

THE CRAFT SOLUTION

A custom designed solution for the *National Sea Rescue Institute*

System Specification

System Specification 1999-06-13-008



Designed as part of the Systems Development Project
of the UCT Full Time Honours in Information Systems Programme
14 June 1999

- [Analysis & Design by Object Sync] -

[*Daniel Chalef*] [*Jason du Preez*] [*Khetan Gajjar*]

[*David Reinhardt*] [*Eden Ridgway*]

- [<http://craft.os.org.za>] -



EXECUTIVE SUMMARY

The CRAFT Solution is a modular extranet knowledge management system. At its core, CRAFT has modules for various functions. CRAFT clients (i.e. the stations, regional head offices and head quarters), making use of CRAFT/client, will communicate with CRAFT/serv via the Internet.

In this document, the specific modules of CRAFT/client and CRAFT/serv are detailed. The results of the process analysis, data analysis and modelling and functional analysis are discussed, and from them, the coding standards to be used throughout the project are formally noted.

The CRAFT/com technical details, including the XML vocabulary, data volume projections, and specific coding standards are explicitly described. The output of the modelling conducted thus far is included in this document, and will be ratified by the NSRI.

This document is to be used in conjunction with the User Requirements Specification, which has been accepted by the NSRI.

TERMS OF REFERENCE

This system specification document has been commissioned as part of the Information System Honours (Full Time) development project. The intent of this document is to detail the technical elements of the CRAFT solution being proposed by the Object Sync team.

This document is to be presented to the Department of Information Systems (University of Cape Town), for academic review in June 1999. Any suggestions made will be deliberated by Object Sync in conjunction with the NSRI, and, where necessary, amendments will be made.

The amended document will then be formally presented to the NSRI. The NSRI will have the opportunity to discuss and query the specific technical details of the CRAFT Solution with Object Sync. Eventually, a final revision will be presented to the NSRI. Once this revision is accepted by all parties, it will be signed. Any amendments thereafter will follow the configuration change procedure, detailed in the *User Requirement Specification*.

TABLE OF CONTENTS

| | |
|--|-----------|
| EXECUTIVE SUMMARY | 2 |
| TERMS OF REFERENCE | 3 |
| TABLE OF CONTENTS..... | 4 |
| TABLES..... | 6 |
| CODE STUBS | 6 |
| THE STATUS OF THIS DOCUMENT | 7 |
| LOGICAL STRUCTURE OF CRAFT | 8 |
| CRAFT/CLIENT..... | 8 |
| <i>CRAFT/ops</i> | 8 |
| <i>CRAFT/gyp</i> | 8 |
| <i>CRAFT/ves</i> | 9 |
| <i>CRAFT/log</i> | 9 |
| <i>CRAFT/spon</i> | 9 |
| <i>CRAFT/man</i> | 9 |
| <i>CRAFT/adm</i> | 10 |
| CRAFT/TECH..... | 10 |
| <i>CRAFT/db</i> | 10 |
| <i>CRAFT/com</i> | 11 |
| <i>CRAFT/serv</i> | 11 |
| CRAFT/DOC | 11 |
| ANALYSES | 12 |
| PROCESS ANALYSIS | 12 |
| DATA ANALYSIS & MODELING | 12 |
| FUNCTIONAL ANALYSIS..... | 13 |
| TECHNICAL ARCHITECTURE | 15 |
| CRAFT/SERV | 15 |
| CRAFT/CLIENT..... | 15 |
| HARDWARