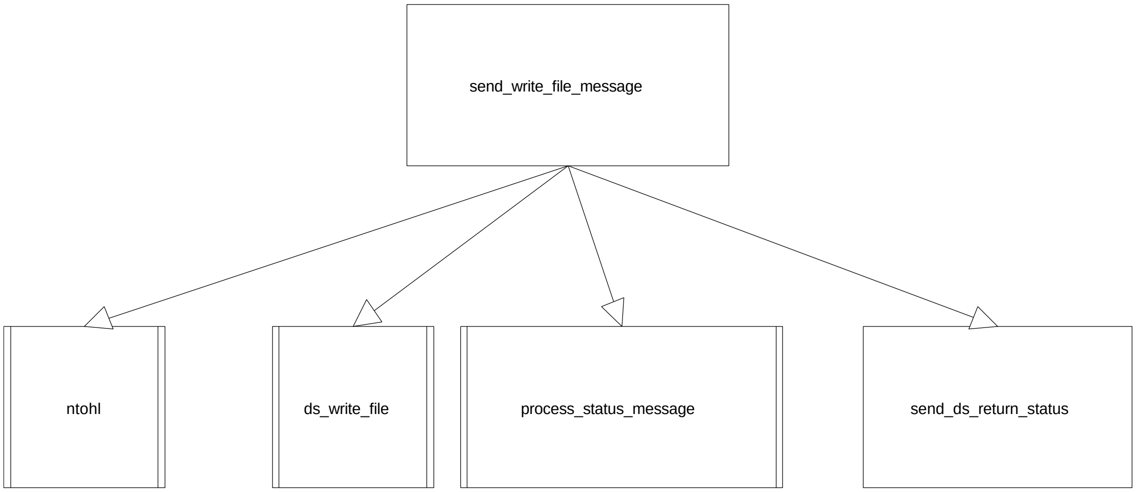

Figure 128 - send_write_file_message Structure Chart

Function	**Description**
ds_write_file	MDI Data Server library routine used to write a file to the Data Server.
ntohl	Network Function used to convert between network and host byte order.
process_status_message	MDI Process Status routine used to log a status message for the specified status type. If the process status library was configured to use a status logger then the message is forwarded to the status logger. Otherwise the message is written to the configured status log file.
send_ds_return_status	send_ds_return_status sends the return status from the data server back to the process that sent the original request to the kiosk_dsif process.

Table 131 - Routines called by send_write_file_message

4.2.1.5.12 send_ds_return_status

send_ds_return_status sends the return status from the data server back to the process that sent the original request to the kiosk_dsif process. The structure chart for send_ds_return_status is depicted in Figure 129. A description of the routines called by send_ds_return_status is provided in Table 132.

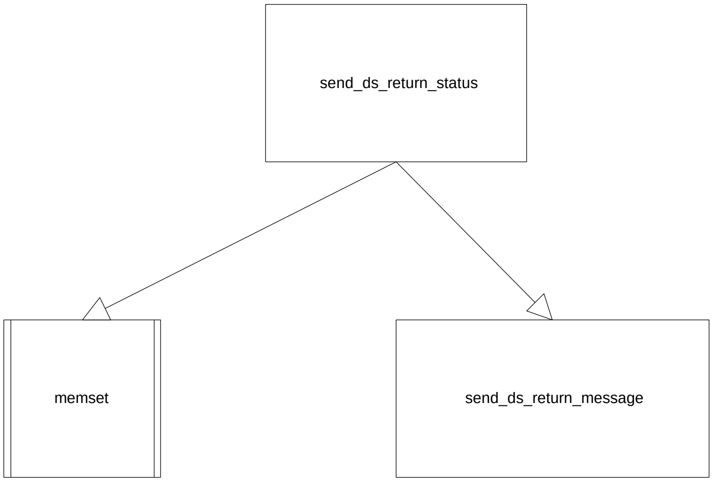

Figure 129 - send_ds_return_status Structure Chart

Function	Description
memset	C Library Function used to set an area of memory to a specified value.
send_ds_return_message	send_ds_return_message sends the information returned by the data server to the process that originally sent the data server message to the kiosk_dsif process.

Table 132 - Routines called by send_ds_return_status

4.2.1.5.13 send_ds_return_message

send_ds_return_message sends the information returned by the data server to the process that originally sent the data server message to the kiosk_dsif process. The structure chart for send_ds_return_message is depicted in Figure 130. A description of the routines called by send_ds_return_message is provided in Table 133.

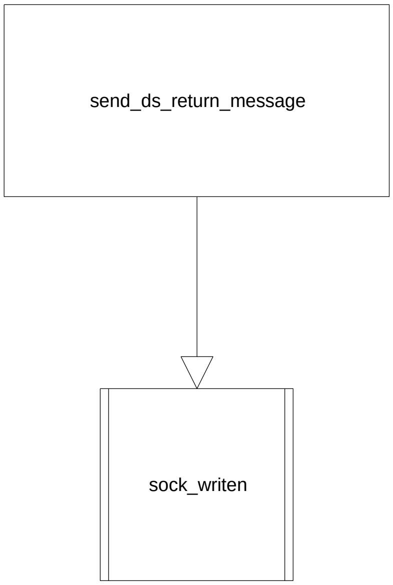

Figure 130 - send_ds_return_message Structure Chart

Function	Description
sock_writen	MDI Socket routine used to write a specified number of bytes to a specified socket.

Table 133 - Routines called by send_ds_return_message

4.2.1.5.14 send_write_equip_status_message

send_write_equip_status_message extracts the specific information from the received message and calls the data server write equipment status routine. The structure chart for send_write_equip_status_message is depicted in Figure 131. A description of the routines called by send_write_equip_status_message is provided in Table 134.

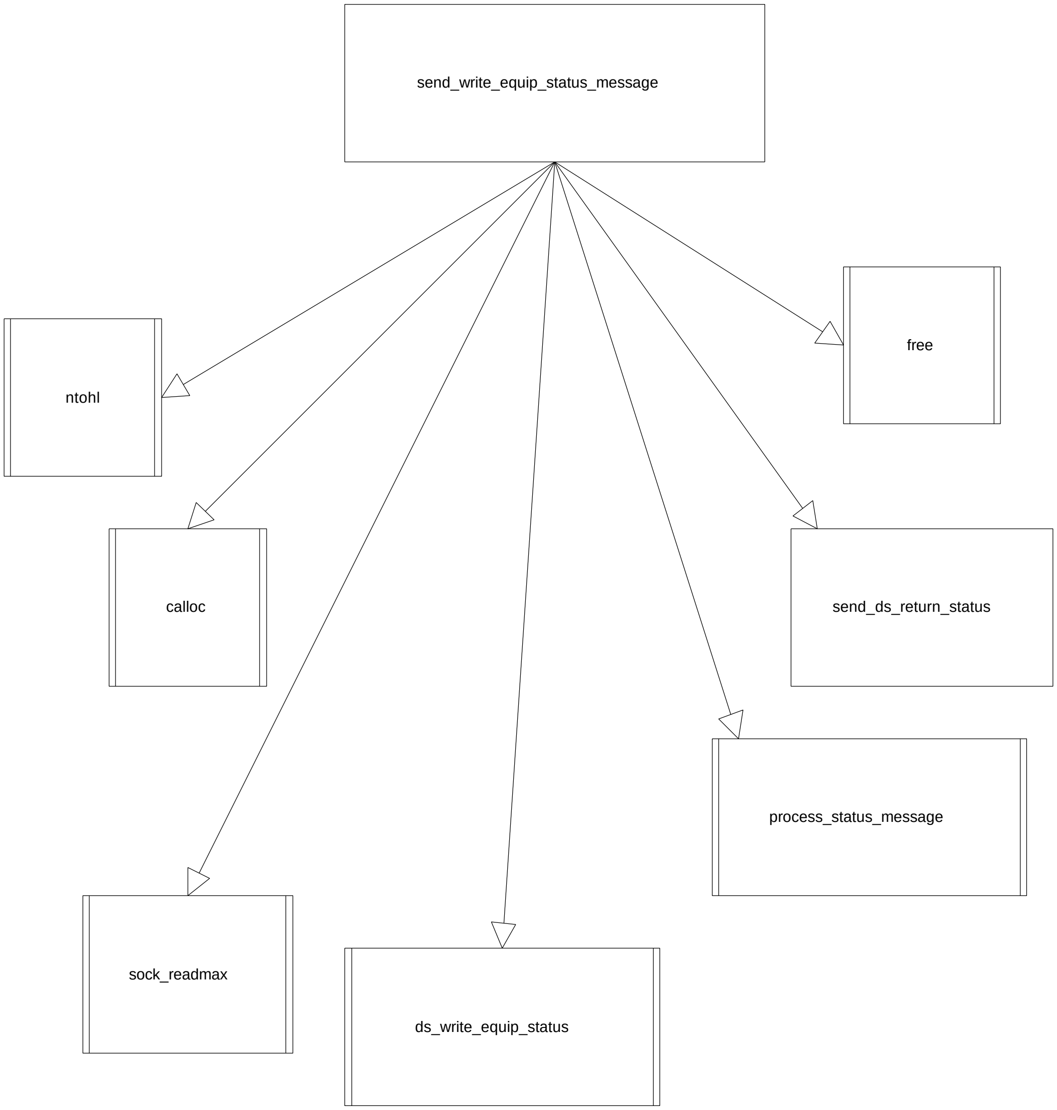

Figure 131 - send_write_equip_status_message Structure Chart

Function	Description
calloc	C Library Function used to allocate the specified amount of space and fill it with zeros.
ds_write_equip_status	MDI Data Server library routine used to write the equipment status of the specified equipment type to the Data Server.
free	C Library Function used to free previously allocated memory and make it available for further allocation.
ntohl	Network Function used to convert between network and host byte order.

Function	Description
process_status_message	MDI Process Status routine used to log a status message for the specified status type. If the process status library was configured to use a status logger then the message is forwarded to the status logger. Otherwise the message is written to the configured status log file.
send_ds_return_status	send_ds_return_status sends the return status from the data server back to the process that sent the original request to the kiosk_dsif process.
sock_readmax	MDI Socket routine that reads a maximum number of bytes from the specified socket using a specified timeout to stop the read when there is no data available for the specified time.

Table 134 - Routines called by send_write_equip_status_message

4.2.1.5.15 send_delete_file_message

send_delete_file_message extracts the specific information from the received message and calls the data server delete file routine. The structure chart for send_delete_file_message is depicted in Figure 132. A description of the routines called by send_delete_file_message is provided in Table 135.

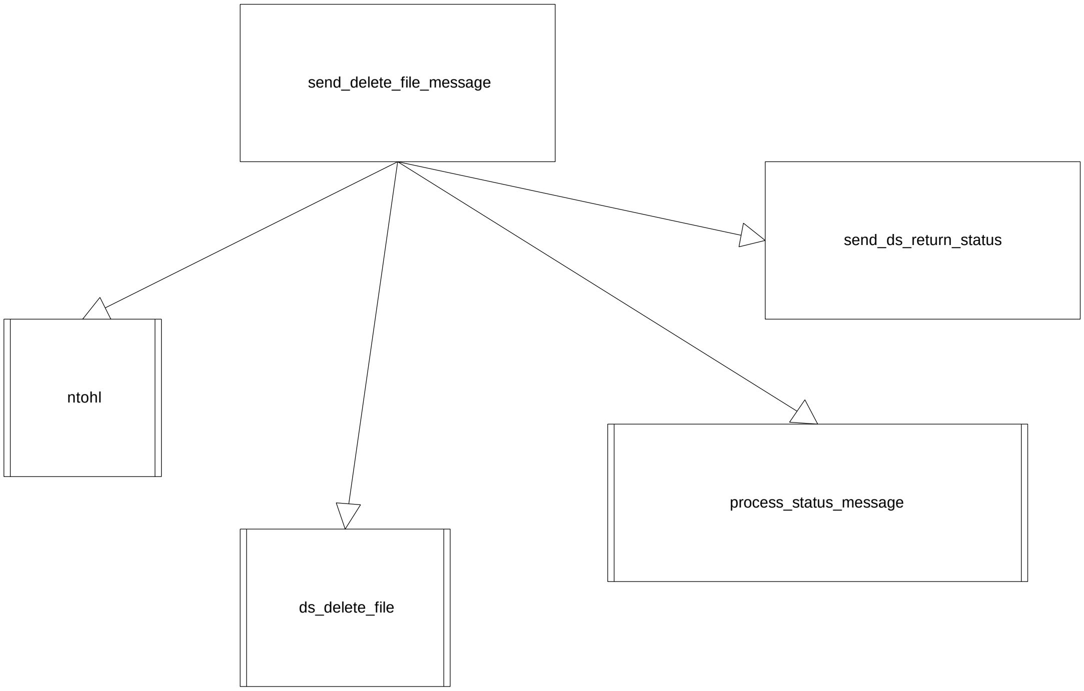

Figure 132 - send_delete_file_message Structure Chart

Function	Description
ds_delete_file	MDI Data Server library routine used to delete the specified file from the Data Server.
ntohl	Network Function used to convert between network and host byte order.
process_status_message	MDI Process Status routine used to log a status message for the specified status type. If the process status library was configured to use a status logger then the message is forwarded to the status logger. Otherwise the message is written to the configured status log file.
send_ds_return_status	send_ds_return_status sends the return status from the data server back to the process that sent the original request to the kiosk_dsif process.

Table 135 - Routines called by send_delete_file_message

4.2.1.5.16 send_file_time_message

send_file_time_message extracts the specific information from the received message and calls the data server file time routine. The structure chart for send_file_time_message is depicted in Figure 133. A description of the routines called by send_file_time_message is provided in Table 136.

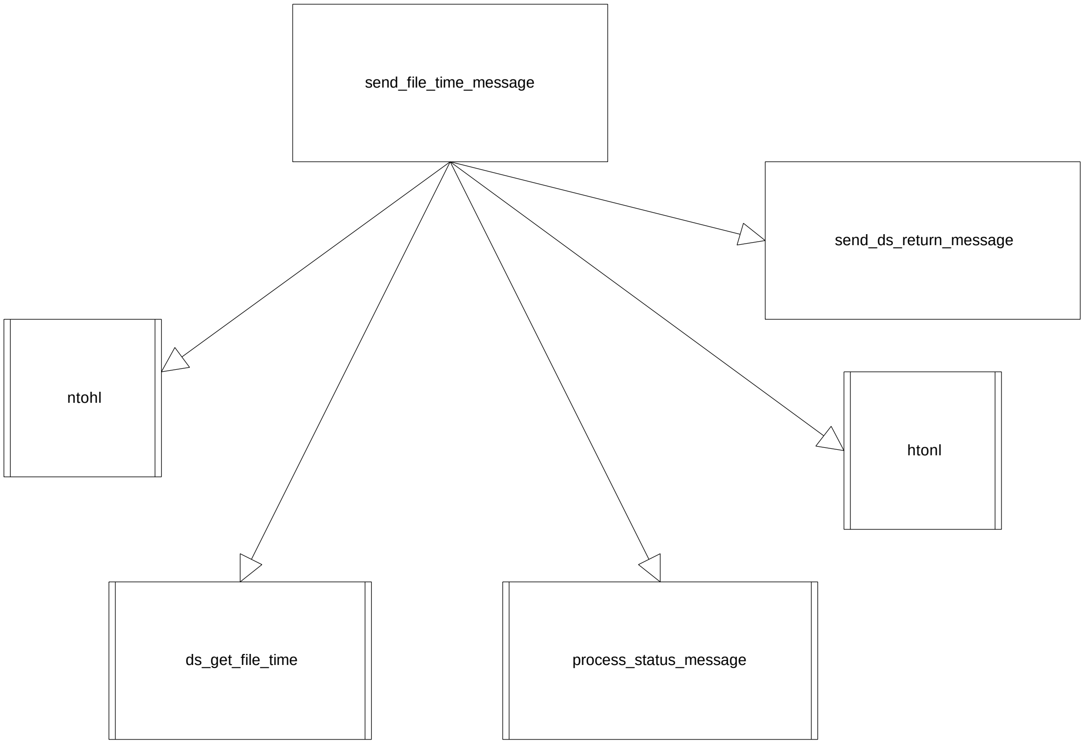

Figure 133 - send_file_time_message Structure Chart

Function	Description
ds_get_file_time	MDI Data Server library routine used to obtain the modification time of the specified file located on the Data Server.
htonl	Network function used to convert from host to network byte formats.
ntohl	Network Function used to convert between network and host byte order.
process_status_message	MDI Process Status routine used to log a status message for the specified status type. If the process status library was configured to use a status logger then the message is forwarded to the status logger. Otherwise the message is written to the configured status log file.
send_ds_return_message	send_ds_return_message sends the information returned by the data server to the process that originally sent the data server message to the kiosk_dsif process.

Table 136 - Routines called by send_file_time_message

4.2.1.5.17 send_get_file_type_time_message

send_get_file_type_time_message extracts the specific information from the received message and calls the data server get file type time routine. The structure chart for send_get_file_type_time_message is depicted in Figure 134. A description of the routines called by send_get_file_type_time_message is provided in Table 137.

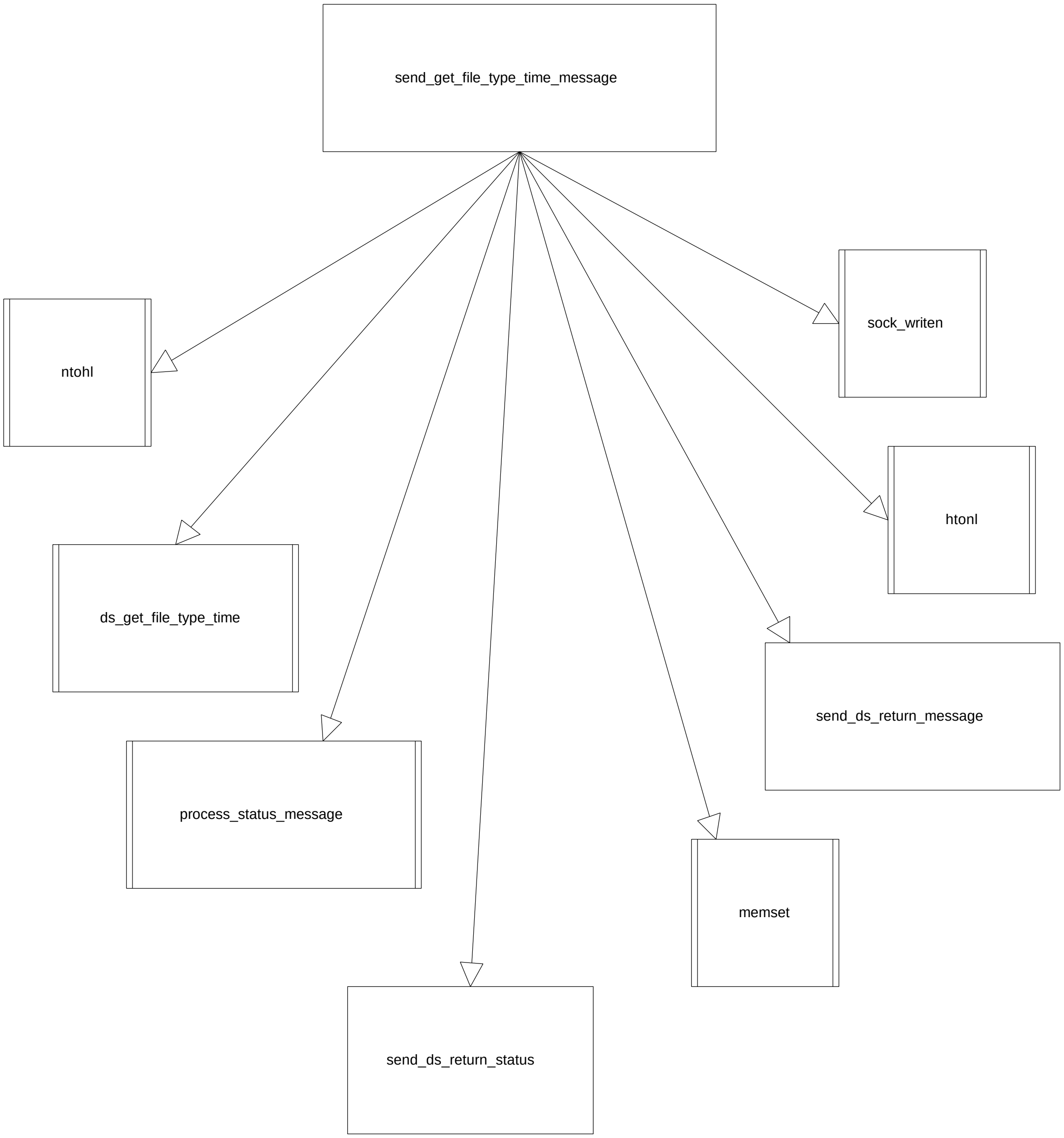

Figure 134 - send_get_file_type_time_message Structure Chart

Function	Description
ds_get_file_type_time	MDI Data Server library routine used to obtain the file modification times for all the files of the specified file type that exist at the Data Server.
htonl	Network function used to convert from host to network byte formats.
memset	C Library Function used to set an area of memory to a specified value.
ntohl	Network Function used to convert between network and host byte order.
process_status_message	MDI Process Status routine used to log a status message for the specified status type. If the process status library was configured to use a status logger then the message is forwarded to the status logger. Otherwise the message is written to the configured status log file.
send_ds_return_message	Send_ds_return_message sends the information returned by the data server to the process that originally sent the data server message to the kiosk_dsif process.
send_ds_return_status	Send_ds_return_status sends the return status from the data server back to the process that sent the original request to the kiosk_dsif process.
sock_writen	MDI Socket routine used to write a specified number of bytes to a specified socket.

Table 137 - Routines called by send_get_file_type_time_message

4.2.1.5.18 send_read_file_message

send_read_file_message extracts the specific information from the received message and calls the data server read file routine. The structure chart for send_read_file_message is depicted in Figure 135. A description of the routines called by send_read_file_message is provided in Table 138.

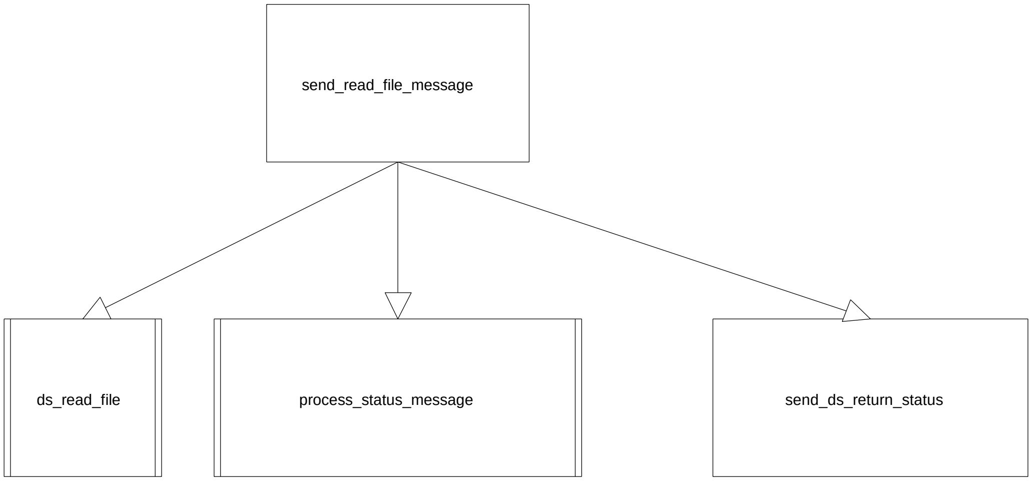

Figure 135 - send_read_file_message Structure Chart

Function	Description
ds_read_file	MDI Data Server library routine used to read the contents of the specified file from the Data Server and place the contents in the specified local file.
process_status_message	MDI Process Status routine used to log a status message for the specified status type. If the process status library was configured to use a status logger then the message is forwarded to the status logger. Otherwise the message is written to the configured status log file.
send_ds_return_status	send_ds_return_status sends the return status from the data server back to the process that sent the original request to the kiosk_dsif process.

Table 138 - Routines called by send_read_file_message

4.2.1.6 Data Server Interface Library

The DSIF Library is composed of a set of functions that can be used by other KMC applications to communicate with the Data Server through DSIF. The following list contains the DSIF Library functions.

- kiosk_dsif_connect,
- kiosk_dsif_delete_file,
- kiosk_dsif_disconnect,
- kiosk_dsif_get_file_time,
- kiosk_dsif_get_file_type_time,
- kiosk_dsif_read_file,
- kiosk_dsif_read_file_list,
- kiosk_dsif_read_status,
- kiosk_dsif_read_timestamp,
- kiosk_dsif_send_heartbeat,
- kiosk_dsif_write_equip_status, and
- kiosk_dsif_write_file.

The functions are described in the sections that follow.

4.2.1.6.1 kiosk_dsif_connect

kiosk_dsif_connect is used to connect to the KIOSK Data Server Interface process. The service name is passed to this routine and is used to make the connection to the appropriate port. The structure chart for kiosk_dsif_connect is depicted in Figure 136. A description of the routines called by kiosk_dsif_connect is provided in Table 139.

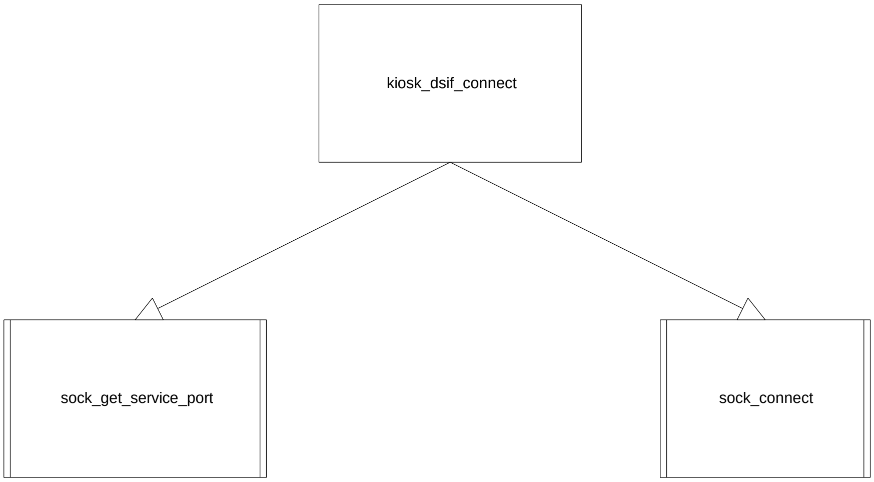

Figure 136 - kiosk_dsif_connect Structure Chart

Function	Description
sock_connect	MDI Socket routine used to create a socket connection to the specified host and port.
sock_get_service_port	MDI Socket routine that returns the port number associated with the specified service name.

Table 139 - Routines called by kiosk_dsif_connect

4.2.1.6.2 kiosk_dsif_delete_file

kiosk_dsif_delete_file packages the delete file data into the appropriate kiosk_dsif message and sends it to the kiosk_dsif process. The structure chart for kiosk_dsif_delete_file is depicted in Figure 137. A description of the routines called by kiosk_dsif_delete_file is provided in Table 140.

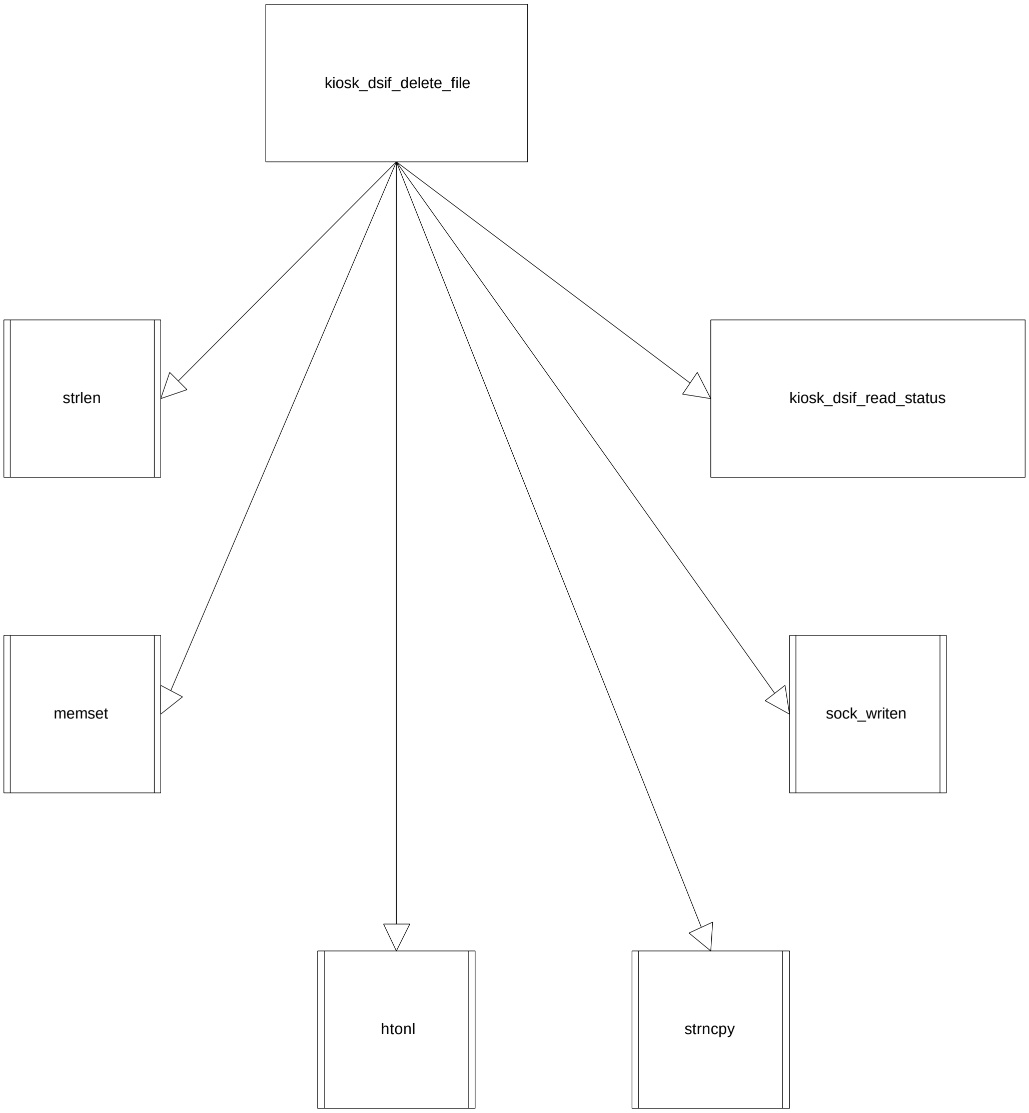

Figure 137 - kiosk_dsif_delete_file Structure Chart

Function	Description
htonl	Network function used to convert from host to network byte formats.
kiosk_dsif_read_status	kiosk_dsif_read_status reads the return status from the Data Server request.
memset	C Library Function used to set an area of memory to a specified value.
sock_written	MDI Socket routine used to write a specified number of bytes to a specified socket.
strlen	UNIX system call that computes the number of characters in a NULL terminated string.
strncpy	C Library Function used to copy a specified number of characters from a source string to a destination string.

Table 140 - Routines called by kiosk_dsif_delete_file

4.2.1.6.3 kiosk_dsif_disconnect

kiosk_dsif_disconnect is used to disconnect a process from the kiosk_dsif process. The structure chart for kiosk_dsif_disconnect is depicted in Figure 138. A description of the routines called by kiosk_dsif_disconnect is provided in Table 141.

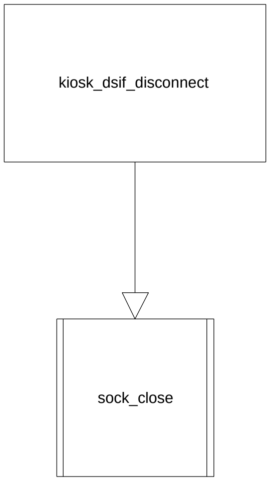

Figure 138 - kiosk_dsif_disconnect Structure Chart

Function	Description
sock_close	MDI Socket routine used to close the specified socket connection.

Table 141 - Routines called by kiosk_dsif_disconnect

4.2.1.6.4 kiosk_dsif_get_file_time

kiosk_dsif_get_file_time packages the get file time data into the appropriate kiosk_dsif message and sends it to the kiosk_dsif process. The structure chart for kiosk_dsif_get_file_time is depicted in Figure 139. A description of the routines called by kiosk_dsif_get_file_time is provided in Table 142.

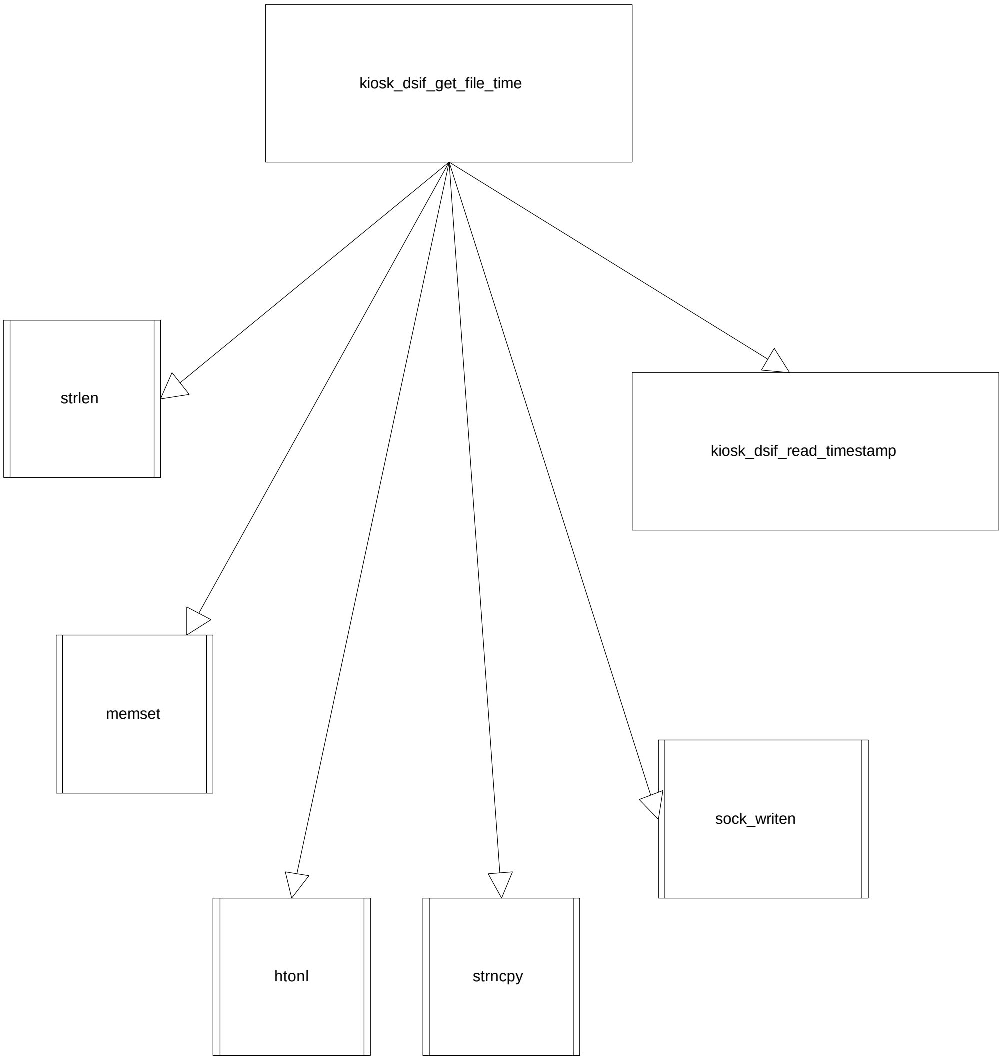

Figure 139 - kiosk_dsif_get_file_time Structure Chart

Function	Description
htonl	Network function used to convert from host to network byte formats.
kiosk_dsif_read_timestamp	kiosk_dsif_read_timestamp reads the returned file time message containing the timestamp returned by the Data Server.
memset	C Library Function used to set an area of memory to a specified value.
sock_writen	MDI Socket routine used to write a specified number of bytes to a specified socket.
strlen	UNIX system call that computes the number of characters in a NULL terminated string.
strncpy	C Library Function used to copy a specified number of characters from a source string to a destination string.

Table 142 - Routines called by kiosk_dsif_get_file_time

4.2.1.6.5 kiosk_dsif_get_file_type_time

kiosk_dsif_get_file_type_time packages the file type time request into the appropriate kiosk_dsif message and sends it to the kiosk_dsif process. Memory is allocated for the file list within this routine, but must be freed by the caller. The structure chart for kiosk_dsif_get_file_type_time is depicted in Figure 140. A description of the routines called by kiosk_dsif_get_file_type_time is provided in Table 143.

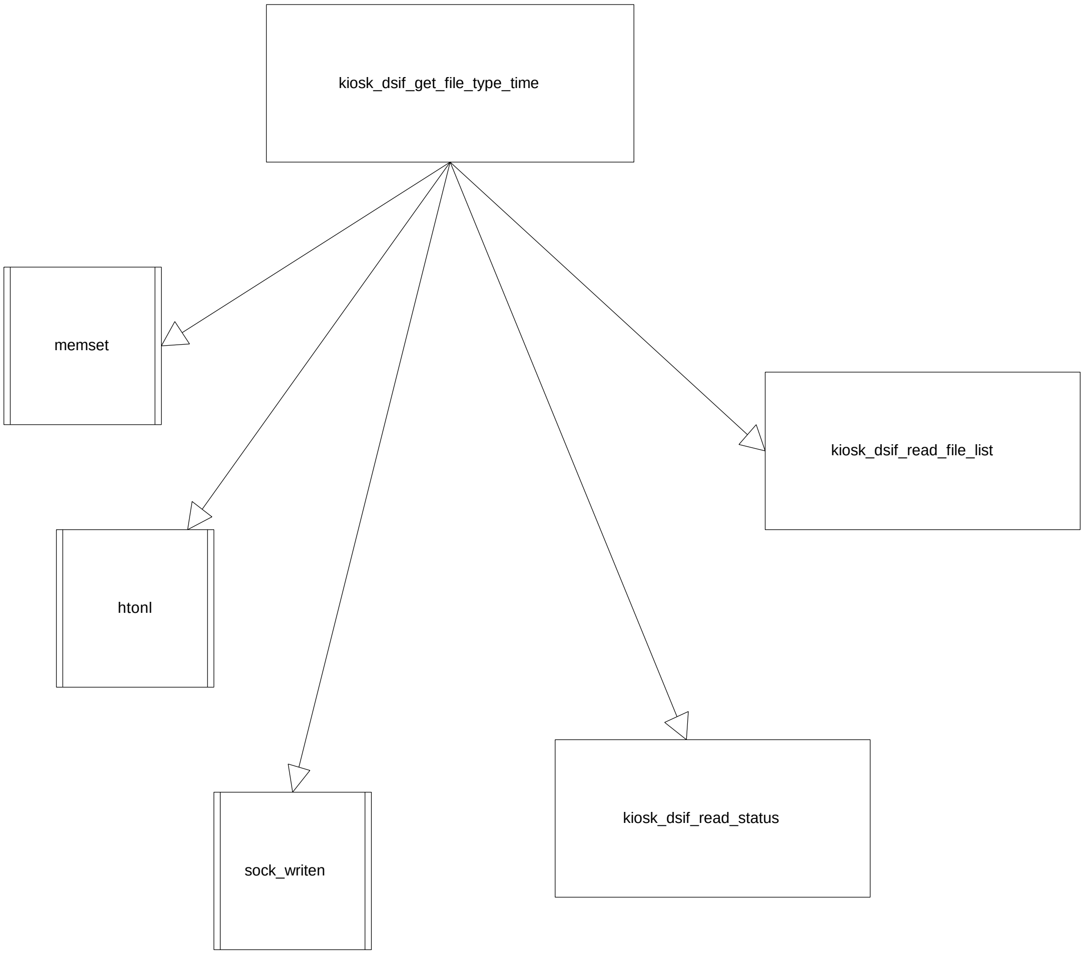

Figure 140 - kiosk_dsif_get_file_type_time Structure Chart

Function	Description
Htonl	Network function used to convert from host to network byte formats.
kiosk_dsif_read_file_list	kiosk_dsif_read_file_list reads the file list information from the kiosk_dsif socket connection. This information includes the number of files in the list and the information for each file in the list.
kiosk_dsif_read_status	kiosk_dsif_read_status reads the return status from the Data Server request.
memset	C Library Function used to set an area of memory to a specified value.
sock_writen	MDI Socket routine used to write a specified number of bytes to a specified socket.

Table 143 - Routines called by kiosk_dsif_get_file_type_time

4.2.1.6.6 kiosk_dsif_read_file

kiosk_dsif_read_file packages the read file request into the appropriate kiosk_dsif message and sends it to the kiosk_dsif process. The structure chart for kiosk_dsif_read_file is depicted in Figure 141. A description of the routines called by kiosk_dsif_read_file is provided in Table 144.

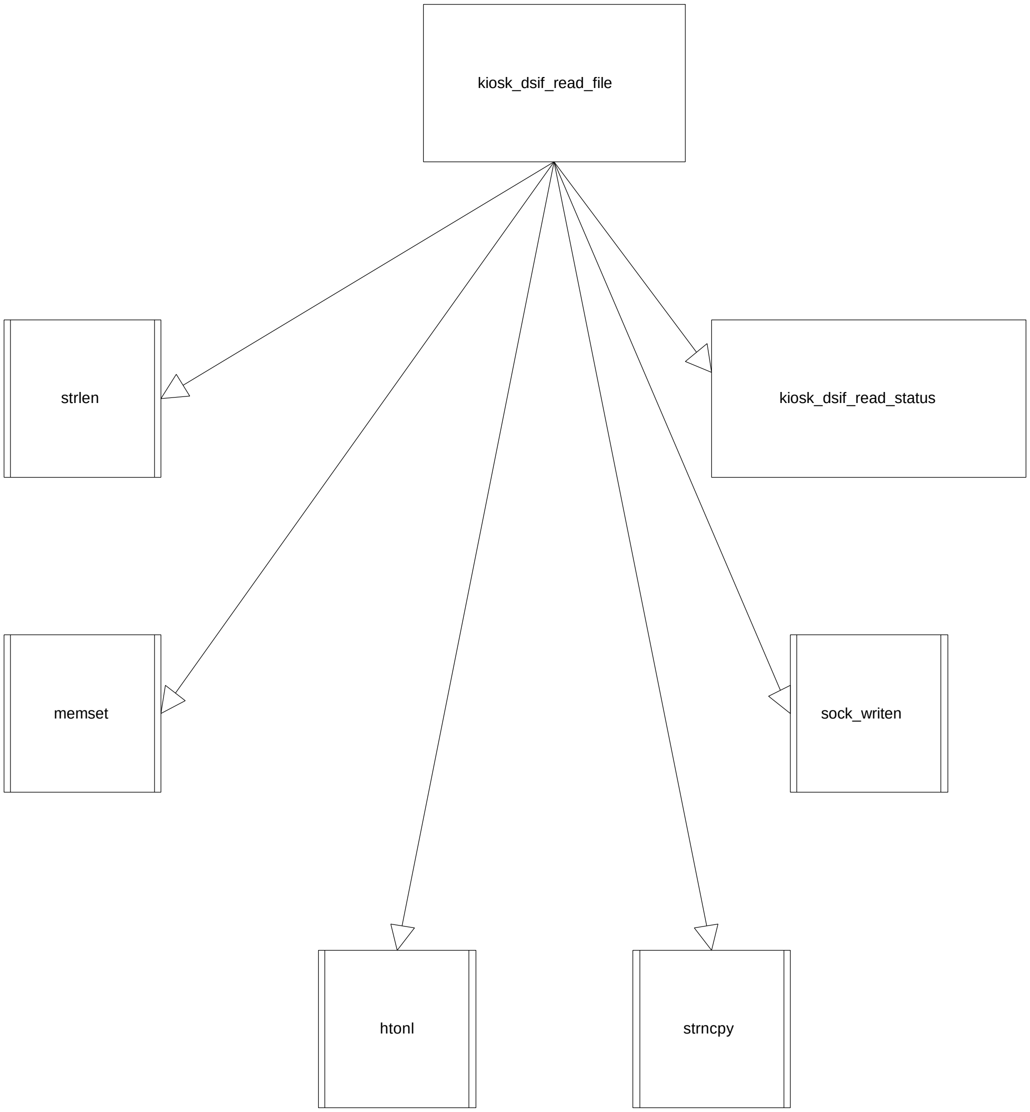

Figure 141 - kiosk_dsif_read_file Structure Chart

Function	Description
htonl	Network function used to convert from host to network byte formats.
kiosk_dsif_read_status	kiosk_dsif_read_status reads the return status from the Data Server request.
memset	C Library Function used to set an area of memory to a specified value.
sock_writen	MDI Socket routine used to write a specified number of bytes to a specified socket.

Function	Description
strlen	UNIX system call that computes the number of characters in a NULL terminated string.
strncpy	C Library Function used to copy a specified number of characters from a source string to a destination string.

Table 144 - Routines called by kiosk_dsif_read_file

4.2.1.6.7 kiosk_dsif_read_file_list

kiosk_dsif_read_file_list reads the file list information from the kiosk_dsif socket connection. This information includes the number of files in the list and the information for each file in the list. The structure chart for kiosk_dsif_read_file_list is depicted in Figure 142. A description of the routines called by kiosk_dsif_read_file_list is provided in Table 145.

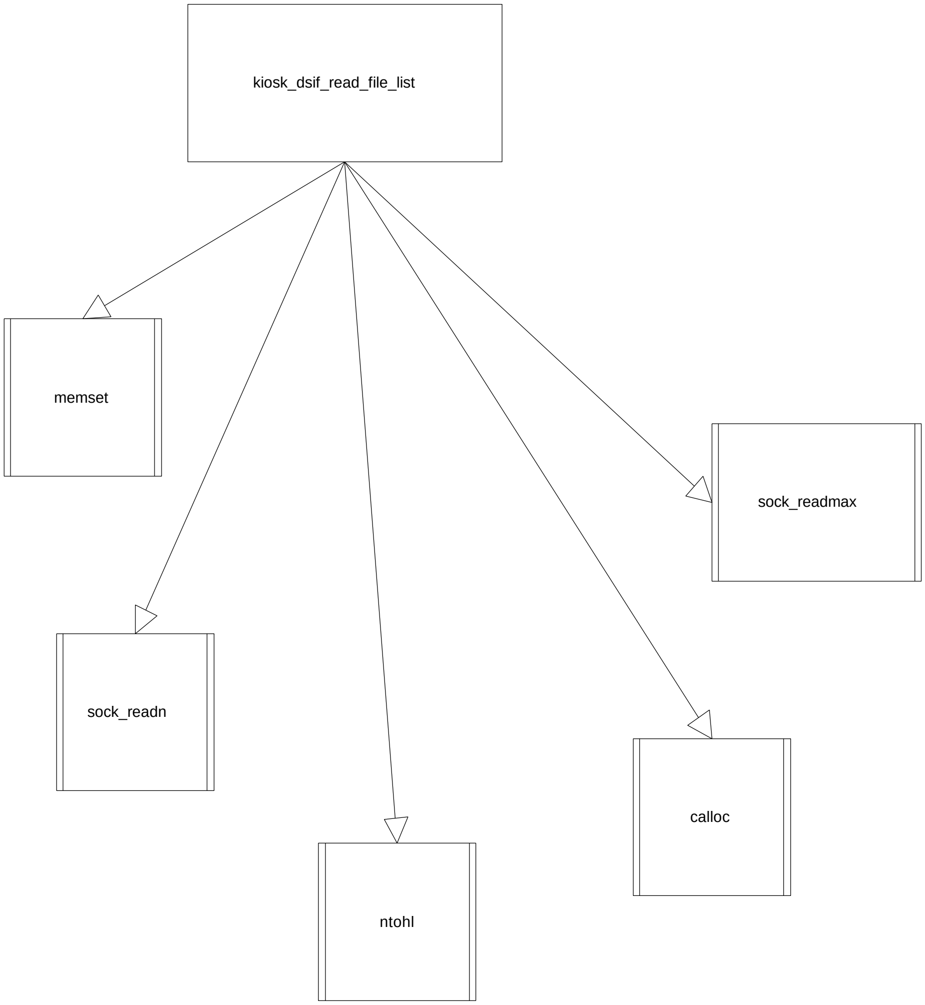

Figure 142 - kiosk_dsif_read_file_list Structure Chart

Function	Description
calloc	C Library Function to allocate the specified amount of space and fill it with zeros.
memset	C Library Function used to set an area of memory to a specified value.
ntohl	Network Function used to convert between network and host byte order.
sock_readmax	MDI Socket routine that reads a maximum number of bytes from the specified socket using a specified timeout to stop the read when there is no data available for the specified time.
sock_readn	MDI Socket routine that reads a specified number of bytes from the specified socket.

Table 145 - Routines called by kiosk_dsif_read_file_list

4.2.1.6.8 kiosk_dsif_read_status

kiosk_dsif_read_status reads the return status from the Data Server request. The structure chart for kiosk_dsif_read_status is depicted in Figure 143. A description of the routines called by kiosk_dsif_read_status is provided in Table 146.

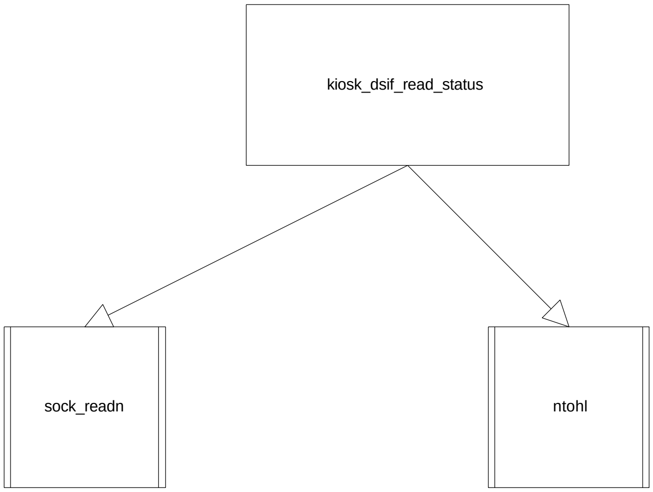

Figure 143 - kiosk_dsif_read_status Structure Chart

Function	Description
ntohl	Network Function used to convert between network and host byte order.
sock_readn	MDI Socket routine that reads a specified number of bytes from the specified socket.

Table 146 - Routines called by kiosk_dsif_read_status

4.2.1.6.9 kiosk_dsif_read_timestamp

kiosk_dsif_read_timestamp reads the returned file time message containing the timestamp returned by the Data Server. The structure chart for kiosk_dsif_read_timestamp is depicted in Figure 144. A description of the routines called by kiosk_dsif_read_timestamp is provided in Table 147.

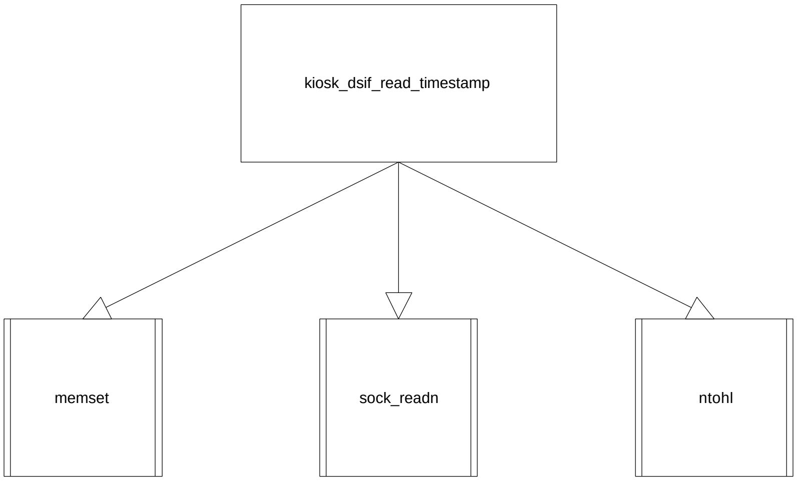

Figure 144 - kiosk_dsif_read_timestamp Structure Chart

Function	Description
memset	C Library Function used to set an area of memory to a specified value.
ntohl	Network Function used to convert between network and host byte order.
sock_readn	MDI Socket routine that reads a specified number of bytes from the specified socket.

Table 147 - Routines called by kiosk_dsif_read_timestamp

4.2.1.6.10 kiosk_dsif_send_heartbeat

kiosk_dsif_send_heartbeat packages the heartbeat message data into the appropriate kiosk_dsif message and sends it to the kiosk_dsif process. The structure chart for kiosk_dsif_send_heartbeat is depicted in Figure 145. A description of the routines called by kiosk_dsif_send_heartbeat is provided in Table 148.

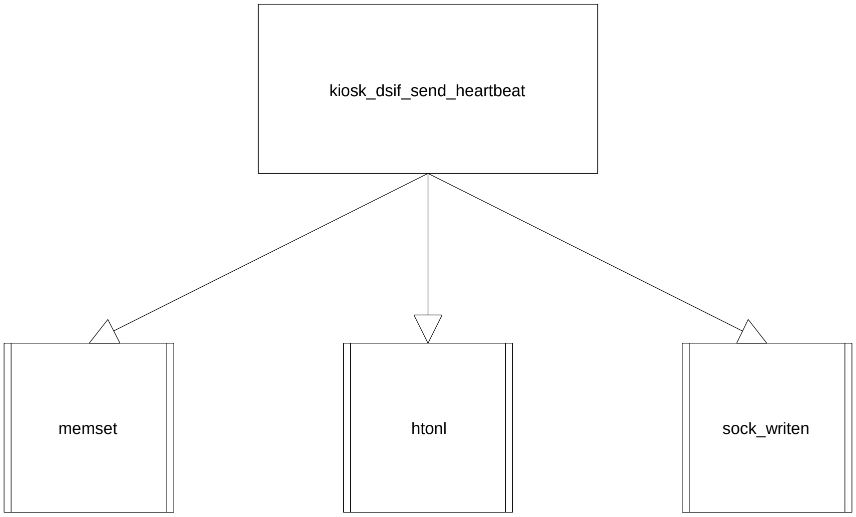

Figure 145 - kiosk_dsif_send_heartbeat Structure Chart

Function	Description
htonl	Network function used to convert from host to network byte formats.
memset	C Library Function used to set an area of memory to a specified value.
sock_writen	MDI Socket routine used to write a specified number of bytes to a specified socket.

Table 148 - Routines called by kiosk_dsif_send_heartbeat

4.2.1.6.11 kiosk_dsif_write_equip_status

kiosk_dsif_write_equip_status packages the write equipment status request into the appropriate kiosk_dsif message and sends it to the kiosk_dsif process. The structure chart for kiosk_dsif_write_equip_status is depicted in Figure 146. A description of the routines called by kiosk_dsif_write_equip_status is provided in Table 149.

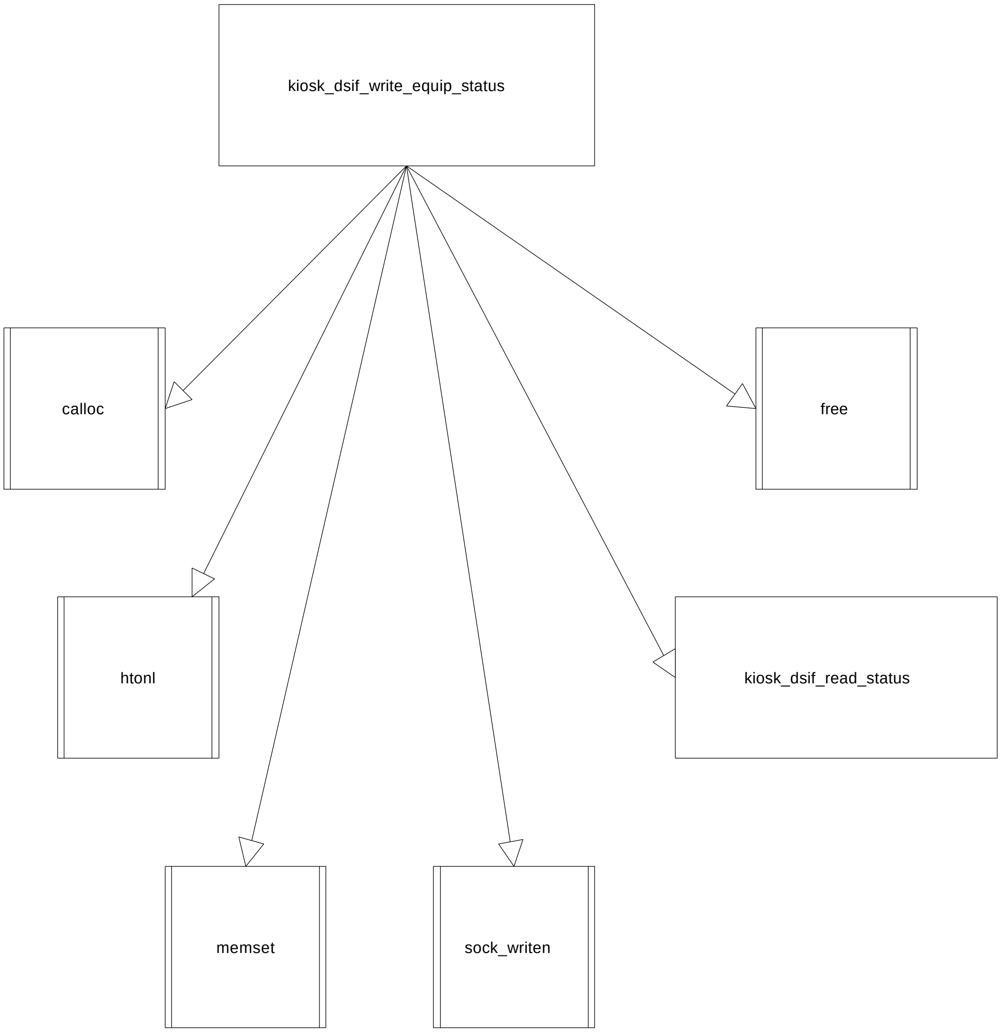

Figure 146 - kiosk_dsif_write_equip_status Structure Chart

Function	Description
calloc	C Library Function used to allocate the specified amount of space and fill it with zeros.
free	C Library Function used to free previously allocated memory and make it available for further allocation.
htonl	Network function used to convert from host to network byte formats.
kiosk_dsif_read_status	kiosk_dsif_read_status reads the return status from the Data Server request.
memset	C Library Function used to set an area of memory to a specified value.
sock_writen	MDI Socket routine used to write a specified number of bytes to a specified socket.

Table 149 - Routines called by kiosk_dsif_write_equip_status

4.2.1.6.12 kiosk_dsif_write_file

kiosk_dsif_write_file packages the write file data into the appropriate kiosk_dsif message and sends it to the kiosk_dsif process. The structure chart for kiosk_dsif_write_file is depicted in Figure 147. A description of the routines called by kiosk_dsif_write_file is provided in Table 150.

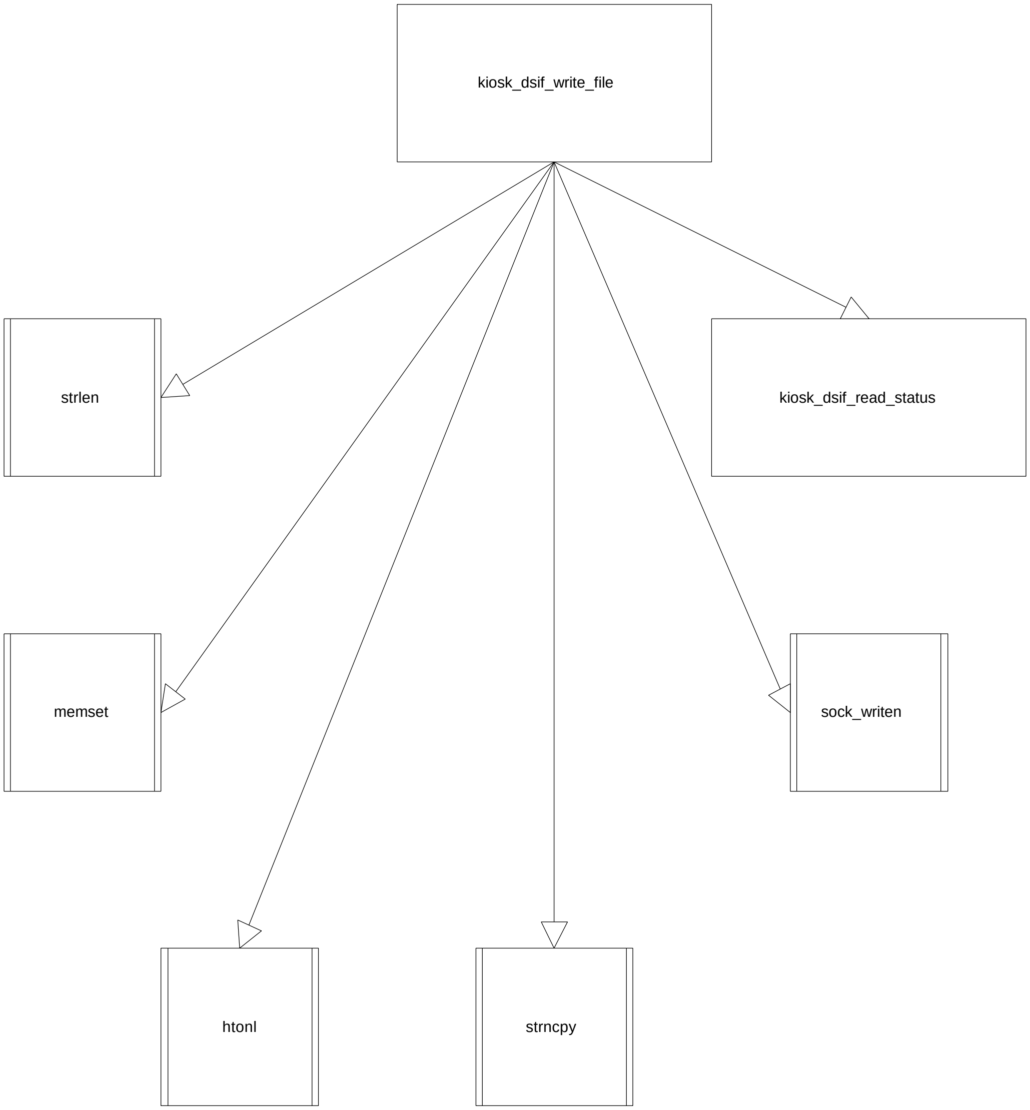

Figure 147 - kiosk_dsif_write_file Structure Chart

Function	Description
htonl	Network function used to convert from host to network byte formats.
kiosk_dsif_read_status	Kiosk_dsif_read_status reads the return status from the Data Server request.
memset	C Library Function used to set an area of memory to a specified value.
sock_writen	MDI Socket routine used to write a specified number of bytes to a specified socket.
strlen	UNIX system call that computes the number of characters in a NULL terminated string.
strncpy	C Library Function used to copy a specified number of characters from a source string to a destination string.

Table 150 - Routines called by kiosk_dsif_write_file

4.2.1.7 Kiosk MC Main

The Kiosk MC Main requests data files from the Data Server (via DSIF) and transmits the files to each of the Kiosk Field Units. The Kiosk MC Main also retrieves status information from each of the Kiosk Field Units. The following structure charts and tables provide the design information for the Kiosk MC Main. Figure 148 depicts the Kiosk MC Main data flows.

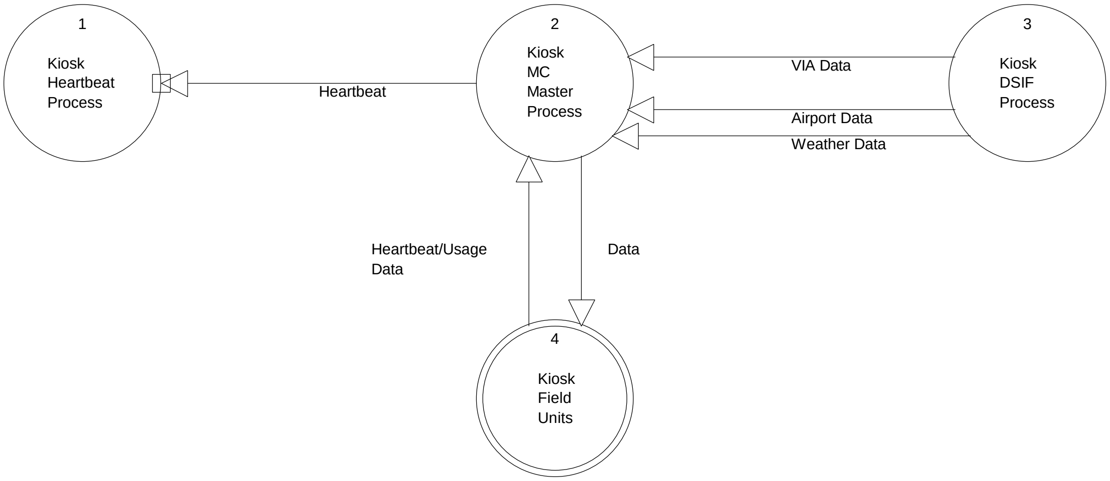

Figure 148 – Kiosk MC Main Data Flows

The VIA, Airport, and Weather files are transmitted to the Kiosk MC Master Process from the DSIF process. These files are then transmitted to the Kiosk Field Units. The Field Unit Heartbeat/Usage data is retrieved from the Field Units and used to update the Detailed Status GUI. The Kiosk MC Master Process periodically reports its own heartbeat data to the Kiosk Heartbeat Process.

4.2.1.7.1 Kiosk MC Main

The Kiosk Master Computer (MC) main process is responsible for interrogating the status of each Kiosk field unit as well as downloading any files that need to be resident on the Kiosk field unit. The main process maintains the current status (of the files on each field unit) using the MDI configuration routines (by storing a filename and a timestamp). This information is kept in configuration files, which are named in the same format as the name of the Kiosk field unit. The main process contains a loop that executes forever checking to see if the Data Server has new files that need to be downloaded, if so, these files are transfered to each active Kiosk field unit. The structure chart for Kiosk MC Main is depicted in Figure 149. A description of the routines called by Kiosk MC Main is provided in Table 151.

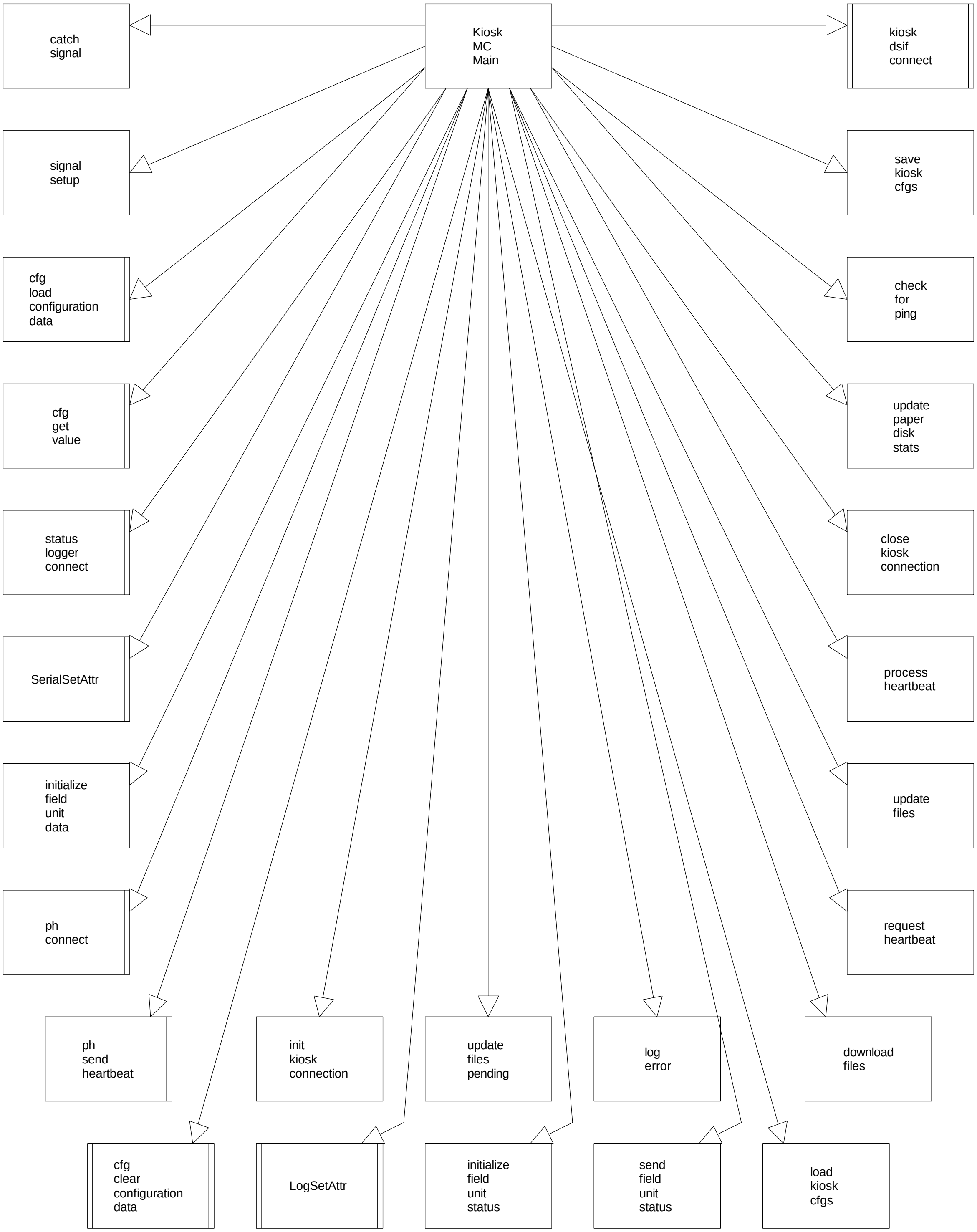

Figure 149 - Kiosk MC Main Structure Chart

Function	Description
catch signal	Kiosk MC function which is invoked when UNIX sends a signal to the Kiosk Master Computer Application.
cfg clear configuration data	MDI Configuration File Common Library routine used to clear the configuration name-pairs loaded from memory.
cfg get value	MDI Configuration File Common Library routine used to return the value of the specified configuration name.
cfg load configuration data	MDI Configuration File Common Library routine used to read the configuration name-pairs from the specified configuration file. These name-value pairs are loaded into memory so they can be accessed on demand by the calling program.
check for ping	Kiosk MC function which will check to see if a "ping" request has been set through the Kiosk detailed status GUI. If a "ping" request has been received, file downloads are temporairly suspended until the Kiosk is "pinged" (and the heartbeat/usage statistic file is retreived).
close kiosk connection	Kiosk MC function which will terminate the modem connection to a Kiosk field unit.
download files	Kiosk MC function which will download the "pending" files to each Kiosk field unit.
init kiosk connection	Kiosk MC function which connects to a specified Kiosk at at a specified phone number.
initialize field unit data	Kiosk MC function which initializes the data structures defined for each Kiosk field unit.
initialize field unit status	Kiosk MC function that is invoked to establish the necessary data structures to transmit the status of each field unit to the Data Server.
kiosk dsif connect	MDI MC Kiosk Master Computer DSIF Library Routine that is invoked to connect to the Kiosk Master Computer process.
load kiosk cfgs	Kiosk MC application routine which will load the last known Kiosk "file download status" (e.g., timestamp for each file downloaded) for each Kiosk field unit.
log error	Kiosk MC function which will log an error message to the status logger.
LogSetAttr	IVN Master Computer function that is used to set the logging levels for the modem and serial port functions.
ph connect	MDI Process Heartbeat Common Library routine which is invoked to connect to the Kiosk MC heartbeat process.
ph send heartbeat	MDI Process Heartbeat Common Library routine which is invoked to send a heartbeat to the Kiosk MC heartbeat process.
process heartbeat	Kiosk MC function which will be post the usage statistics to shared memory for use by the Kiosk detailed status GUI.
request heartbeat	Kiosk MC function which is invoked once the Kiosk is "on-line" with the master computer. The function will request the heartbeat/usage statistics file from the Kiosk filed unit.
save kiosk cfgs	Kiosk MC function which will save the current configuration information (i.e., what version of each file is stored on each Kiosk field unit) to a configuration file. This information is used on startup of the Kiosk MC main process to allow the process to be placed in the same state it was it when it was terminated.
send field unit status	Kiosk MC function that is invoked to transmit the current status of each field unit to the Data Server.

Function	Description
SerialSetAttr	MDI In-Vehicle Navigation Common Library routine that allows the attributes of the specified serial port to be modified.
signal setup	Kiosk MC function which is invoked to setup the UNIX signal handler for each type of UNIX signal that the application can receive.
status logger connect	MDI Status Logger Common Library Routine that is invoked to connect to the status logger process.
update files	Kiosk MC function which will update the local copies of files from the Data Server (if the version on the Data Server is different from the local copy).
update files pending	Kiosk MC application routine which will determine if any files need to be downloaded to the Kiosk field units. The routine will compare the timestamps of the files on the dataserver to the timestamp of the file last downloaded to the Kiosk for each Kiosk field unit.
update paper disk stats	Function which will determine if the "low paper" or "disk space low" indicator should be set (which the Kiosk detailed status GUI uses to warn the user of these conditions). Data from the configuration file is used to set the threshold of when these values are to be set.

Table 151 - Routines called by Kiosk MC Main

4.2.1.7.2 catch signal

catch signal is the Kiosk MC function that traps signals from the UNIX operating system. The structure chart for signal setup is depicted in Figure 150. A description of the routines called by signal setup is provided in Table 152.

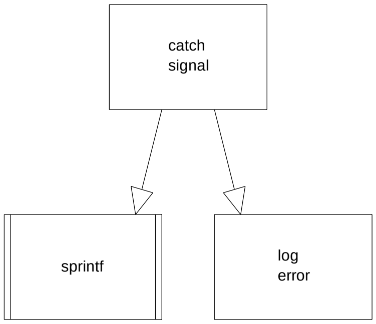

Figure 150 - catch signal Structure Chart

Function	Description
log error	Kiosk MC function which will log an error message to the status logger.
Sprintf	C Library Function that provides printf capabilities to a character string.

Table 152 - Routines Called by catch signal

4.2.1.7.3 signal setup

signal setup is the Kiosk MC function that sets up the UNIX signal handler for each type of UNIX signal that the application can receive. The structure chart for signal setup is depicted in Figure 151. A description of the routines called by signal setup is provided in Table 153.

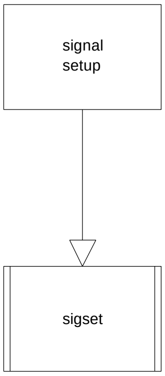

Figure 151 - signal setup Structure Chart

Function	Description
sigset	UNIX system call that associates a function with a specific signal; that is, when the system generates the signal the specified function will be invoked.

Table 153 - Routines called by signal setup

4.2.1.7.4 initialize field unit data

initialize field unit data is the Kiosk MC function that initializes the data structures for each Kiosk Field Unit. The structure chart for initialize field unit data is depicted in Figure 152. A description of the routines called by initialize field unit data is provided in Table 154.

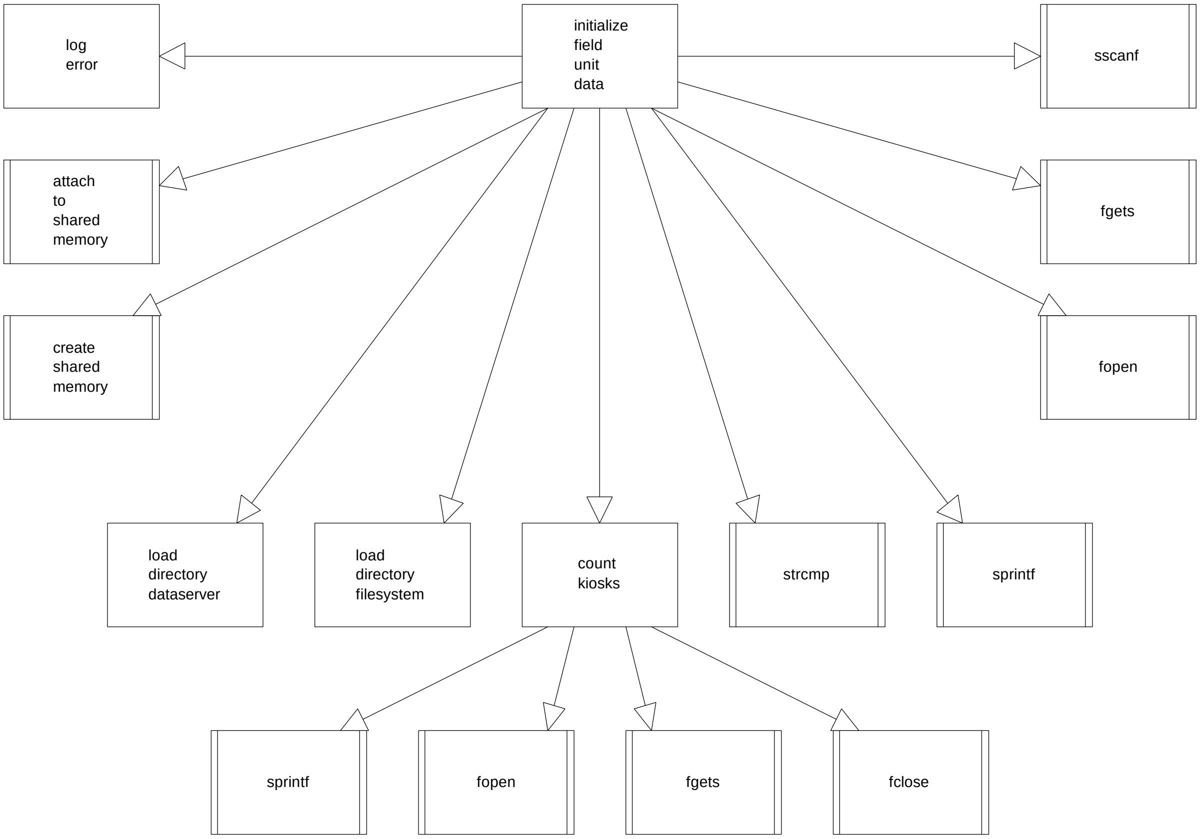

Figure 152 - initialize field unit data Structure Chart

Function	Description
attach to shared memory	MDI Data Server support utility that will attach to an existing UNIX shared memory segment.
count kiosks	Kiosk MC utility function will count how many kiosks are defined in the configuration file. This information is useful when "sizing" data structures.
create shared memory	MDI Data Server support utility that will create a UNIX shared memory segment.
fclose	C Library Function used to close an open file.
fgets	C Library Function used to read a line of text from a file.
fopen	C Library Function that opens the specified file using the specified access mode.
load directory data server	Kiosk MC function that will determine which files from the data server have been updated. The function will set any update status in the appropriate data structure; this information will then be used to determine if the files need to be downloaded to the Kiosk field units.

Function	Description
load directory filesystem	Kiosk MC function that will determine which files in the specified directory have been updated. The function will set any update status in the appropriate data structure; this information will then be used to determine if the files need to be downloaded to the Kiosk field units.
log error	Kiosk MC function which will log an error message to the status logger.
sprintf	C Library Function that provides printf capabilities to a character string.
sscanf	UNIX function call that will allow formatted "input" from a NULL terminated character string.
strcmp	UNIX function that will compare the contents of two NULL terminated character strings.

Table 154 - Routines called by initialize field unit data

4.2.1.7.5 log error

log error is the Kiosk MC function that transmits an error message to the status logger. The structure chart for log error is depicted in Figure 153. A description of the routines called by log error is provided in Table 155.

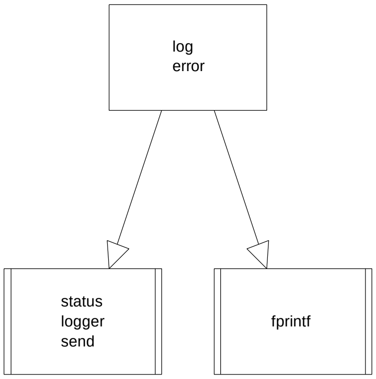

Figure 153 - log error Structure Chart

Function	Description
fprintf	UNIX system call to print formatted data to a file stream.
status logger send	Kiosk MC Library routine that will send messages to the Status Logger process.

Table 155 - Routines called by log error

4.2.1.7.6 load directory data server

load directory data server is the Kiosk MC function that determines which Kiosk data files have been updated in the Data Server. The function will set any update status in the appropriate data structure; this information will then be used to determine if the files need to be downloaded to the Kiosk field units. The structure chart for load directory data server is depicted in Figure 154. A description of the routines called by load directory data server is provided in Table 156.

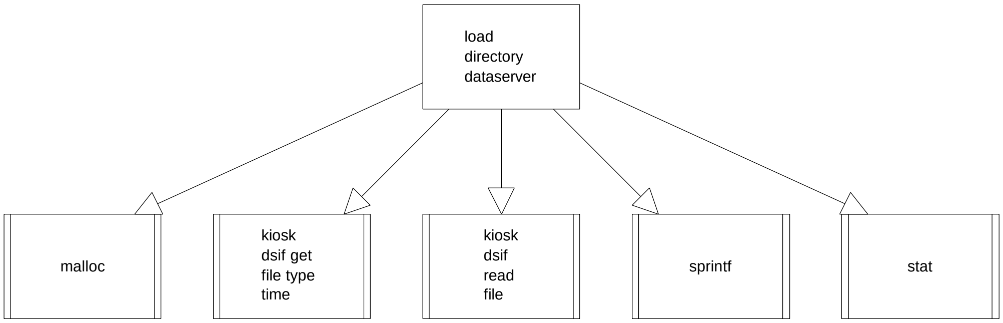

Figure 154 - load directory data server Structure Chart

Function	Description
kiosk dsif get file type time	MDI Kiosk MC library function that will retrieve a list of active filenames (and their timestamps) from the Data Server.
kiosk dsif read file	MDI Kiosk MC library function that will retrieve a file from the Data Server.
malloc	UNIX system call which will allocate the specified number bytes of memory.
sprintf	C Library Function that provides printf capabilities to a character string.
Stat	UNIX system call that will return file attributes for a specified file.

Table 156 - Routines Called by load directory data server

4.2.1.7.7 load directory filesystem

load directory filesystem is the Kiosk MC function that checks the specified directory and determines which files have been updated. The function will set any update status in the appropriate data structure; this information will then be used to determine if the files need to be downloaded to the Kiosk field units. The structure chart for init kiosk connection is depicted in Figure 155. A description of the routines called by init kiosk connection is provided in Table 157.

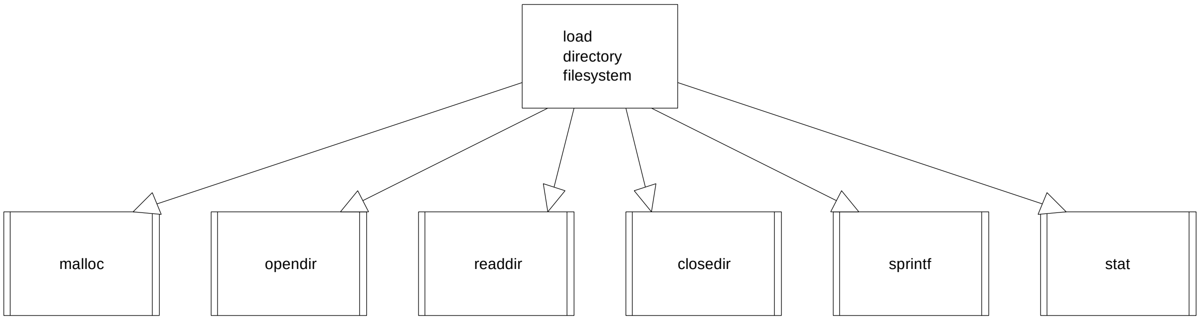

Figure 155 - load directory filesystem Structure Chart

Function	Description
closedir	UNIX system call that will "close" a directory (which had been previously opened for programmatic reading).
malloc	UNIX system call which will allocate the specified number bytes of memory.
opendir	UNIX system call that will "open" a directory for programmatic reading of the contents.
readdir	UNIX system call which reads the next entry in the directory.
sprintf	C Library Function that provides printf capabilities to a character string.
stat	UNIX system call that will return file attributes for a specified file.

Table 157 - Routines Called by load directory filesystem

4.2.1.7.8 init kiosk connection

init kiosk connection is the Kiosk MC function that performs the connection to the specified Kiosk using the specified phone number. The structure chart for init kiosk connection is depicted in Figure 156. A description of the routines called by init kiosk connection is provided in Table 158.

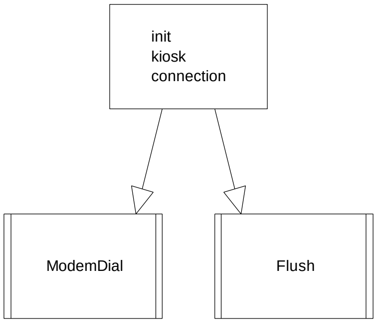

Figure 156 - init kiosk connection Structure Chart

Function	Description
Flush	MDI In-Vehicle Navigation Common Library routine that will perform a UNIX "flush" on the specified serial port. This is invoked to assure that all data is flushed from the serial UART.
ModemDial	MDI In-Vehicle Navigation Common Library routine that will dial a specified phone number, at a specified baud rate, using the specified serial port (i.e., modem).

Table 158 - Routines called by init kiosk connection

4.2.1.7.9 update files pending

update files pending is the Kiosk MC routine that determines if any files need to be downloaded to the Kiosk field units. The routine will compare the timestamps of the files on the data server to the timestamp of the file last downloaded to the Kiosk for each Kiosk field unit. The structure chart for update files pending is depicted in Figure 157. A description of the routines called by update files pending is provided in Table 159.

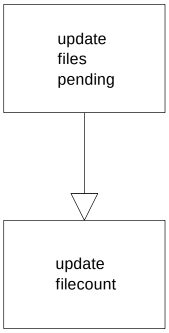

Figure 157 - update files pending Structure Chart

Function	Description
update filecount	Kiosk MC function which will determine which files need to be updated on each Kiosk field unit.

Table 159 - Routines called by update files pending

4.2.1.7.10 update filecount

update filecount is the Kiosk MC function that determines the files that need to be updated on each Kiosk field unit. The structure chart for update filecount is depicted in Figure 158. A description of the routines called by update filecount is provided in Table 160.

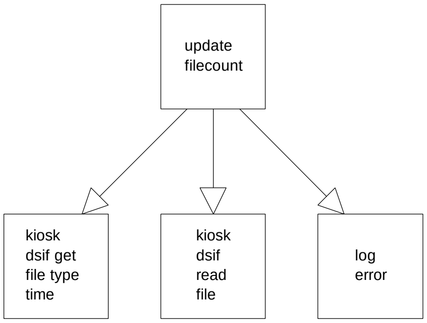

Figure 158 - update filecount Structure Chart

Function	Description
kiosk dsif get file type time	MDI Kiosk MC library function that will retrieve a list of active filenames (and their timestamps) from the Data Server.
kiosk dsif read file	MDI Kiosk MC library function that will retrieve a file from the Data Server.
log error	Kiosk MC function which will log an error message to the status logger.

Table 160 - Routines called by update filecount

4.2.1.7.11 load kiosk cfgs

load kiosk cfgs is the Kiosk MC routine that loads the last known Kiosk "file download status" (e.g., timestamp for each file downloaded) of each Kiosk field unit. The structure chart for load kiosk cfgs is depicted in Figure 159. A description of the routines called by load kiosk cfgs is provided in Table 161.

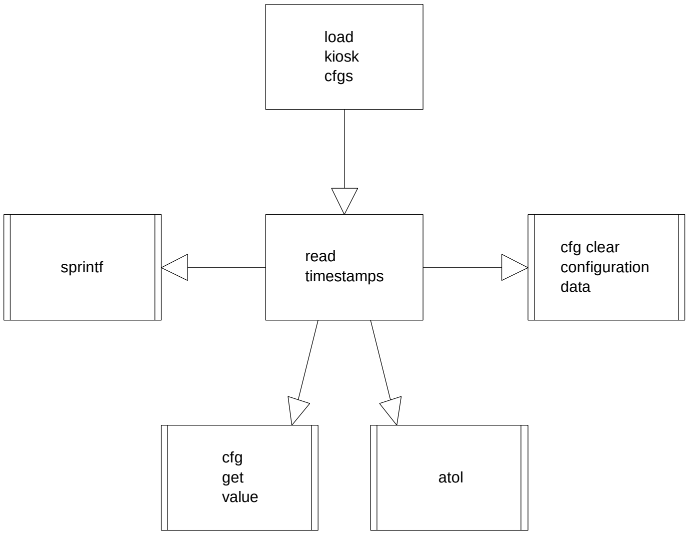

Figure 159 - load kiosk cfgs Structure Chart

Function	Description
read timestamps	Kiosk MC function which will write timestamps to a configuration file.
atol	UNIX system call that converts a NULL terminated character string to a long integer.
cfg clear configuration data	MDI Configuration File Common Library routine used to clear the configuration name-pairs loaded from memory.
cfg get value	MDI Configuration File Common Library routine used to return the value of the specified configuration name.
sprintf	UNIX system call to print formatted data to character string.

Table 161 - Routines called by load kiosk cfgs

4.2.1.7.12 download files

download files is the Kiosk MC function that downloads the "pending" files to each Kiosk field unit. The structure chart for download files is depicted in Figure 160. A description of the routines called by download files is provided in Table 162.

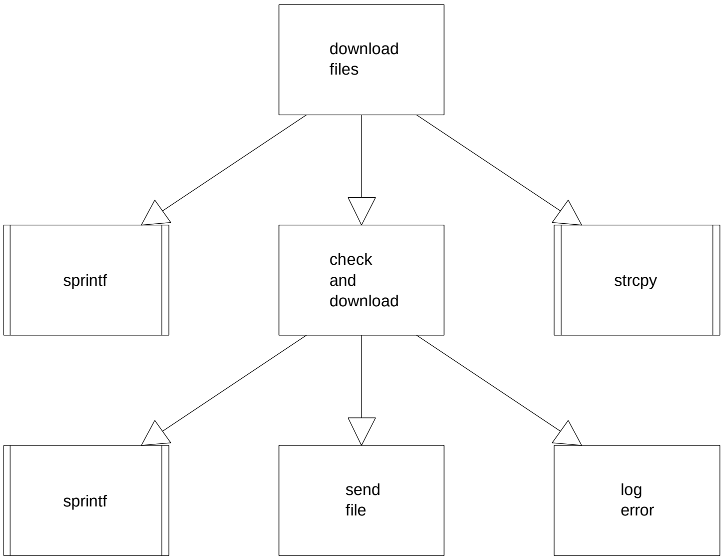

Figure 160 - download files Structure Chart

Function	Description
check and download	Kiosk MC function that will download any "pending" files to a Kiosk field unit that requires an updated file.
log error	Kiosk MC function which will log an error message to the status logger.
send file	Kiosk MC function which is invoked to transfer a file to the Kiosk field unit.
sprintf	UNIX system call to print formatted data to character string.
strcpy	UNIX system call to copy a NULL terminated character string from one variable to another.

Table 162 - Routines called by download files

4.2.1.7.13 send file

send file is the Kiosk MC function that is invoked to transfer a file from the Master Computer to the Kiosk field unit. The structure chart for send file is depicted in Figure 161. A description of the routines called by send file is provided in Table 163.

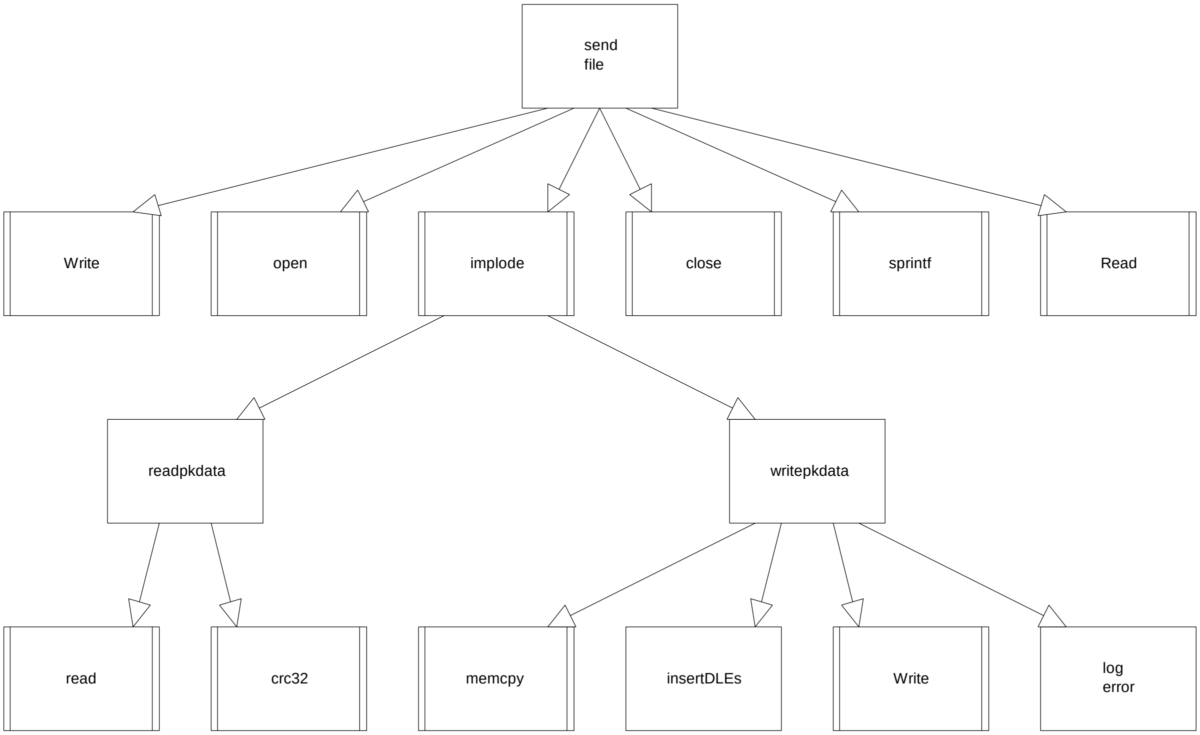

Figure 161 - send file Structure Chart

Function	Description
insertDLEs	A Kiosk MC function that will insert ASCII DLEs before any ASCII DLE, SOH, or EOT. Once the DLE is inserted, the DLE, SOH, or EOT byte is then incremented by one so that the byte is not interpreted by the communications software as a "control character". The data receiving program on the Kiosk field unit will reverse the insertDLE function so that the data stream is correct.
log error	Kiosk MC function which will log an error message to the status logger.
readpkdata	The function is utilized in the PKWare implode function to "supply" data to the PKWare compression engine. The function is utilized to read data from a data file and supply the requested number of bytes to the PKWare engine.
writepkdata	The function is utilized in the PKWare implode function to "dispose" of the data as it is compressed by the PKWare compression engine. The function will format the data and write it to the modem port that is currently attached to the Kiosk field unit.
close	UNIX system call to close a previously opened file descriptor.
crc32	PKWare supplied function that computes a 32 bit CRC for the data supplied to the function. The CRC will be transmitted to the Kiosk field unit so that the integrity of the data transmitted can be verified.
implode	PKWare supplied function that provides programmatic support to compress data streams using PKWare's technology (PKWare developed pkzip and pkunzip). The implode function includes as arguments function definitions which provide "data provider" (data to be compressed) and "data disposer" (how to save the compressed data).
memcpy	UNIX system call that copies data from one place in memory to another place in memory.

Function	Description
open	UNIX system call to open a file; the function will return an integer file descriptor that can be used with the UNIX read, write, and close system calls.
Read	MDI In-Vehicle Navigation Common Library routine that will read a specified number of characters from the specified serial port (i.e. modem). This function will invoke the UNIX read system call.
sprintf	UNIX system call to print formatted data to character string.
Write	MDI In-Vehicle Navigation Common Library routine that will write a specified number of characters to the specified serial port (i.e. modem). This function will invoke the UNIX write system call.

Table 163 - Routines called by send file

4.2.1.7.14 request heartbeat

request heartbeat is the Kiosk MC function that requests the heartbeat/usage statistics file from the Kiosk filed unit. This function is invoked once the Kiosk is "on-line" (connected via the modem) with the Master Computer. The structure chart for request heartbeat is depicted in Figure 162. A description of the routines called by request heartbeat is provided in Table 164.

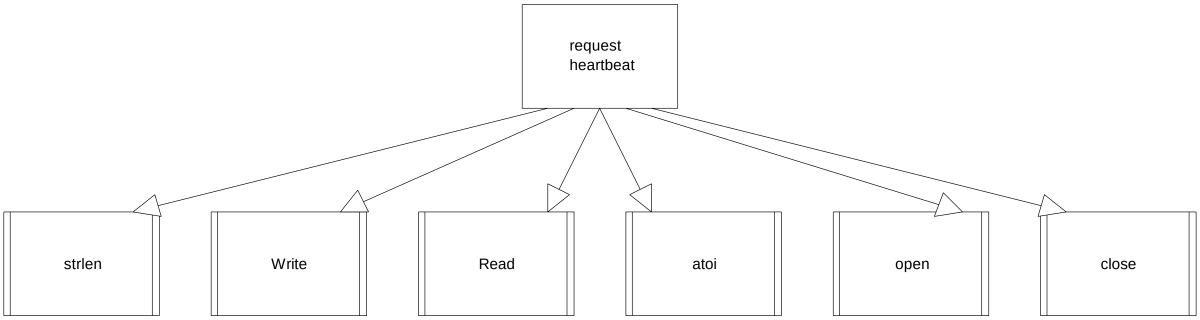

Figure 162 - request heartbeat Structure Chart

Function	Description
atoi	UNIX system call that converts a NULL terminated character string to an integer value.
close	UNIX system call to close a previously opened file descriptor.
open	UNIX system call to open a file; the function will return an integer file descriptor that can be used with the UNIX read, write, and close system calls.
Read	MDI In-Vehicle Navigation Common Library routine that will read a specified number of characters from the specified serial port (i.e. modem). This function will invoke the UNIX read system call.

Function	Description
strlen	UNIX system call that computes the number of characters in a NULL terminated string.
Write	MDI In-Vehicle Navigation Common Library routine that will write a specified number of characters to the specified serial port (i.e., modem). This function will invoke the UNIX write system call.

Table 164 - Routines called by request heartbeat

4.2.1.7.15 update files

update files is the Kiosk MC function which updates the local copies of files from the Data Server (if the version on the Data Server is different from the local copy). The structure chart for update files is depicted in Figure 163. A description of the routines called by update files is provided in Table 165.

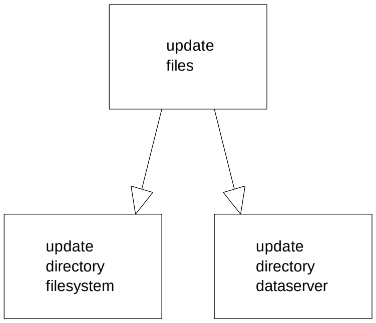

Figure 163 - update files Structure Chart

Function	Description
update directory data server	Kiosk MC function that is invoked to update the master file data structures with a list (and timestamps) of each file obtained from the data server.
update directory filesystem	Kiosk MC function that is invoked to update the master file data structures with a list (and timestamps) of each file contained in the target directory.

Table 165 - Routines called by update files

4.2.1.7.16 update directory data server

update directory data server is the Kiosk MC function that updates the master file data structures with a list (and timestamps) of each file obtained from the data server. The structure chart for update files is depicted in Figure 164. A description of the routines called by update files is provided in Table 166.

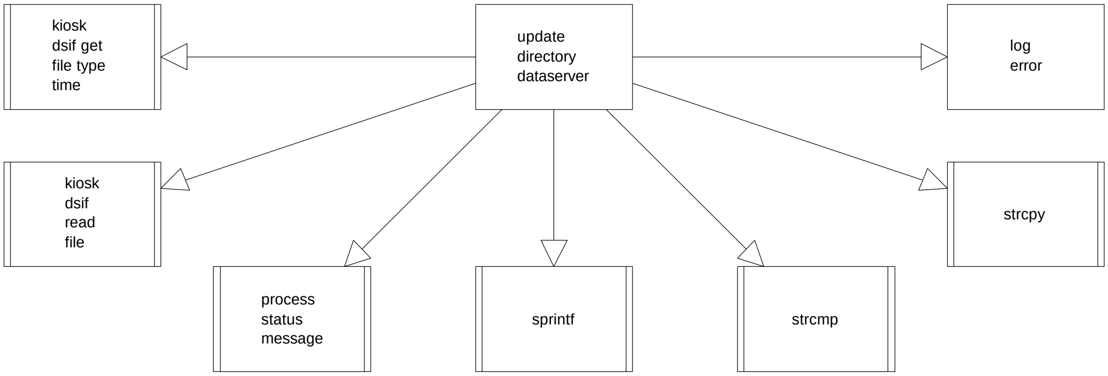

Figure 164 - update directory data server structure chart

Function	Description
kiosk dsif get file type time	MDI Kiosk MC library function that will retrieve a list of active filenames (and their timestamps) from the Data Server.
kiosk dsif read file	MDI Kiosk MC library function that will retrieve a file from the Data Server.
log error	Kiosk MC function which will log an error message to the status logger.
process status message	MDI Data Server utility routine which will send a message (which will be logged) to the Status Logger.
sprintf	C Library Function that provides printf capabilities to a character string.
strcmp	UNIX function that will compare the contents of two NULL terminated character strings.
strcpy	C Library Function used to copy a source string to a destination string.

Table 166 - Routines Called by update directory dataserver

4.2.1.7.17 update directory filesystem

update directory filesystem is the Kiosk MC function that updates the master file data structures with a list (and timestamps) of each file contained in the target directory. The structure chart for update files is depicted in Figure 165. A description of the routines called by update files is provided in Table 167.

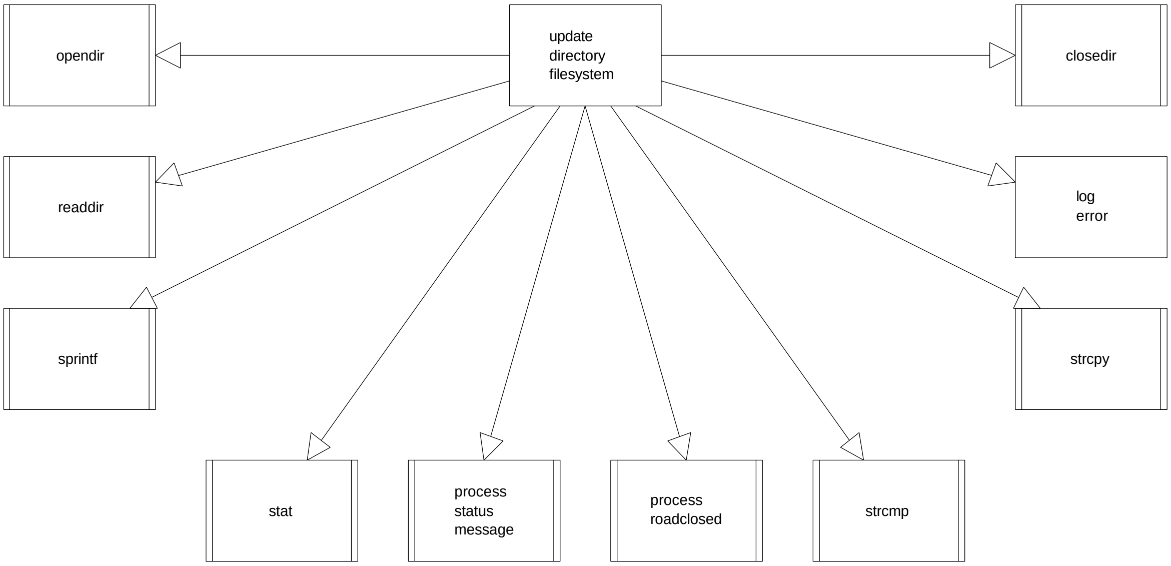

Figure 165 - update directory filesystem Structure Chart

Function	Description
closedir	UNIX system call that will "close" a directory (which had been previously opened for programmatic reading).
log error	Kiosk MC function which will log an error message to the status logger.
opendir	UNIX system call that will "open" a directory for programmatic reading of the contents.
process roadclosed	Transferfiles function that will convert the State of Texas provided road closed datafile to a format required by the Data Server requirements.
process status message	MDI Data Server utility routine which will send a message (which will be logged) to the Status Logger.
readdir	UNIX system call which reads the next entry in the directory.
sprintf	C Library Function that provides printf capabilities to a character string.
stat	UNIX system call that will return file attributes for a specified file.
strcmp	UNIX function that will compare the contents of two NULL terminated character strings.
strcpy	C Library Function used to copy a source string to a destination string.

Table 167 - Routines Called by update directory filesystem

4.2.1.7.18 process heartbeat

process heartbeat is the Kiosk MC function which posts the usage statistics to shared memory for use by the Kiosk detailed status GUI. The structure chart for process heartbeat is depicted in Figure 166. A description of the routines called by process heartbeat is provided in Table 168.

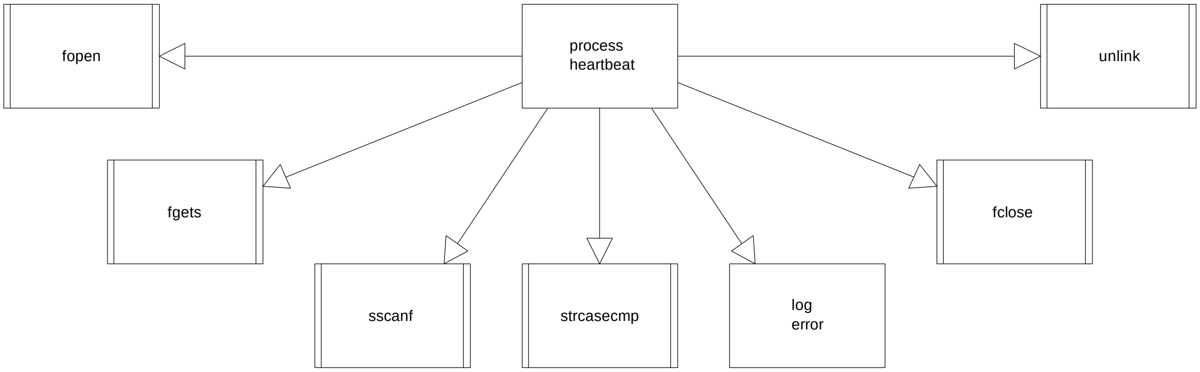

Figure 166 - process heartbeat Structure Chart

Function	Description
log error	Kiosk MC function which will log an error message to the status logger.
fclose	UNIX system call to close a previously opened file stream pointer.
fgets	UNIX system call that will get one line of data from a stream based file descriptor.
fopen	UNIX system call to open a file stream pointer to the specified file.
sscanf	UNIX function call that will allow formatted "input" from a NULL terminated character string.
strcasecmp	UNIX system call that will compare two NULL terminated character strings while ignoring differences in case.
unlink	UNIX system call that will remove the specified file from the file system.

Table 168 - Routines called by process heartbeat

4.2.1.7.19 close kiosk connection

close kiosk connection is the Kiosk MC function that terminates the modem connection to a Kiosk field unit. The structure chart for close kiosk connection is depicted in Figure 167. A description of the routines called by close kiosk connection is provided in Table 169.

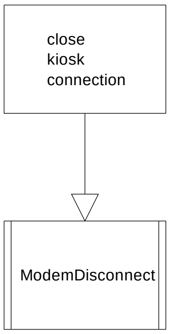

Figure 167 - close kiosk connection Structure Chart

Function	Description
ModemDisconnect	MDI In-Vehicle Navigation Common Library routine that will disconnect the specified modem port (i.e., hang up the line).

Table 169 - Routines called by close kiosk connection

4.2.1.7.20 update paper disk stats

The update paper disk stats function determines if the "low paper" or "disk space low" indicator should be set (used on the detailed status GUI to warn the user of these conditions). Data from the configuration file is used to set the threshold of when these values are to be set. The structure chart for update paper disk stats is depicted in Figure 168. A description of the routines called by update paper disk stats is provided in Table 170.

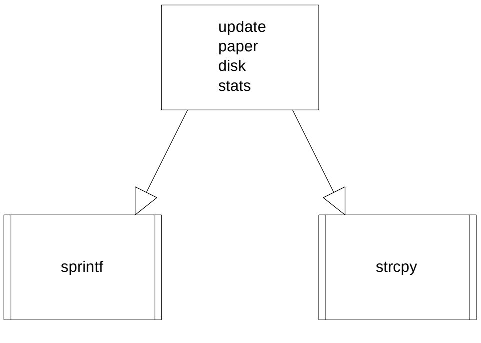

Figure 168 - update paper disk stats Structure Chart

Function	Description
sprintf	UNIX system call to print formatted data to character string.
strcpy	UNIX system call to copy a NULL terminated character string from one variable to another.

Table 170 - Routines called by update paper disk stats

4.2.1.7.21 check for ping

check for ping is the Kiosk MC function that checks to see if a "ping" (see if the Kiosk is responding) request has been set through the Kiosk detailed status GUI. If a "ping" request has been received, file downloads are temporarily suspended until the Kiosk is "pinged" (and the heartbeat/usage statistic file is retrieved). The structure chart for check for ping is depicted in Figure 169. A description of the routines called by check for ping is provided in Table 171.

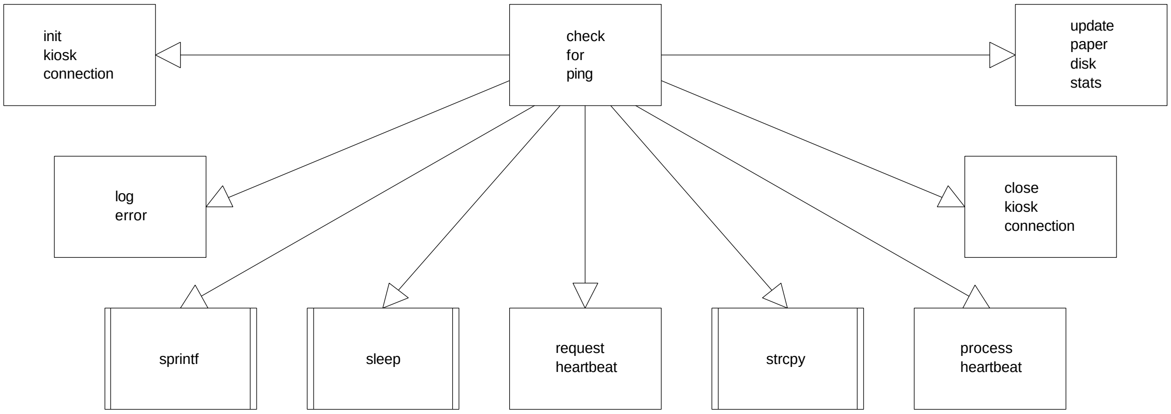

Figure 169 - check for ping Structure Chart

Function	Description
close kiosk connection	Kiosk MC function which will terminate the modem connection to a Kiosk field unit.
init kiosk connection	Kiosk MC function which connects to a specified Kiosk at a specified phone number.
log error	Kiosk MC function which will log an error message to the status logger.
process heartbeat	Kiosk MC function which will be post the usage statistics to shared memory for use by the Kiosk detailed status GUI.
request heartbeat	Kiosk MC function which is invoked once the Kiosk is "on-line" with the master computer. The function will request the heartbeat/usage statistics file from the Kiosk filed unit.

Function	Description
update paper disk stats	Function which will determine if the "low paper" or "disk space low" indicator should be set (which the Kiosk detailed status GUI uses to warn the user of these conditions). Data from the configuration file is used to set the threshold of when these values are to be set.
sleep	UNIX system call that is invoked to suspend execution of a program for a specified number of seconds.
sprintf	UNIX system call to print formatted data to character string.
strcpy	UNIX system call to copy a NULL terminated character string from one variable to another.

Table 171 - Routines called by check for ping

4.2.1.7.22 init kiosk connection

init kiosk connection is the Kiosk MC function that connects to the specified Kiosk at the specified phone number. The structure chart for init kiosk connection is depicted in Figure 170. A description of the routines called by init kiosk connection is provided in Table 172.

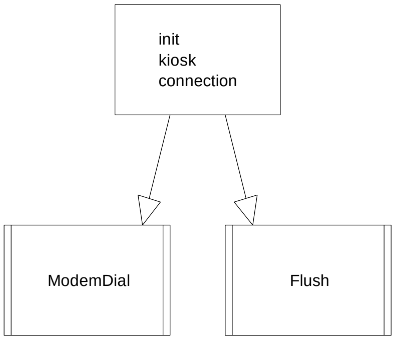

Figure 170 - init kiosk connection Structure Chart

Function	Description
Flush	MDI In-Vehicle Navigation Common Library routine that will perform a UNIX "flush" on the specified serial port. This is invoked to assure that all data is flushed from the serial UART.
ModemDial	MDI In-Vehicle Navigation Common Library routine that will dial a specified phone number, at a specified baud rate, using the specified serial port (i.e., modem).

Table 172 - Routines called by init kiosk connection

4.2.1.7.23 close kiosk connection

close kiosk connection is the Kiosk MC function that terminates the modem connection with a Kiosk field unit. The structure chart for close kiosk connection is depicted in Figure 171. A description of the routines called by close kiosk connection is provided in Table 173.

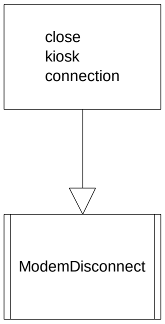

Figure 171 - close kiosk connection Structure Chart

Function	Description
ModemDisconnect	MDI In-Vehicle Navigation Common Library routine that will disconnect the specified modem port (i.e., hang up the line).

Table 173 - Routines called by close kiosk connection

4.2.1.7.24 save kiosk cfgs

save kiosk cfgs is the Kiosk MC function that saves the current configuration information (i.e., what version of each file is stored on each Kiosk field unit) to the appropriate configuration file. This information is used on startup of the Kiosk MC main process to allow the process to be placed in the same state it was in when it was terminated. The structure chart for save kiosk cfgs is depicted in Figure 172. A description of the routines called by save kiosk cfgs is provided in Table 174.

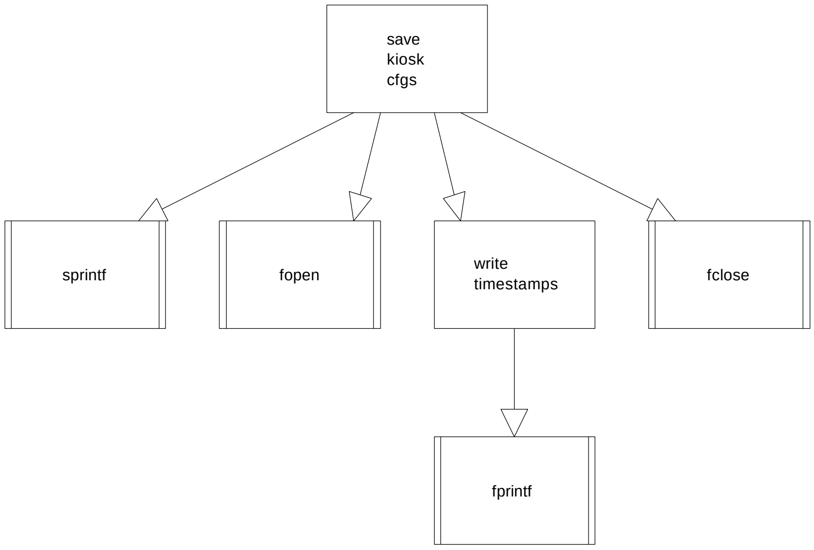

Figure 172 - save kiosk cfgs Structure Chart

Function	Description
write timestamps	Kiosk MC function which will read timestamps to a configuration file (which has been previously created by the write_timestamps function).
fclose	UNIX system call to close a previously opened file stream pointer.
fopen	UNIX system call to open a file stream pointer to the specified file.
fprintf	UNIX system call to print formatted data to a file stream.
sprintf	UNIX system call to print formatted data to character string.

Table 174 - Routines called by save kiosk cfgs

4.2.1.7.25 initialize field unit status

initialize field unit status is the Kiosk MC function that initializes each of the Kiosk field unit’s data structures. The structure chart for initialize field unit status is depicted in Figure 173. A description of the routines called by initialize field unit status is provided in Table 175.

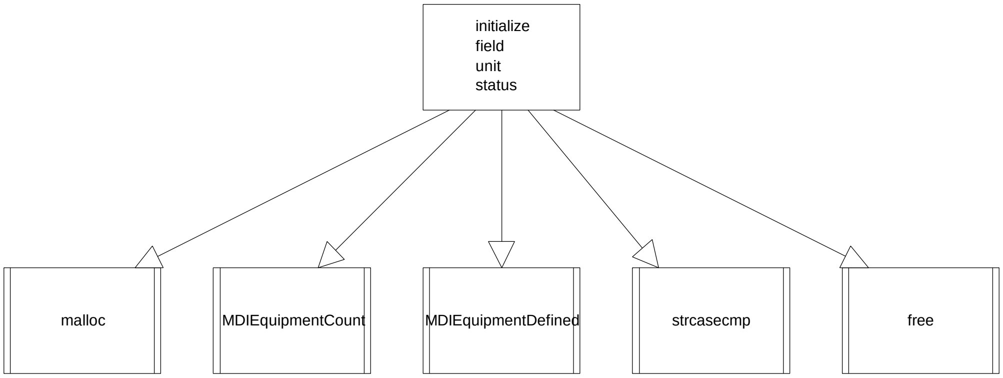

Figure 173 - initialize field unit status Structure Chart

Function	Description
free	C Library Function used to free previously allocated memory and make it available for further allocation.
malloc	UNIX system call which will allocate the specified number bytes of memory.
MDIEquipmentCount	Real-Time Subsystem function that will access the MDI configuration files and return the number of Kiosk Field Units defined within the real-time subsystem.
MDIEquipmentDefined	Real-Time Subsystem function that will access the MDI configuration files and return a sorted list of Kiosk Field Units defined within the real-time subsystem.
strcasecmp	UNIX system call that will compare two NULL terminated character strings while ignoring differences in case.

Table 175 - Routines Called by initialize field unit status

4.2.1.7.26 send field unit status

send field unit status is the Kiosk MC function that transmits the current status of each field unit to the Data Server. The structure chart for send field unit status is depicted in Figure 174. A description of the routines called by send field unit status is provided in Table 176.

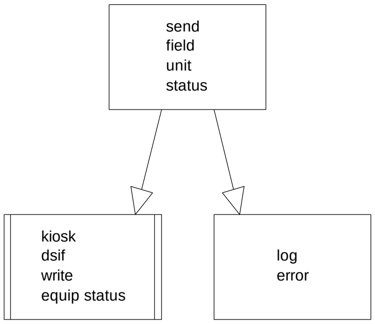

Figure 174 - send field unit status Structure Chart

Function	Description
kiosk dsif write equip status	Kiosk DSIF function that allows the status of all Kiosk Field Units to be transferred to the Data Server.
log error	Kiosk MC function which will log an error message to the status logger.

Table 176 - Routines Called by send field unit status

4.2.1.8 Transfer Data Files

Once an hour, the Transfer Data Files application runs on the Data Server that connects to the NT server to get the current versions of the VIA data files. This application constructs a temporary version control file that contains the VIA data file information. It then compares the version of each data file in the temporary version control file with the master version control file retrieved from the Data Server. The application then constructs a list of files that are out-of-date, reconnects to the NT server, and retrieves the files and stores the files on the Data Server. The updated VIA data files are downloaded by System Maintenance to the Kiosk Field Units.

Once an hour, the Transfer Data Files application retrieves the current weather conditions and five-day forecast information from the Transguide Web Server. The weather provider, Alex Garcia of KABB Channel 29, updates a San Antonio area radar map, a current weather conditions file and a five-day forecast file once an hour. The files are retrieved from the Web Server and stored on the Data Server. The data flows for Transfer Data Files are depicted in Figure 175.

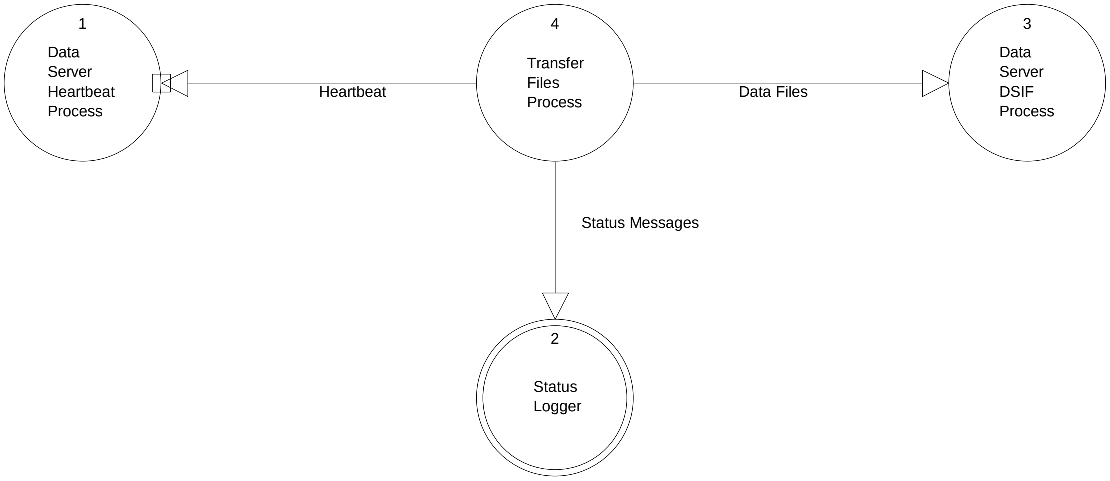

Figure 175 – Transfer Data Files Data Flows

The Transfer Files Process data flows are supplying Weather and Via files to the Data Server, logging status messages to the Status Logger, and providing a periodic heartbeat to the Data Server Heartbeat Process.

4.2.1.8.1 Transfer Data File Main

The Transfer Data File program will continuously monitor a single directory and the program will update the Data Server with changed files. The program is written generically so that on startup a source directory is specified. This implies that multiple instances of the Transfer Data File program execute in the MDI environment. The structure chart for Transfer Data File Main is depicted in Figure 176. A description of the routines called by Transfer Data File Main is provided in Table 177.

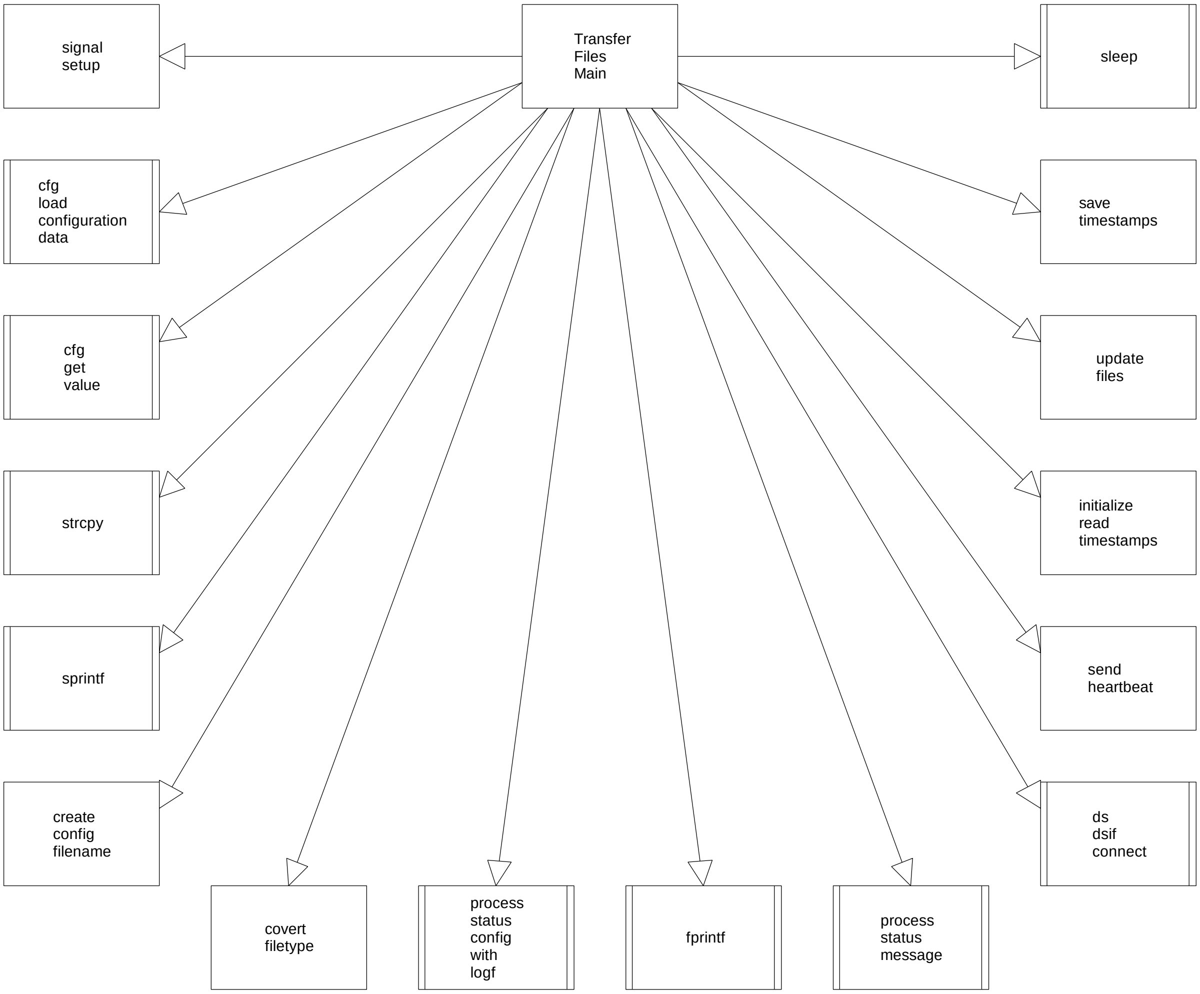

Figure 176 - Transfer Data File Main Structure Chart

Function	**Description**
covert filetype	Transferfiles function which will convert a character string process type (e.g., VIA, Weather, Screen Saver or Road Closed) to a C enumerated type.
create config filename	Transferfiles function that will create a full pathname to the configuration file (which holds process configuration data).
initialize read timestamps	The function will initialize its local data structure and then read the appropriate data configuration file to "reset" the program's status to the state it was in when it was last executing.
save timestamps	Transferfiles function that will save the current file timestamp information to the appropriate configuration file (this information is used on process restart).
send heartbeat	Transferfiles function that is invoked to send a heartbeat to the Data Server DSIF process.

Function	Description
signal setup	Kiosk MC function which is invoked to setup the UNIX signal handler for each type of UNIX signal that the application can receive.
update files	Kiosk MC function which will update the local copies of files from the Data Server (if the version on the Data Server is different from the local copy).
cfg get value	MDI Configuration File Common Library routine used to return the value of the specified configuration name.
cfg load configuration data	MDI Configuration File Common Library routine used to read the configuration name-pairs from the specified configuration file. These name-value pairs are loaded into memory so they can be accessed on demand by the calling program.
ds dsif connect	MDI Data Server utility routine which will "connect" to the Data Server DSIF process. This will allow heartbeat and data requests to be sent between the invoking process and the Data Server.
fprintf	UNIX system call to print formatted data to a file stream.
process status config with logf	MDI Data Server utility routine which will "attach" the invoking process to the Status Logger. A logfile will also be opened.
process status message	MDI Data Server utility routine which will send a message (which will be logged) to the Status Logger.
sleep	UNIX system call that is invoked to suspend execution of a program for a specified number of seconds.
sprintf	UNIX system call to print formatted data to character string.
strcpy	UNIX system call to copy a NULL terminated character string from one variable to another.

Table 177 - Routines called by Transfer Data File Main

4.2.1.8.2 signal setup

The signal setup function sets up the UNIX signal handler for each type of UNIX signal that the application can receive. The structure chart for signal setup is depicted in Figure 177. A description of the routines called by signal setup is provided in Table 178.

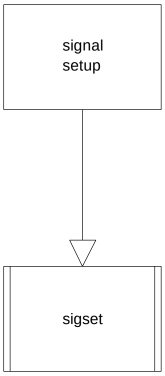

Figure 177 - signal setup Structure Chart

Function	Description
sigset	UNIX system call that associates a function with a specific signal; that is, when the system generates the signal the specified function will be invoked.

Table 178 - Routines called by signal setup

4.2.1.8.3 save timestamps

The save timestamps function saves the current file timestamp information to the appropriate configuration file (this information is used on process restart). The structure chart for save timestamps is depicted in Figure 178. A description of the routines called by save timestamps is provided in Table 179.

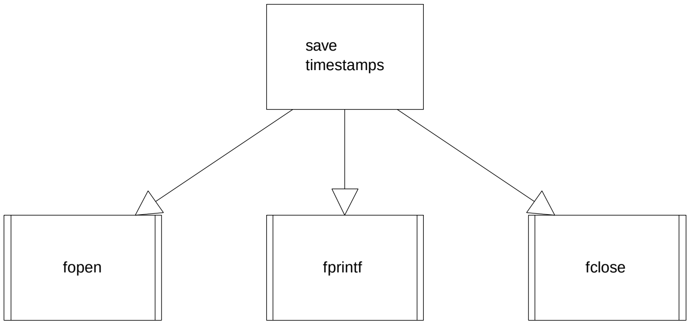

Figure 178 - save timestamps Structure Chart

Function	Description
fclose	UNIX system call to close a previously opened file stream pointer.
fopen	UNIX system call to open a file stream pointer to the specified file.
fprintf	UNIX system call to print formatted data to a file stream.

Table 179 - Routines called by save timestamps

4.2.1.8.4 update files

The update files function updates the local copies of files from the Data Server (if the version on the Data Server is different from the local copy). The structure chart for update files is depicted in Figure 179. A description of the routines called by update files is provided in Table 180.

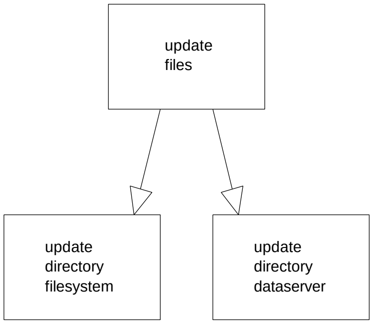

Figure 179 - update files Structure Chart

Function	Description
update directory dataserver	Kiosk MC function that is invoked to update the master file data structures with a list (and timestamps) of each file obtained from the data server.
update directory filesystem	Kiosk MC function that is invoked to update the master file data structures with a list (and timestamps) of each file contained in the target directory.

Table 180 - Routines called by update files

4.2.1.8.5 update directory filesystem

update directory filesystem is the Kiosk MC function that updates the master file data structures with a list (and timestamps) of each file contained in the target directory. The structure chart for update directory filesystem is depicted in Figure 180. A description of the routines called by update directory filesystem is provided in Table 181.

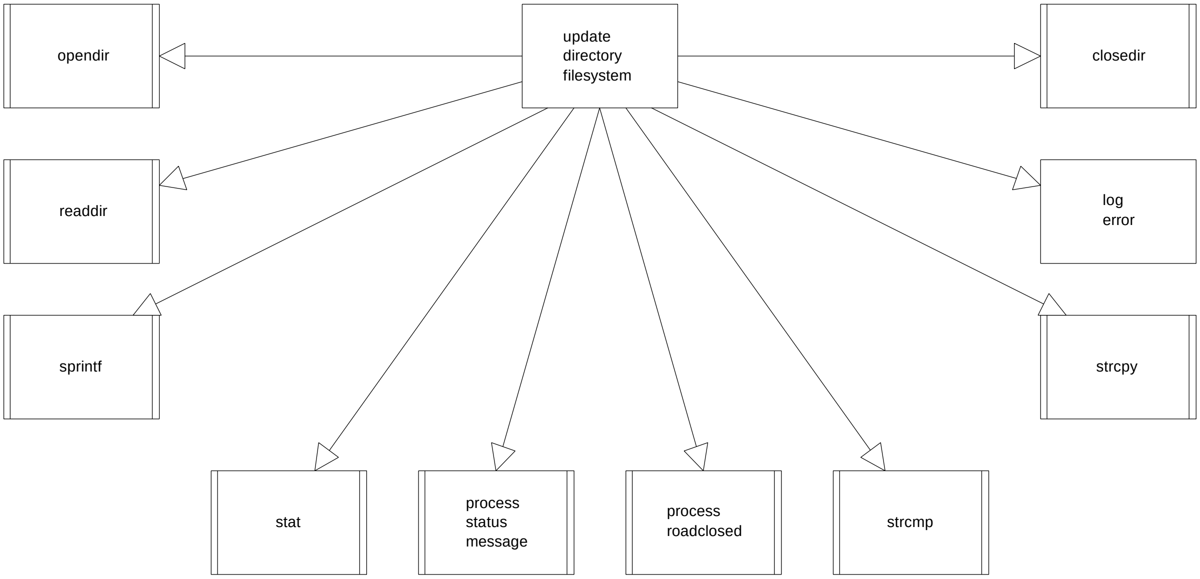

Figure 180 - update directory filesystem Structure Chart

Function	Description
closedir	UNIX system call that will "close" a directory (which had been previously opened for programmatic reading).
log error	Kiosk MC function which will log an error message to the status logger.
opendir	UNIX system call that will "open" a directory for programmatic reading of the contents.
process roadclosed	Transferfiles function that will convert the State of Texas provided road closed datafile to a format required by the Data Server requirements.
process status message	MDI Data Server utility routine which will send a message (which will be logged) to the Status Logger.
readdir	UNIX system call which reads the next entry in the directory.
sprintf	C Library Function that provides printf capabilities to a character string.
stat	UNIX system call that will return file attributes for a specified file.
strcmp	UNIX function that will compare the contents of two NULL terminated character strings.
strcpy	C Library Function used to copy a source string to a destination string.

Table 181 - Routines called by update directory filesystem

4.2.1.8.6 update directory dataserver

This Kiosk MC function updates the master file data structures with a list (and timestamps) of each file obtained from the data server. The structure chart for update directory dataserver is depicted in Figure 181. A description of the routines called by update directory dataserver is provided in Table 182.

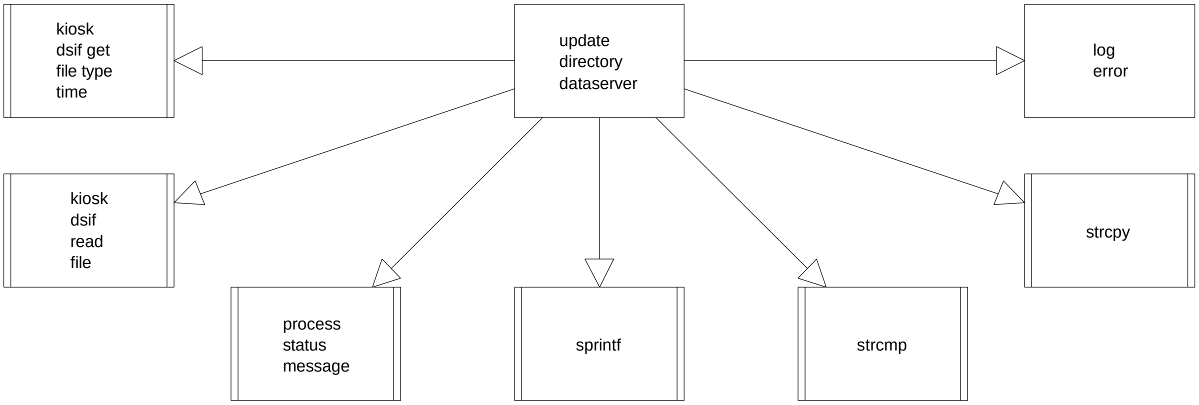

Figure 181 - update directory dataserver Structure Chart

Function	Description
kiosk dsif get file type time	MDI Kiosk MC library function that will retrieve a list of active filenames (and their timestamps) from the Data Server.
kiosk dsif read file	MDI Kiosk MC library function that will retrieve a file from the Data Server.
log error	Kiosk MC function which will log an error message to the status logger.
process status message	MDI Data Server utility routine which will send a message (which will be logged) to the Status Logger.
sprintf	C Library Function that provides printf capabilities to a character string.
strcmp	UNIX function that will compare the contents of two NULL terminated character strings.
strcpy	C Library Function used to copy a source string to a destination string.

Table 182 - Routines called by update directory dataserver

4.2.1.8.7 initialize read timestamps

This function will initialize its local data structure and then read the appropriate data configuration file to "reset" the program's status to the state it was in when it was last executing. The structure chart for initialize read timestamps is depicted in Figure 182. A description of the routines called by initialize read timestamps is provided in Table 183.

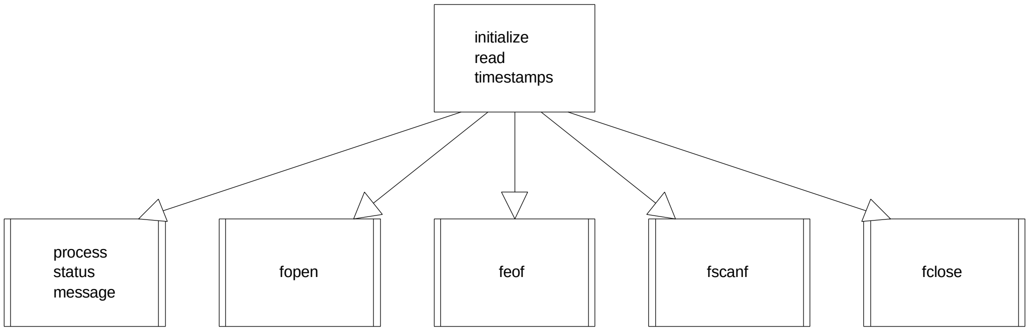

Figure 182 - initialize read timestamps Structure Chart

Function	Description
fclose	UNIX system call to close a previously opened file stream pointer.
feof	UNIX system call which is used to determine if the end of file condition is true for a file stream pointer.
fopen	UNIX system call to open a file stream pointer to the specified file.
fscanf	UNIX system call to read formatted data from a file stream pointer.
process status message	MDI Data Server utility routine which will send a message (which will be logged) to the Status Logger.

Table 183 - Routines called by initialize read timestamps

4.2.1.8.8 send heartbeat

This function sends a heartbeat to the Data Server DSIF process. The structure chart for send heartbeat is depicted in Figure 183. A description of the routines called by send heartbeat is provided in Table 184.

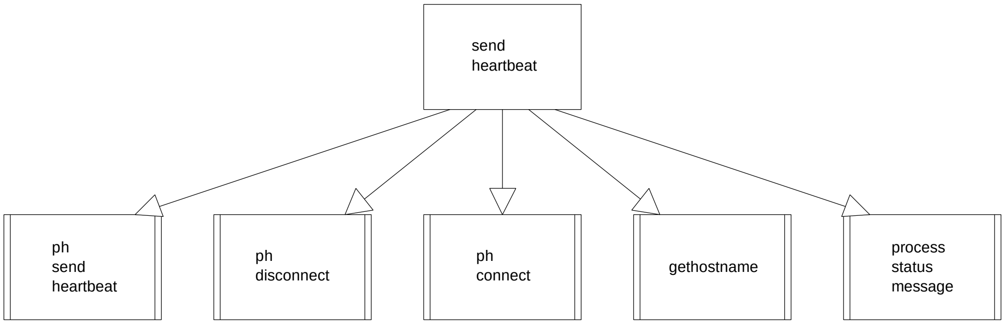

Figure 183 - send heartbeat Structure Chart

Function	Description
gethostname	UNIX system call that returns the hostname of the machine on which the function is executing.
ph connect	MDI Process Heartbeat Common Library routine which is invoked to connect to the Kiosk MC heartbeat process.
ph disconnect	MDI Process Heartbeat Common Library routine which is invoked to disconnect from the Kiosk MC heartbeat process.
ph send heartbeat	MDI Process Heartbeat Common Library routine which is invoked to send a heartbeat to the Kiosk MC heartbeat process.
process status message	MDI Data Server utility routine which will send a message (which will be logged) to the Status Logger.

Table 184 - Routines called by send heartbeat

4.2.2 Kiosk Field Unit

A total of forty (40) Kiosk Field Units are located in areas of the city with high pedestrian and vehicle traffic. Thirty-six (36) Kiosk Field Units are located indoors at various shopping malls and businesses, and four (4) are located outdoors at various tourist attractions and points of interest.

The high level design and data flows of the Kiosk Field Units is shown in Figure 184. Data is received by a Kiosk Field Unit using two communications methods. Real-time data (traffic conditions) are received via FM STIC broadcasts. Non-real-time data (VIA information, airport data, weather data, and screen-saver files) are received via an internal modem. The modem is also used to transfer status information from each Kiosk Field Unit to the KMC.

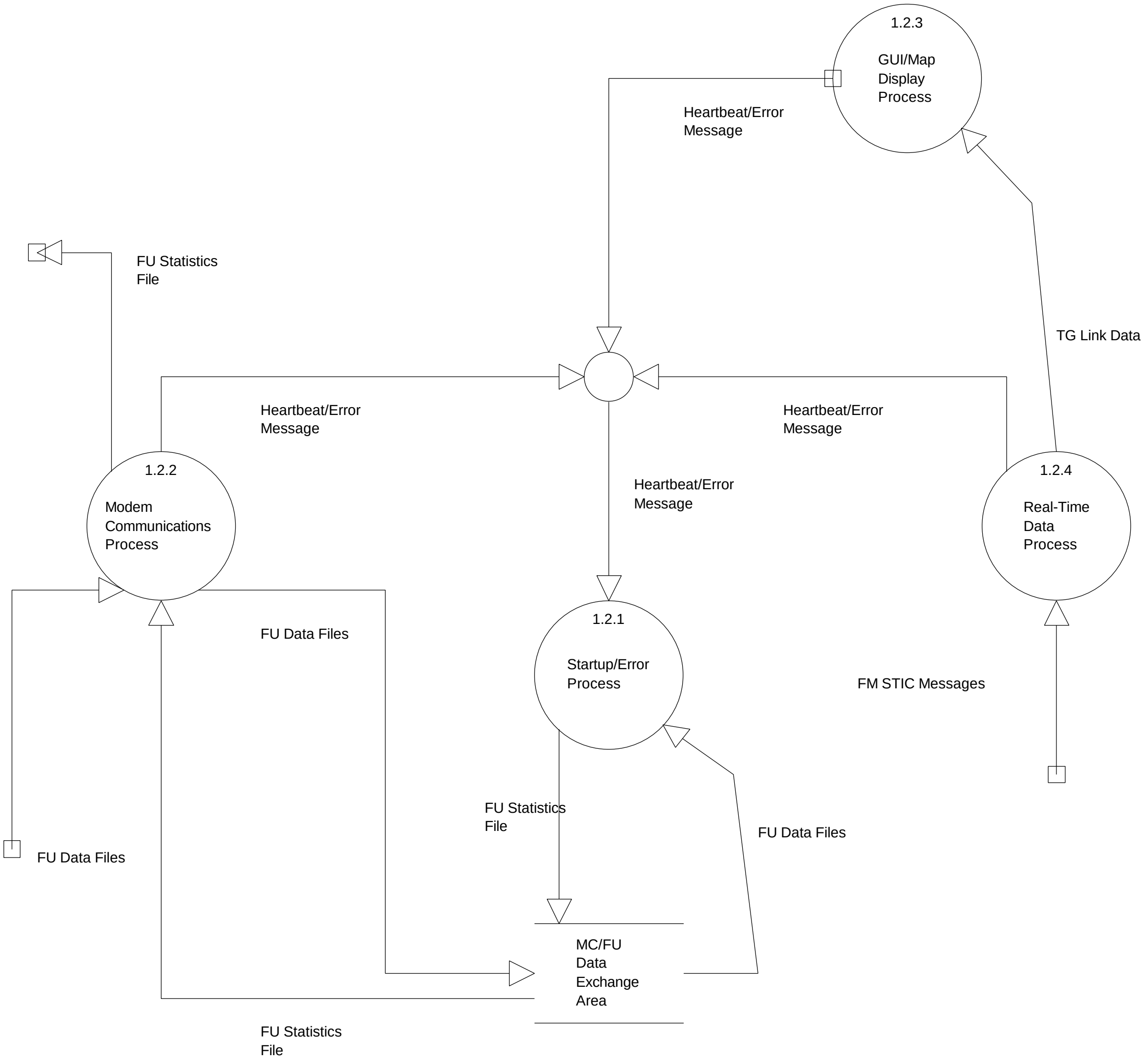

Figure 184. Kiosk Field Unit High Level Design and Data Flows

The Kiosk Field Unit GUI receives VIA, Airport, and weather information from the data communications interface and map data stored locally on the disk. A Kiosk user is able to interactively request a map display, route guidance, airport information, transit information, and weather information at a touch screen terminal. The information requested may be viewed on the terminal screen or printed in a black and white representation.

4.2.2.1 Startup/Error Server Process

The Startup/Error Server Process is started at boot up of the Kiosk Field Unit (KFU). This process is responsible for starting the other KFU processes, monitoring the activity (heartbeat) of the processes it starts, logging errors and messages from the other KFU processes, and moving data files (VIA, Airport, Weather, and Screensaver) from the KMC Received directory to the appropriate Production directory. Figure 185 depicts the events that occur within the Start/Error Server Process and Table 185 provides a description of these events and their triggers.

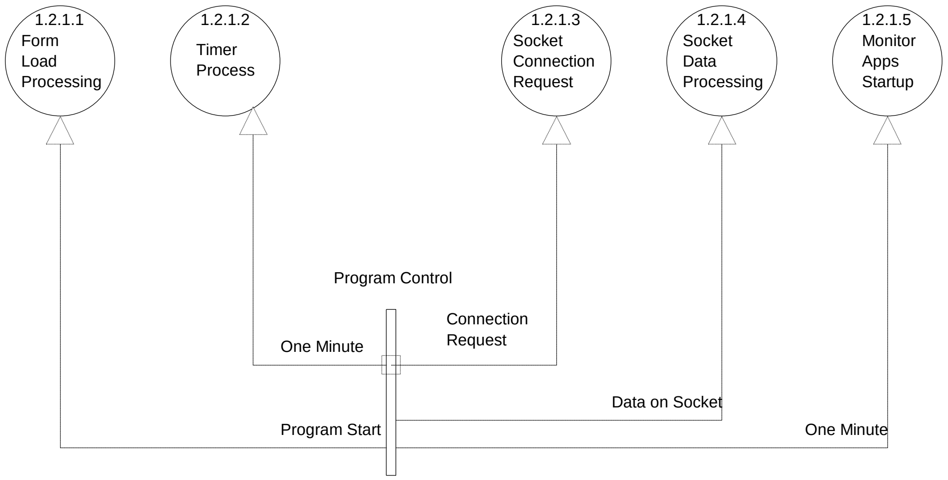

Figure 185 – Startup/Error Events and Event Triggers Structure Chart

Event/Event Trigger	Description
Connection Request	When another application accepts the socket connection to this application, the Socket Connection event is triggered. This event allows the outside application to connect to the socket so that it can send data to this application.
Data on Socket	When data is written to the socket by another application, the Data Arrival event is triggered and the data is processed.
Form Load Processing	This is the initial action performed by the application, when it is started.
Monitor Apps Startup	This event occurs during startup of the Field Unit applications. After an application is started, the Error Server waits for the application to send a heartbeat to indicate that it has successfully started. This timer event monitors how long it has been since the application was started and compares it to the maximum time allowed for an application to start. If the application does not heartbeat within the prescribed time, the Error Server stops existing applications, logs a fatal error, and terminates itself.
One Minute	The minute timer is set to run once every sixty seconds. When the timer executes, the heartbeat status is updated, the Received Data directories are updated, the hour timer is checked and the midnight timer is checked.
Program Control	The control of the events and activities that can occur while an application is executing.

Event/Event Trigger	Description
Program Start	When the program is initiated, the first action of the application is to load the startup form and perform the form load functions.
Socket Connection Request	This event occurs when one application requests a socket connection to another application so that data can be exchanged. The maximum number of connections is incremented. The new connection is initialized and accepted.
Socket Data Processing	This event is triggered when data is sent from one application to another application that is listening for data. When data is detected on the socket, this event is started and the data is processed.
Timer Process	This event is used to control actions that occur on a periodic basis.

Table 185 - Startup/Error Events and Event Triggers Descriptions

The Timer Event within the Startup/Error Server Process is used to cause several other events to take place at varying times. The timer is used to determine when one minute has passed, one hour has elapsed, and when it is midnight. The Timer Event activities are depicted in Figure 186. A description of these activities and the events that trigger these activities is provided in Table 186.

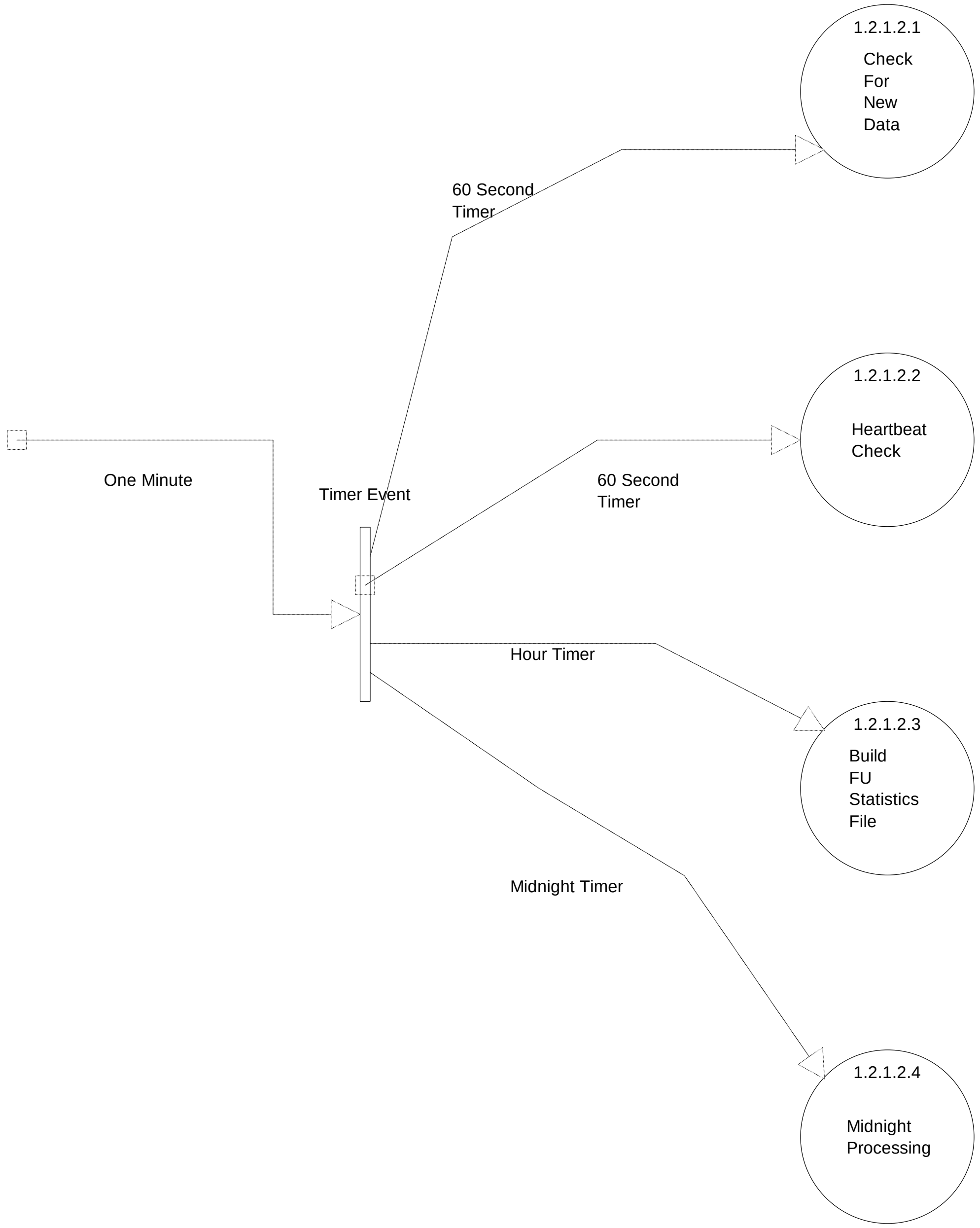

Figure 186 – Startup/Error Server Timer Events

Function	Description
60 Second Timer	This timer is triggered once a minute and the functions scheduled to occur once a minute are performed.
Build FU Statistics File	Builds the Field Unit Statistics file that is used to transmit the Field Unit's current status to the Master Computer.
Check For New Data	Determines whether new data files have been received from the Master Computer. When new data files are found, the files are moved into the production directories.
Heartbeat Check	This activity monitors the applications that were started by the Error Server. The Heartbeat Check ensures that each application has transmitted a heartbeat message within the prescribed heartbeat time as defined in the configuration file.
Hour Timer	This timer is triggered once an hour and the functions scheduled to occur once an hour are performed.
Midnight Processing	Performs a daily set of activities for the Field Unit at midnight.
Midnight Timer	This timer is triggered once a day and the functions scheduled to occur once a day are performed.
One Minute	The minute timer is set to run once every sixty seconds. When the timer executes, the heartbeat status is updated, the Received Data directories are updated, the hour timer is checked and the midnight timer is checked.
Timer Event	This timer is used to control a series of events that occur periodically. These events are checking for the arrival of new data files, checking the heartbeat status of each application, building the field unit statistics file, and performing midnight processing.

Table 186 - Startup/Error Server Timer Event Descriptions

4.2.2.1.1 Form Load

This event is activated when the application is started. The Kiosk Out of Service form is displayed, the Configuration file is processed, the error port is initialized and setup to listen, the error log file is opened, the startup activity is logged, the applications are started, and the Kiosk Out of Service form is unloaded. The structure chart for Form Load is depicted in Figure 187. A description of the routines called by Form Load is provided in Table 187.

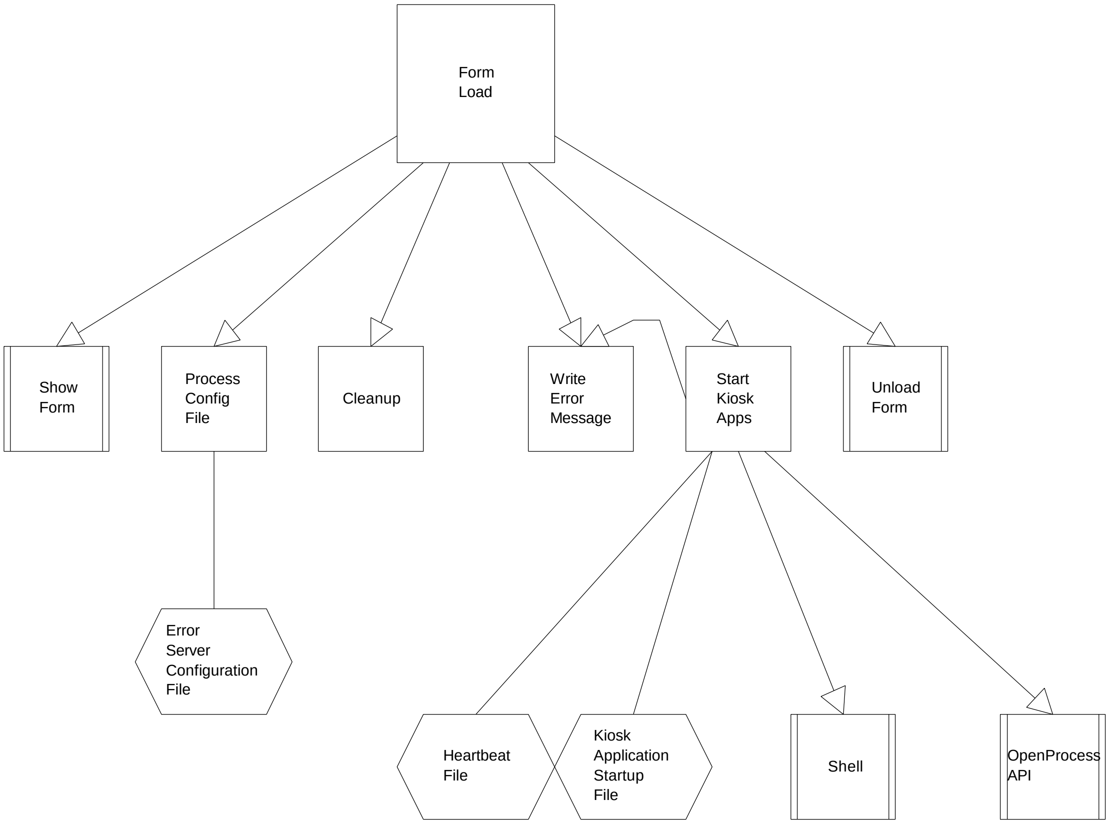

Figure 187 - Form Load Structure Chart

Function	Description
Cleanup	The routine obtains a list of existing files in the window temporary directory and attempts to delete them. This activity is performed to prevent temporary files from building up.
Form Load	This event is activated when the application is started. The Kiosk Out of Service form is displayed, the Configuration file is processed, the error port is intialized and setup to listen, the error log file is opened, the startup activity is logged, the applications are started, and the Kiosk Out of Service form is unloaded.
OpenProcess API	The Open Process Windows API is used to retrieve an application Process Handle. The API accepts the application's process id that is returned by the Shell command. The Process Handle is what is used to terminate the application.
Process Config File	This function opens the configuration file, reads in the configuration items into an array, and closes the file. The format of each line is a text identifier of the item followed by a colon and the value for that item. Each item is then loaded into the appropriate variable for use.
Shell	The Shell is used to start Field Unit applicaitons. The command accepts the application name and window style as input and returns the process id for the application or returns 0 if the application is not started successfully.
Show Form	Visual Basic Event that displays the specified form.

Function	Description
Start Kiosk Apps	This function starts up the Field Unit applications specified in the application startup file. If the heartbeat file exist, it is deleted. The heartbeat file is then opened so that the application data can be written to the file as each application is started. The application startup file is also opened. As each application is read from the application startup file, the heartbeat data structure is loaded, the application is started using the Shell command, the starting of the application is logged, the success of the application starting is tested. If the application starts successfully, the process id is retrieved using the Open Process API, the process id is loaded into the heartbeat structure and the heartbeat structure is written to the heartbeat file. If the application fails to start, the loop is exited and an error is logged. The process continues all until the applications in the application startup file have been processed. If an error occurs, the error is logged and the function is setup to return failure.
Unload Form	Visual Basic Event that removes the specified form from the display and closes it. The form and its fields are no longer accessible until another load or show form is executed.
Write Error Message	The function writes a message into the error log file. The current time and data are prepended to the message and the record is written to the file.

Table 187 - Routines called by Form Load

4.2.2.1.2 Data Arrival

The Data Arrival event is triggered by the arrival of data on the socket. The data is read from the socket and processed. The record is parsed from the data and the first two bytes are used to determine what type of record it is. If the record is a heartbeat record, Process Heartbeat is called. Otherwise, it is an error message and Write Error Message is called to log the message. The structure chart for Data Arrival is depicted in Figure 188. A description of the routines called by Data Arrival is provided in Table 188.

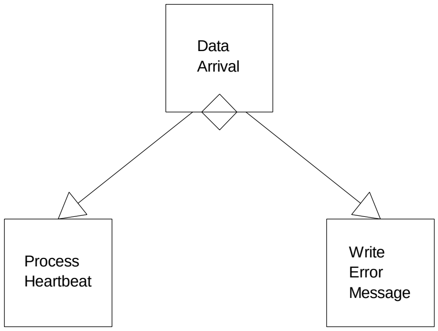

Figure 188 - Data Arrival Structure Chart

Function	Description
Process Heartbeat	This function receives the heartbeat string and parses out the application id. The Heartbeat file is opened and each record read. As the records are read, the application id contained in the record is compared to the one passed into the function. If the ids match, the current time is stored in the application time portion of the record, the record is written back to the file, and the heartbeat is logged. If none of the records contain the application id passed into the function, then an error is logged. The heartbeat file is closed and the function returns.
Write Error Message	The function writes a message into the error log file. The current time and data are prepended to the message and the record is written to the file.

Table 188 - Routines called by Data Arrival

4.2.2.1.3 Check for New Data

The Check for New Data function is designed to move data files from the Received Data directories into the Production directories and delete the specified data files from the Production directories. The function processes data files for transit, screensaver, weather and airport. These file formats are described in the subsections below. The processing for transit and screensaver is unique in that these entities not only have files to move into production, but may also have files that need to be deleted from production. For the transit entity, the temporary directory is checked to see if there is a delete file that still needs to be processed and if found, the Delete File subroutine is called to process the file. Next, the Received Data transit directory is checked for a delete file and if found the Delete File subroutine is called to process the file. The Received Data transit directory is tested for data files. If one or more files are present, the Move Files To Prod function is called to process the files. The Received Data weather directory is tested for files. If one or more files are present, the Move Files To Prod function is called to process the files. The Received Data airport directory is tested for files. If one or more files are present, the Move Files To Prod function is called to process the files. For the screensaver entity, the temporary directory is checked to see if there is a delete file that still needs to be processed and if found, the Delete File subroutine is called to process the file. Next, the Received Data screensaver directory is checked for a delete file and if found the Delete File subroutine is called to process the file. The Received Data screensaver directory is tested for data files. If one or more files are present, the Move Files To Prod function is called to process the files. The structure chart for Check for New Data is depicted in Figure 189. A description of the routines called by Check for New Data is provided in Table 189.

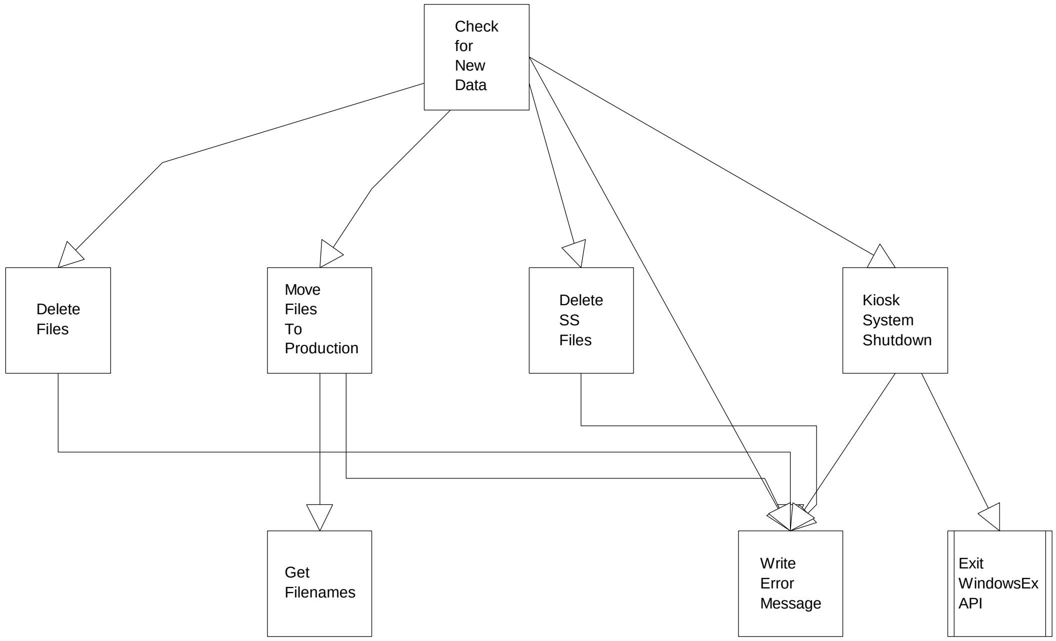

Figure 189 - Check for New Data Structure Chart

Function	Description
Delete Files	Using the delete filename and directory path passed into the routine, the filenames contained in the delete file are deleted from the specified directory path. The delete file is opened. As each filename is read, the directory path is appended and the file is deleted. If a file cannot be deleted, the filename is stored into a temporary delete file. If the temporary delete file contains filenames, it is written to the directory and filename passed into the routine. If an error occurs during normal processing, the error is logged and the routine exits. If an error occurs while trying to delete a file, the error is logged and if the error is NOT file not found, the filename is saved into an array. This array is written to the temporary delete file.
Delete SS Files	This subroutine compares the contents of the screen saver control file with the files in the screen saver directory. If the file in the screen saver directory is not in the control file, then the file is deleted. First, the filenames from the control file are read into an array. Next, the filenames are retrieved from the screen saver directory one by one and compared to each of the filenames from the control file. If a match is not found, then delete the file from the screen saver directory. If an attempt to delete a file fails, store the filename into a temporary delete file that is processed at a later time.
Exit WindowsEx API	A Windows API that provides the capability to shutdown the PC in a number of ways. Error Server used the PowerOff and Shutdown and Reboot options.
Get Filenames	Using the directory path passed into the function, the directory is searched for the existence of files. If one or more files are found, the filenames are written to an array and the number of files is counted. Once the filenames are saved into the array, the function returns the array and the number of files found. If an error occurs during execution, the error is logged and the function returns failure.
Kiosk System Shutdown	This routine shutdowns and reboots the Field Unit. The Kiosk applications are terminated and a message is logged that the system is shutdown down. Next, the Windows API ExitWindosEx is called with the shutdown type set to shutdown and restart the system.

Function	Description
Move Files To Production	Using the Master Computer (MC) directory path and the Production directory path, this function moves files from the MC directory to the Production directory. The function, Get Filenames is called to determine the filenames that need to be moved into the Production directory. If Get Filenames returns failure, Move Files To Prod set the failure code and returns. Using the filenames returned by Get Filenames, Move Files To Prod move each file from the MC directory to the Production directory. The file in the Production directory is deleted. The file in the MC directory is opened and locked for read write so that no other application can access the file while it is being processed. The Production file is opened for read write so that no other application can access the file while it is being processed. The records are read from the file in the MC directory and written to the file in the Production directory. This process continues until the entire file is written to the production directory. Both files are closed and made available to other applications. The file in the MC directory is then deleted. This process is performed for each file in the MC directory. If an occurs while deleting the production file, the error is logged. If the error is file not found, the processing resumes at the next statement. If the error is something other than file not found, the next file is processed. If an error occurs at a location other than the deleting of the production file, the error is logged and the next file is processed.
Write Error Message	The function writes a message into the error log file. The current time and data are prepended to the message and the record is written to the file.

Table 189 - Routines called by Check for New Data

4.2.2.1.3.1 VIA Data File Formats

The information that VIA provides to the kiosk system includes:

- General Information,
- Fare Information,
- Special Events Information,
- VIATrans Information, and
- Route Schedule Information.

VIA stores this information on a Windows NT server in a directory named VIAInfo. Under this directory are subdirectories that contain data files VIA provides as listed above. The following sections describe the data fields that comprises the VIA data. In addition, the descriptions, formats and byte counts are defined.

4.2.2.1.3.1.1 General Information

The VIA General Information file is supplied by VIA in a gif file format. In order to accommodate large amounts of data, there may be multiple files containing general information. The naming convention for these files is as follows:

- GIxx - where GI stands for General Information and xx stands for the page number of the information up to 99 pages (e.g., GI01 is the first page of containing general information).

VIA is responsible for the content and naming of these files.

4.2.2.1.3.1.2 Fare and Pass Information

The VIA Fare and Pass Information file is supplied by VIA in a gif file format. In order to accommodate large amounts of data, there may be multiple files containing fare and pass information. The naming convention for these files is as follows:

- FPxx - where FP stands for Fare and Pass Information and xx stands for the page number of the information up to 99 pages (e.g., FP01 is the first page of containing fare and pass information).

VIA is responsible for the content and naming of these files.

4.2.2.1.3.1.3 Special Events Information

The VIA Special Events Information file is supplied by VIA in a gif file format. In order to accommodate large amounts of data, there may be multiple files containing special event information. The naming convention for these files is as follows:

- SExx - where SE stands for Special Event Information and xx stands for the page number of the information up to 99 pages (e.g., SE01 is the first page of containing special event information).

VIA is responsible for the content and naming of these files.

4.2.2.1.3.1.4 VIATrans Information

The VIATrans Information file is supplied by VIA in a gif file format. In order to accommodate large amounts of data, there may be multiple files containing disability information. The naming convention for these files is as follows:

- DSxx - where DS stands for VIATrans (disability) Information and xx stands for the page number of the information up to 99 pages (e.g., DS01 is the first page of containing disability information).

VIA is responsible for the content and naming of these files.

4.2.2.1.3.1.5 Route Schedule Information

The first set of files will contain information about the bus routes. Each file in this set will contain information for one route only. Also, there will be a corresponding image file (gif) that contains a graphic layout of the route. The file name will be of the format x.txt and x.gif where x is the route number

(1 < x < 999). The format of the text file is described below.

A) Record 1: Effective Date (date route is first used)

e.g. (970524) On May 24, 1997, route became an active route

B) Record 2: Attribute List (currently wheelchair only-if applicable)

e.g. WHEELCHAIR

C) Record 3: Direction

e.g. I (Inbound) / O (Outbound)
N (Northbound) / S (Southbound)
E (Eastbound) / W (Westbound)
C (Clockwise) / CW (Counter Clockwise)

D) Record 4: Service Number

e.g. 1 (Monday through Friday)
2 (Saturday and certain holidays)
3 (Sunday and certain holidays)
4 (Special Service Schedule)
5 (Special Service Schedule)

E) Record 5: Bus Stop Numbers (time points only for the given bus route)

e.g. 34347 (fixed length of 5 digits)

F) Record 6-n, where n = the last set of bus stop times for a given direction and/or service number: Bus Stop Times (one for each bus stop number). The format is hh:mmXX, where XX is a code that succeeds the bus stop time if applicable, and hh:mm represents the time based on a 24-hour clock. No time present indicates the bus does not stop at that particular bus stop. If no code is present then blanks will be present. An example of the codes is G (garage) and FG (from garage). Repeat D-F for each service number and Repeat C-F when a new direction is encountered. See attached file.

The second file is a cross-reference file that lists bus stop numbers against the following fields:

1) full location names,
2) time point (1 or 0),
3) list of associated bus routes that service the bus stop.

The fields are comma delimited and the values are enclosed in quotes. The record format is as follows:

Columns 1-8 bus stop numbers,
Columns 9-51 bus stop names,
Columns 52-55 time points, and
Columns 56+ comma separated bus routes.

The third file is a master service file, servXXXX.txt where XXXX is the year, that consists of 366 (367 for leap years) lines. The first line is a header line indicating routes that follow a normal or special service schedule (e.g. 0 in the first position represents a default service schedule for all routes with the exception of those routes following 0). Each line thereafter consists of a single digit which reflects the service number for that day of the year (e.g. line 2 will have the number 3 which represents a Sunday service schedule since line 2 corresponds to January 1, New Year’s Day, and New Year’s Day follows a Sunday schedule). This method will handle holiday schedules and weekends in a simpler manner.

The fourth file is a route deletion file, viadel.txt, which lists each bus route and associated image file that is to be deleted. For example, 16 indicates that route file (16.txt) and image file (16.gif) are to be deleted.

4.2.2.1.3.2 Weather Data File Formats

The Weather Data is provided by KABB Channel 29 meteorologist, Alex Garcia. Mr. Garcia was awarded the weather provider contract by TxDOT. The Weather Data consists of three files containing a radar image of San Antonio and the surrounding area, the current San Antonio conditions, and the San Antonio five day forecast. These files are supplied by the weather provider in a gif file format. The files are transmitted from the weather provider to the TxDOT Web Server on an hourly basis. Examples of these files can be found in the description of the Field Unit GUI design later in this document.

4.2.2.1.3.3 Airport Data File Formats

The airport information is manually maintained through the System Maintenance GUI, which executes on the KMC. The airport information consists of the following information:

- Airline Terminal Information,
- Rental Car Agency Information , and
- Airport Parking Fees.

This data is maintained in individual files, which are described in the following sections.

4.2.2.1.3.3.1 Airline Information

Table 190 shows the data that comprises airline information. The data is stored in a file named airline.txt.

Field	**Description**	**Format**	**# of Bytes**
Name	Airline Name	ASCII	20
Local Number	Local Telephone Number	ASCII	12
TollFree	1-800 Telephone Number	ASCII	12
Terminal	Airport Terminal Servicing the Airline	ASCII	1

Table 190. Airline Data Fields

4.2.2.1.3.3.2 Rental Car Agency Information

Table 191 shows the data that comprises rental car agency information. The data is stored in a file named Rental.txt.

Field	**Description**	**Format**	**# of Bytes**
Name	Rental Car Agency Name	ASCII	20
Local Number	Local Telephone Number	ASCII	12
TollFree	1-800 Telephone Number	ASCII	12

Table 191. Rental Car Agency Data Fields

4.2.2.1.3.3.3 Parking Fee Information

Table 192 shows the data that comprises parking fee information. The data is stored in a file named parking.txt.

Field	**Description**	**Format**	**# of Bytes**
Location	Parking Lot (Short/Long Term)	ASCII	11
Fee	Price	ASCII	7

Table 192. Airport Parking Fee Data Fields

4.2.2.1.3.4 Screensaver Files

The screensaver files are composed of a control file and the graphics, audio, and video files to be played. The control file is maintained on the KMC, using the System Maintenance GUI. The graphics, audio, and video files are copied to the KMC from an external source and added to the control file through the System Maintenance GUI. The files are maintained separately and described below

4.2.2.1.3.4.1 ScreenSaver Control File

The file contains the name of the file(s) to execute, the file(s) type, and how long the file(s) are to be played. Table 193 contains the format of the control file.

Field	**Description**	**Format**	**# of Bytes**
Filename	Name of file(s) to be played	ASCII	8
Type	Specifies the type of file(s) to be played, 1 – bitmap only 2 – bitmap and audio file(s) together 3 – video file	ASCII	1
Length	Amount of time in milliseconds to play the file(s)	ASCII	5

Table 193 – Screensaver Control File Data Fields

4.2.2.1.3.4.2 Graphics, Audio, and Video Files

The graphics, audio, and video files are copied to the KMC from an external source (e.g., floppy disk). The format of the graphics is required to be .bmp. The format of the audio files is required to be .wav. The format of the video files is required to be .avi.

4.2.2.1.4 Perform Heartbeat Check

This routine monitors the applications that were started by the application. The routine opens the heartbeat file, reads the application information from the file, and compares the last heartbeat time to current time. If the difference between the two times is greater than or equal to the heartbeat timeout (e.g., 5 minutes), then the Kiosk Out of Service form is displayed, the heartbeat file is closed, the applications are stopped, the applications are restarted, the Kiosk Out of Service form is removed and the loop is exited. If the difference between the two times is less than the heartbeat timeout, the next application is checked. When the check application loop is complete, the routine returns. The structure chart for Perform Heartbeat Check is depicted in Figure 190. A description of the routines called by Perform Heartbeat Check is provided in Table 194.

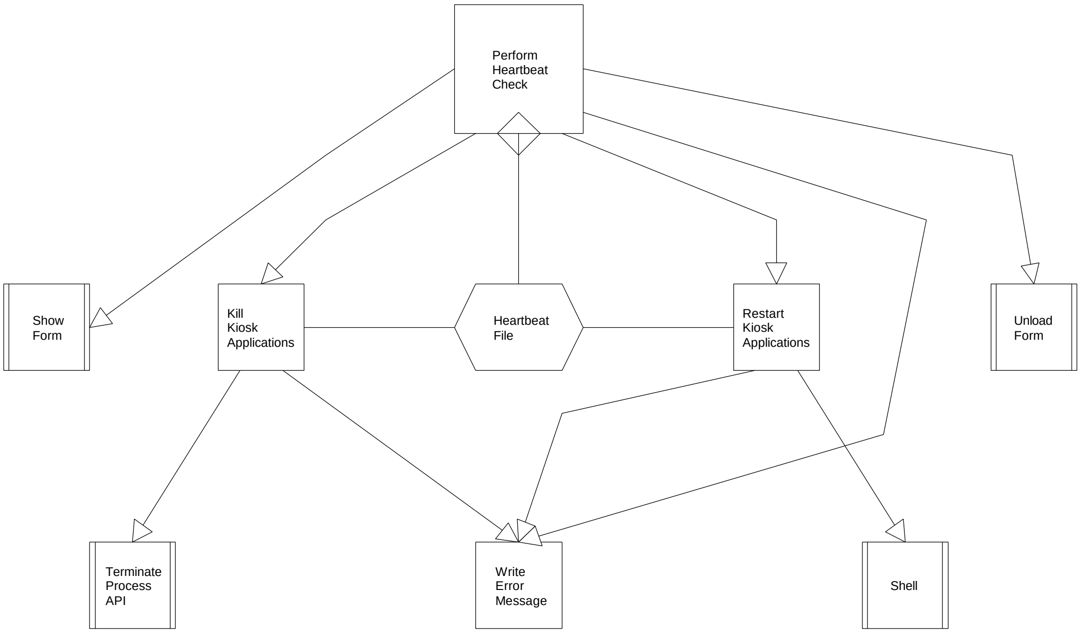

Figure 190 - Perform Heartbeat Check Structure Chart

Function	Description
Kill Kiosk Applications	This function terminates the applications started by this application. The heartbeat file is opened and each started application's information is read from the file into the heartbeat status data structure. From this structure, the application's process id is read and used to terminate the application. A Windows API call is made to the TerminateProcess API passing the application's process id. When the API returns, a message is logged that the application was terminated and the next application is processed. Once the applications have been terminated the heartbeat file is closed and the function returns success.
Restart Kiosk Applications	This function restarts the Field Unit applications specified in the heartbeat status file. Heartbeat monitoring is disabled and the program state is set to startup. The heartbeat status file is opened, the heartbeat records are read from the file, and the file is closed. For each record (application), the application name and window style are read from the record and used to restart the given application using the Shell command. If the application starts successfully, the process id is retrieved using the Open Process API. The new process id and current time are loaded into the heartbeat structure. Next, the application id passed into this function is compared with the application id that is being restarted. If these application ids match, the number of times the application has been restarted is incremented by one and stored into the heartbeat structure. The heartbeat status file is then opened, the record is written back to the file and the file is closed. The function then waits until the application has successfully started by monitoring the startup flag and the application timeout flag. If the application times out, the function returns failure. Otherwise, the application returns success.

Function	Description
Shell	The Shell is used to start Field Unit applications. The command accepts the application name and window style as input and returns the process id for the application or returns 0 if the application is not started successfully.
Show Form	Visual Basic Event that displays the specified form.
Terminate Process API	This function is a Windows 32 API that terminates an application based on the handle (process) id that is passed to the function. The application is immediately terminated without warning.
Unload Form	Visual Basic Event that removes the specified form from the display and closes it. The form and its fields are no longer accessible until another load or show form is executed.
Write Error Message	The function writes a message into the error log file. The current time and data are prepended to the message and the record is written to the file.

Table 194 - Routines called by Perform Heartbeat Check

4.2.2.1.5 Build HB Stats File

The Build HB Stats File routine builds the Field Unit Statistics file. If the previous Statistics file exists, it is deleted. The Statistics file is opened and locked for Read/Write, so that no other applications can access the file while it is being built. The status of the Field Unit applications and the estimated number of pages printed are determined. Next the Usage Statistics file is opened for Read/Write, so that no other applications can access the file while it is being processed. The usage statistics are read from the file and stored locally. Using the GetDiskFreeSpace Window API, the amount and percentage of available disk space is calculated. The number of times the kiosk has been restarted is determined and stored locally. The statistics data is written to the Field Units Statistics File. The structure chart for Build HB Stats File is depicted in Figure 191. A description of the routines called by Build HB Stats File is provided in Table 195.

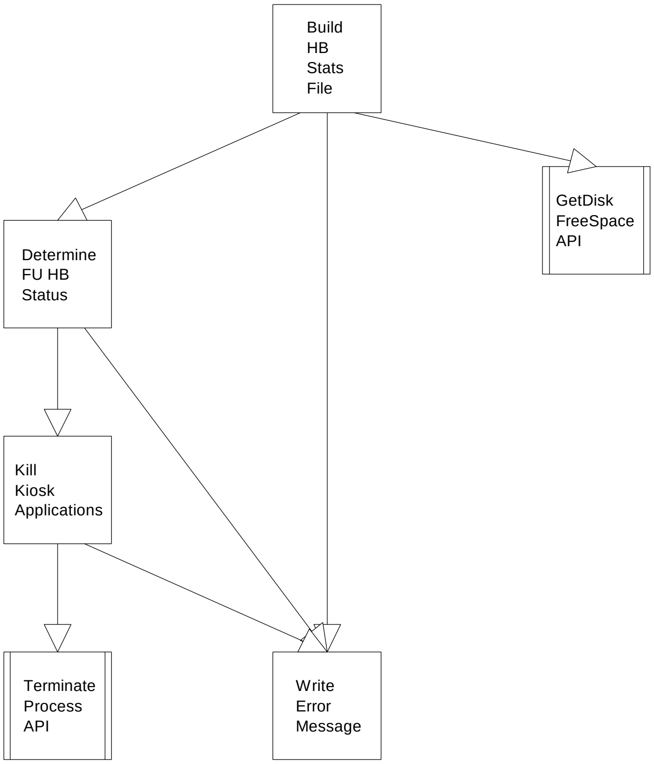

Figure 191 - Build HB Stats File Structure Chart

Function	**Description**
Determine FU HB Status	The function determines the overall status of the Field Unit, the status of each application, and the estimated number of pages printed. The Heartbeat File is opened and each application's heartbeat data is read using the Heartbeat Status Data Structure. The number of times the application has been restarted is compared to the restart limits. If the application has been restarted more than once, but less than 3 times, the overall Field Unit status is updated to warning (Yellow). If the application has been started more the 3 times, the overall Field Unit status is updated to Out of Service (Red). The status information for the Field Unit and each application are saved and returned. In order to determine the estimated number of pages printed, the Paperlow File is opened and the estimated number of pages printed is read from the file. If the number of pages printed is greater than the Paper Threshold (700), the overall Field Unit status is updated to Warning. The Heartbeat and Paperlow files are closed and the status and paperlow statuses are returned.
GetDisk FreeSpace API	The API is provided using the Windows 32 Application Program Interfaces. The function is passed the Drive and Directory path to be used to determine the disk free space. The function returns the sectors per cluster, bytes per sector, number of free clusters, and the total clusters. The values are used to calculate the number of bytes available.
Kill Kiosk Applications	This function terminates the applications started by this application. The heartbeat file is opened and each started application's information is read from the file into the heartbeat status data structure. From this structure, the application's process id is read and used to terminate the application. A Windows API call is made to the TerminateProcess API passing the application's process id. When the API returns, a message is logged that the application was terminated and the next application is processed. Once the applications have been terminated the heartbeat file is closed and the function returns success.

Function	Description
Terminate Process API	This function is a Windows 32 API that terminates an application based on the handle (process) id that is passed to the function. The application is immediately terminated without warning.
Write Error Message	The function writes a message into the error log file. The current time and data are prepended to the message and the record is written to the file.

Table 195 - Routines called by Build HB Stats File

4.2.2.1.6 Midnight Processing

The Midnight Processing activity is spawned by the minute timer. Each time the timer executes, the routine determines if midnight has passed. If it is midnight or later, the routine performs the midnight activities. These activities are cleaning up the error log file, deleting temporary files from the windows temp directory, and shutting down the system and rebooting The structure chart for Midnight Processing is depicted in Figure 192. A description of the routines called by Midnight Processing is provided in Table 196.

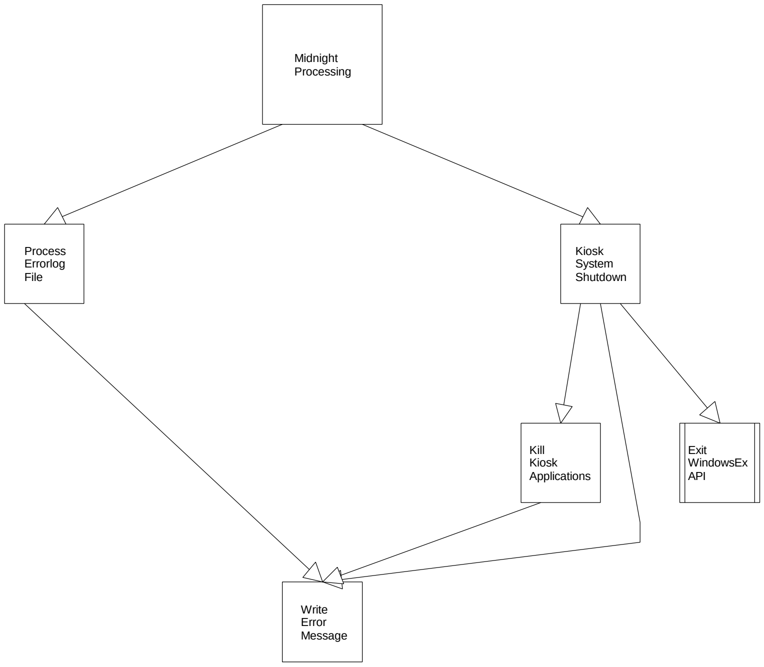

Figure 192 - Midnight Processing Structure Chart

Function	Description
Exit WindowsEx API	A Windows API that provides the capability to shutdown the PC in a number of ways. Error Server used the PowerOff and Shutdown and Reboot options.
Kill Kiosk Applications	This function terminates the applications started by this application. The heartbeat file is opened and each started application's information is read from the file into the heartbeat status data structure. From this structure, the application's process id is read and used to terminate the application. A Windows API call is made to the TerminateProcess API passing the application's process id. When the API returns, a message is logged that the application was terminated and the next application is processed. Once the applications have been terminated the heartbeat file is closed and the function returns success.
Kiosk System Shutdown	This routine shutdowns and reboots the Field Unit. The Kiosk applications are terminated and a message is logged that the system is shutdown. Next, the Windows API ExitWindosEx is called with the shutdown type set to shutdown and restart the system.
Process Errorlog File	This routine renames the current error log file using the day of the week as the extension. A week's worth of error log files are maintained.
Write Error Message	The function writes a message into the error log file. The current time and data are prepended to the message and the record is written to the file.

Table 196 - Routines called by Midnight Processing

4.2.2.2 Graphical User Interface

Kiosk Field Units contain a GUI based display that allows a user to navigate the system by touch. The user can access real-time traffic conditions, a San Antonio Area map display, airport information, weather information, VIA bus information, and route guidance through a series of GUIs. Icons, buttons, and images for the GUI of the Kiosk Field Units are custom designed using a San Antonio/Texas theme. The artwork is stored in one of the following formats: BMP or GIF. The primary events that occur and the actions that trigger these events are depicted in Figure 193.

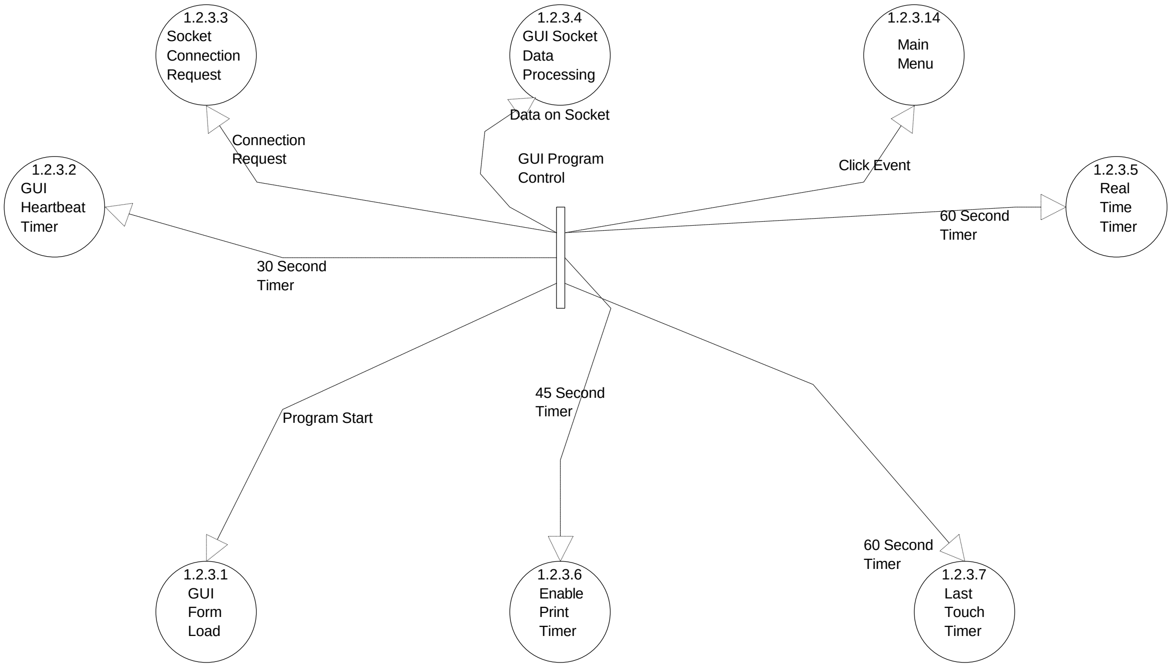

Figure 193 - GUI Events and Event Triggers Structure Chart

Function	Description
30 Second Timer	This event is set to occur every thirty seconds. When the event is triggered, the application will perform the actions specified in the timer.
45 Second Timer	This event is set to occur every forty-five seconds. When the event is triggered, the application will perform the actions specified in the timer.
60 Second Timer	This timer is triggered once a minute and the functions scheduled to occur once a minute are performed.
Click Event	This event is activated when the user touches a button on the GUI.
Connection Request	When another application accepts the socket connection to this application, the Socket Connection event is triggered. The event allows the outside application to connect to the socket so that it can send data to this application.
Data on Socket	When data is written to the socket by another application, the Data Arrival event is triggered and the data is processed.
Enable Print Timer	This timer disables itself and enables the printer capability after the print button has being repeatedly selected too many times.

Function	Description
GUI Form Load	When the form load event is activated, the configuration file is processed, the command line parameters are retrieved, the screensaver state is set to false, the error logging capabilities are initialized, the color arrays are loaded, the primary forms are loaded, the last touch time to determine the length of time between touches is initialized to the current time, the usage counts are initialized, the printer is enabled, the screensaver and real time timers are initialized, the real time socket is initialized, the volume is initialized, a heartbeat is transmitted, and a successful startup message is transmitted.
GUI Heartbeat Timer	This event is triggered every thirty seconds to transmit the heartbeat record to the Error Server and to check for new weather, airport and transit files.
GUI Socket Data Processing	This event is triggered when data is sent from the real time application to the GUI application that is listening for data. When data is detected on the socket, this event is started and the data is processed. The data is retrieved from the socket and broken into records. The records are divided by a record delimiter. Each record contains either speed or incident data. The record type, starting longitude, starting latitude, starting level, ending longitude, ending latitude, ending level, and speed or incident code are extracted from each record and loaded into the global real time data array.
Last Touch Timer	This timer first checks to see if the screensaver is already executing. If the screensaver is not executing, the last touch time is subtracted from the current time and the difference is compared to the screensaver timeout. If the difference exceeds the timeout, then the current form is either unloaded or hidden (this is based on the forms that are loaded at startup), the timer is disabled, and the screensaver control file is read. If there is a problem with the screensaver control file, the screensaver is not executed. Otherwise, the first file(s) are processed.
Main Menu	This event is triggered when the user touches one of the buttons on the SA Online Main Menu. The possible choices are Help, Volume, SA Map, Weather, Transit (Via) and Weather.
Program Start	When the program is initiated, the first action of the application is to load the startup form and perform the form load functions.
Real Time Timer	When this timer is activated, the timer activate event is disabled to prevent the timer from being triggered while the code inside the timer is executing. The current form and the screensaver state are checked. If the current form is the SA Map form or the screensaver state is true and real time data is available, then the SA Map is updated with the most current real time data. Once the timer completes execution, the timer is enabled.
Socket Connection Request	This event occurs when one application requests a socket connection to another application so that data can be exchanged. The maximum number of connections is incremented. The new connection is initialized and accepted.

Table 197 - GUI Events and Event Triggers Descriptions

4.2.2.2.1 Main Menu Screen

The Main Menu Screen defines a hierarchy of options available to the user. The user can select to view the San Antonio Area Map, view Airport information, view Weather information, view VIA information, perform Route Guidance, and obtain Help. Each of these options are further divided into one or more informational screens. The structure chart for Main Menu Screen is depicted in Figure 194. A description of the routines called by Main Menu Screen is provided in Table 198.

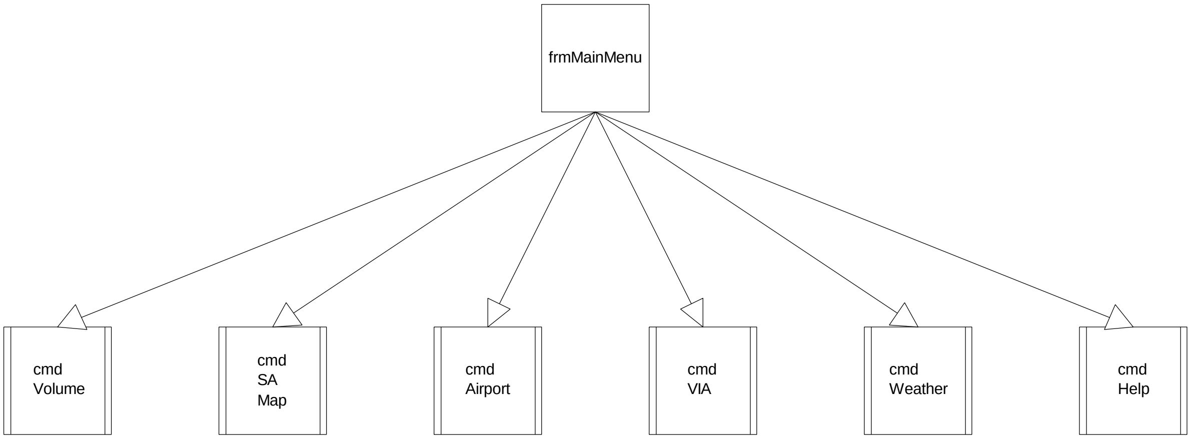

Figure 194 - Main Menu Screen Structure Chart

Function	Description
cmd Airport	This button increments the number of airport accesses by one and activates the airport GUI.
cmd Help	The button displays help for the specified GUI from which it is selected. The appropriate bitmap file is retrieved and displayed.
cmd SA Map	This button increments the number of San Antonio Map accesses by one and activates the San Antonio Map display.
cmd VIA	This button increments the number of transit accesses by one and displays the transit GUI.
cmd Volume	This button changes the volume of the Kiosk. Each time the button is selected the volume is raised until it reaches maximum volume. The next selection of the button mutes the volume and another selection of the button sets the volume to it lowest audible setting.
cmd Weather	This button increments the number of weather accesses by one and activates the weather GUI.
frmMainMenu	This is the initial form that is displayed for the user.

Table 198 - Routines called by Main Menu Screen

4.2.2.2.2 San Antonio Area Map

The San Antonio Street Map Display is the primary method of displaying data to the user at the Kiosk Field Unit. The map, which is based on the Navigation Technologies San Antonio Region database, displays major arterials, city streets, residential streets, road labels, highway signage, hospitals, schools, parks, and airports. The user may use touch-screen input to zoom or pan the map display.

The map distinguishes road classification by line segment size and/or color and distinguishes real-time travel speeds on instrumented roadways using color-coded line segments. For example, roadways with normal traffic speeds are indicated in green, while roadways with below-normal traffic speeds are indicated in yellow and red, respectively.

The map displays icons indicating current traffic incidents and lane closures. The icons reveal detailed information about the incident and lane closures when the icon is touched by the user. The map also indicates in icon form, at the request of the user, the location of selected Points of Interest such as automated teller machines (ATMs), shopping centers, restaurants, gas stations, and tourist attractions. The structure charts for Main Menu Screen are depicted in Figure 195, Figure 196, and Figure 197. A description of the routines called by Main Menu Screen is provided in Table 199.

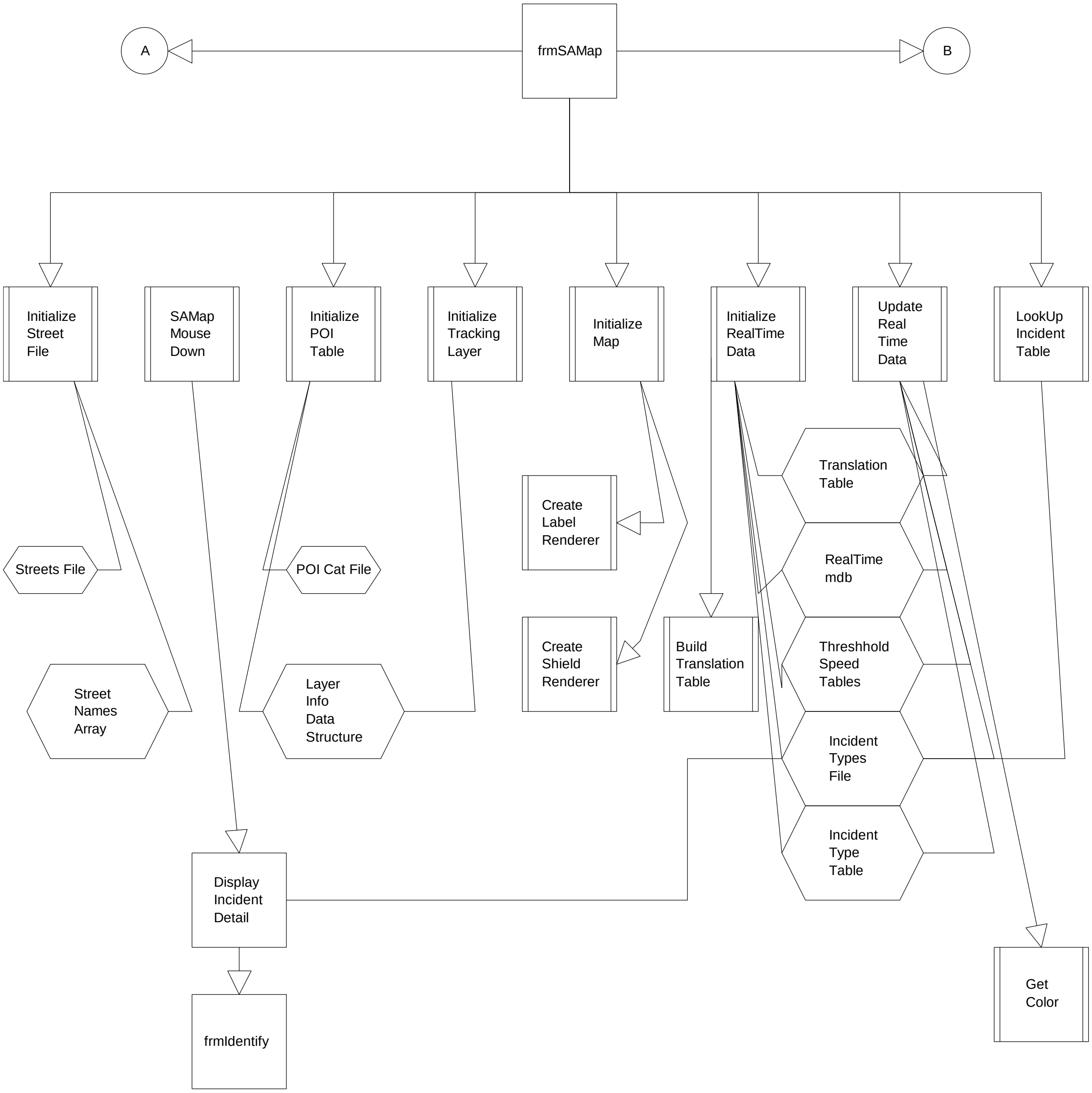

Figure 195 - San Antonio Area Map Structure Chart 1 of 3

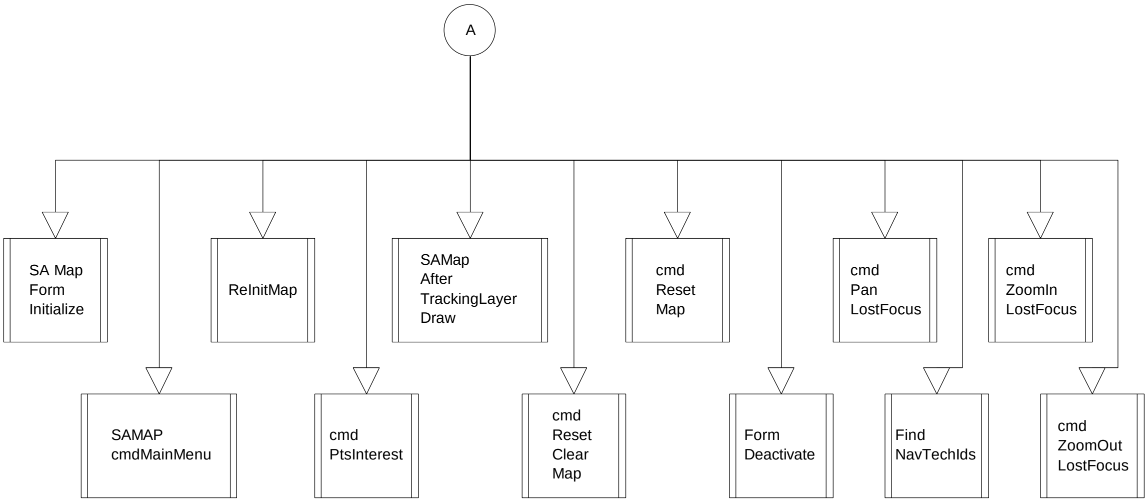

Figure 196 - San Antonio Area Map Structure Chart 2 of 3

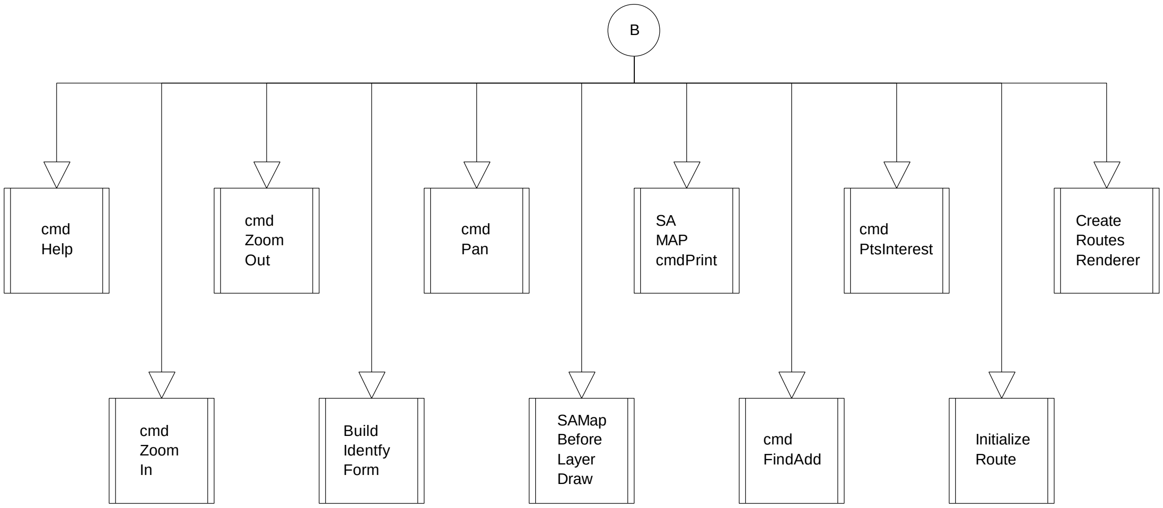

Figure 197 - San Antonio Area Map Structure Chart 3 of 3

Function	Description
Build Identify Form	This subroutine creates field values for the Identify Form and determines if the GetMeThere button should be displayed. Load the Identify form with name, address, and phone number. Set display criteria for form depending on the values of fields. Make the form large enough to display the larger of the name or address field. If address can be geocoded, then add GetMeThere button to the form. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError.
Build Translation Table	This function loads the translation table file into the Access database, translation_table.mdb. The Translation Table file (trns_tbl.dat) is opened and the first record (number of Transguide links) is read. Next the Transguide link information and each NavTech link ID associated with the Transguide links information is read. The data for each Transguide link is terminated by a record containing only "-1". For each Transguide Link, perform the following: read the Transguide Link ID, starting longitude, starting latitude, starting level, ending longitude, ending latitude, ending level, and street name. Read the Navtech IDs and bearings until a "-1" is encountered, build a NavTech ID string by concatenating the Navtech IDs and bearings with a space between each and incrementing the NavTech Count with each. Build the string that will be used as the primary index by concatenating the starting latitude, starting longitude, starting level, ending latitude, ending longitude, and ending level. Replace any apostrophes in the Transguide Link ID with underscores. Insert into the Translation Table Database the following: starting latitude, starting longitude, starting level, ending latitude, ending longitude, ending level, number of NavTech IDs, NavTech ID string, bearing, index string, and Transguide Link ID. After all Transguide Links have been processed, sort the database (by index string) and count the number of records. Build the Transguide-to-NavTech table by reading each record from the database into the table. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError.
cmd FindAdd	Read a list of street names from a file and add the names to the Find Address form.
cmd Help	The button displays help for the specified GUI from which it is selected. The appropriate bitmap file is retrieved and displayed.
cmd Pan	If current scale is 1, then instruct user that map can not be moved and disable map. Otherwise, instruct user to touch map and move in any direction.
cmd Pan LostFocus	This event occurs when the user has selected Pan and moved their finger along the map. If current scale is 1, then instruct user that map can not be moved. Otherwise, reset label instructions to default (Touch Any Button to Continue). While the map is redrawing, this routine displays the Please Wait message.
cmd PtsInterest	Determine which label instructions should be displayed. If there are more layers displayed that street layers plus shields (implying that there is a POI layer), then instruct the user to touch any button or icon. Otherwise, instruct the user to touch any button.
cmd Reset Clear Map	Re-initialize map (ReInitMap), set current scale to 1, and display map at its fullest extent.
cmd Reset Map	Resize the map to its full extent, but do not clear the layers.
cmd Zoom In	Reset label instructions based on current scale. If map is at maximum scale, then instruct user that map can not be scaled and disable map. Otherwise, instruct user to touch map where more detail is desired.
cmd Zoom Out	Reset label instructions based on current scale. If current scale is 1, then instruct user that map can not be scaled and disable map. Otherwise, instruct user to touch map where less detail is desired.
cmd ZoomIn LostFocus	This event occurs when the user has selected Zoom In and touched an area on the map. If current scale is at maximum scale, then instruct user that map can not be scaled. Otherwise, reset label instructions to default (Touch Any Button to Continue). While the map is redrawing, this routine displays the Please Wait message.
cmd ZoomOut LostFocus	This event occurs when the user has selected Zoom Out and touched an area on the map. If current scale is 1, then instruct user that map can not be scaled. Otherwise, reset label instructions to default (Touch Any Button to Continue). While the map is redrawing, this routine displays the Please Wait message.

<table>
<tr><th>Function</th><th>Description</th></tr>
<tr><td>Create Label Renderer</td><td>This subroutine creates and displays street labels. The renderers are assigned to the all_rds and fast_rds layers. For each renderer, assign values to the label placers for link classes 1, 2, 3, 4, and 9. NOTE: This renderer (LabelPlacer) is part of MoPlus, a set of unsupported objects. These objects have not been beta-tested and ESRI does not provide support for them.</td></tr>
<tr><td>Create Routes Renderer</td><td>This subroutine renders the street segments with colors representing realtime speeds.
Create a strings collection with each possible color value from the colors array. For each color (green, lime, yellow, orange, red), concatenate a "1" for half the values and concatenate a "2" for the others. The strings collection now contains the following values: green1, lime1, yellow1, orange1, red1, green2, lime2, yellow2, orange2, & red2.
For each value in the link_rds shapefile (access database related to it), the specific values are displayed as follows:
<table>
<tr><th>color code</th><th>Color</th><th>Line Thickness</th></tr>
<tr><td>green1</td><td>DarkGreen</td><td>1</td></tr>
<tr><td>green2</td><td>DarkGreen</td><td>2</td></tr>
<tr><td>lime1</td><td>Green</td><td>1</td></tr>
<tr><td>lime2</td><td>Green</td><td>2</td></tr>
<tr><td>yellow1</td><td>Yellow</td><td>2</td></tr>
<tr><td>yellow2</td><td>Yellow</td><td>4</td></tr>
<tr><td>orange1</td><td>Maroon</td><td>2</td></tr>
<tr><td>orange2</td><td>Maroon</td><td>3</td></tr>
<tr><td>red1</td><td>Red</td><td>2</td></tr>
<tr><td>red2</td><td>Red</td><td>3</td></tr>
<tr><td>all others</td><td>Black</td><td>1</td></tr>
</table></td></tr>
<tr><td>Create Shield Renderer</td><td>This function creates the highway shields for the SA Map and places them on the Shields layer. This is done only for the top level map.</td></tr>
<tr><td>Display Incident Detail</td><td>Compare the point where user touched against incident points in the realtime table. If the user touched a point "close enough" to an actual incident, then display the incident type. Find location where the screen was touched. Convert the touched xy coordinates to strings and compare to the location of each current incident. Calculate the distance between that point and any incident icon and consider that distance to be the shortest and that incident to be the closest. For each incident, calculate the distance between the incident and the point touched, and if the distance is shorter than the shortest distance, make that incident the closest. If the shortest distance is no more than a predefined distance (assigned to be .005/current scale) then look up the incident type and write that type to the Identify form. (refer to frmIdentify.Identify).</td></tr>
<tr><td>Find NavTechIds</td><td>This function looks up the Transguide linkid and returns the associated NavTech linkids, the number of links, the bearing and the color for each link. The search key is built from the update real time array based on the pointer passed to the function. The key is composed of the starting longitude, starting latitude, starting level, ending longitude, ending latitude, and ending level. The function using a binary search, searches through the Link Id table looking for a match on the Transguide Linkid key. When a match is found, the number of links and the pointer into the NavTech link id is read from the Translation table. For each NavTech linkid associated with the Transguide linkid, the linkid, color and bearing are retrieved. When the information has been assimilated, the data is returned.</td></tr>
<tr><td>Form Deactivate</td><td>Reset map and caption if any form other than frmPtsInterest, frmIdentify, frmFindPOI, or frmFindAdd are displayed.</td></tr>
<tr><td>FrmIdentify</td><td>This form displays the name and address of either the Point of Interest or address selected by the user.</td></tr>
<tr><td>FrmSAMap</td><td>This GUI displays the San Antonio Street Map, real time traffic data if it is available, and the user interactive functions for the map (i.e., Zoom In, Zoom Out, Pan, Find Address, Find Point of Interest, Reset Map, and Reset and Clear Map).</td></tr>
</table>

Function	Description
Get Color	This function returns a color-width code with which to color each map link segment. Based on the link class input, determine which speed threshold table to use. There is a speed threshold table for each link class of 1 (roads with average speed of greater than 45 mph), link class 2 (roads with average speed greater than 30 mph), and link class 3 (road with average speed greater than 20 mph). Once the speed threshold table is selected, check speed input against each array value. If the input speed is greater than the array value, select the color by indexing into the colors array with the corresponding array index from the thresholds table. If the link class is "1", concatenate a "2" to the color value, otherwise concatenate a "1". This value will be used as the value of the thickness of the line drawn on the map. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError.
Initialize Map	This subroutine creates the San Antonio street map street layers and label renderers. Clear Instruction Label, set current scale level to 1, initialize RealTimeData flag to False, and set the map background color. Connect to the database and add layers corresponding to different zoom levels of the map. Create a layer that consists of 2 separate layers to display streets that have thickness: create one layer of thickness 5 and color black to display the outside edge of the streets and create one layer of thickness 3 and color white to mark the inside. Both layers are created from the "All_rds" shapefile. Create a layer to display streets that average more than 30 mph. This layer is created from the "Fast_rds" shapefile. Create a layer to display major arteries from the "Maj_rds" shapefile. Create a layer to display highway shields at the top layer of the map. This layer is created from the "Shields" shapefile. Render this layer to display different shields for different roads. Create a layer to display real-time data from the shapefile, "Lnk_rds". Render the labels for the all_rds layer and fast_rds layer. Find the address for the kiosk and add event to the tracking layer. Display map at its fullest extent. If an error occurs, disable the SA Map option from the Main Menu, build an error message using the error number and error message (if known) and send it to LogError.
Initialize POI Table	This subroutine builds the Points of Interest table in memory. Open the POI Category file. This file contains the parameters for each type of POI category. Read each record into a globally-defined data structure. Based on the value of the color read from the file, assign a MapObjects color constant. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError.
Initialize RealTime Data	This subroutine builds a Transguide-to-Navtech-Link Lookup table, relates the lnk_rds shapefile (realtime links) to an Access database table, initializes the speed threshold arrays, and initializes the incident types array. Read the translation table into memory. If an error occurs, ensure that realtime data will not be used. Perform an AddRelate to initialize the relationship between the lnk_rds shapefile and Access database. Open the database and create a recordset of all records. Initialize each record to null or false. Initialize the speed threshold arrays. Initialize the incident types data structure by reading the incident text file. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError.
Initialize Route	This subroutine prepares the network engine for route guidance. Call AddNetLayer to create new layer. Set global variable ROUTE_GUIDE_VALID to true if new net layer was created and if the kiosk has a valid location.
Initialize Street File	If the streets file does not exist, then build it from the all_rds shapefile as follows: Build a recordset of all the records in the all_rds shapefile. Create a strings collection of each full street name. A strings collection is used to avoid duplicate names. Write the contents of the strings collection to the streets file. Write the street names to the FindAdd form and street name array for later use in Find Address function. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError.
Initialize Tracking Layer	This subroutine initializes the tracking layer to display the kiosk location icon, incident icons, and user-specified POI and address icons. Set up the icon properties. The total number of tracking layer icons is equal to the number of POI icons identified in the POI category file, plus the kiosk icon, plus any incident icons displayed, plus a user-selected POI or address destination icon if active. Initialize incident icons (red circle around the letter "i"). Initialize kiosk location icon (navy blue star). Initialize user-specified destination (dark green asterisk). Initialize icon properties for each POI category defined in POI categories file. These properties were previously read into LAYER_INFO data structure. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError.
LookUp Incident Table	This subroutine searches the incident table to determine if the incident exists or to delete the record. From realtime data, incidents can be identified by a point (lat/lon) or a segment (identified by starting lat/lon and ending lat/lon). If the former, the incident is displayed on the map by displaying an icon at the point specified. If the latter, the incident is displayed on the map by displaying an icon at the starting point of the link. Therefore, incidents are always displayed at the starting lat/lon of the realtime record. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError.

Function	Description
ReInitMap	This subroutine reinitializes the San Antonio street map display by removing all POI layers and all icons. The kiosk location icon and incident icons are redisplayed. If there is a route line, it is cleared and set to nothing. If there are layers displayed other than street layers, delete them. Clear the tracking layer. Redisplay the kiosk location. Redisplay incidents. Clear the route line. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError.
SA MAP cmdPrint	This subroutine prints the SA Map, NavTech Logo and disclaimer as a minimum. If the identify form is currently displayed, this routine will also print the name, address, and phone number. Finally, if a route was calculated, the turn by turn instructions will also be displayed.
SA Map Form Initialize	Initialize Streets File, POI Table, Tracking Layer, Map, Real Time Data, and Route Guidance.
SAMap After TrackingLayer Draw	Executes after Tracking Layer has been drawn which is done after all other map layers have been drawn. If there exists output from the Route Guidance application, display the line. Reset label instructions.
SAMap Before Layer Draw	Executes before each layer is drawn. Resizes Incident icons, Points of Interest icons for Tracking Layer, Points of Interest icons for map layers, and Highway shields. Sets all layers to false and determines, based on current scale of map which should be visible. For example, when the full extent of the map is displayed, the current scale is 1 and only the TopView and Shields are displayed. When the map is fully zoomed in, the current scale is equal to the max scale and the Black and White 1604 layers are visible.
SAMAP cmdMainMenu	Hide the SA Map GUI and show the SA Online Main Menu.
SAMap Mouse Down	Activated when the screen in touched, or the mouse is clicked. Based on which button is active, perform the following: Zoom In - If the current scale is less than the maximum scale, the center is set to the location touched and the map is scaled to 1/scale factor. Scale Factor is set to 3.5. Maximum scale is set to 5 Zoom Out - If the current scale is at least 3, then the center is set to the location touched and the map is scaled by the scale factor. If the current scale is 2, then the full extent of the map is displayed. Pan - If the current scale is more than 1, then the MapObject property, Pan is invoked. If there are no active buttons, then the user has touched the map. Perform the following based on the state of the map: If the Points of Interest layer is displayed (i.e., there is a layer consisting of points other than the Shields layer), then call frmIdentify.Identify with the parameters x and y (point touched). If any incidents are displayed (incidents are displayed in the tracking layer and are identified by NUMBER_INC_RECORDS > 0) then call DisplayIncidentDetail.
Update Real Time Data	This subroutine processes the real time data that has been received. In order to update incident icons, delete the previously defined incident icons. Next, the latest real time data is copied to a working array, so that the latest incidents and speeds can be processed. For each Transguide Linkid that contains a speed, the associated NavTech Linkids and their current color is retrieved. For each NavTech linkid, the current speed is converted to a color and the color is compared to the color retrieved. If the colors are different, the color is updated. For each Transguide Linkid that contains a negative incident id, the incident is removed from the incident list. For each Transguide Linkid that contains incident data, the incident is added to the incident table. Finally if the SA Map is currently being displayed, the map is refreshed.

Table 199 - Routines called by San Antonio Area Map

4.2.2.2.3 Find Address

The Find Address GUI allows the user to enter an address whose location is to be displayed. The user selects a street name from a file created from NavTech data and then enters the street number. If the location is found, it is marked on the map with a dark green asterisk. The structure chart for Find Address is depicted in Figure 198. A description of the routines called by Find Address is provided in Table 200.

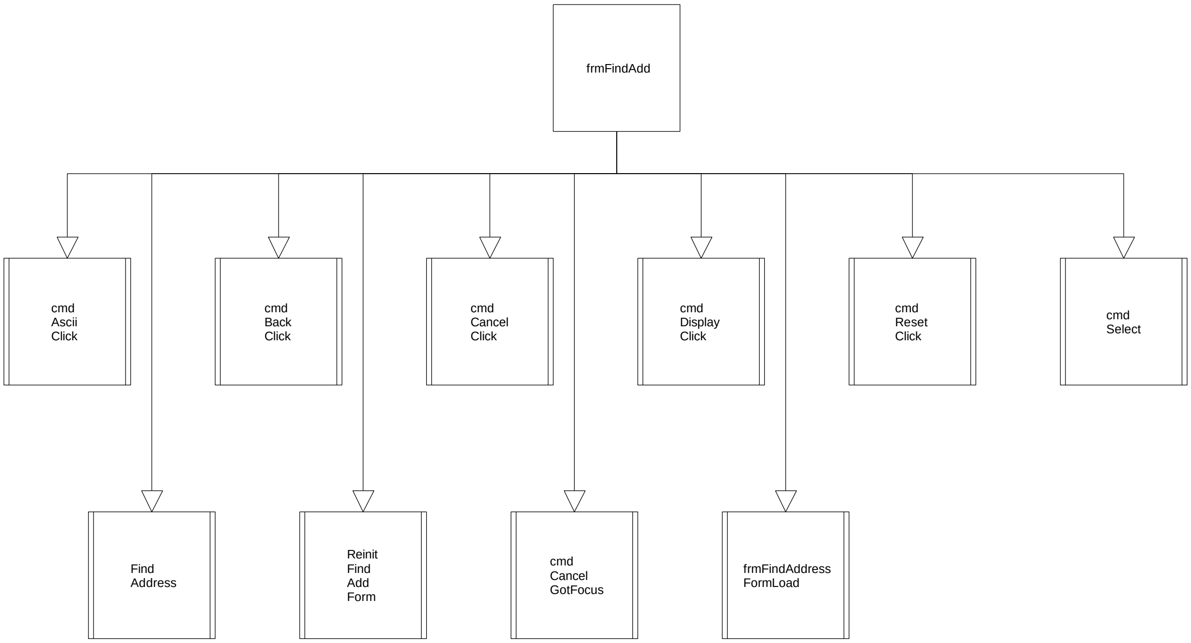

Figure 198 - Find Address Structure Chart

Function	Description
cmd ASCII Click	This subroutine determines which button was selected (letter, number, or space) from the FindAdd form and finds the street name that matches the current street name or number concatenated with the value of the selected button. The routine first plays the click sound and resets the last touch time. Next, determine which button was selected by the user. Index values from 0 to 25 correspond to the 26 letters of the alphabet from A to Z, index values from 26 to 35 correspond to numeric digits from 0 to 9, and index value 36 corresponds to the space character. Determine which part of the address (street name or street address) is being built. If the current street name is blank, then the street name field is being built. Otherwise, the street number is being built. If the street name field is being built and a character is being input, determine if a street name exists that begins with the character input. If not, display a warning message to the user that no street begins with that character and allow the user to try again. Otherwise, add the character input to the street name field and exit. If the street number field is being built, add the numeric input to the street number field and exit. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError.

Function	Description
cmd Back Click	This subroutine erases the last number entered in the street number field. Make click sound and reset Last_Touch_Timer. If the length of the street number field is greater than 0, remove the last number.
cmd Cancel Click	This subroutine cancels the Address Lookup function. Make click sound, reset Last Touch Time, display a message to the user to select any button to continue (or to select any button or icon to continue), and unload the PtsInterest form.
cmd Cancel GotFocus	This subroutine saves the address previously selected by the user and sets the reset string equal to the street name field.
cmd Display Click	Displays icons for the Point of Interest category selected by the user. Make click sound, reset Last_Touch_Timer, reinitialize the map to remove previously displayed POI icons, and blank the message to the user. Using the index of the line selected by the user, read into the POI array to determine the type of POI selected. Create a new layer based on this POI. If the user selected the category, "Transportation", build a point renderer to display different transportation types by different icons, display a message to the user to select any button or icon to continue, and unload the PtsInterest form. If the user selected any category other than "Transportation", build a geodataset of facilities included in the POI category selected by the user, get the icon font, color, and style from the POI Layer Info data structure, add the layer to the SA Map, display a message to the user to select any button or icon to continue, and unload the PtsInterest form. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError.
cmd Reset Click	This subroutine resets either the street name field or the street number field. Make click sound and reset Last_Touch_Timer. Determine if the user is resetting the street name or the street number field. If street name, then reset the street name list to the top and clear the street name field. If the user is clearing the street number field, clear the street number field. Reset the caption.
cmd Select	This subroutine accepts the street name from the user and prepares the form for street number entry. The Find Address form is divided into 2 parts; the top area is used to allow the user to enter a street name and the bottom area is used to allow the user to enter an address number. When the top area is active, the bottom area is disabled, and when the bottom area is active, the top area is disabled. This routine enables the top portion of the form and clears the bottom portion. Make click sound and reset Last_Touch_Timer. Save the street name; if the street name is blank, default to the first street name in the street name list. In order to prepare the form for number entry, change the instruction label, disable name-selection objects, disable the selection of letters and space, move the numbers to the bottom of the form, move the dividing line to above the numbers, move the reset button to the bottom of the form, respace the remaining buttons, enable the backspace button and display the button.
Find Address	This function converts and returns the MapObjects address location object based on the street address selected by the user. Create a MapObjects AddressMatcher object and correlate its matching fields ("LeftFromField", "StreetField", etc to those used in the shapefiles ("LREF_ADDRE", "ST_NAME", etc). Index the table (if not already indexed) to allow searching. Locate the address using MapObject's MatchAddress method. Return the results. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError.
frmFindAddress FormLoad	The Back and Display buttons are enabled.

Function	Description
Reinit Find Add Form	This subroutine resets the Find Address Form to allow the user to enter a street name. The Find Address form is divided into 2 parts; the top area is used to allow the user to enter a street name and the bottom area is used to allow the user to enter an address number. When the top area is active, the bottom area is disabled, and when the bottom area is active, the top area is disabled. This routine enables the top portion of the form and clears the bottom portion. Clear the street name text field and initialize the street name list to the top position. Enable the alphabetic characters and space button. Move the numbers area of the form up and move the line that divides the two portions of the form down so that the numbers are included in the top portion of the form. Move the reset button to the top portion of the form and re-display the remaining buttons to space them equally. Enable the backspace and display buttons. Reset the caption. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError.

Table 200 - Routines called by Find Address

4.2.2.2.4 Points of Interest List

The Points of Interest List GUI displays the list of Points of Interest from which the user may select and view on the map. The structure chart for Points of Interest List is depicted in Figure 199. A description of the routines called by Points of Interest List is provided in Table 201.

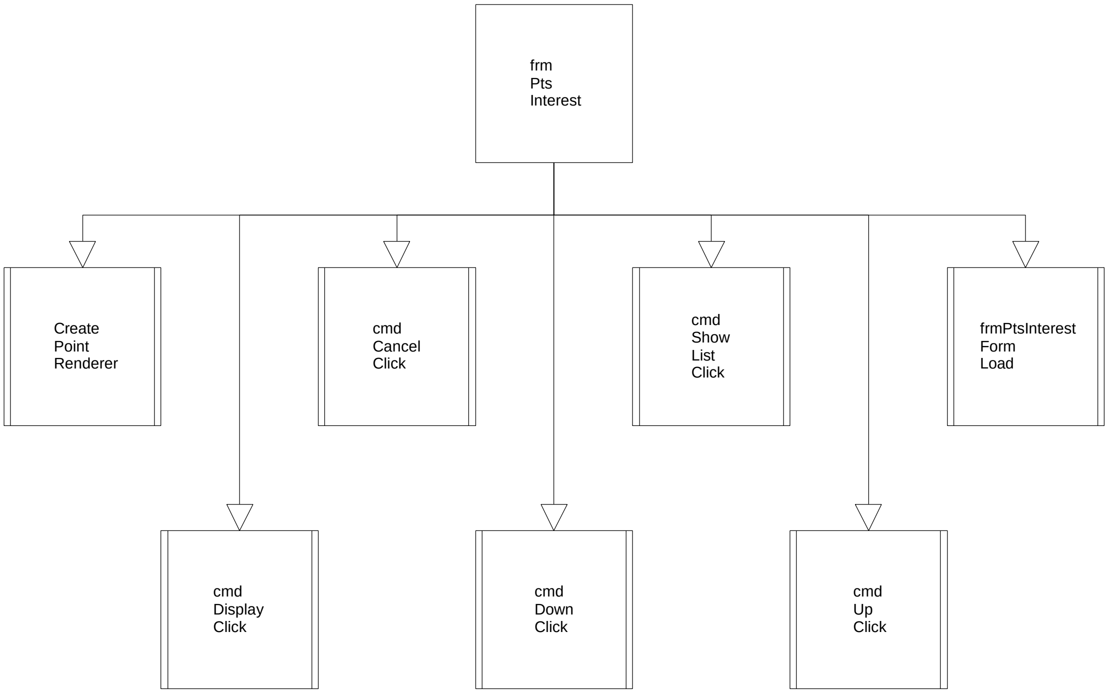

Figure 199 - Points of Interest List Structure Chart

Function	Description
cmd Cancel Click	This subroutine cancels the Address Lookup function. Make click sound, reset Last Touch Time, display a message to the user to select any button to continue (or to select any button or icon to continue), and unload the PtsInterest form.
cmd Display Click	Displays icons for the Point of Interest category selected by the user. Make click sound, reset Last_Touch_Timer, reinitialize the map to remove previously displayed POI icons, and blank the message to the user. Using the index of the line selected by the user, read into the POI array to determine the type of POI selected. Create a new layer based on this POI. If the user selected the category, "Transportation", build a point renderer to display different transportation types by different icons, display a message to the user to select any button or icon to continue, and unload the PtsInterest form. If the user selected any category other than "Transportation", build a geodataset of facilities included in the POI category selected by the user, get the icon font, color, and style from the POI Layer Info data structure, add the layer to the SA Map, display a message to the user to select any button or icon to continue, and unload the PtsInterest form. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError.
cmd Down Click	If not at the bottom of the Points of Interest list, advance to the next item on the list. Make click sound, reset LAST_TOUCH_TIME. If the item currently selected in the list is not the last item, make the next item active.
cmd Show List Click	Displays a list of Point of Interest facility names based on the category selected by the user. Make click sound, reset Last_Touch_Time, and reinitialize the map to remove any previously displayed POI icons.
cmd Up Click	If not at the top of the Points of Interest list, advance to the previous item on the list. Make click sound and reset LAST_TOUCH_TIME. If the item currently selected in the list is not the first item, make the previous item active.
Create Point Renderer	This subroutine displays different fonts for transportation based on type (airport, bus station, or train station). Select font for transportation icons. Create a strings collection and populate it with the facility codes for airport, bus station, and train station. For each value defined, assign font index, color, and size.
frmPtsInterest Form Load	Add the Points of Interest items to the list box and highlight the first item.

Table 201 - Routines called by Points of Interest List

4.2.2.2.5 Find Point of Interest

The Find Point of Interest GUI displays the Points of Interest list and provides the capability for the user to scroll through the list and select the desired Point of Interest. The structure chart for Find Point of Interest is depicted in Figure 200. A description of the routines called by Find Point of Interest is provided in Table 202.

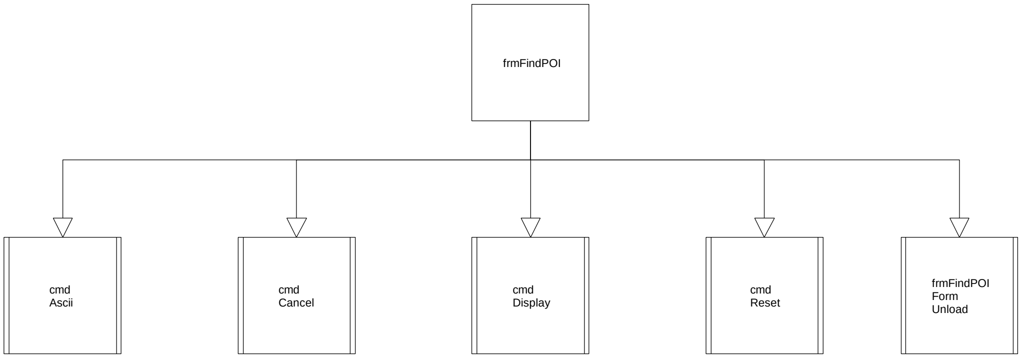

Figure 200 - Find Point of Interest Structure Chart

Function	Description
cmd Ascii	This subroutine determines which button was selected (letter, number, or space) and then finds the first point of interest that matches the selected button. If no point of interest begins with the selected button, a warning message is displayed. Make click sound and reset Last_Touch_Timer. Determine which button was selected by the user. Index values from 0 to 25 correspond to the 26 letters of the alphabet from A to Z, index values from 26 to 35 correspond to numeric digits from 0 to 9, and index value 36 corresponds to the space character. Build the POI name by concatenating the value of the index to the POI string.
cmd Cancel	This subroutine allows the user to cancel the Select POI function. Make click sound, reset Last_Touch_Timer, reinitialize the map, and delete any POI icons displayed.
cmd Display	This subroutine displays the facilities for one POI category or displays the single POI facility selected by the user. Make click sound, reset Last_Touch_Timer, and clear instruction label on map. Get facility name from POI list. Temporarily replace apostrophes in the name with the underscore symbol. This is done to ensure that an accurate query is performed. This will be reset later. Search the currently selected POI shapefile for the facility name and put the results in a recordset. (The current POI file is determined by looking up the POI index in the POI category list). Replace underscores in the name with apostrophes to restore the facility names. Set the recordset to the first record. If more than one name was found (multiple facilities for same name, e.g., McDonalds restaurants), display each record as an icon on the tracking layer, but also create a map layer for later searching. If only one record was found, display the icon, build the Identify form, and display it. NOTE: The characteristics for the icons to be displayed are found by looking up the POI form index in the POI_Categories file. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError.
cmd Reset	This subroutine resets the POI string. Make click sound, reset Last_Touch_Timer, reset the POI form to the top of the list, and clear the POI string.
frmFindPOI Form Unload	Reset the default values for the GUI.

Table 202 - Routines called by Find Point of Interest

4.2.2.2.6 Identify GUI

The Identify GUI provides a function that takes the x,y coordinates from the MouseDown event, finds the features that are at or near that point, and populates the Identify form. The input parameter location is converted to a MapObject point and the POI layer is checked to see if it is active. If the layer is invisible, then the actual layer being displayed to the user is the tracking layer; this is used when POI's with the same name (e.g., McDonald's) are being displayed. If this is the case, then compare the distance between each point in the active layer with the point that was touched by the user, saving the closest point with each comparison. After finding the closest point, determine if the point is close enough to have been touched. If so, flash the point and display the Identify form with the POI name, address, and phone number, as available. If the layer is displayed on the map as a map layer, create a dataset of records in the POI shapefile that is displayed, and search for a POI within a specified search tolerance. If one is found, flash the point and display the Identify form with the POI name, address, and phone number, as available. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError. The structure chart for Identify GUI is depicted in Figure 201. A description of the routines called by Identify GUI is provided in Table 203.

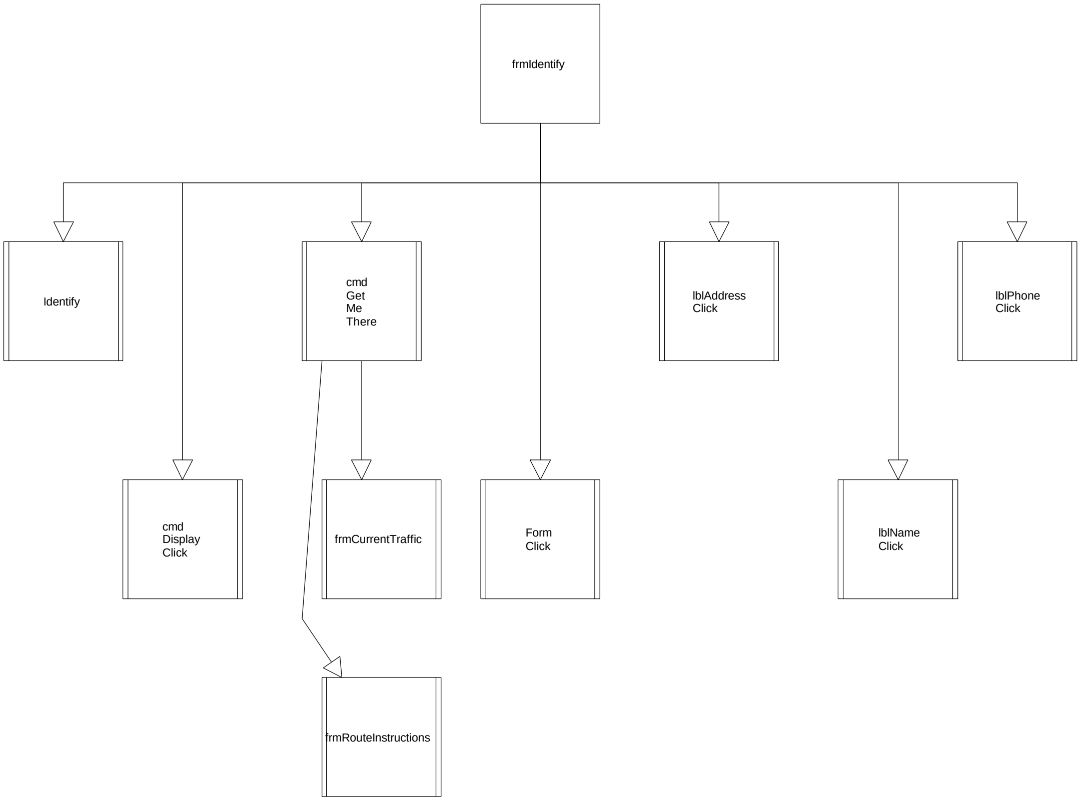

Figure 201 - Identify GUI Structure Chart

Function	Description
cmd Display Click	Displays icons for the Point of Interest category selected by the user. Make click sound, reset Last_Touch_Timer, reinitialize the map to remove previously displayed POI icons, and blank the message to the user. Using the index of the line selected by the user, read into the POI array to determine the type of POI selected. Create a new layer based on this POI. If the user selected the category, "Transportation", build a point renderer to display different transportation types by different icons, display a message to the user to select any button or icon to continue, and unload the PtsInterest form. If the user selected any category other than "Transportation", build a geodataset of facilities included in the POI category selected by the user, get the icon font, color, and style from the POI Layer Info data structure, add the layer to the SA Map, display a message to the user to select any button or icon to continue, and unload the PtsInterest form. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError.
cmd Get Me There	This routine sets the origin and destination points, invokes the PathFinder routine, and prepares turn-by-turn instructions for the specified route. Make Click Sound and reset LAST_TOUCH_TIME. If real-time data is available, determine if the user wants real-time traffic conditions to be considered when building the route. Inform the user that this will take some extra time. If the user wants to use real-time data, update the speed table (MakeSpeedTable) and update the impedances (ROUTE_GUIDE updateImpedances). Define the source and destination points, define the parameters required by ROUTE_GUIDE, and find the path (ROUTE_GUIDE FindPath). Refresh the tracking layer to display the route and display the turn-by-turn instructions. If an error occurs, build an error message using the error number and error message (if known) and send it to LogError.
Form Click	Make Click Sound, reset LastTouchTime, and unload frmIdentify.
frmCurrentTraffic	This GUI is displayed after the user has selected the Get Me There button. The GUI determines whether or not the user wants to include the current traffic conditions in the route guidance calculations.
frmIdentify	This form displays the name and address of either the Point of Interest or address selected by the user.
frmRouteInstructions	This GUI lists the turn by turn instructions, after the user requested route has been calculated. In addition, the route is drawn on the SA Map.
lblAddress Click	Make Click Sound, reset LastTouchTime, and unload frmIdentify.
lblName Click	Make Click Sound, reset LastTouchTime, and unload frmIdentify.
lblPhone Click	Make Click Sound, reset LastTouchTime, and unload frmIdentify.

Table 203 - Routines called by Identify GUI

4.2.2.2.7 Current Traffic GUI

The Current Traffic GUI is displayed after the user has selected the Get Me There button. The GUI determines whether or not the user wants to include the current traffic conditions in the route guidance calculations. The structure chart for Current Traffic GUI is depicted in Figure 202. A description of the routines called by Current Traffic GUI is provided in Table 204.

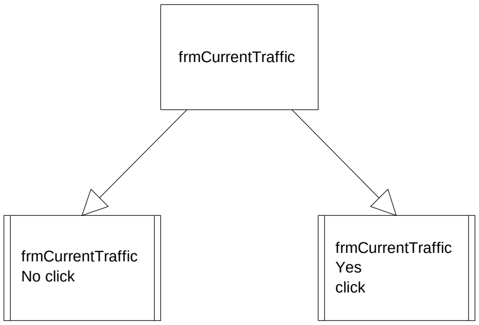

Figure 202 - Current Traffic GUI Structure Chart

Function	Description
frmCurrentTraffic No click	Set the response flag to False and unloads the Current Traffic GUI.
frmCurrentTraffic Yes click	Set the response flag to True and unloads the Current Traffic GUI.

Table 204 - Routines called by Current Traffic GUI

After the user has selected whether or not to use real time traffic conditions, the route is calculated using DLL calls developed by Environmental System Research Institute (ESRI). These calls calculate the route, return a layer that when rendered displays the route on the San Antonio Street Map, and return a list of turn by turn instructions that are displayed in a list box next to the San Antonio Street Map.

4.2.2.2.8 Airport Data Screen

When selected by the user, the “Airport” button on the Main Menu Screen allows the display of a predefined zoom area displaying the traffic in the area of the San Antonio International Airport centering on the airport. The user can select to view Airline Carrier information, Rental Car Agency information, or Airport Parking information. The structure chart for the Airport Data GUI is depicted in Figure 203. A description of the routines called by the Airport Data GUI is provided in Table 205.

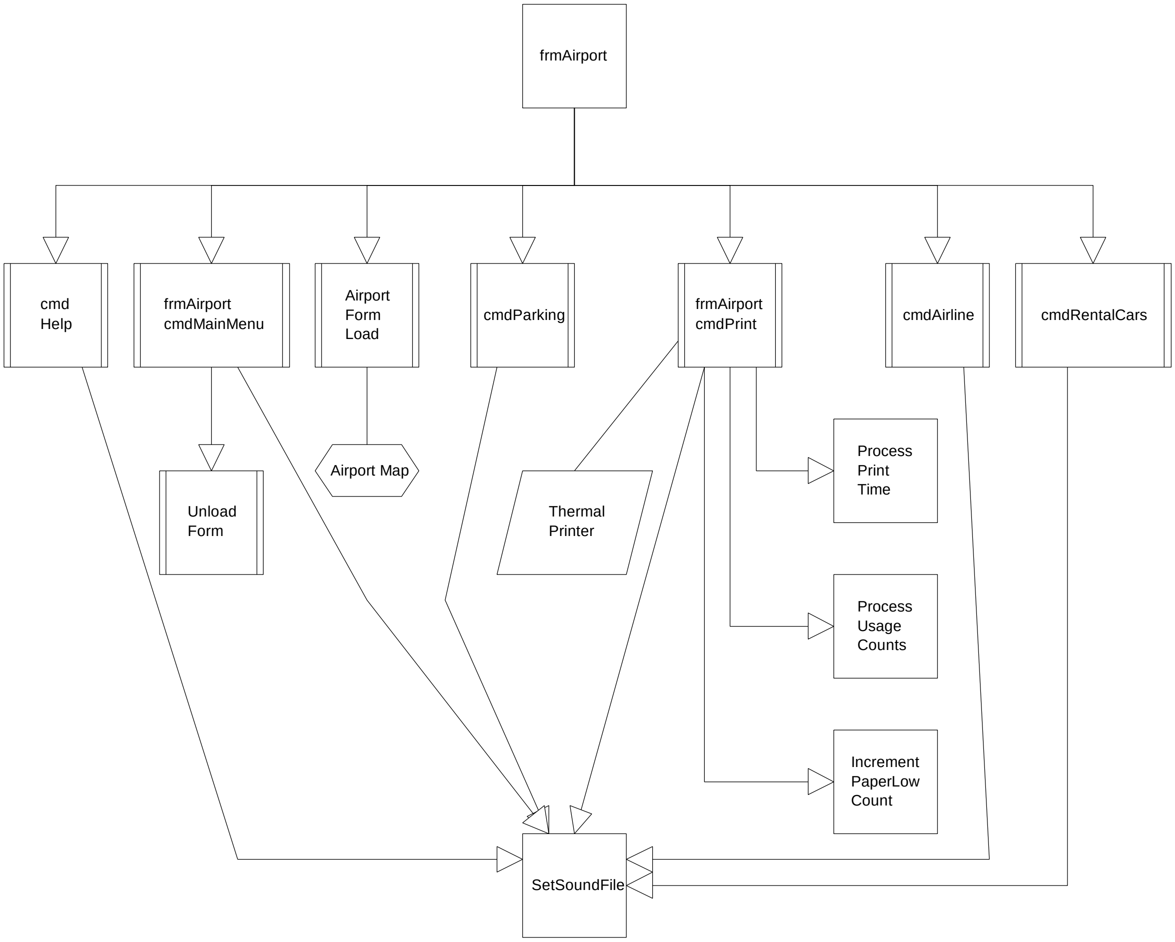

Figure 203 - Airport Data Screen Structure Chart

Function	Description
Airport Form Load	Loads the airport map from the airport shapefile.
cmd Help	The button displays help for the specified GUI from which it is selected. The appropriate bitmap file is retrieved and displayed.
cmdAirline	This event is triggered when the user touches the Airlines button; causing the Airline information to be displayed.
cmdParking	This event is triggered when the user touches the Parking button; causing the Parking information to be displayed.
cmdRentalCars	This event is triggered when the user touches the Rental Cars button; causing the Rental Car information to be displayed.
frmAirport cmdMainMenu	When selected, the click soundfile is play, the last touch time is updated, the SA Online Main Menu is displayed and the Airport GUI is hidden.
frmAirport cmdPrint	After sound file is played, set LAST_TOUCH_TIME to current time. If ENABLE_PRINT is true, the airport map is printed. Otherwise nothing is printed. ProcessPrintTime by checking if the print button has been selected more than 5 times in the last 30 seconds. IncrementPaperLowCount and store in status file. ProcessUsageCount by incrementing PRINT_COUNT variable in status file. Set printer properties such as orientation and length by calling the printer dll (pprtr.dll) and print the map.
Increment PaperLow Count	Increments the paper low counter, contained in the paper low file, every time a print is requested by the user.
Process Print Time	Stores the last print time into the Print Time array and then determines if the print button has exceeded its threshold (e.g., more that 5 prints in 30 seconds).
Process Usage Counts	Saves the current usage counts for Main Menu, SA Map, Airport, Weather, VIA and Route Guidance to the usage file.
SetSoundFile	Passes the sound file, that is called when any click event occurs, as a parameter to the call to the Windows API PlaySound
Unload Form	Visual Basic Event that removes the specified form from the display and closes it. The form and its fields are no longer accessible until another load or show form is executed.

Table 205 - Routines called by Airport Data Screen

4.2.2.2.9 Airline Carriers

The Airline Carriers GUI displays the airline carrier information contained in the airlines.txt file. The structure chart for Airline Carriers is depicted in Figure 204. A description of the routines called by Airline Carriers is provided in Table 206.

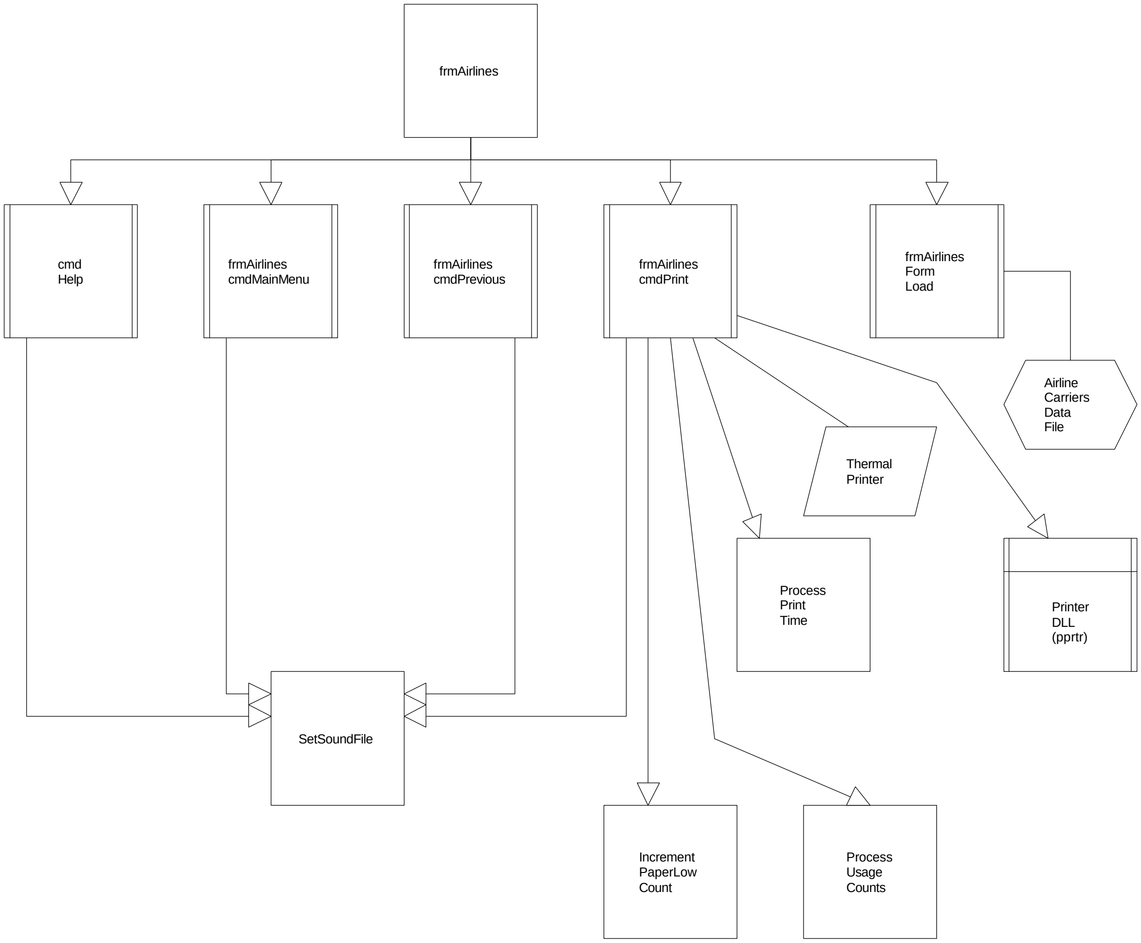

Figure 204 - Airline Carriers Structure Chart

Function	Description
cmd Help	The button displays help for the specified GUI from which it is selected. The appropriate bitmap file is retrieved and displayed.
frmAirlines cmdMainMenu	After sound file is played, set LAST_TOUCH_TIME to current time, show frmMainMenu, set lblErrorMsg visibility property to false and unload frmAirlines GUI.
frmAirlines cmdPrevious	After sound file is played, set LAST_TOUCH_TIME to current time, show frmAirport GUI, set lblErrorMsg visibility property to false, and unload frmAirlines GUI.
frmAirlines cmdPrint	After sound file is played, set LAST_TOUCH_TIME to current time. If ENABLE_PRINT is true and grdAirlines contains data continue with printing procedures, otherwise discontinue printing. ProcessPrintTime by checking if the print button has been selected more than 5 times in the last 30 seconds. IncrementPaperLowCount and store in status file. ProcessUsageCount by incrementing PRINT_COUNT variable in status file. Set printer properties such as orientation and length by calling the printer dll (pprtr.dll) and send airlines grid data to the thermal printer for printing.
frmAirlines Form Load	Set up grdAirlines parameters. Get and load airline agency data into grdAirlines. If unable to load airline agency data (open error or file missing) then set lblErrorMsg visibility property to true, close the data file, and exit subroutine.
Increment PaperLow Count	Increments the paper low counter, contained in the paper low file, every time a print is requested by the user.
Printer DLL (pprtr)	Changes thermal printer properties, such as orientation and paper length, through a dll call.
Process Print Time	Stores the last print time into the Print Time array and then determines if the print button has exceeded its threshold (e.g., more that 5 prints in 30 seconds).
Process Usage Counts	Saves the current usage counts for Main Menu, SA Map, Airport, Weather, VIA and Route Guidance to the usage file.
SetSoundFile	Passes the sound file, that is called when any click event occurs, as a parameter to the call to the Windows API PlaySound.

Table 206 - Routines called by Airline Carriers

4.2.2.2.10 Rental Car Agency GUI

The Rental Car Agency GUI displays the Rental Car information contained in the rental.txt file. The structure chart for Rental Car Agency GUI is depicted in Figure 205. A description of the routines called by Rental Car Agency GUI is provided in Table 207.

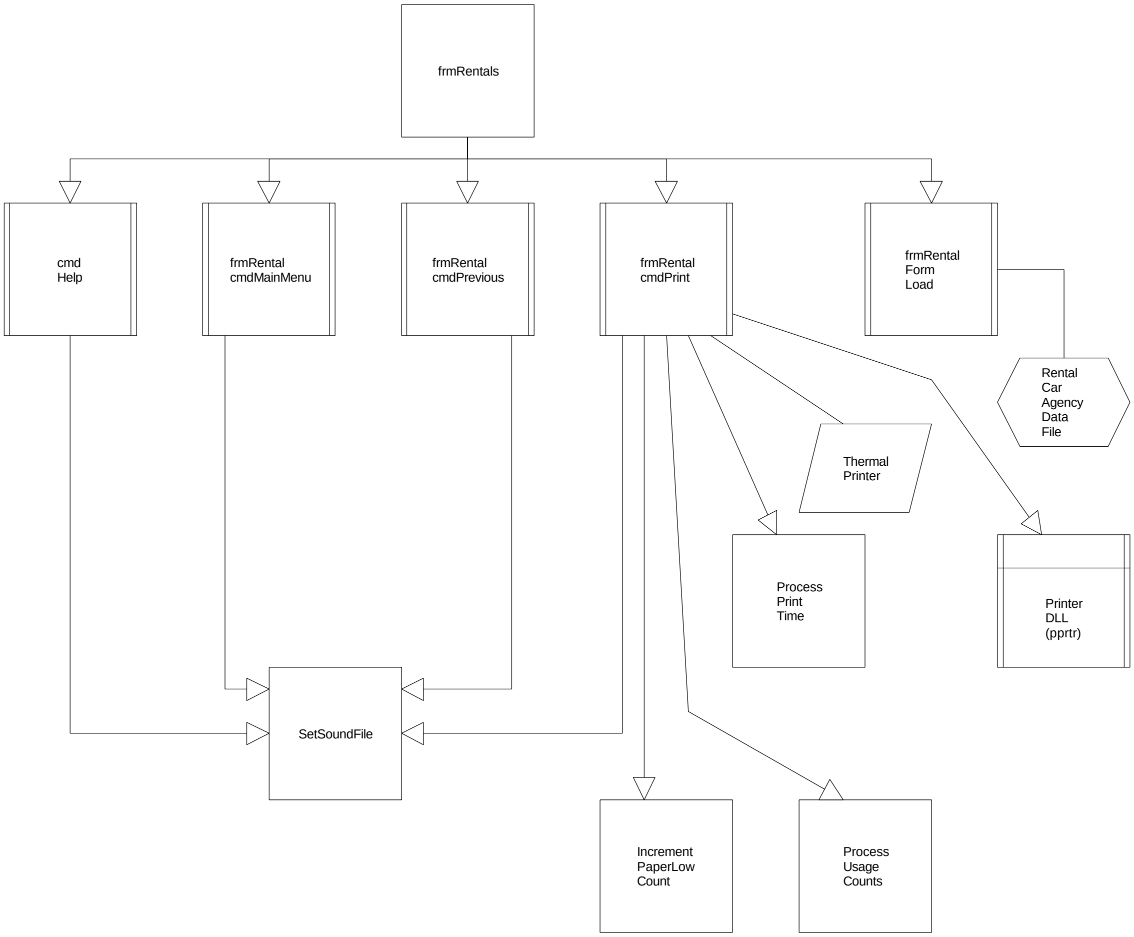

Figure 205 - Rental Car Agency GUI Structure Chart

Function	Description
cmd Help	The button displays help for the specified GUI from which it is selected. The appropriate bitmap file is retrieved and displayed.
frmRental cmdMainMenu	After sound file is played, set LAST_TOUCH_TIME to current time, show frmMainMenu, set lblErrorMsg visibility property to false, and hide frmRentals GUI.
frmRental cmdPrevious	After sound file is played, set LAST_TOUCH_TIME to current time, show frmAirport GUI, set lblErrorMsg visibility property to false, and hide frmRentals GUI.
frmRental cmdPrint	After sound file is played, set LAST_TOUCH_TIME to current time. If ENABLE_PRINT is true and grdRentals contains data, continue with printing procedures, otherwise discontinue printing. ProcessPrintTime by checking if the print button has been selected more than 5 times in the last 30 seconds. IncrementPaperLowCount and store in status file. ProcessUsageCount by incrementing PRINT_COUNT variable in status file. Set printer properties such as orientation and length by calling the printer dll (pprtr.dll). Send rental car agency grid data to the thermal printer for printing.
frmRental Form Load	Set up grdRentalCars parameters. Get and load rental car agency data into grdRentalCars. If unable to load rental car agency data (open error or file missing) then set lblErrorMsg visibility property to true, close the data file and exit the subroutine.
Increment PaperLow Count	Increments the paper low counter, contained in the paper low file, every time a print is requested by the user.
Printer DLL (pprtr)	Changes thermal printer properties, such as orientation and paper length, through a dll call.
Process Print Time	Stores the last print time into the Print Time array and then determines if the print button has exceeded its threshold (e.g., more that 5 prints in 30 seconds).
Process Usage Counts	Saves the current usage counts for Main Menu, SA Map, Airport, Weather, VIA and Route Guidance to the usage file.
SetSoundFile	Passes the sound file, that is called when any click event occurs, as a parameter to the call to the Windows API PlaySound.

Table 207 - Routines called by Rental Car Agency GUI

4.2.2.2.11 Airport Parking Fee GUI

The Airport Parking Fee GUI displays the parking data contained in the parking.txt file. The structure chart for Airport Parking Fee GUI is depicted in Figure 206. A description of the routines called by Airport Parking Fee GUI is provided in Table 208.

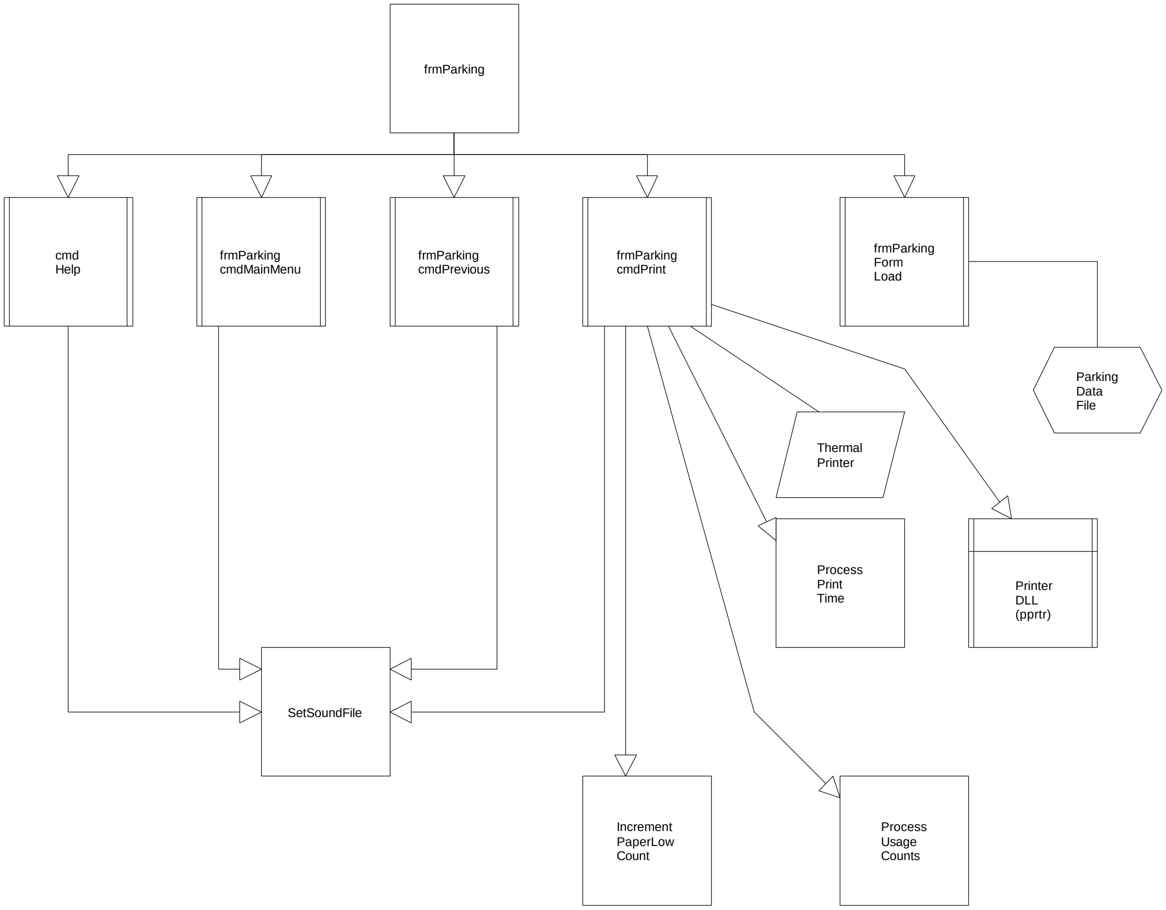

Figure 206 - Airport Parking Fee GUI Structure Chart

Function	Description
cmd Help	The button displays help for the specified GUI from which it is selected. The appropriate bitmap file is retrieved and displayed.
frmParking cmdMainMenu	After sound file is played, set LAST_TOUCH_TIME to current time, show frmMainMenu, set lblErrorMsg visibility property to false, and hide frmParking GUI.
frmParking cmdPrevious	After sound file is played, set LAST_TOUCH_TIME to current time, show frmAirport GUI, set lblErrorMsg visibility property to false, and hide frmParking GUI.
frmParking cmdPrint	After sound file is played, set LAST_TOUCH_TIME to current time. If ENABLE_PRINT is true and grdParking contains data, continue with printing procedures, otherwise discontinue printing. ProcessPrintTime by checking if the print button has been selected more than 5 times in the last 30 seconds. IncrementPaperLowCount and store in status file. ProcessUsageCount by incrementing PRINT_COUNT variable in status file. Set printer properties such as orientation and length by calling the printer dll (pprtr.dll) and send parking fees grid data to the thermal printer for printing.
frmParking Form Load	If any errors occur, send error message to error logging routine. Set up grdParking parameters. Get and load parking data into grdParking. If unable to load parking data (open error or file missing) then set lblErrorMsg visibility property to true and close data file. Exit subroutine.
Increment PaperLow Count	Increments the paper low counter, contained in the paper low file, every time a print is requested by the user.
Printer DLL (pprtr)	Changes thermal printer properties, such as orientation and paper length, through a dll call.
Process Print Time	Stores the last print time into the Print Time array and then determines if the print button has exceeded its threshold (e.g., more that 5 prints in 30 seconds).
Process Usage Counts	Saves the current usage counts for Main Menu, SA Map, Airport, Weather, VIA and Route Guidance to the usage file.
SetSoundFile	Passes the sound file, that is called when any click event occurs, as a parameter to the call to the Windows API PlaySound.

Table 208 - Routines called by Airport Parking Fee GUI

4.2.2.2.12 San Antonio Weather Screen

The San Antonio Weather GUI displays the latest radar map received from the Master Computer and provides the capability to access the Five Day Forecast and Current Conditions GUIs. The structure chart for San Antonio Weather Screen is depicted in Figure 207. A description of the routines called by San Antonio Weather Screen is provided in Table 209.

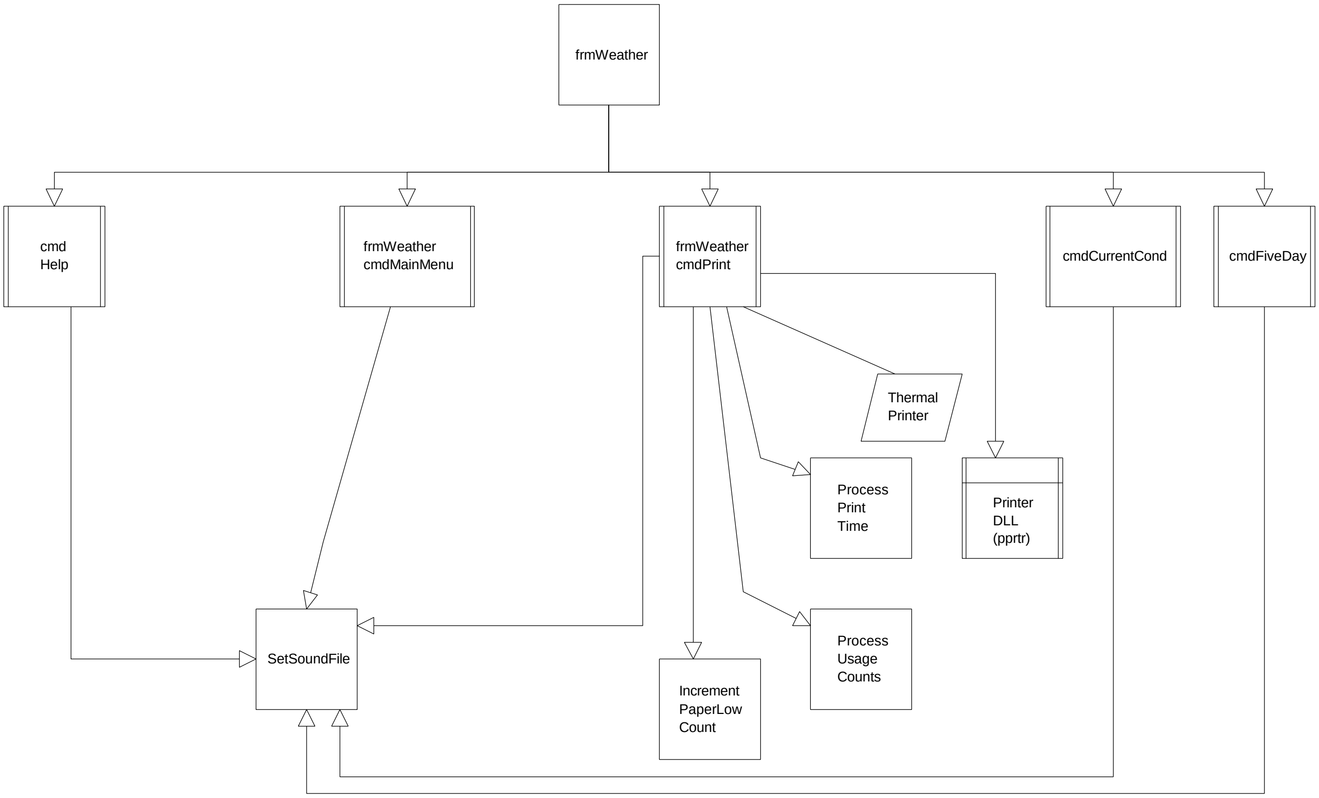

Figure 207 - San Antonio Weather Screen Structure Chart

Function	Description
cmd Help	The button displays help for the specified GUI from which it is selected. The appropriate bitmap file is retrieved and displayed.
cmdCurrentCond	After sound file is played, set LAST_TOUCH_TIME to current time, show frmCurrentCond , and hide frmWeather.
cmdFiveDay	After sound file is played, set LAST_TOUCH_TIME to current time, show frmFiveDay, and hide frmWeather.
FrmWeather cmdMainMenu	After sound file is played, set LAST_TOUCH_TIME to current time, show frmMainMenu, and hide frmWeather GUI.
frmWeather cmdPrint	After sound file is played, set LAST_TOUCH_TIME to current time. If ENABLE_PRINT is true and Image1 contains a graphic, continue with printing procedures, otherwise discontinue printing. ProcessPrintTime by checking if the print button has been selected more than 5 times in the last 30 seconds. IncrementPaperLowCount and store in status file. ProcessUsageCount by incrementing PRINT_COUNT variable in status file. Set printer properties such as orientation and length by calling the printer dll (pprtr.dll) and send radar map graphic to the thermal printer for printing.
Increment PaperLow Count	Increments the paper low counter, contained in the paper low file, every time a print is requested by the user.
Printer DLL (pprtr)	Changes thermal printer properties, such as orientation and paper length, through a dll call.
Process Print Time	Stores the last print time into the Print Time array and then determines if the print button has exceeded its threshold (e.g., more that 5 prints in 30 seconds).
Process Usage Counts	Saves the current usage counts for Main Menu, SA Map, Airport, Weather, VIA and Route Guidance to the usage file.
SetSoundFile	Passes the sound file, that is called when any click event occurs, as a parameter to the call to the Windows API PlaySound.

Table 209 - Routines called by San Antonio Weather Screen

4.2.2.2.13 Current Conditions GUI

The Current Conditions GUI displays the latest current conditions bitmap received from the Master Computer. The structure chart for Current Conditions GUI is depicted in Figure 208. A description of the routines called by Current Conditions GUI is provided in Table 210.

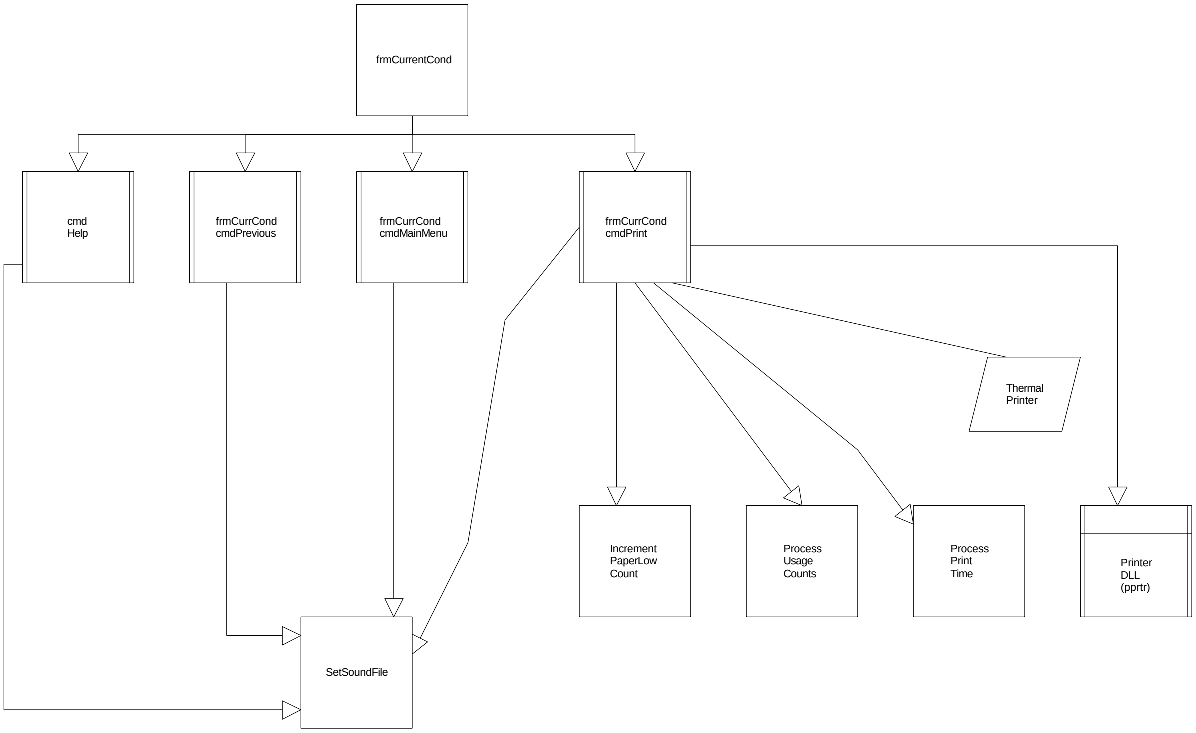

Figure 208 - Current Conditions GUI Structure Chart

Function	Description
cmd Help	The button displays help for the specified GUI from which it is selected. The appropriate bitmap file is retrieved and displayed.
frmCurrCond cmdMainMenu	After sound file is played, set LAST_TOUCH_TIME to current time, show frmMainMenu GUI, and hide frmCurrentCond GUI.
frmCurrCond cmdPrevious	After sound file is played, set LAST_TOUCH_TIME to current time, show frmWeather GUI, and hide frmCurrentCond GUI.
frmCurrCond cmdPrint	After sound file is played, set LAST_TOUCH_TIME to current time. If ENABLE_PRINT is true and Image1 contains a graphic, continue with printing procedures, otherwise discontinue printing. ProcessPrintTime by checking if the print button has been selected more than 5 times in the last 30 seconds. IncrementPaperLowCount and store in status file. ProcessUsageCount by incrementing PRINT_COUNT variable in status file. Set printer properties such as orientation and length by calling the printer dll (pprtr.dll) and send current conditions graphic to the thermal printer for printing.
Increment PaperLow Count	Increments the paper low counter, contained in the paper low file, every time a print is requested by the user.
Printer DLL (pprtr)	Changes thermal printer properties, such as orientation and paper length, through a dll call.
Process Print Time	Stores the last print time into the Print Time array and then determines if the print button has exceeded its threshold (e.g., more that 5 prints in 30 seconds).

Function	Description
Process Usage Counts	Saves the current usage counts for Main Menu, SA Map, Airport, Weather, VIA and Route Guidance to the usage file.
SetSoundFile	Passes the sound file, that is called when any click event occurs, as a parameter to the call to the Windows API PlaySound.

Table 210 - Routines called by Current Conditions GUI

4.2.2.2.14 Five-Day Forecast GUI

The Five-Day Forecast GUI displays the last five-day forecast bitmap received from the Master Computer. The structure chart for Five-Day Forecast GUI is depicted in Figure 209. A description of the routines called by Five-Day Forecast GUI is provided in Table 211.

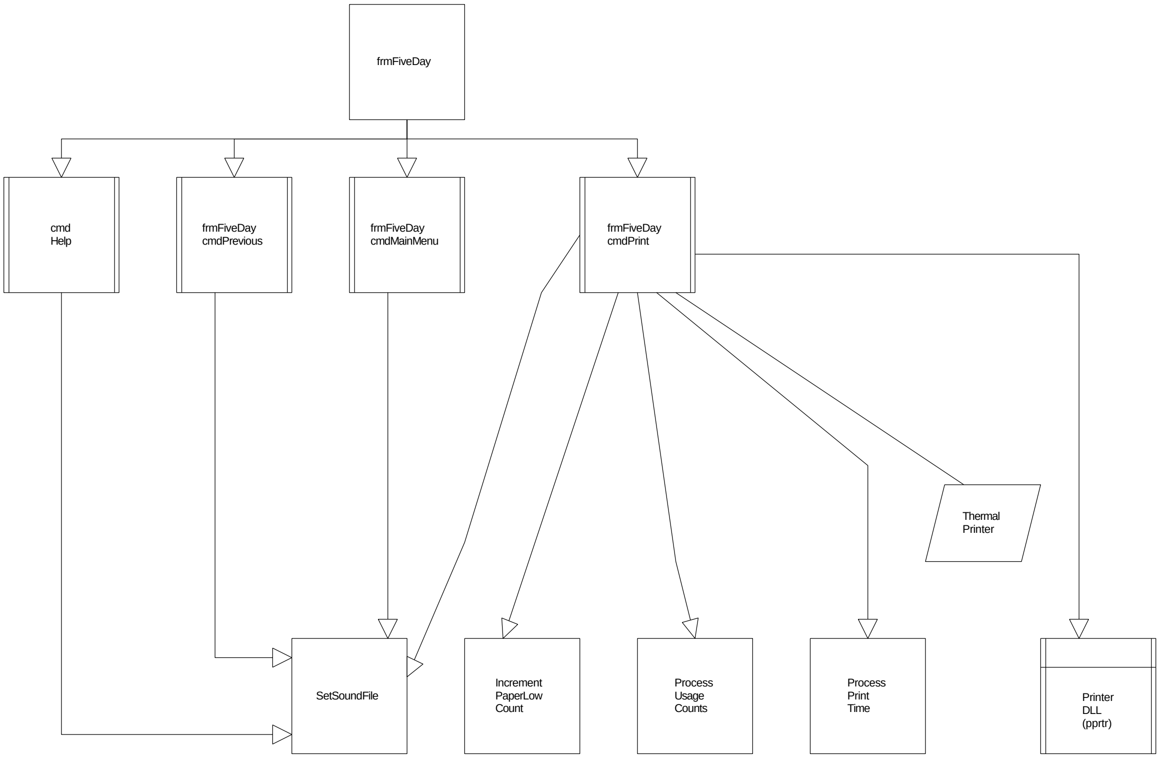

Figure 209 - Five Day Forecast GUI Structure Chart

Function	Description
cmd Help	The button displays help for the specified GUI from which it is selected. The appropriate bitmap file is retrieved and displayed.
frmFiveDay cmdMainMenu	After sound file is played, set LAST_TOUCH_TIME to current time, show frmMainMenu GUI, and hide frmFiveDay GUI.
frmFiveDay cmdPrevious	After sound file is played, set LAST_TOUCH_TIME to current time, show frmWeather GUI, and hide frmFiveDay GUI.
frmFiveDay cmdPrint	After sound file is played, set LAST_TOUCH_TIME to current time. If ENABLE_PRINT is true and Image1 contains a graphic, continue with printing procedures, otherwise discontinue printing. ProcessPrintTime by checking if the print button has been selected more than 5 times in the last 30 seconds. IncrementPaperLowCount and store in status file. ProcessUsageCount by incrementing PRINT_COUNT variable in status file. Set printer properties such as orientation and length by calling the printer dll (pprtr.dll) and send five day graphic to the thermal printer for printing.
Increment PaperLow Count	Increments the paper low counter, contained in the paper low file, every time a print is requested by the user.
Printer DLL (pprtr)	Changes thermal printer properties, such as orientation and paper length, through a dll call.
Process Print Time	Stores the last print time into the Print Time array and then determines if the print button has exceeded its threshold (e.g., more that 5 prints in 30 seconds).
Process Usage Counts	Saves the current usage counts for Main Menu, SA Map, Airport, Weather, VIA and Route Guidance to the usage file.
SetSoundFile	Passes the sound file, that is called when any click event occurs, as a parameter to the call to the Windows API PlaySound.

Table 211 - Routines called by Five Day Forecast GUI

4.2.2.2.15 VIA Transit Screen

The Via Transit GUI provides the ability to access the VIA Transit data. Buttons are provided to access route schedules, special event information, general information, fare and pass information, and VIATrans information. The structure chart for VIA Transit Screen is depicted in Figure 210. A description of the routines called by VIA Transit Screen is provided in Table 212.

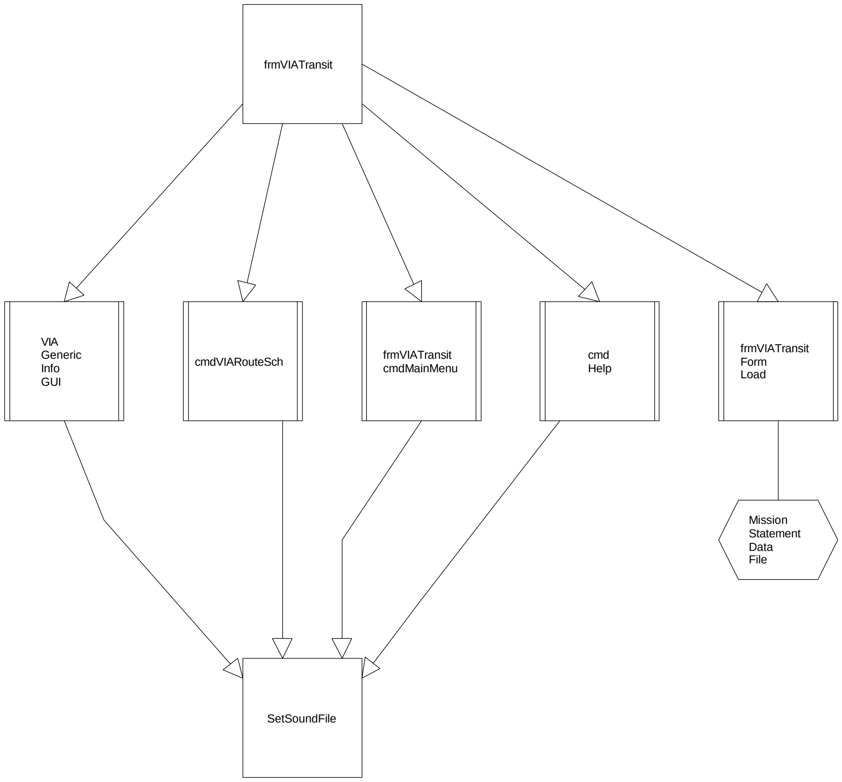

Figure 210 - VIA Transit Screen Structure Chart

Function	Description
cmd Help	The button displays help for the specified GUI from which it is selected. The appropriate bitmap file is retrieved and displayed.
cmdVIARouteSch	After sound file is played, set LAST_TOUCH_TIME to current time, show frmVIARouteSch GUI, and hide frmVIATransit GUI.
frmVIATransit cmdMainMenu	After sound file is played, set LAST_TOUCH_TIME to current time, show frmMainMenu GUI, and hide frmVIATransit GUI.
frmVIATransit Form Load	Get and display the VIA mission statement. If unable to load mission statement (open error or file missing), then send error message, close the data file, and exit the subroutine.
SetSoundFile	Passes the sound file, that is called when any click event occurs, as a parameter to the call to the Windows API PlaySound
VIA Generic Info GUI	This GUI is used to display the VIA information gif files (Fares and Passes, General Information, Special Events, and VIATrans) to the user. The same GUI is used to display fare and pass information, general information, special event information, and VIATrans information. After sound file is played, the LAST_TOUCH_TIME is set to the current time, the VIA Generic Info GUI is displayed with the selected VIA information, and the frmVIATransit GUI is hidden.

Table 212 - Routines called by VIA Transit Screen

4.2.2.2.16 VIA Bus Stop GUI

The VIA Bus Stop GUI displays a map containing the bus stops surrounding the location of the kiosk and provides the capability to view routes associated with those bus stops. The structure chart for VIA Bus Stop GUI is depicted in Figure 211. A description of the routines called by VIA Bus Stop GUI is provided in Table 213.

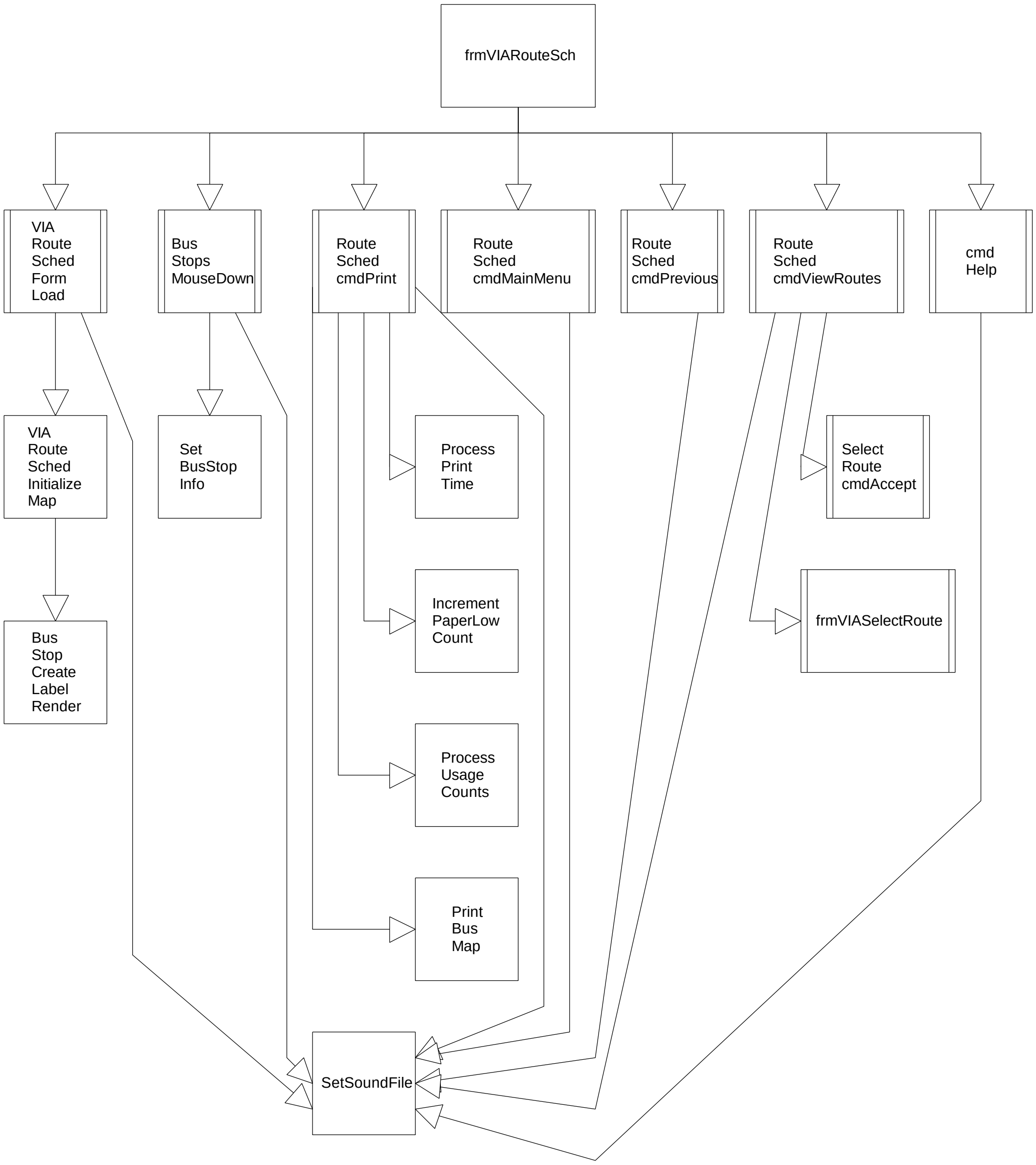

Figure 211 - VIA Bus Stop GUI Structure Chart

Function	**Description**
Bus Stop Create Label Render	This routine creates the labels for the bus stop map. The MoPlus Label Placer is used to label the street names and the font is set to Arial. Finally, the renderer is then setup to display the most detailed street layout.
Bus Stops MouseDown	The event is triggered when the user touches the map and determines which bus stop the user is selecting. The coordinates of the location the user touched are retrieved and the search tolerances are set. Using the SearchByDistance Map Objects method, determine if there is a bus stop with the search area. If a bus stop is found, get the bus stop number and flash the icon to indicate that it was selected.
cmd Help	The button displays help for the specified GUI from which it is selected. The appropriate bitmap file is retrieved and displayed.
FrmVIASelectRoute	Provides a list of bus routes associated with the selected bus stop from which the user can select one.
Increment PaperLow Count	Increments the paper low counter, contained in the paper low file, every time a print is requested by the user.
Print Bus Map	This routine prints the bus stop map that is displayed on the Via Route Schedule GUI. Using the PPrtr DLL, the printer sets and prints the name of the map, the bus stop map, and the Navigation Technologies logo.
Process Print Time	Stores the last print time into the Print Time array and then determines if the print button has exceeded its threshold (e.g., more that 5 prints in 30 seconds).
Process Usage Counts	Saves the current usage counts for Main Menu, SA Map, Airport, Weather, VIA and Route Guidance to the usage file.
Route Sched cmdMainMenu	This routine invokes the touch sound, updates the last touch time, Shows the SA Online Main Menu, and Hides the Via Route Schedule form.
Route Sched cmdPrevious	This routine plays the touch sound file, updates the last touch time, shows the VIATransit form and hides the VIA Route Schedule form.
Route Sched cmdPrint	This routine prints the bus stop map. The touch sound is invoked and the last touch time is updated. If printing is enabled, the last print time is updated, the paper low count is incremented, the print usage count is updated and the bus stop map is printed.
Route Sched cmdViewRoutes	This routine displays either the selected route of the Select Route form. If there is only one route associated with the bus stop, the schedule for that route is retrieved, the Display Route form is displayed, and the VIA Route Schedule form is hidden. If there is more than one route associated with the bus stop, build the list of routes that can be selected and display the routes on the Select Route form.

Function	Description
Select Route cmdAccept	This routine finds and displays the bus route schedule information. The last touch time is updated and a message that the route schedule is being processed is displayed. Next, the bus schedule grids are cleared. Determine the day of the year and search through the service file until the day is found. Determine if the route follows a typical service schedule or an alternative service schedule. If the route number is in the header record, then the route follows an alternate service schedule. Otherwise, the route follows the typical service schedule. Next, set the Select Route label to the route number. Open the route file and examine the service type. If the service type is zero, then the route does not provide services for this day. Otherwise, begin to load the route schedule into two grids. Use the bus stop numbers to retrieve the associated text name from the Cross Reference array. Use the numbers and names to build a legend for printing and load the text names as headings in the grids. Load the stop times into the grids. Determine if these are bus route attributes and if one is found, display the attribute. Finally, determine the directions for each of the grids and display them. After the grids are completed, the associated bitmap file is loaded. The SelectRoute list is cleared and the form is hidden. The DisplayRoute form is loaded and the VIA Route Schedule form is hidden.
Set BusStop Info	This routine finds the bus stop text name and associated bus route(s) for the bus stop that was selected by the user. The name and route(s) are displayed on the Bus Stop Data label. The bus stop number is used to search the Cross Reference file for a match. When found, build a string containing the stop name and associated bus routes and assign it to the Bus Stop Data label. If no match is found, display an error message on the Bus Stop Data label.
SetSoundFile	Passes the sound file, that is called when any click event occurs, as a parameter to the call to the Windows API PlaySound.
VIA Route Sched Form Load	This loads the Via Route Schedule Form, including a map of the area surrounding the Kiosk.
VIA Route Sched Initialize Map	This function sets up a map of the area surrounding the kiosk. The location of the kiosk and nearby bus stops are displayed. The map layer that is going to display the area around the kiosk is prepared and the label rendered is defined. Next, the map layer containing the bus stop icons is prepared and the layer containing the kiosk location icon is prepared. The size of the search area for the bus stops is defined. If there are not at least five bus stops in the search area, increase the area until there are five or more bus stops. Center the kiosk location on the map display and size the map. Finally, render the map containing the kiosk location and associated bus stops.

Table 213 - Routines called by VIA Bus Stop GUI

4.2.2.2.17 VIA Select Route GUI

The VIA Select Route GUI provides a list of bus routes associated with the selected bus stop from which the user can select one. The structure chart for VIA Select Route GUI is depicted in Figure 212. A description of the routines called by VIA Select Route GUI is provided in Table 214.

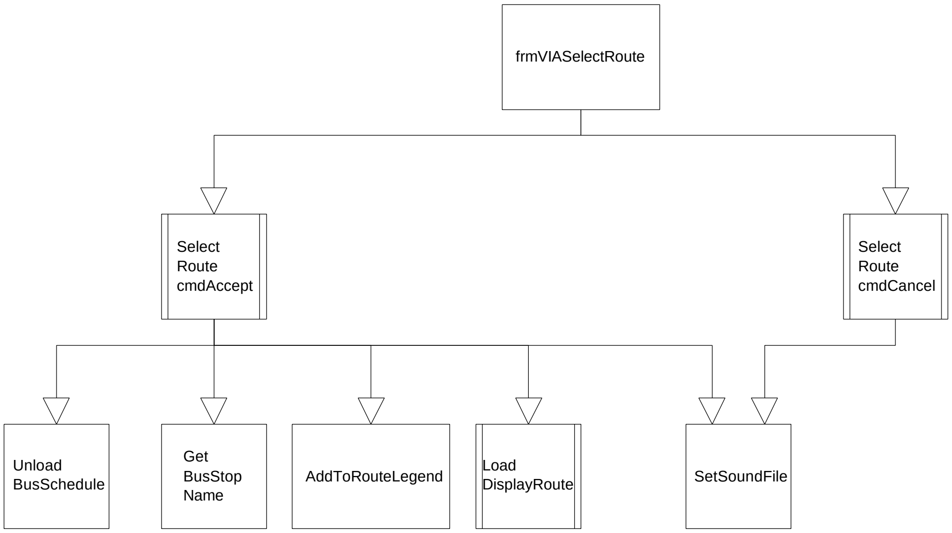

Figure 212 - VIA Select Route GUI Structure Chart

Function	Description
AddToRouteLegend	This routine adds the bus stop number and corresponding bus stop name to the route legend grid on the Display Route form.
Get BusStop Name	This function retrieves the bus stop text name from the Cross Reference array using the bus stop number as the key.
Load DisplayRoute	This method loads the Display Route form.
Select Route cmdAccept	This routine finds and displays the bus route schedule information. The last touch time is updated and a message that the route schedule is being processed is displayed. Next, the bus schedule grids are cleared. Determine the day of the year and search through the service file until the day is found. Determine if the route follows a typical service schedule or an alternative service schedule. If the route number is in the header record, then the route follows an alternate service schedule. Otherwise, the route follows the typical service schedule. Next, set the Select Route label to the route number. Open the route file and examine the service type. If the service type is zero, then the route does not provide services for this day. Otherwise, begin to load the route schedule into two grids. Use the bus stop numbers to retrieve the associated text name from the Cross Reference array. Use the numbers and names to build a legend for printing and load the text names as headings in the grids. Load the stop times into the grids. Determine if there are bus route attributes and if one is found, display the attribute. Finally, determine the directions for each of the grids and display them. After the grids are completed, the associated bitmap file is loaded. The SelectRoute list is cleared and the form is hidden. The DisplayRoute form is loaded and the VIA Route Schedule form is hidden.
Select Route cmdCancel	This event occurs when the user selects the Cancel button on the Select Route form. The touch soundfile is played, the last touch time is updated, the bus route list is cleared and the Select Route form is hidden.
SetSoundFile	Passes the sound file, that is called when any click event occurs, as a parameter to the call to the Windows API PlaySound.
Unload BusSchedule	This routine clears the two route schedule grids using the RemoveItem method.

Table 214 - Routines called by VIA Select Route GUI

4.2.2.2.18 VIA Display Route Schedule

The VIA Display Route Schedule GUI displays the route schedule grids and bitmap. The user can also select to print the route schedule information and/or return to the VIA Route Schedule GUI. The structure chart for VIA Display Route Schedule is depicted in Figure 213. A description of the routines called by VIA Display Route Schedule is provided in Table 215.

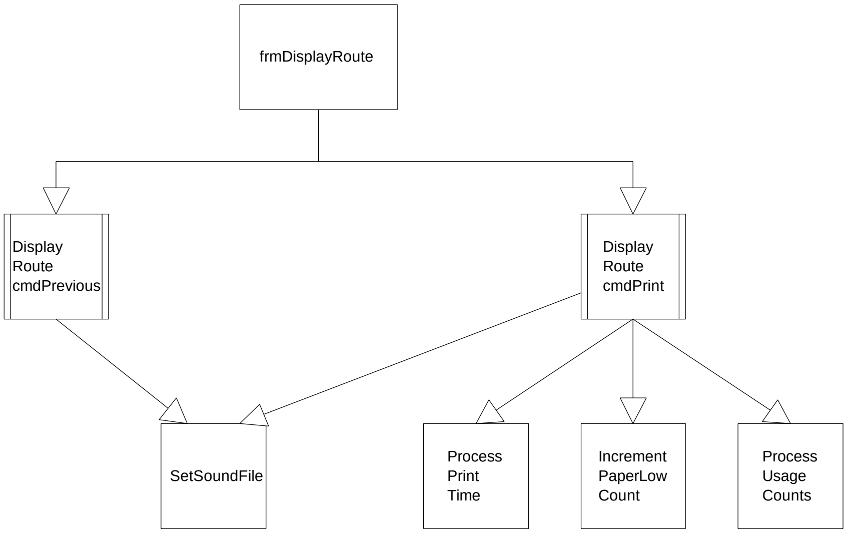

Figure 213 - VIA Display Route Schedule Structure Chart

Function	Description
Display Route cmdPrevious	This event is triggered when the user selects the previous button. The touch soundfile is played, the last touch time is updated, displayed route attributes are cleared, the VIA Route Schedule GUI is shown, and the Display Route GUI is hidden.
Display Route cmdPrint	This event is triggered when the user selects the print button. The touch soundfile is played and the last touch time is updated. If printing is enabled, then the bus route schedule, the bus route bitmap and the bus stop legend are printed. In order to prepare for printing the route information, the print time is updated, the paper low counter is incremented, and the print usage counter is incremented. Using the PPrtr DLL the two route grids, the route bitrmap, and the route legend are printed.
Increment PaperLow Count	Increments the paper low counter, contained in the paper low file, every time a print is requested by the user.
Process Print Time	Stores the last print time into the Print Time array and then determines if the print button has excedded its threshold (e.g., more that 5 prints in 30 seconds).
Process Usage Counts	Saves the current usage counts for Main Menu, SA Map, Airport, Weather, VIA and Route Guidance to the usage file.
SetSoundFile	Passes the sound file, that is called when any click event occurs, as a parameter to the call to the Windows API PlaySound.

Table 215 - Routines called by VIA Display Route Schedule

4.2.2.2.19 VIA Generic Information GUI

The VIA Generic Information GUI is used to display the VIA information gif files to the user. The GUI is used to display fare and pass information, general information, special event information, and VIATrans information. The structure chart for VIA Generic Information GUI is depicted in Figure 214. A description of the routines called by VIA Generic Information GUI is provided in Table 216.

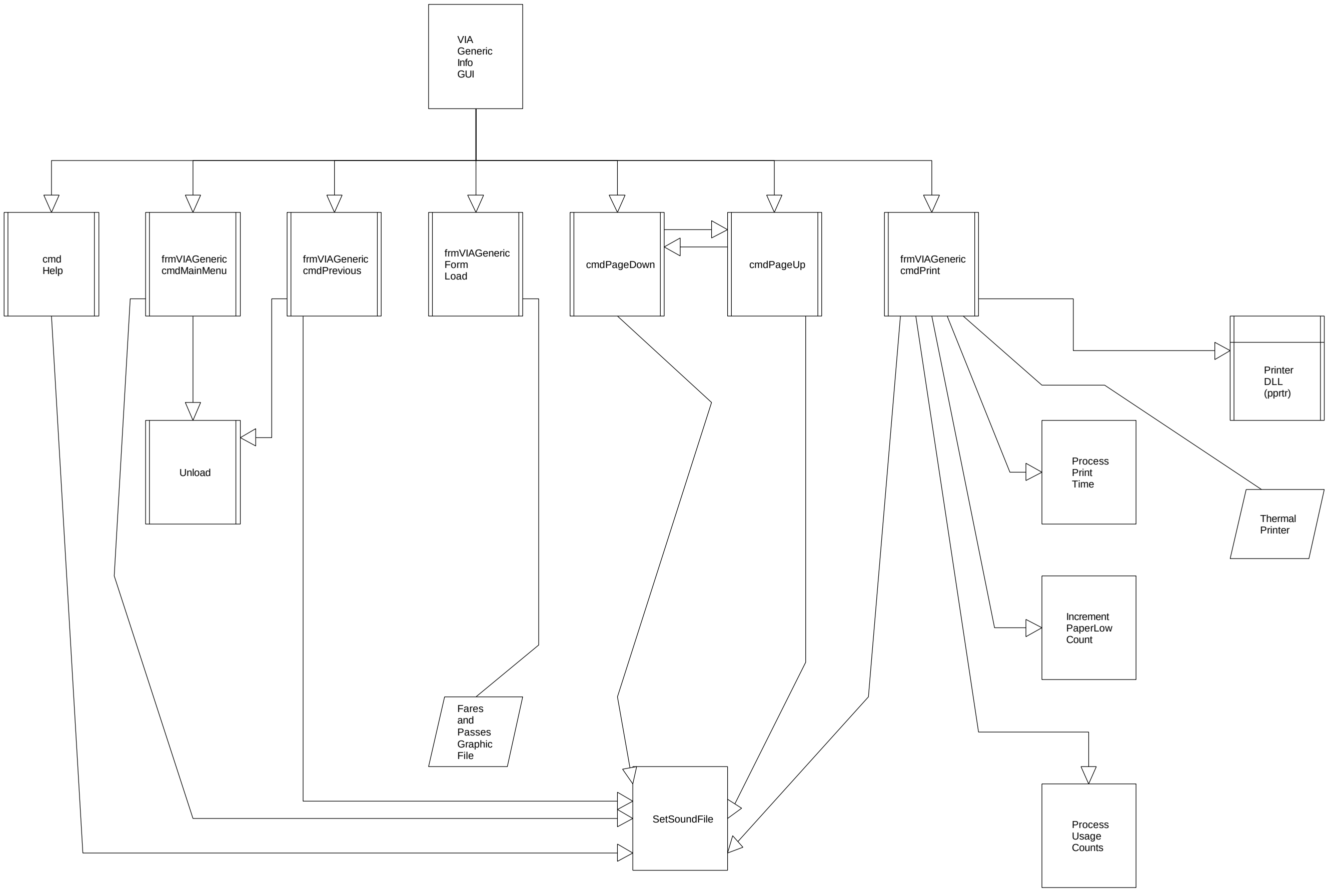

Figure 214 - VIA Information GUIs Structure Chart

Function	Description
cmd Help	The button displays help for the specified GUI from which it is selected. The appropriate bitmap file is retrieved and displayed.
CmdPageDown	After sound file is played, set LAST_TOUCH_TIME to current time. If any errors occur, send an error message to error logging routine. Set lblErrorMsg visibility property to false. Set HScroll1 visibility property to true. If there is another graphic to display then set Picture2 to next graphic and set cmdPageUp enabled property to true. Also, if it is the last picture to be displayed then set cmdPageDown enabled property to false. If unable to load next graphic, send an error message, set Picture2 to no graphic, set lblErrorMsg to "Page (applicable page number) Currently Unavailable" and visibility property to true, set HScroll1 visibility property to false and continue for successive pages (if any).
CmdPageUp	After sound file is played, set LAST_TOUCH_TIME to current time. If any errors occur, send an error message to error logging routine. Set lblErrorMsg visibility property to false. Set HScroll1 visibility property to true. If there is another graphic to display then set Picture2 to previous graphic and set cmdPageDown enabled property to true. Also, if it is the first picture to be displayed then set cmdPageUp enabled property to false. If unable to load previous graphic, send an error message, set Picture2 to no graphic, set lblErrorMsg to "Page (applicable page number) Currently Unavailable" and visibility property to true, set HScroll1 visibility property to false and continue for successive pages (if any).
FrmVIAGeneric cmdMainMenu	After sound file is played, set LAST_TOUCH_TIME to current time, show frmMainMenu, delete the contents of the file list array to reclaim Windows resources, and unload the VIA Generic Info GUI.
FrmVIAGeneric cmdPrevious	After sound file is played, set LAST_TOUCH_TIME to current time, show frmVIATransit, delete the contents of the file list array to reclaim Windows resources, and unload the VIA Generic Info GUI.
FrmVIAGeneric cmdPrint	If ENABLE_PRINT is true and Picture2 contains a graphic, continue with printing procedures, otherwise discontinue printing. ProcessPrintTime by checking if the print button has been selected more than 5 times in the last 30 seconds. IncrementPaperLowCount and store in status file. ProcessUsageCount by incrementing PRINT_COUNT variable in status file. Change Picture2 scalemode to twips(1) for printing only. Set printer properties such as orientation and length by calling the printer dll (pprtr.dll) and send special events graphic to the thermal printer for printing. Change Picture2 scalemode back to default, pixel, for display.
FrmVIAGeneric Form Load	Setup initial form properties: (1)set scalemode to pixel for frmVIAGeneric, Picture1, and Picture2, (2)set Picture2 autosize property to true, (3)remove Picture1 and Picture2 borders. Load the selected VIA graphics into an array for display purposes. If no graphics are found then set lblErrorMsg to "Information Currently Unavailable", set its visibility property to true, and hide all graphic controls (cmdPageUp, cmdPageDown, Picture1, Picture2, HScroll1, VScroll1). If only the first graphic is missing then set lblErrorMsg to "Page 1 is Currently Unavailable" and setup a default Picture1 and Picture2 size. Otherwise, load first graphic into Picture2. If only one graphic file is available set cmdPageUp and cmdPageDown enabled property to false. Setup remaining graphic properties: (1)set Picture1 and Picture2 locations and (2)set HScroll1 and VScroll1 position and Max property to appropriate width and height (if scroll bars are needed).
Increment PaperLow Count	Increments the paper low counter, contained in the paper low file, every time a print is requested by the user.
Printer DLL (pprtr)	Changes thermal printer properties, such as orientation and paper length, through a dll call.
Process Print Time	Stores the last print time into the Print Time array and then determines if the print button has exceeded its threshold (e.g., more that 5 prints in 30 seconds).
Process Usage Counts	Saves the current usage counts for Main Menu, SA Map, Airport, Weather, VIA and Route Guidance to the usage file.

Function	Description
SetSoundFile	Passes the sound file, that is called when any click event occurs, as a parameter to the call to the Windows API PlaySound.
Unload	Visual Basic Function - Unloads the appropriate form from memory and reclaims Windows resources.

Table 216 - Routines called by VIA Generic Information GUI

4.2.2.2.20 Screen Saver

The Screen Saver function is invoked when the last touch time has exceeded the maximum amount of time between user touches. The function executes similar to a screen saver in that pictures and videos are displayed until someone touches the GUI, thus ending the screen saver and displaying the SA Online Main Menu. The structure chart for Screen Saver is depicted in Figure 215. A description of the routines called Screen Saver is provided in Table 217.

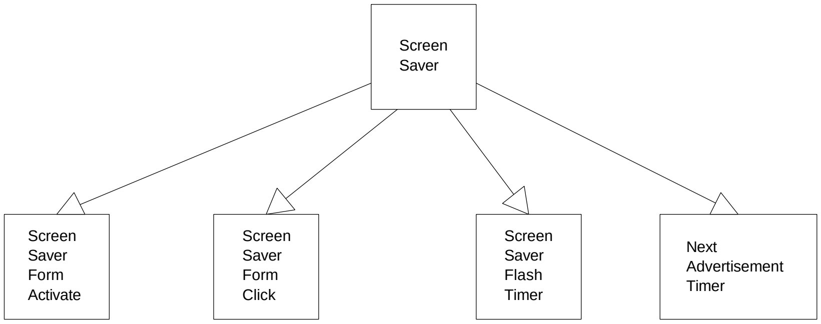

Figure 215 - Screen Saver Structure Chart

Function	Description
Next Advertisement Timer	This routine determines if the current advertisement's timer has expired. If the advertisement's time has expired, then the display of the advertisement is terminated and the next advertisement is loaded and displayed.
Screen Saver Flash Timer	This routine changes the color of the message displayed at the bottom of the screen saver once a second.
Screen Saver Form Activate	This routine enables the next advertisement and flash timers and sets the flash timer interval to one second.
Screen Saver Form Click	This event is triggered when the user touches the GUI. Pictures or movies that are currently being displayed are stopped and the SA Online Main Menu is displayed.

Table 217 - Routines called by Screen Saver

4.2.2.3 Real Time Data Application

The Real Time Data Application reads, processes and transmit the real time traffic data being broadcast from the FM STIC. The application reads the real time traffic data protocol from a serial port, which has a specially modified FM receiver that receives the real time data broadcast. The data is then converted into Transguide Link Ids and speeds or incident codes. The Transguide Link Ids and associated speed or incident codes are then transmitted to the GUI for display on the San Antonio Street Map and to be used in the calculation of routes. For more information about the FM STIC and the real time data protocol, consult the In-Vehicle Navigation Design Document. The remainder of this section describes the Real Time Data Application design using structure charts and tables. The primary events that occur and the actions that trigger these events are depicted in Figure 216 and are described in Table 218.

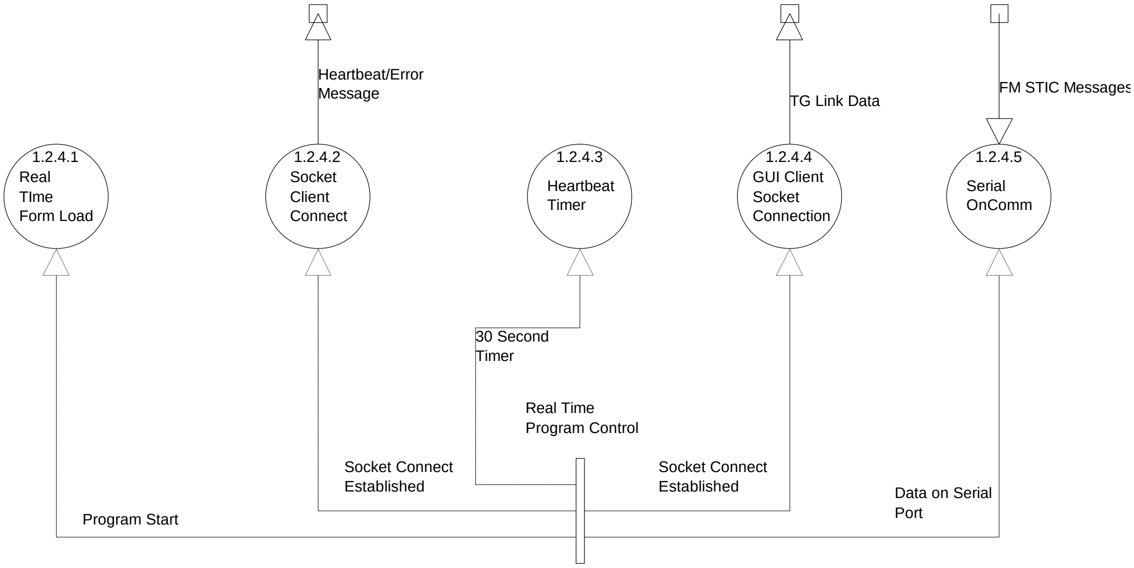
Heartbeat/Error Message
TG Link Data
FM STIC Messages
1.2.4.1 Real TIme Form Load
1.2.4.2 Socket Client Connect
1.2.4.3 Heartbeat Timer
1.2.4.4 GUI Client Socket Connection
1.2.4.5 Serial OnComm
30 Second Timer
Real Time Program Control
Socket Connect Established
Socket Connect Established
Data on Serial Port
Program Start

Figure 216 –Real Time Data Process Events and Event Triggers

Function	Description
30 Second Timer	This event is set to occur every thirty seconds. When the event is triggered, the application will perform the actions specified in the timer.
Data on Serial Port	This is an indication that data has arrived on the serial port and needs to be processed.
FM STIC Messages	The FM STIC Messages are broadcast by the In-Vehicle Navigation project and the structure of the messages can be found in the In-Vehicle Navigation Design Document.
GUI Client Socket Connection	This routine transmits the Transguide Link Ids and their associated speed or incident codes to the GUI application.
Heartbeat Timer	This event is triggered every thirty seconds to transmit the heartbeat record to the Error Server.
Heartbeat/Error Message	A record containing either an application heartbeat or an error message. The record is composed of a 2 byte identifier, the message, and a record delimiter.
Program Start	When the program is initiated, the first action of the application is to load the startup form and perform the form load functions.
Real Time Form Load	This event takes place at program start and displays the real time form used for diagnostic purposes.
Real Time Program Control	The control of the events and activities that can occur while an application is executing.
Serial OnComm	This is the event that is triggered when data is available on the serial port. The data is read from the port and processed by the real time process.
Socket Client Connect	This event is triggered when the Error Server accepts the socket request. The socket ready flag is set to True to indicate that data may now be transmitted over the socket.

Function	Description
Socket Connect Established	The server application has accepted the socket connection request and the communication between the programs is initiated.
TG Link Data	Data containing the Transguide Link Id and its associated speed or incident code.

Table 218 – Real Time Data Process Event and Event Triggers Description

4.2.2.3.1 Real Time Form Load

The Real Time Form Load routine performs the initialization for the real time data process including loading the configurable parameters, initializing process variables, and opening the connection to the error server. The structure chart for Real Time Form Load is depicted in Figure 217. A description of the routines called by Real Time Form Load is provided in Table 219.

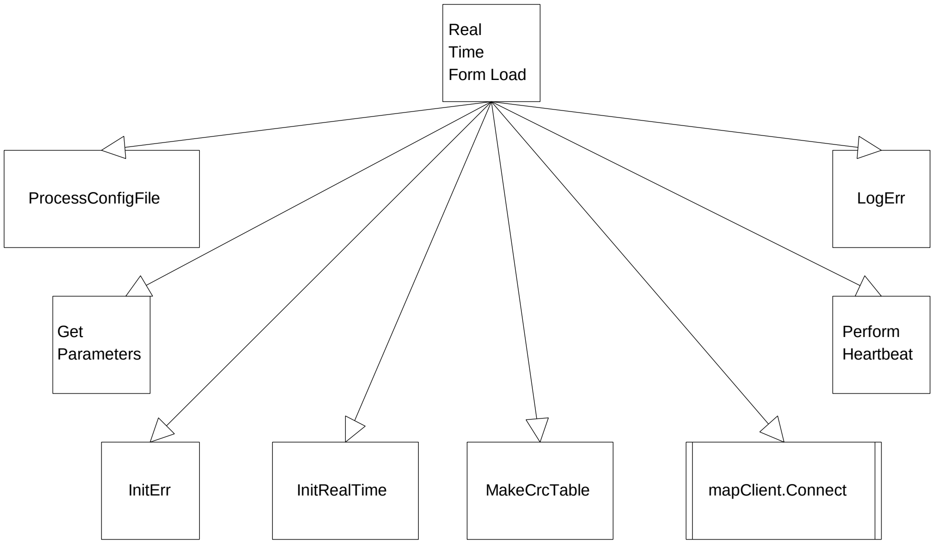

Figure 217 - Real Time Form Load Structure Chart

Function	Description
Get Parameters	The function retrieves the command line parameters passed from Error Server. The function first checks to see if the application is being run in local or remote mode. If the application is in local mode, the application id is defaulted to one. If the application is in remote mode, the application id is retrieved from the command line parameters using Command$ command. The returned command line is parsed. The application id is identified by a "-i" followed by the application id number. The id is stored in a global variable for use by the Heartbeat process. If the application id is not found, an error is logged.
InitErr	This function sets up the socket connection to the Error Server for error reporting. The remote host IP address and remote host port number are stored in a Winsock object to identify how Modem Communications will communicate with the Error Server. Next, the Connect method is used to establish the socket connection between the two programs. Initerr must wait until the connection is established before proceeding. The Winsock object Connect event will be triggered when the connection is successfully established. The local Winsock object is then assigned to a global variable for use by other routines.
InitRealTime	Initializes the global variables used by the real time data process including global flags and counters, the list of processed sequence numbers and list of incidents.
LogErr	This function transmits the given message to the Error Server. The message is passed into the function as a parameter and a record delimiter is appended to the message. The message is then transmitted to the Error Server, using the Winsock object SendData method. Due to the unknown length of the message, the message is broken into ten byte segments and transmitted piecemeal until the entire message is sent.
MakeCrcTable	Initializes the entries in the reverse CCITT CRC-16 look up table. This table provides the reverse CCITT CRC-16 value for each of the 255 possible byte values and is used by the function CalcCrcWithLookup to determine the CRC values for individual bytes in a string.
Perform Heartbeat	This function is called by the heartbeat timer, when it is time to send a heartbeat message to the Error Server. The heartbeat message is built and a record delimiter is appended to the end of the record. The message is transmitted to the Error Server via the Winsock object SendData method.
ProcessConfigFile	Reads and assigns from disk file the values of the configurable parameters for the real time data process.
mapClient.Connect	Establishes the socket connection to the error logging process.

Table 219 - Routines called by Real Time Form Load

4.2.2.3.2 MakeCrcTable

The MakeCrcTable routine initializes the entries in the reverse CCITT CRC-16 look up table. This table provides the reverse CCITT CRC-16 value for each of the 255 possible byte values and is used by the function CalcCrcWithLookup to determine the CRC values for individual bytes in a string. The structure chart for MakeCrcTable is depicted in Figure 218. A description of the routines called by MakeCrcTable is provided in Table 220.

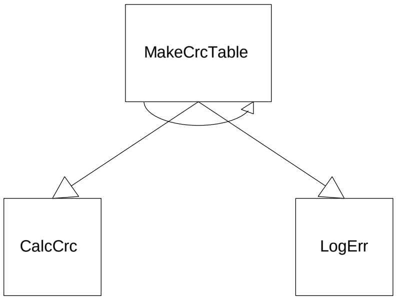

Figure 218 - MakeCrcTable Structure Chart

Function	Description
CalcCrc	Calculates the reverse CCITT CRC-16 on the passed byte, resulting in an update to a 16-bit CRC value. Additional data can be included in the CRC computed by repeatedly calling the routine with the new data and the current value of the accumulator.
LogErr	This function transmits the given message to the Error Server. The message is passed into the function as a parameter and a record delimiter is appended to the message. The message is then transmitted to the Error Server, using the Winsock object SendData method. Due to the unknown length of the message, the message is broken into ten byte segments and transmitted piecemeal until the entire message is sent.

Table 220 - Routines called by MakeCrcTable

4.2.2.3.3 Serial OnComm

The Serial OnComm routine reads and processes serial data from the STIC receiver. The function is invoked on serial port interrupt. Data is read and added to a message buffer which is processed when a complete message has been received. The function operates as a state machine in which the states include 1) detection of the start of message flag, 2) reading the message byte count field, 3) reading the message data, and 4) processing the message. The structure chart for Serial OnComm is depicted in Figure 219. A description of the routines called by Serial OnComm is provided in Table 221.

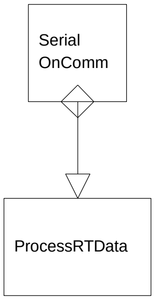

Figure 219 - Serial OnComm Structure Chart

Function	Description

Function	Description
ProcessRTData	Processes a complete message from the STIC receiver. Processing includes 1) validating the message by verifying the CRC, 2) decoding the message header to determine the message type, 3) decoding the message data to extract the link speed or incident data contained in the message, 4) transmitting the extracted data to the map client.

Table 221 - Routines called by Serial OnComm

4.2.2.3.4 ProcessRTData

The ProcessRTData routine processes a complete message from the STIC receiver. Processing includes 1) validating the message by verifying the CRC, 2) decoding the message header to determine the message type, 3) decoding the message data to extract the link speed or incident data contained in the message, and 4) transmitting the extracted data to the map client. The structure chart for ProcessRTData is depicted in Figure 220. A description of the routines called by ProcessRTData is provided in Table 222.

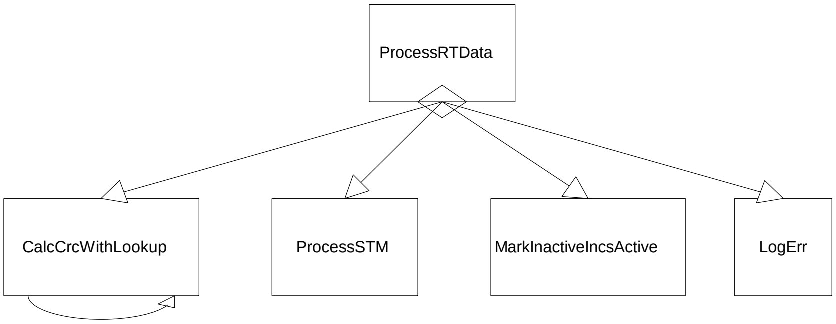

Figure 220 - ProcessRTData Structure Chart

Function	Description
CalcCrcWithLookup	Calculates the reverse CCITT CRC-16 on the passed string using the CRC look up table generated at system initialization.
LogErr	This function transmits the given message to the Error Server. The message is passed into the function as a parameter and a record delimiter is appended to the message. The message is then transmitted to the Error Server, using the Winsock object SendData method. Due to the unknown length of the message, the message is broken into ten byte segments and transmitted piecemeal until the entire message is sent.
MarkInactiveIncsActive	Sets the status of all inactive incidents in the global incident list to active. This routine is called in the event that an error is detected which prevents processing of an STM in order to prevent deletion of an active incident from the global incident list.
ProcessSTM	Processes a single STIC Transmission Message (STM). Each STM contains one or more Traffic Information Messages (TIMs). Each TIM contains either link speed or incident data for a number of links or incidents. The array of STMs which contain the complete set of link speed and incident data is referred to as the STM superframe. Refer to the Transguide In Vehicle Navigation System High Speed FM Subcarrier Communications Protocol for a complete description of the STM message format. This function extracts the header information from the STM, detects the start of a superframe, performs any special processing required at the beginning of a superframe, and calls the function which processes the TIMs contained within the STM.

Table 222 - Routines called by ProcessRTData

4.2.2.3.5 ProcessSTM

The ProcessSTM routine processes a single STIC Transmission Message (STM). Each STM contains one or more Traffic Information Messages (TIMs). Each TIM contains either link speed or incident data for a number of links or incidents. The array of STMs which contain the complete set of link speed and incident data is referred to as the STM superframe. Refer to the Transguide In Vehicle Navigation System High Speed FM Subcarrier Communications Protocol for a complete description of the STM message format. This function extracts the header information from the STM, detects the start of a superframe, performs any special processing required at the beginning of a superframe, and calls the function which processes the TIMs contained within the STM. The structure chart for ProcessSTM is depicted in Figure 221. A description of the routines called by ProcessSTM is provided in Table 223.

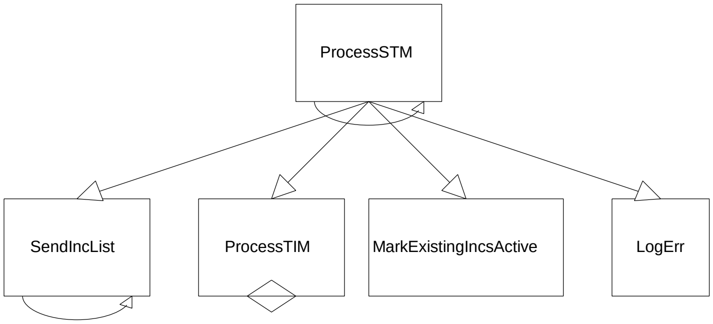

Figure 221 - ProcessSTM Structure Chart

Function	Description
LogErr	This function transmits the given message to the Error Server. The message is passed into the function as a parameter and a record delimiter is appended to the message. The message is then transmitted to the Error Server, using the Winsock object SendData method. Due to the unknown length of the message, the message is broken into ten byte segments and transmitted piecemeal until the entire message is sent.
MarkExistingIncsActive	Searches the global incident list for incidents with the specified sequence number and sets the status of those incidents to active. This routine is called to set as active the incidents in STMs which have been previously processed (as indicated by the sequence number) and so are still active.
ProcessTIM	Determines the type of TIM (link speed or incident) and, according to the TIM type, calls the appropriate TIM processing function to extract the data from the TIM.
SendIncList	Sends all new incidents and all inactive incidents in the global incident list to the map client. Once sent, inactive incidents are deleted from the incident list. All other incidents are marked as inactive.

Table 223 - Routines called by ProcessSTM

4.2.2.3.6 SendIncList

The SendIncList routine sends all new incidents and all inactive incidents in the global incident list to the map client. Once sent, inactive incidents are deleted from the incident list. All other incidents are marked as inactive. The structure chart for SendIncList is depicted in Figure 222. A description of the routines called by SendIncList is provided in Table 224.

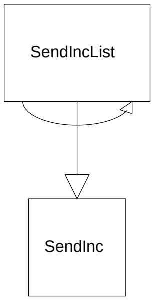

Figure 222 - SendIncList Structure Chart

Function	Description
SendInc	Formats the data for a single incident into an ASCII string and transmits the data to the map client. The string transmitted includes the starting and ending longitude, latitude and street level code of the incident location, the incident type code, and the incident begin time.

Table 224 - Routines called by SendIncList

4.2.2.3.7 ProcessTIM

The ProcessTIM routine determines the type of TIM (link speed or incident) and, according to the TIM type, calls the appropriate TIM processing function to extract the data from the TIM. The structure chart for ProcessTIM is depicted in Figure 223. A description of the routines called by ProcessTIM is provided in Table 225.

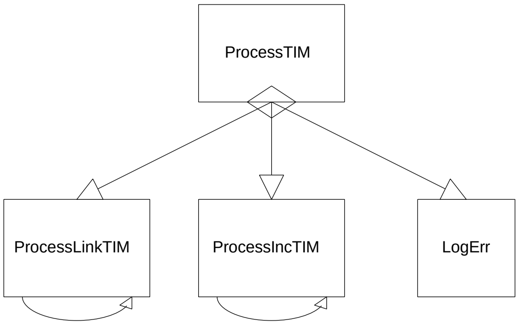

Figure 223 - ProcessTIM Structure Chart

Function	Description
LogErr	This function transmits the given message to the Error Server. The message is passed into the function as a parameter and a record delimiter is appended to the message. The message is then transmitted to the Error Server, using the Winsock object SendData method. Due to the unknown length of the message, the message is broken into ten byte segments and transmitted piecemeal until the entire message is sent.
ProcessIncTIM	Processes an incident data TIM. This includes 1) extracting the data from the location reference header, 2) extracting the location and data for each incident, and 3) storing the location and data for each incident to the global incident list.
ProcessLinkTIM	Processes a link speed data TIM. This includes 1) extracting the data from the location reference header, 2) extracting the location and speed for each link, and 3) formatting the location and speed data for each link into ASCII strings and transmitting the strings to the map client. Each string transmitted to the map client contains the data for one link, including the starting and ending longitude, latitude and street level code of the link, and the speed for the link.

Table 225 - Routines called by ProcessTIM

4.2.2.3.8 ProcessLinkTIM

The ProcessLinkTIM routine processes a link speed data TIM. This includes 1) extracting the data from the location reference header, 2) extracting the location and speed for each link, and 3) formatting the location and speed data for each link into ASCII strings and transmitting the strings to the map client. Each string transmitted to the map client contains the data for one link, including the starting and ending longitude, latitude and street level code of the link, and the speed for the link. The structure chart for ProcessLinkTIM is depicted in Figure 224. A description of the routines called by ProcessLinkTIM is provided in Table 226.

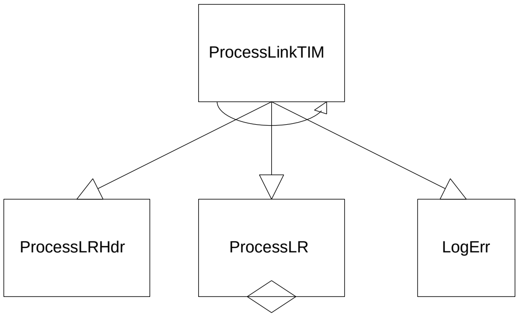

Figure 224 - ProcessLinkTIM Structure Chart

Function	Description
LogErr	This function transmits the given message to the Error Server. The message is passed into the function as a parameter and a record delimiter is appended to the message. The message is then transmitted to the Error Server, using the Winsock object SendData method. Due to the unknown length of the message, the message is broken into ten byte segments and transmitted piecemeal until the entire message is sent.
ProcessLR	Processes a location reference by determining the location reference type (local or global point or link) and calling the appropriate processing function to extract the location data for that type.
ProcessLRHdr	Extracts the data from location reference header. This data includes the longitude and latitude of the origin location and the location reference type (either local or global).

Table 226 - Routines called by ProcessLinkTIM

4.2.2.3.9 ProcessIncTIM

The ProcessIncTIM routine processes an incident data TIM. This includes 1) extracting the data from the location reference header, 2) extracting the location and data for each incident, and 3) storing the location and data for each incident to the global incident list. The structure chart for ProcessIncTIM is depicted in Figure 225. A description of the routines called by ProcessIncTIM is provided in Table 227.

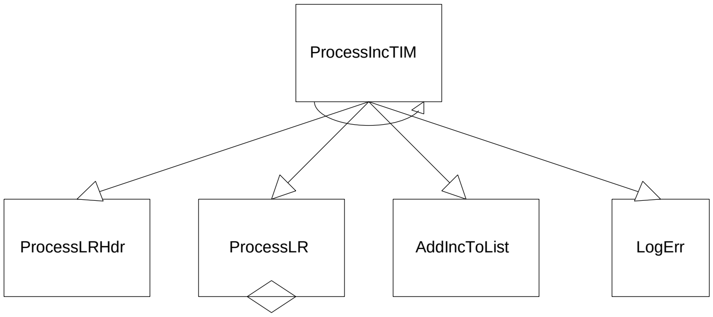

Figure 225 - ProcessIncTIM Structure Chart

Function	Description
AddIncToList	Subroutine adds the passed incident to the global list of active incidents. The incident is added only if there is space available and the incident is not already in the list.
LogErr	This function transmits the given message to the Error Server. The message is passed into the function as a parameter and a record delimiter is appended to the message. The message is then transmitted to the Error Server, using the Winsock object SendData method. Due to the unknown length of the message, the message is broken into ten byte segments and transmitted piecemeal until the entire message is sent.
ProcessLR	Processes a location reference by determining the location reference type (local or global point or link) and calling the appropriate processing function to extract the location data for that type.
ProcessLRHdr	Extracts the data from location reference header. This data includes the longitude and latitude of the origin location and the location reference type (either local or global).

Table 227 - Routines called by ProcessIncTIM

4.2.2.3.10 AddIncToList

The AddIncToList routine adds the passed incident to the global list of active incidents. The incident is added only if there is space available and the incident is not already in the list. The structure chart for AddIncToList is depicted in Figure 226. A description of the routines called by AddIncToList is provided in Table 228.

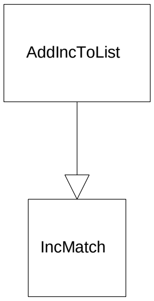

Figure 226 - AddIncToList Structure Chart

Function	Description
IncMatch	Determines if two incidents match by comparing each of the fields for the two incidents. The function returns 1 if all fields in both incidents match, 0 otherwise.

Table 228 - Routines called by AddIncToList

4.2.2.3.11 ProcessLR

The ProcessLR routine processes a location reference by determining the location reference type (local or global point or link) and calling the appropriate processing function to extract the location data for that type. The structure chart for ProcessLR is depicted in Figure 227. A description of the routines called by ProcessLR is provided in Table 229.

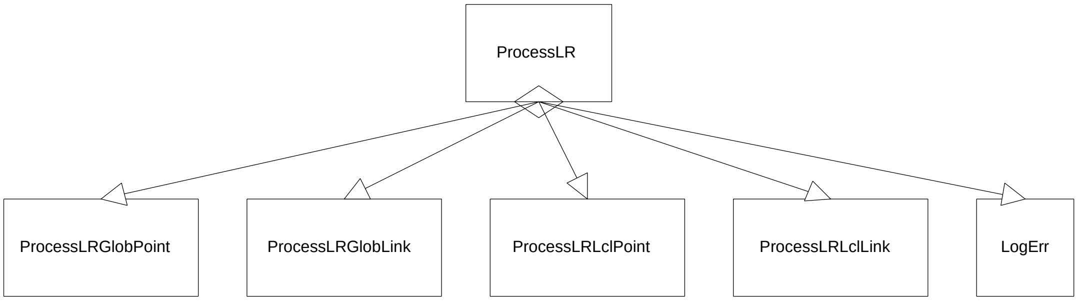

Figure 227 - ProcessLR Structure Chart

Function	Description
LogErr	This function transmits the given message to the Error Server. The message is passed into the function as a parameter and a record delimiter is appended to the message. The message is then transmitted to the Error Server, using the Winsock object SendData method. Due to the unknown length of the message, the message is broken into ten byte segments and transmitted piecemeal until the entire message is sent.
ProcessLRGlobLink	Processes a global link location reference. This includes extracting the starting and ending latitude, longitude and street level and if available the street name for globally referenced link location reference.
ProcessLRGlobPoint	Processes a global point location reference. This includes extracting the latitude, longitude, street level and if available the street name for globally referenced point location reference.
ProcessLRLclLink	Processes a local link location reference. This includes extracting the starting and ending latitude, longitude and street level and if available the street name for a locally referenced link location reference and converting the local start/end locations to absolute locations.
ProcessLRLclPoint	Processes a local point location reference. This includes extracting the local latitude, longitude, street level and if available the street name for locally referenced point location reference and converting the locally referenced location to an absolute location.

Table 229 - Routines called by ProcessLR

4.2.2.3.12 ProcessLRGlobPoint

The ProcessLRGlobPoint routine processes a global point location reference. This includes extracting the latitude, longitude, street level and if available the street name for globally referenced point location reference. The structure chart for ProcessLRGlobPoint is depicted in Figure 228. A description of the routines called by ProcessLRGlobPoint is provided in Table 230.

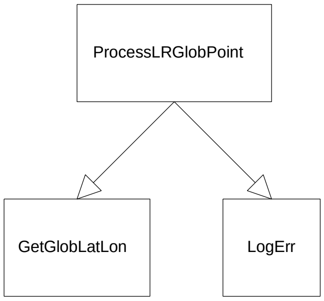

Figure 228 - ProcessLRGlobPoint Structure Chart

Function	Description
GetGlobLatLon	Extracts the latitude or longitude from a global location reference and converts the value from microdegrees to Transguide units (10's of microdegrees).
LogErr	This function transmits the given message to the Error Server. The message is passed into the function as a parameter and a record delimiter is appended to the message. The message is then transmitted to the Error Server, using the Winsock object SendData method. Due to the unknown length of the message, the message is broken into ten byte segments and transmitted piecemeal until the entire message is sent.

Table 230 - Routines called by ProcessLRGlobPoint

4.2.2.3.13 ProcessLRGlobLink

The ProcessLRGlobLink routine processes a global link location reference. This includes extracting the starting and ending latitude, longitude and street level and if available the street name for globally referenced link location reference. The structure chart for ProcessLRGlobLink is depicted in Figure 229. A description of the routines called by ProcessLRGlobLink is provided in Table 231.

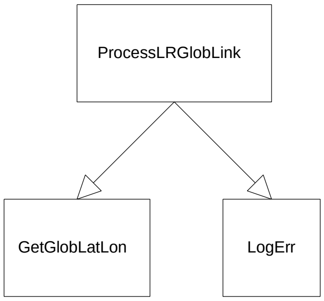

Figure 229 - ProcessLRGlobLink Structure Chart

Function	Description
GetGlobLatLon	Extracts the latitude or longitude from a global location reference and converts the value from microdegrees to Transguide units (10's of microdegrees).
LogErr	This function transmits the given message to the Error Server. The message is passed into the function as a parameter and a record delimiter is appended to the message. The message is then transmitted to the Error Server, using the Winsock object SendData method. Due to the unknown length of the message, the message is broken into ten byte segments and transmitted piecemeal until the entire message is sent.

Table 231 - Routines called by ProcessLRGlobLink

4.2.2.3.14 ProcessLRLclPoint

The ProcessLRLclPoint routine processes a local point location reference. This includes extracting the local latitude, longitude, street level and if available the street name for locally referenced point location reference and converting the locally referenced location to an absolute location. The structure chart for ProcessLRLclPoint is depicted in Figure 230. A description of the routines called by ProcessLRLclPoint is provided in Table 232.

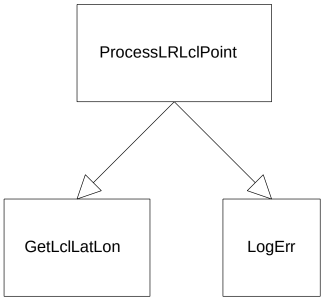

Figure 230 - ProcessLRLclPoint Structure Chart

Function	Description
GetLclLatLon	Extracts the latitude or longitude from a local location reference, converts the value from microdegrees to Transguide units (10's of microdegrees), and converts the locally referenced value to an absolute location.
LogErr	This function transmits the given message to the Error Server. The message is passed into the function as a parameter and a record delimiter is appended to the message. The message is then transmitted to the Error Server, using the Winsock object SendData method. Due to the unknown length of the message, the message is broken into ten byte segments and transmitted piecemeal until the entire message is sent.

Table 232 - Routines called by ProcessLRLclPoint

4.2.2.4 Modem Communications Application

The Modem Communication Application communicates with the Kiosk Master Computer through a modem to receive data files (Via, Airport, Weather, and Screensaver) and transmits status. The application waits for an incoming phone call. When an incoming phone call is detected, the application establishes a communication connection with the Kiosk Master Computer. Once the connection is established, the Modem Communications and the Kiosk Master Computer exchange data. Figure 231 depicts the events that occur within the Modem Communications Application and Table 233 provides a description of these events and their triggers.

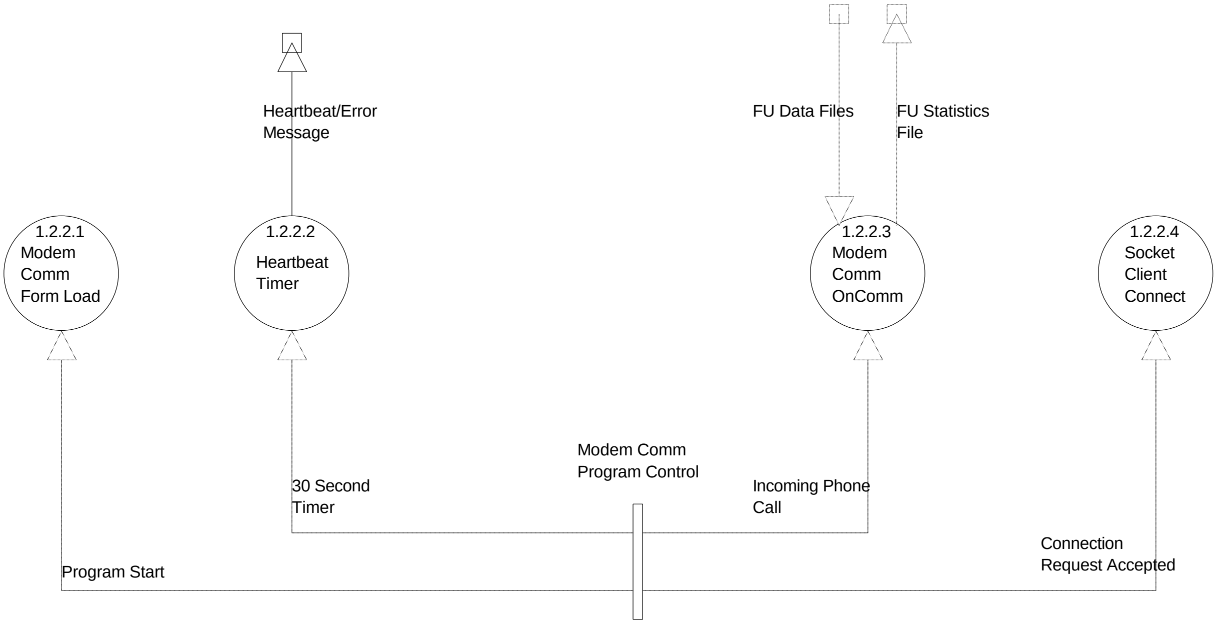
Heartbeat/Error Message
FU Data Files
FU Statistics File
1.2.2.1 Modem Comm Form Load
1.2.2.2 Heartbeat Timer
1.2.2.3 Modem Comm OnComm
1.2.2.4 Socket Client Connect
Modem Comm Program Control
30 Second Timer
Incoming Phone Call
Connection Request Accepted
Program Start

Figure 231 – Modem Communications Events and Event Triggers

Function	Description
Modem Comm Program Control	The control of the events and activities that can occur while an application is executing.
30 Second Timer	This event is set to occur every thirty seconds. When the event is triggered, the application will perform the actions specified in the timer.
Connection Request Accepted	This event occurs when the server accepts the socket connection; thus enabling communications between the two applications.
Incoming Phone Call	Action that occurs when an incoming phone call is detected on the modem.
Program Start	When the program is initiated, the first action of the application is to load the startup form and perform the form load functions.
FU Data Files	These files are received from the Master Computer and placed into the production directories. These files includes Transit files, Weather files, Airport files, and Screen Saver files.

Function	Description
FU Statistics File	Contains the current status of the Field Unit and its applications.
Heartbeat/Error Message	A record containing either an application heartbeat or an error message. The record is composed of a 2 byte identifier, the message, and a record delimiter.
Heartbeat Timer	This event is triggered every thirty seconds to transmit the heartbeat record to the Error Server.
Modem Comm Form Load	This event occurs when the application is initially started.

Function	Description
Modem Comm OnComm	When this event occurs, the event value is returned by the OnComm Event. This value is used to determine the event or error that has occurred. This list below are the values and their meaning that can occur: Eventbreak - Break Signal Received EventCDTO - Carrier Detect Timeout EventCTSTO - CTS Timeout EventDSRTO - DSR Timeout EventFrame - Framing Error EventOverrun - Data Lost EventRxOver - Receive buffer overflow EventRxParity - Parity Error EventTxFull - Transmit buffer full EventDCB - Unexpected error retrieving DCB EvCD - Change in CD line EvCTS - Change in CTS line EvDSR - Change in DSR line EvRing - Change in Ring Indicator EvReceive - Received Threshold number of characters; Threshold number is determined by the Threshold property. EvSend - Send Threshold number of characters in the transmit buffer; Send Threshold number is determined by the Send Threshold property. EvEOF - An EOF character was detected in the input stream All events values should be detected, however not all of the event values will have meaning. For the values that are not being processed, log a message with the error handler that the event occurred. For events that are processed, the event and what actions occur are described below: EvCD - Display the current time, set the flags that control the incoming file, heartbeat file, and down loaded data, and the CRC down load flags to false. EvRing - Display the incoming phone call message and the current time EvReceive - The is the primary event that will get triggered, since this is how the incoming data files will be processed. When this event is triggered, read the data from the buffer and process it a byte at a time. If a Start of Message byte (ASCII 01) is detected, this could mean that a filename is being downloaded, a heartbeat requested is being downloaded, or data is being written to a file. Set the appropriate tracking flags to indicate these possibilities. If an End of Message byte (ASCII 04) is detected, then a file download, a CRC download, a heartbeat file upload, or an incoming filename downloaded may be in process. If a file is being downloaded, close the file. If the CRC has been downloaded, use the PX Extract function (PKExtractFile2File) and adjust the CRC to match an Unsigned Long that is computed on the Master Computer. Next, compare the downloaded CRC with the computed CRC. If these CRCs match, send a file received successful to the Master Compute; otherwise send a file received unsuccessful to the Master Computer. If the heartbeat file is being uploaded, open the heartbeat file, read the data, and transmit it to the Master Computer. If the heartbeat file does not exist, create a default file and transmit the default file to the Master Computer. If there is an incoming filename, display the name of the file. Finally, reset the processing flags. If a DLE Pending (Hex 10) is detected, this indicates the next byte needs have one added to it. This is done to prevent the data bytes from being misinterpreted. If the byte is not one of the above cases, it is assumed that is must be data. The number of bytes read is incremented by one. If a start of message has been received, then the processing flags are tested to determined what is being processed. If the data is about to be downloaded, a temporary file is opened that will contain the data and the downloaddata flag is set to true. If a new filename is about to be downloaded, the variable that will contain the filename is set to null. and the incoming filename flag is set to true. If a CRC is about to be downloaded, set the variable that will contain the CRC value to null and the CRC download flag to true. If the heartbeat file is about to be uploaded, the upload heartbeat flag is set to true. If the byte is not one of the above values, an error has occurred and it is ignored. If a start of message is not received, then write the byte to the temporary file or concatenate the byte to the filename or concatenate the byte to the CRC string based on the values of the processing flags. EvSend - The number of Receive Threshold bytes is recorded. Ev EOF - The character is ignored. If the character is not one of the above event values, an informational error is recorded and the character is ignored.
Socket Client Connect	This event is triggered when the Error Server accepts the socket request. The socket ready flag is set to True to indicate that data may now be transmitted over the socket.

Table 233 - Modem Communications Events and Event Triggers Descriptions

4.2.2.4.1 Modem Comm Form Load

The Modem Comm Form Load event occurs when the application is initially started to perform initialization activities. These activities are:

- Process the configuration file
- Get the command line parameters
- Initialize the Field Unit Status Array
- Connect to the Error Server if errors are being logged remotely
- Setup the communications port for the modem
  - Use the communications port, baud rate, parity, data, and stop bits read from the configuration file
  - Set the input mode to binary
  - Set the comm control to read the entire buffer
  - Set the buffer size to 1024 bytes (the larger this value the more memory required)
  - Open the port
  - Set the receive threshold to 1 byte (the larger this value the more memory required)
  - Set the transmit threshold to 0 bytes (the larger this value the more memory required)
  - Send the attention command to the modem
  - Set the modem volume to off
  - Reset the modem
  - Initialize to factory setting (including answering after one ring)
- Perform an initial heartbeat
- Log a successful startup message.

The structure chart for Modem Comm Form Load is depicted in Figure 232. A description of the routines called by Modem Comm Form Load is provided in Table 234.

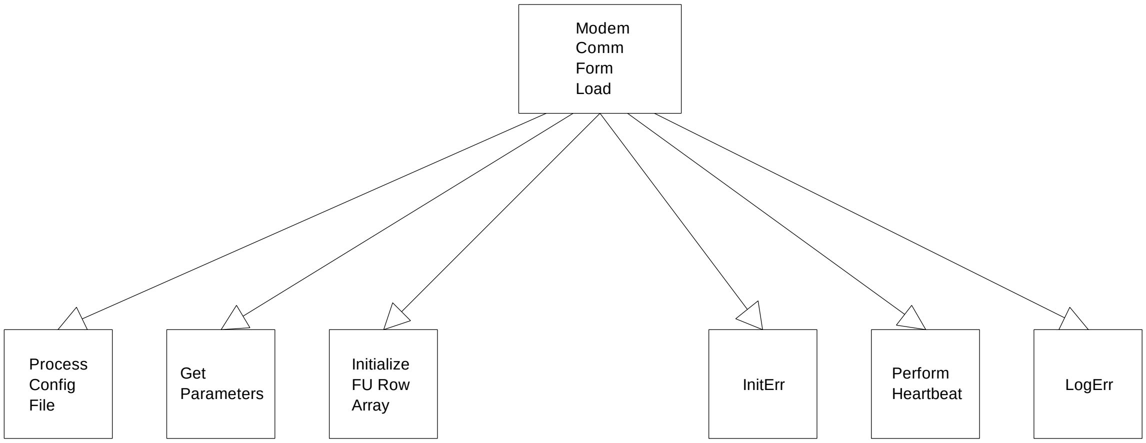

Figure 232 - Modem Comm Form Load Structure Chart

Function	Description
Get Parameters	The function retrieves the command line parameters passed from Error Server. The function first checks to see if the application is being run in local or remote mode. If the application is in local mode, the application id is defaulted to one. If the application is in remote mode, the application id is retrieved from the command line parameters using Command$ command. The returned command line is parsed. The application id is identified by a "-i" followed by the application id number. The id is stored in a global variable for use by the Heartbeat process. If the application id is not found, an error is logged.
InitErr	This function sets up the socket connection to the Error Server for error reporting. The remote host IP address and remote host port number are stored in a Winsock object to identify how Modem Communications will communicate with the Error Server. Next, the Connect method is used to establish the socket connection between the two programs. Initerr must wait until the connection is established before proceeding. The Winsock object Connect event will be triggered when the connection is successfully established. The local Winsock object is then assigned to a global variable for use by other routines.
Initialize FU Row Array	If the Field Unit Statistics do not exist, the field unit stats array is used to load the file. This array is initialized to default values by this routine. Each array element is loaded with the name of the record and a default count of zero.
LogErr	This function transmits the given message to the Error Server. The message is passed into the function as a parameter and a record delimiter is appended to the message. The message is then transmitted to the Error Server, using the Winsock object SendData method. Due to the unknown length of the message, the message is broken into ten byte segments and transmitted piecemeal until the entire message is sent.
Perform Heartbeat	This function is called by the heartbeat timer, when it is time to send a heartbeat message to the Error Server. The heartbeat message is built and a record delimiter is appended to the end of the record. The message is transmitted to the Error Server via the Winsock object SendData method.
Process Config File	This function opens the configuration file, reads in the configuration items into an array, and closed the file. The format of each line is a text identifier of the item followed by a colon and the value for that item. Each item is then loaded into the appropriate variable for use.

Table 234 - Routines called by Modem Comm Form Load

4.2.2.4.2 Heartbeat Timer

The Heartbeat Timer event is set to trigger every thirty seconds to transmit a heartbeat message to the error server The structure chart for Heartbeat Timer is depicted in Figure 233. A description of the routines called by Heartbeat Timer is provided in Table 235.

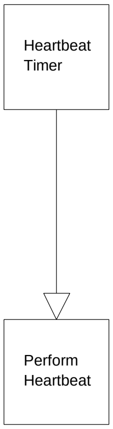

Figure 233 - Heartbeat Timer Structure Chart

Function	Description
Perform Heartbeat	This function is called by the heartbeat timer, when it is time to send a heartbeat message to the Error Server. The heartbeat message is built and a record delimiter is appended to the end of the record. The message is transmitted to the Error Server via the Winsock object SendData method.

Table 235 - Routines called by Heartbeat Timer

4.2.3 Build Translation Table (MapMatch)

The MapMatch application provides a GUI that allows the user to develop the Translation Table used by the Kiosk Field Units to make the conversion from Transguide Link Ids to NavTech LinkIds. The GUI provides two maps of the San Antonio area. One map is constructed of NavTech Links and the other map is constructed of Transguide Links. The user selects a Transguide Link on the Transguide map and then selects the associated NavTech Links from the NavTech map. MapMatch stores the association between the Transguide Link and the NavTech Links. The subsections below provide the design of MapMatch. Each subsection contains a structure chart and table describing the structure chart routines.

4.2.3.1 MapMatch

The MapMatch routine defines all environment variables and initiates the first draw of all four windows. The structure chart for MapMatch is depicted in Figure 234. A description of the routines called by MapMatch is provided in Table 236.

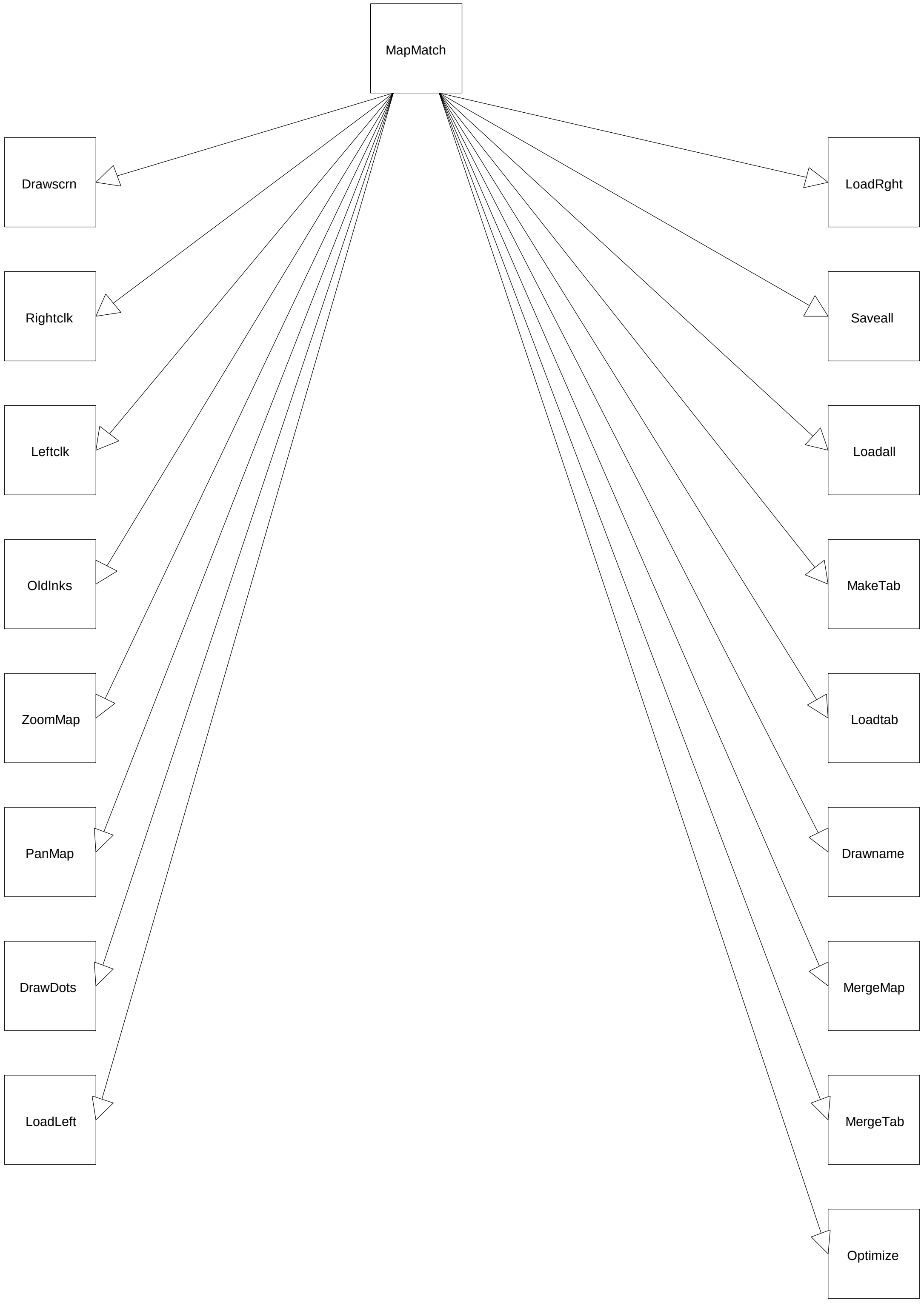

Figure 234 - MapMatch Structure Chart

Function	Description
DrawDots	Changes the endpoint display attribute of the slave and/or master maps and redraws the map windows.
Drawname	Changes the name display attribute of the slave and/or master maps and redraws the map windows.
Drawscrn	Draws all four windows and contents. Defines the callback functions for the controls in the Map Controls window as well as the map windows.
Leftclk	Changes the selected master map link as a response to a mouse click on master map. Highlights the selected master map link and removes highlighting from the previous master map link. Highlights the associated slave map links and removes highlighting from the previously associated slave map links. Updates the Link Status display.
Loadall	Loads a .mat file saved previously by SAVEALL. Redraws the all four windows.
LoadLeft	Uses LoadMap to load the master map data from a user specified file.
LoadRght	Uses LoadMap to load slave map data from a user specified file.
Loadtab	Loads associations between the master and slave map from a previously created map association look-up table.
MakeTab	Creates the look-up table of link associations.
MapMatch	Defines all environment variables and initiates the first draw of all four windows
MergeMap	Merges the contents of two ASCII map files discarding duplicate entries.
MergeTab	Creates the look-up table of link associations.
Oldlnks	Associates the currently selected master map link with the same slave map links as the previously selected master map link.
Optimize	Optimizes MatLab workspace memory using the MatLab "pack" function.
PanMap	Changes the map view by shifting the axes. Allows the user to pan left, right, up, and down.
Rightclk	Dassociates or disassociates the selected slave map link with the selected master map link as a response to a mouse click on slave map. The selected slave map link is either highlighted or unhighlighted depending on its current statue. Updates the Link Status window.
Saveall	Creates a binary .mat file that contains the values of all the current environment variables.
ZoomMap	Changes the map view by scaling the axes. Allows the user to zoom in, zoom out, zoom to show all map links, zoom to show all selected links, or zoom the slave map to the same view as the master map.

Table 236 - Routines called by MapMatch

4.2.3.2 DrawScrn

The DrawScrn routine draws all four windows and contents. Defines the callback functions for the controls in the Map Controls window as well as the map windows. The structure chart for DrawScrn is depicted in Figure 235. A description of the routines called by DrawScrn is provided in Table 237.

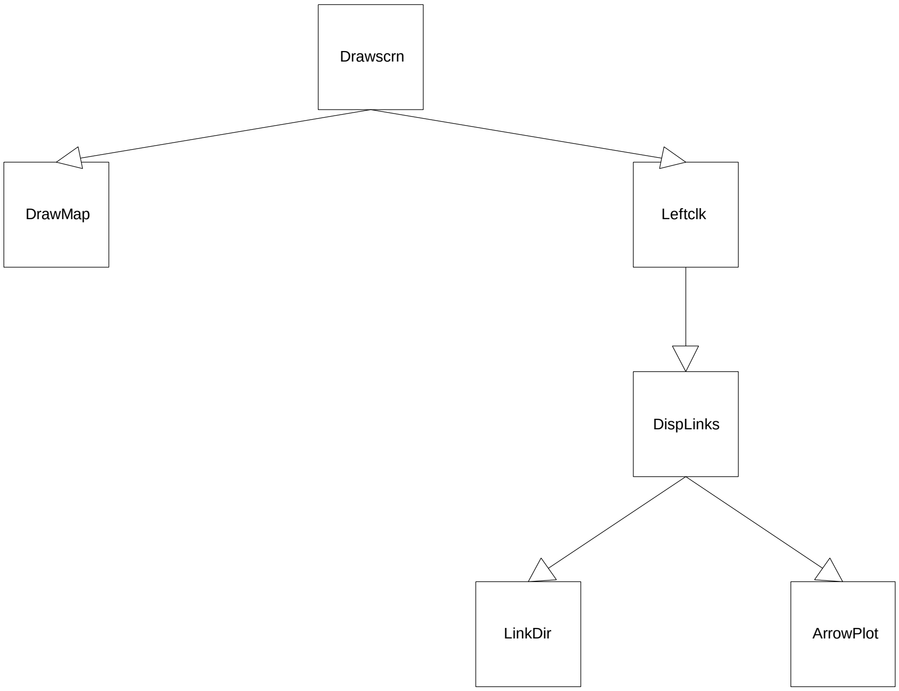

Figure 235 - DrawScrn Structure Chart

Function	Description
ArrowPlot	Draws a line with an arrowhead.
DispLinks	Draws all the contents of the Link Status window. Displays information on the selected links and generates a selected link vector plot.
DrawMap	Draws the contents of a map window and defines the callback functions that are executed as a response to a mouse click on a map segment.
Leftclk	Changes the selected master map link as a response to a mouse click on master map. Highlights the selected master map link and removes highlighting from the previous master map link. Highlights the associated slave map links and removes highlighting from the previously associated slave map links. Updates the Link Status display.
LinkDir	Computes the relative direction of a slave map link to a master map link.

Table 237 - Routines called by DrawScrn

4.2.3.3 RightClk

The RightClk routine associates or disassociates the selected slave map link with the selected master map link as a response to a mouse click on slave map. Highlights or removes highlighting from the selected slave map link. Updates the Link Status window. The structure chart for RightClk is depicted in Figure 236. A description of the routines called by RightClk is provided in Table 238.

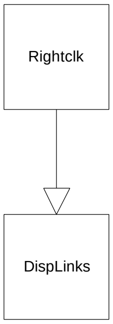

Figure 236 - RightClk Structure Chart

Function	Description
DispLinks	Draws all the contents of the Link Status window. Displays information on the selected links and generates a selected link vector plot.

Table 238 - Routines called by RightClk

4.2.3.4 Oldlnks

The Oldlnks routine associates the currently selected master map link with the same slave map links as the previously selected master map link. The structure chart for Oldlnks is depicted in Figure 237. A description of the routines called by Oldlnks is provided in Table 239.

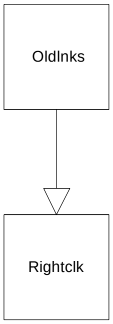

Figure 237 - Oldlnks Structure Chart

Function	Description
Rightclk	Associates or disassociates the selected slave map link with the selected master map link as a response to a mouse click on slave map. The selected slave map link is either highlighted or unhighlighted depending on its current state. Updates the Link Status window.

Table 239 - Routines called by Oldlnks

4.2.3.5 Drawdots

The Drawdots routine changes the endpoint display attribute of the slave and/or master maps and redraws the map windows. The structure chart for Drawdots is depicted in Figure 238. A description of the routines called by Drawdots is provided in Figure 238.

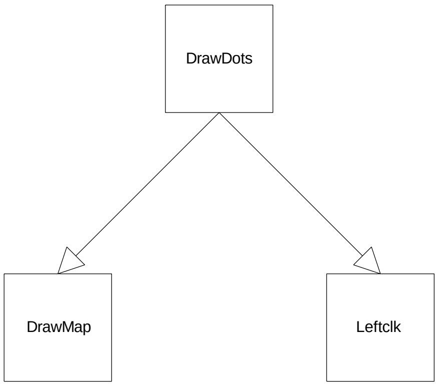

Figure 238 - Drawdots Structure Chart

Function	Description
DrawMap	Draws the contents of a map window and defines the callback functions that are executed as a response to a mouse click on a map segment.
Leftclk	Changes the selected master map link as a response to a mouse click on master map. Highlights the selected master map link and removes highlighting from the previous master map link. Highlights the associated slave map links and removes highlighting from the previously associated slave map links. Updates the Link Status display.

Table 240 - Routines called by Drawdots

4.2.3.6 LoadLeft

The LoadLeft routine uses LoadMap to load the master map data from a user specified file. The structure chart for LoadLeft is depicted in Figure 239. A description of the routines called by LoadLeft is provided in Table 241.

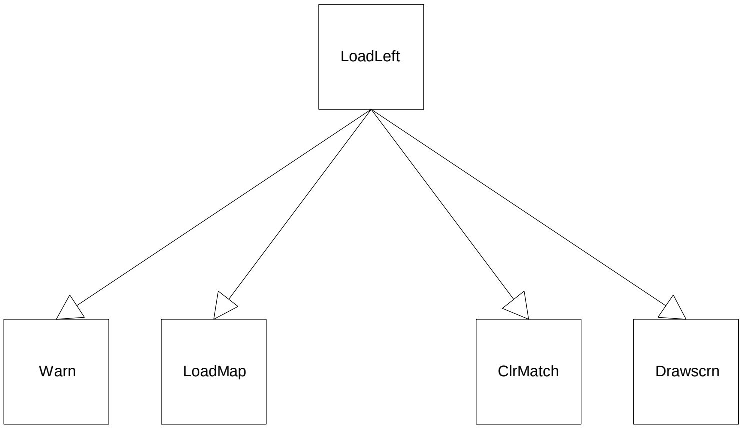

Figure 239 - LoadLeft Structure Chart

Function	Description
ClrMatch	Clears all link associations between the master and slave maps.
Drawscrn	Draws all four windows and contents. Defines the callback functions for the controls in the Map Controls window as well as the map windows.
LoadMap	Loads map data from an ASCII file.
Warn	Creates a warning message window.

Table 241 - Routines called by LoadLeft

4.2.3.7 Loadmap

The Loadmap routine loads map data from an ASCII file. The structure chart for Loadmap is depicted in Figure 240. A description of the routines called by Loadmap is provided in Table 242.

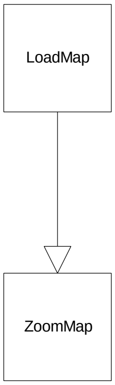

Figure 240 - Loadmap Structure Chart

Function	Description
ZoomMap	Changes the map view by scaling the axes. Allows the user to zoom in, zoom out, zoom to show all map links, zoom to show all selected links, or zoom the slave map to the same view as the master map.

Table 242 - Routines called by Loadmap

4.2.3.8 LoadRght

The LoadRght routine uses LoadMap to load slave map data from a user specified file. The structure chart for LoadRght is depicted in Figure 241. A description of the routines called by LoadRght is provided in Table 243.

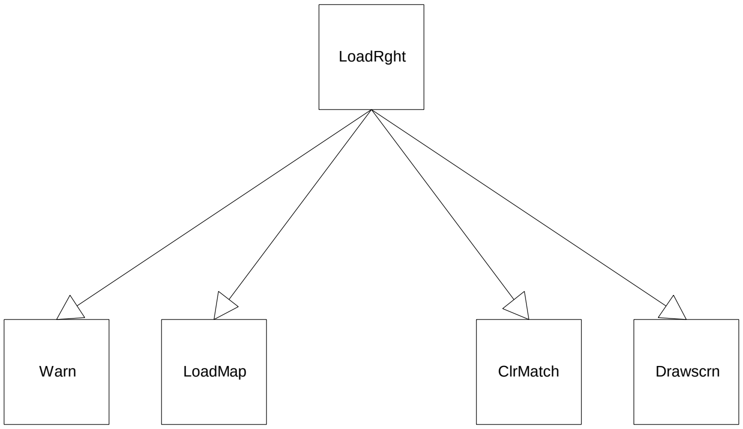

Figure 241 - LoadRght Structure Chart

Function	Description
ClrMatch	Clears all link associations between the master and slave maps.
Drawscrn	Draws all four windows and contents. Defines the callback functions for the controls in the Map Controls window as well as the map windows.
LoadMap	Loads map data from an ASCII file.
Warn	Creates a warning message window.

Table 243 - Routines called by LoadRght

4.2.3.9 Loadall

The Loadall routine loads a .mat file saved previously by SAVEALL. Redraws the all four windows. The structure chart for Loadall is depicted in Figure 242. A description of the routines called by Loadall is provided in Table 244.

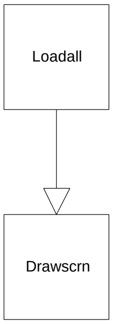

Figure 242 - Loadall Structure Chart

Function	Description
Drawscrn	Draws all four windows and contents. Defines the callback functions for the controls in the Map Controls window as well as the map windows.

Table 244 - Routines called by Loadall

4.2.3.10 MakeTab

The MakeTab routine creates the look-up table of link associations. The structure chart for MakeTab is depicted in Figure 243. A description of the routines called by MakeTab is provided in Table 245.

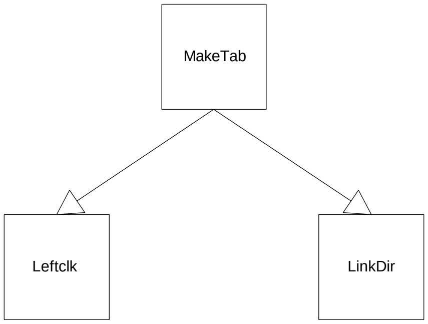

Figure 243 - MakeTab Structure Chart

Function	Description
Leftclk	Changes the selected master map link as a response to a mouse click on master map. Highlights the selected master map link and removes highlighting from the previous master map link. Highlights the associated slave map links and removes highlighting from the previously associated slave map links. Updates the Link Status display.
LinkDir	Computes the relative direction of a slave map link to a master map link.

Table 245 - Routines called by MakeTab

4.2.3.11 Loadtab

The Loadtab routine loads associations between the master and slave map from a previously created map association look-up table. The structure chart for Loadtab is depicted in Figure 244. A description of the routines called by Loadtab is provided in Table 246.

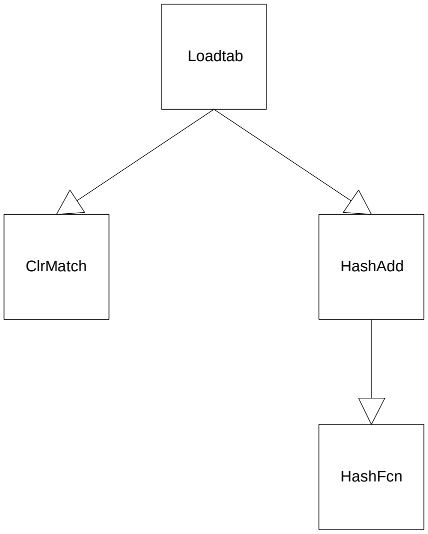

Figure 244 - Loadtab Structure Chart

Function	Description
ClrMatch	Clears all link associations between the master and slave maps.
HashAdd	Adds an entry into a hash table.
HashFcn	Computes an index to a hash table from an input string.

Table 246 - Routines called by Loadtab

4.2.3.12 Drawname

The Drawname routine draws the contents of a map window and defines the callback functions that are executed as a response to a mouse click on a map segment. The structure chart for Drawname is depicted in Figure 245. A description of the routines called by Drawname is provided in Table 247.

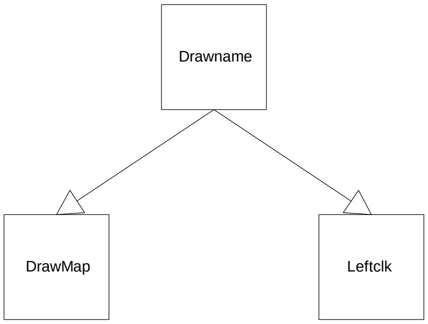

Figure 245 - Drawname Structure Chart

Function	Description
Drawname	Changes the name display attribute of the slave and/or master maps and redraws the map windows.
Leftclk	Changes the selected master map link as a response to a mouse click on master map. Highlights the selected master map link and removes highlighting from the previous master map link. Highlights the associated slave map links and removes highlighting from the previously associated slave map links. Updates the Link Status display.

Table 247 - Routines called by Drawname

4.2.3.13 MergeMap

The MergeMap routine merges the contents of two ASCII map files discarding duplicate entries. The structure chart for MergeMap is depicted in Figure 246. A description of the routines called by MergeMap is provided in Table 248.

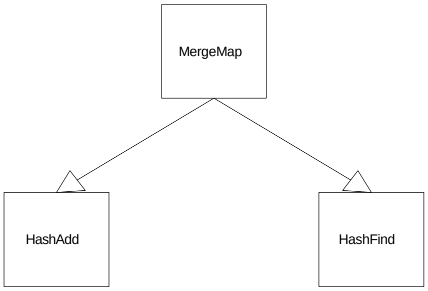

Figure 246 - MergeMap Structure Chart

Function	Description
HashAdd	Adds an entry into a hash table.
HashFind	Searches a hash table for an entry.

Table 248 - Routines called by MergeMap

4.2.3.14 Optimize

The Optimize routine optimizes MatLab workspace memory using the MatLab "pack" function. The structure chart for Optimize is depicted in Figure 247. A description of the routines called by Optimize is provided in Table 249.

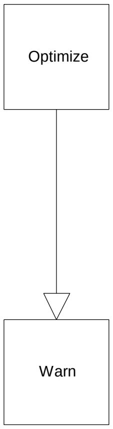

Figure 247 - Optimize Structure Chart

Function	Description
Warn	Creates a warning message window.

Table 249 - Routines called by Optimize

5. Traceability Matrix

The traceability for the Kiosk Traveler Information System is presented in this section. It lists the requirements of the system that were presented in Section 3 of this document. Along with each requirement is the source of the requirement, the design element it was assigned to, the level at which it will be tested, and the method that will be used to verify the requirement.

This table will be used throughout the design, development, and test of the system to ensure that the requirements have been met. It will continually be updated as requirements and design elements are refined. During development of the Acceptance Test Plan (ATP), sections of the test plan will be referenced in the *TEST LEVEL* column of this table to cross-reference to the ATP.

The columns of the traceability table have the following meanings:

- Requirement Number: A unique identifier (with embedded level) requirement is assigned
- Requirement: A brief description of the requirement
- Source: A paragraph reference in RFO, proposal, or other source for this requirement
- Design Element: The design element to which the requirement is allocated
- Test Case(s): The test cases from the Acceptance Test Plan that test the requirement

The Traceability Matrix begins on the following page.

Requirement Number	Requirement	Source	Des
KSK-PY-1	The Kiosk Master Computer shall be a Sun Microsystems Ultra SPARCStation with the following configuration: • 167 MHZ SPARC (RISC) CPU, • 4.2 Gigabyte hard disk, • 128 Megabytes RAM, • Floppy Disk, • CD-ROM, • Turbo GX+ Graphics, • 20 Inch color monitor, • 8 port modem server (SCSI) attached, • Dual Ethernet Interface, and • Dual SCSI Channels.	P-2.3.2.4.1	N/A
KSK-PY-2	The Indoor and Outdoor Kiosk Field Unit computers shall have, at a minimum, the following configuration: • Windows 95, • 120 MHz processor clock speed, • 32 MB RAM, • 1.6 GB hard disk drive, • 3.5 inch 1.44 MB floppy drive, • 8X CD-ROM drive, • 1 RS-232 asynchronous communication port, • 1 bi-directional parallel port, • 101 key enhanced keyboard, • 2 button mouse, and • an internal modem.	P-2.3.2.4.2	N/A

Requirement Number	Requirement	Source	Des
KSK-PY-4	The Indoor Kiosk shall include the following: • Antenna/receiver assembly, • Processor with keyboard, • Touch-screen monitor, • Speakers, • Printer, • Power strip, • Cooling fan, • UL & FCC certification, • Rated to operate at an ambient temperature range from 60 to 85 degrees Fahrenheit, • Rated to operate at a non-condensing humidity range from 35 to 80 percent relative humidity.	P-2.3.2.4.3	N/A
KSK-PY-5	The Outdoor Kiosk shall include the following: • Antenna/receiver assembly, • Processor with keyboard, • Touch-screen monitor, • Speakers, • Printer, • Modem, • Heating/cooling system, • UL & FCC certification, • Rated to operate at an ambient temperature range from -10 to 115 degrees Fahrenheit, • Rated to operate at a non-condensing humidity range from 20 to 100 percent relative humidity.	P-2.3.2.4.3	N/A

Requirement Number	Requirement	Source	Des
KSK-PY-6	The Indoor Kiosk enclosure shall be rated at the following environment specifications: • Ambient temperature range of 60 to 85 degrees Fahrenheit. • Non-condensing humidity range from 35 to 80 percent relative humidity.	P-2.3.2.2.3.1	N/A
KSK-PY-7	The Outdoor Kiosk enclosure shall be rated at the following environment specifications: • Ambient temperature range of –10 to 115 degrees Fahrenheit. • Non-condensing humidity range from 20 to 100 percent relative humidity.	P-2.3.2.2.3.2	N/A
KSK-IF-1	The Kiosk System shall interface with the Data Server.	P-2.3.1	4.2. 4.2.
KSK-IF-1.1a	The Kiosk System shall be capable of submitting the San Antonio area weather conditions to the Data Server.	P-2.3.1	4.2.
KSK-IF-1.1b	The Kiosk System shall be capable of submitting the San Antonio area weather forecast to the Data Server.	P-2.3.1	4.2.
KSK-IF-1.1c	The Kiosk System shall be capable of submitting the current San Antonio area radar map to the Data Server.	P-2.3.1	4.2.
KSK-IF-1.1d	The Kiosk System shall be capable of retrieving the San Antonio area weather conditions from the Data Server.	P-2.3.1	4.2. 4.2.
KSK-IF-1.1e	The Kiosk System shall be capable of retrieving the San Antonio area weather forecast from the Data Server.	P-2.3.1	4.2. 4.2.

Requirement Number	Requirement	Source	Des
KSK-IF-1.1f	The Kiosk System shall be capable of retrieving the current San Antonio area radar map from the Data Server.	P-2.3.1	4.2.
KSK-IF-1.2a	The Kiosk System shall be capable of submitting airline and airport terminal information to the Data Server.	P-2.3.1	4.2.
KSK-IF-1.2b	The Kiosk System shall be capable of submitting airport rental agency information to the Data Server.	P-2.3.1	4.2.
KSK-IF-1.2c	The Kiosk System shall be capable of submitting airport parking lot information to the Data Server.	P-2.3.1	4.2.
KSK-IF-1.2d	The Kiosk System shall be capable of retrieving airline and airport terminal information from the Data Server.	P-2.3.1	4.2.
KSK-IF-1.2e	The Kiosk System shall be capable of retrieving airport rental agency information from the Data Server.	P-2.3.1	4.2. 4.2.
KSK-IF-1.2f	The Kiosk System shall be capable of retrieving airport parking lot information from the Data Server.	P-2.3.1	4.2. 4.2.
KSK-IF-1.3a	The Kiosk System shall be capable of submitting route schedules to the Data Server.	P-2.3.1	4.2.
KSK-IF-1.3b	The Kiosk System shall be capable of submitting standard and special fares to the Data Server.	P-2.3.1	4.2.
KSK-IF-1.3c	The Kiosk System shall be capable of submitting park & ride locations to the Data Server.	P-2.3.1	4.2.

Requirement Number	Requirement	Source	Des
KSK-IF-1.3d	The Kiosk System shall be capable of submitting special bus events and the associated schedules to the Data Server.	P-2.3.1	4.2.
KSK-IF-1.3e	The Kiosk System shall be capable of submitting VIA handicapped bus dispatch (VIATrans) services to the Data Server.	P-2.3.1	4.2.
KSK-IF-1.3f	The Kiosk System shall be capable of submitting general VIA information to the Data Server.	P-2.3.1	4.2.
KSK-IF-1.3g	The Kiosk System shall be capable of submitting graphical displays of selected bus routes data to the Data Server.	P-2.3.1	4.2.
KSK-IF-1.3h	The Kiosk System shall be capable of retrieving route schedules from the Data Server.	P-2.3.1	4.2. 4.1.
KSK-IF-1.3i	The Kiosk System shall be capable of retrieving standard and special fares from the Data Server.	P-2.3.1	4.2. 4.1.
KSK-IF-1.3j	The Kiosk System shall be capable of retrieving park & ride locations from the Data Server.	P-2.3.1	4.2. 4.1.
KSK-IF-1.3k	The Kiosk System shall be capable of retrieving special bus events and the associated schedules from the Data Server.	P-2.3.1	4.2. 4.1.
KSK-IF-1.3l	The Kiosk System shall be capable of retrieving VIA handicapped bus dispatch (VIATrans) services from the Data Server.	P-2.3.1	4.2. 4.1.
KSK-IF-1.3m	The Kiosk System shall be capable of retrieving general VIA information from the Data Server.	P-2.3.1	4.2. 4.1.

Requirement Number	Requirement	Source	Des
KSK-IF-1.3n	The Kiosk System shall be capable of retrieving displays of selected bus routes data from the Data Server.	P-2.3.1	4.2. 4.1.
KSK-IF-2	The Kiosk System shall interface with the In-Vehicle Navigation system data stream being transmitted utilizing the STIC communication system for real-time traffic conditions data.	P-2.3.2.2.6	4.2.
KSK-IF-2.1	The Kiosk Field Unit shall receive the real-time traffic condition data broadcast from the STIC communication network.	P-2.3.2.2.9	4.2.
KSK-IF-3	The Kiosk System shall interface with the weather data source.	P-2.3.2.2.4	4.2.
KSK-IF-3.1a	The Kiosk System shall be capable of retrieving the San Antonio area weather conditions from the weather data source.	P-2.3.2.2.7	4.2.
KSK-IF-3.1b	The Kiosk System shall be capable of retrieving the San Antonio area weather forecast from the weather data source.	P-2.3.2.2.7	4.2.
KSK-IF-3.1c	The Kiosk System shall be capable of retrieving the current San Antonio area radar map data from the weather data source.	P-2.3.2.2.7	4.2.
KSK-IF-4	The Kiosk System shall interface with the airport data source.	P-2.3.2.2.4	4.2.
KSK-IF-4.1	The Kiosk Master Computer shall be capable of receiving airport terminal, airport rental agency, and airport parking lot data from the airport data source.	P-2.3.2.2.7	4.2. 4.2.
KSK-IF-5	The Kiosk System shall interface with the VIA data source.	P-2.3.2.2.4	4.2.

Requirement Number	**Requirement**	**Source**	**Des**
KSK-IF-5.1a	The Kiosk Master Computer shall be capable of receiving route schedules from the VIA data source.	P-2.3.2.2.7	4.2.
KSK-IF-5.1b	The Kiosk Master Computer shall be capable of receiving standard and special fares from the VIA data source.	P-2.3.2.2.7	4.2.
KSK-IF-5.1c	The Kiosk Master Computer shall be capable of receiving park & ride locations from the VIA data source.	P-2.3.2.2.7	4.2.
KSK-IF-5.1d	The Kiosk Master Computer shall be capable of receiving special bus events and the associated schedules from the VIA data source.	P-2.3.2.2.7	4.2.
KSK-IF-5.1e	The Kiosk Master Computer shall be capable of receiving VIA handicapped bus dispatch (VIATrans) services from the VIA data source.	P-2.3.2.2.7	4.2.
KSK-IF-5.1f	The Kiosk Master Computer shall be capable of receiving general VIA information from the VIA data source.	P-2.3.2.2.7	4.2.
KSK-IF-5.1g	The Kiosk Master Computer shall be capable of receiving graphical displays of selected bus routes from the VIA data source.	P-2.3.2.2.7	4.2.
KSK-IF-6	The Kiosk System shall interface with screen saver data source(s).	P-2.3.2.2.4	4.2.
KSK-IF-6.1	The Kiosk Master Computer shall be capable of receiving screen saver files.	P-2.3.2.2.7	4.2. 4.2.
KSK-IF-7	The Kiosk System shall interface with the Kiosk Field Units.	P-2.3.1	4.2.

Requirement Number	Requirement	Source	Des
KSK-IF-7.1a	The Kiosk Master Computer shall be capable of transmitting the San Antonio area weather conditions to the Kiosk Field Units.	P-2.3.2.2.7	4.2.
KSK-IF-7.1b	The Kiosk Master Computer shall be capable of transmitting the San Antonio area weather forecast to the Kiosk Field Units.	P-2.3.2.2.7	4.2.
KSK-IF-7.1c	The Kiosk Master Computer shall be capable of transmitting the current San Antonio area radar map data to the Kiosk Field Units.	P-2.3.2.2.7	4.2.
KSK-IF-7.2a	The Kiosk Master Computer shall be capable of transmitting airport terminal data to the Kiosk Field Units.	P-2.3.2.2.7	4.2.
KSK-IF-7.2b	The Kiosk Master Computer shall be capable of transmitting airport rental agency data to the Kiosk Field Units.	P-2.3.2.2.7	4.2.
KSK-IF-7.2c	The Kiosk Master Computer shall be capable of transmitting airport parking lot data to the Kiosk Field Units.	P-2.3.2.2.7	4.2.
KSK-IF-7.3a	The Kiosk Master Computer shall be capable of transmitting route schedules to the Kiosk Field Units.	P-2.3.2.2.7	4.2.
KSK-IF-7.3b	The Kiosk Master Computer shall be capable of transmitting standard and special fares, park & ride locations to the Kiosk Field Units.	P-2.3.2.2.7	4.2.
KSK-IF-7.3c	The Kiosk Master Computer shall be capable of transmitting special bus events and the associated schedules to the Kiosk Field Units.	P-2.3.2.2.7	4.2.
KSK-IF-7.3d	The Kiosk Master Computer shall be capable of transmitting VIA handicapped bus dispatch (VIATrans) services to the Kiosk Field Units.	P-2.3.2.2.7	4.2.

Requirement Number	**Requirement**	**Source**	**Des**
KSK-IF-7.3e	The Kiosk Master Computer shall be capable of transmitting general VIA information to the Kiosk Field Units.	P-2.3.2.2.7	4.2.
KSK-IF-7.3f	The Kiosk Master Computer shall be capable of transmitting graphical displays of selected bus routes data to the Kiosk Field Units.	P-2.3.2.2.7	4.2.
KSK-IF-7.3g	The Kiosk Master Computer shall be capable of transmitting park & ride locations to the Kiosk Field Units.	P-2.3.2.2.7	4.2.
KSK-IF-7.4	The Kiosk Master Computer shall be capable of transmitting screen saver files to the Kiosk Field Units.	P-2.3.2.2.7	4.1.
KSK-IF-8	The Kiosk System shall interface with the general public through a touchscreen, using a Graphical User Interface.	P-2.3.1	4.2.
KSK-IF-8.1	The Kiosk Field Unit shall provide touchscreen interaction for users to interface with the Map Display.	P-2.3.2.2.8	4.2.
KSK-IF-8.2	The Kiosk Field Unit shall provide touchscreen interaction for users to interface with the Transit Display.	P-2.3.2.2.8	4.2.
KSK-IF-8.3	The Kiosk Field Unit shall provide touchscreen interaction for users to interface with the Airport Display.	P-2.3.2.2.8	4.2.
KSK-IF-8.4	The Kiosk Field Unit shall provide touchscreen interaction for users to interface with the Weather Display.	R-27.1.5	4.2.

Requirement Number	Requirement	Source	Des
KSK-IF-8.5	The Kiosk Field Unit shall provide touchscreen interaction for users to interface with the Route Guidance Display.	P-2.3.2.2.8	4.2.
KSK-FN-1	The Kiosk System shall display the real-time traffic conditions of the highways/roadways monitored by TransGuide.	P-2.3.2.2.8	4.2.
KSK-FN-1.2	The Kiosk Field Unit shall be capable of displaying real-time traffic data using a San Antonio Map Display.	P-2.3.2.2.8	4.2.
KSK-FN-1.3	The Kiosk Field Unit map shall display traffic conditions using color-coding.	P-2.3.2.2.8 R-27.1.2	4.2.
KSK-FN-1.4	The Kiosk Field Unit map shall display incidents and lane closures utilizing icons.	P-2.3.2.2.8	4.2.
KSK-FN-1.5	The Kiosk Field Unit shall provide additional information about an incident or lane closure when the respective icon is touched.	P-2.3.2.2.8	4.2.
KSK-FN-1.7	The Kiosk Field Unit map shall display current airport traffic conditions for instrumented sections of highway around the San Antonio International Airport.	P-2.3.2.2.8	4.2.
KSK-FN-1.8	The Kiosk Field Unit map shall identify city streets, residential streets, and highways.	P-2.3.2.2.8	4.2.
KSK-FN-1.9	The Kiosk Field Unit map shall have the capability to zoom in and out of the San Antonio Street Map Display utilizing touch screen input.	P-2.3.2.2.8	4.2.
KSK-FN-1.10	The Kiosk Field Unit map shall have the capability to pan the San Antonio Street Map Display utilizing touch screen input.	P-2.3.2.2.8	4.2.

Requirement Number	Requirement	Source	Des
KSK-FN-1.11	The Kiosk Field Unit map shall display icons indicating locations of automated teller machines (ATMs), shopping centers, restaurants, gas stations, tourist attractions, hospitals, schools, parks, airports, and bus stops.	P-2.3.2.2.8	4.2.
KSK-FN-1.12	The Kiosk Field Unit San Antonio Street Map Display software shall integrate data from the Navigation Technologies San Antonio Region database with real-time data from the Data Server.	P-2.3.2.2.8	4.2.
KSK-FN-1.13	The Kiosk Field Unit map real-time traffic conditions shall be updated at least every five (5) minutes.	R-27.3.3	4.2.
KSK-FN-2	The Kiosk System shall display weather data.	P-2.3.2.2.8	4.2.
KSK-FN-2.1	The Kiosk Field Unit shall display the current San Antonio weather conditions.	P-2.3.2.2.8	4.2.
KSK-FN-2.2	The Kiosk Field Unit shall display the local San Antonio forecast.	P-2.3.2.2.8	4.2.
KSK-FN-2.3	The Kiosk Field Unit shall display a San Antonio area radar map.	P-2.3.2.2.8	4.2.
KSK-FN-2.4	The Kiosk Field Unit current weather conditions shall be updated when updates are provided by the weather data source.	P-2.3.2.2.8	4.2. 4.2. 4.2.
KSK-FN-2.5	The Kiosk Field Unit San Antonio area radar map shall be updated when updates are provided by the weather data source.	P-2.3.2.2.8	4.2. 4.2. 4.2.
KSK-FN-2.6	The Kiosk Field Unit local San Antonio forecast shall be updated when updates are provided by the weather data source.	P-2.3.2.2.8	4.2. 4.2. 4.2.

Requirement Number	Requirement	Source	Des
KSK-FN-3	The Kiosk System shall display airport data.	P-2.3.2.2.8	4.2.
KSK-FN-3.1	The Kiosk Field Unit shall display the traffic conditions for the sections of instrumented highway that surround the airport.	R-27.3.3	4.2.
KSK-FN-3.2	The Kiosk Field Unit shall display a listing of local airline names, their phone numbers and the terminal in which they are located.	P-2.3.2.2.8	4.2.
KSK-FN-3.3	The Kiosk Field Unit shall display a listing of local rental car agencies and their phone numbers located at the San Antonio International Airport.	P-2.3.2.2.8	4.2.
KSK-FN-3.4	The Kiosk Field Unit shall display a listing of the location and cost of airport parking lots.	P-2.3.2.2.8	4.2.
KSK-FN-4	The Kiosk System shall display VIA data.	P-2.3.2.2.8	4.2.
KSK-FN-4.1	The Kiosk Field Unit shall display route schedules and graphical displays of the routes that are available.	P-2.3.2.2.8	4.2.
KSK-FN-4.2	The Kiosk Field Unit shall provide scheduled times for major bus stops on a selected route.	P-2.3.2.2.8	4.2.
KSK-FN-4.3	The Kiosk Field Unit shall display a description of standard and special fares.	P-2.3.2.2.8	4.2.
KSK-FN-4.4	The Kiosk Field Unit shall display a description of park & ride locations.	P-2.3.2.2.8	4.2.
KSK-FN-4.5	The Kiosk Field Unit shall display a description of special bus events and the associated schedules.	P-2.3.2.2.8	4.2.

Requirement Number	**Requirement**	**Source**	**Des**
KSK-FN-4.6	The Kiosk Field Unit shall display information about VIA handicapped bus dispatch (VIATrans) services.	P-2.3.2.2.8	4.2.
KSK-FN-4.7	The Kiosk Field Unit shall display general VIA information.	VIA	4.2.
KSK-FN-5	The Kiosk System shall display screen saver (advertisements) files when the Kiosk is not being accessed by a user.	P-2.3.2.2.8	4.2.
KSK-FN-5.1a	The Kiosk Master Computer shall accept bitmap (.bmp) files for the displaying of graphical displays on the Kiosk Field Unit.	R-27.3.2	4.2.
KSK-FN-5.1b	The Kiosk Master Computer shall accept wave (.wav) files for the playing of audio files on the Kiosk Field Unit.	R-27.3.2	4.2.
KSK-FN-5.1c	The Kiosk Master Computer shall accept audio video interleaved (.avi) files for playing video clips on the Kiosk Field Unit.	R-27.3.2	4.2.
KSK-FN-5.2	The Kiosk Field Units shall be capable of receiving screen saver files from the Master Computer and updating the existing screen saver.	P-2.3.1	4.2.
KSK-FN-5.3	The Kiosk Field Units shall be capable of executing the screen saver.	P-2.3.1	4.2.
KSK-FN-6	The Kiosk System shall provide system diagnostics.	P-2.3.2.2.8	4.2.
KSK-FN-6.1	A Kiosk Master Computer Diagnostic Status GUI shall be implemented that displays the last known status of the Kiosk Field Units.	P-2.3.2.2.7	4.2.
KSK-FN-6.2	The Kiosk Master Computer shall automatically interrogate the Kiosk Field Units.	P-2.3.2.2.7	4.2.

Requirement Number	**Requirement**	**Source**	**Des**
KSK-FN-6.3	The Kiosk Master Computer shall provide the capability to manually interrogate individual Kiosk Field Units.	P-2.3.2.2.7	4.2.
KSK-FN-6.4	The Kiosk Master Computer shall store the interrogation status results.	P-2.3.2.2.7	4.2.
KSK-FN-6.6	The Kiosk Master Computer shall have the capability to download data and screen saver files.	P-2.3.2.2.7	4.2.
KSK-FN-6.7	The Kiosk Master Computer shall upload Kiosk Field Unit usage statistics.	P-2.3.2.2.7	4.2.
KSK-FN-6.7a	The Kiosk Master Computer shall upload Kiosk Field Unit statistics on the number of times the Kiosk is used.	P-2.3.2.2.7	4.2.
KSK-FN-6.7b	The Kiosk Master Computer shall upload Kiosk Field Unit statistics on the number of times the San Antonio Map is accessed.	P-2.3.2.2.7	4.2.
KSK-FN-6.7c	The Kiosk Master Computer shall upload Kiosk Field Unit statistics on the number of times Airport information is accessed.	P-2.3.2.2.7	4.2.
KSK-FN-6.7d	The Kiosk Master Computer shall upload Kiosk Field Unit statistics on the number of times Weather information is accessed.	P-2.3.2.2.7	4.2.
KSK-FN-6.7e	The Kiosk Master Computer shall upload Kiosk Field Unit statistics on the number of times VIA Transit information is accessed.	P-2.3.2.2.7	4.2.
KSK-FN-6.7f	The Kiosk Master Computer shall upload Kiosk Field Unit statistics on the number of times Route Guidance is accessed.	P-2.3.2.2.7	4.2.

Requirement Number	Requirement	Source	Des
KSK-FN-6.9	The Kiosk Field Unit shall be capable of reporting status to the Kiosk Master Computer.	P-2.3.2.2.8	4.2.
KSK-FN-6.10	The Kiosk Field Unit diagnostic software shall accept non-real-time file updates from the Kiosk Master Computer.	P-2.3.2.2.8	4.2.
KSK-FN-6.12	The Kiosk Field Unit shall keep usage statistics.	P-2.3.2.2.8	4.2.
KSK-FN-6.12a	The Kiosk Field Unit shall keep statistics on the number of times the Kiosk is used.	P-2.3.2.2.8	4.2.
KSK-FN-6.12b	The Kiosk Field Unit shall keep statistics on the number of times the San Antonio Map is accessed.	P-2.3.2.2.8	4.2.
KSK-FN-6.12c	The Kiosk Field Unit shall keep statistics on the number of times Airport information is accessed.	P-2.3.2.2.8	4.2.
KSK-FN-6.12d	The Kiosk Field Unit shall keep statistics on the number of times Weather information is accessed.	P-2.3.2.2.8	4.2.
KSK-FN-6.12e	The Kiosk Field Unit shall keep statistics on the number of times VIA Transit information is accessed.	P-2.3.2.2.8	4.2.
KSK-FN-6.12f	The Kiosk Field Unit shall keep statistics on the number of times Route Guidance information is accessed.	P-2.3.2.2.8	4.2.
KSK-FN-7	The Kiosk System shall provide route guidance.	P-2.3.2.2.8	4.2. 4.2.

Requirement Number	**Requirement**	**Source**	**Des**
KSK-FN-7.1	The Kiosk Field Units shall convert the real-time traffic condition data stream into data that can be interpreted by the Navigation Technologies database and the Route Guidance application.	R-27.3.3	4.2.
KSK-FN-7.2	The Kiosk Field Unit shall be capable of displaying route guidance using the Navigation Technologies database.	R-27.3.3	4.2.
KSK-FN-7.3	The Kiosk Field Unit shall provide a graphical display of the route from the kiosk's location to the selected destination.	P-2.3.2.2.8	4.2.
KSK-FN-7.4	The Kiosk Field Unit shall allow the user to select a route from the Kiosk Field Unit's location to a selected Point of Interest.	P-2.3.2.2.8	4.2.
KSK-FN-7.5	The Kiosk Field Unit shall allow the user to select their destination from a list of the points of interest retrieved from the Navigation Technologies database.	P-2.3.2.2.8	4.2.
KSK-FN-7.6	The Kiosk Field Unit shall allow the user to enter the address of the destination.	P-2.3.2.2.8	4.2.
KSK-FN-7.7	The Kiosk Field Unit shall utilize a color-coded line segment on the San Antonio Street Map to indicate the calculated route.	P-2.3.2.2.8	4.2.
KSK-FN-7.8	The Kiosk Field Unit shall utilize real-time speed information to calculate travel time to the selected destination.	P-2.3.2.2.8	4.2.
KSK-FN-7.9	The Kiosk Field Unit shall display the estimated travel time and speed for the selected route.	P-2.3.2.2.8	4.2.

Requirement Number	Requirement	Source	Des
KSK-FN-7.10	The Kiosk Field Unit shall display turn-by-turn instructions for a calculated route.	P-2.3.2.2.8	4.2.
KSK-FN-8	The Kiosk System shall be capable of printing user selected items.	P-2.3.2.2.8	4.2.
KSK-FN-8.2	The Kiosk Field Unit shall be capable of printing the route map and instructions.	P-2.3.2.2.8	4.2.
KSK-FN-8.3	The Kiosk Field Unit shall be capable of printing the transit information.	P-2.3.2.4.3	4.2. 4.2.
KSK-FN-8.4	The Kiosk Field Unit shall be capable of printing the airport information.	P-2.3.2.4.2	4.2. 4.2. 4.2.
KSK-FN-8.5	The Kiosk Field Unit shall be capable of printing the local weather conditions, the local forecast and the radar map.	P-2.3.2.4.2	4.2. 4.2. 4.2.
KSK-FN-9	The Kiosk Field Unit shall provide help to assist the user in the operation of the Kiosk application.	P-2.3.2.2.8	4.2.

Requirement Number	Requirement	Source	Des
KSK-FN-9.1	The Kiosk Field Unit shall provide Help buttons to provide information on how to use the GUI currently displayed.	P-2.3.2.2.8	4.2.
KSK-FN-10	The Kiosk System shall provide monitoring and restarting of its processes.	P-2.3.2.2.7	4.2.
KSK-FN-10.2	The Master Computer subsystem shall provide monitoring and restarting of its applications.	P-2.3.2.2.7	4.2.
KSK-FN-10.3	The Kiosk Field Unit subsystem unattended applications shall automatically startup at boot-up.	P-2.3.2.2.7	N/A
KSK-FN-10.4	The Kiosk Field Unit subsystem shall provide monitoring and restarting of its applications.	P-2.3.2.2.7	4.2.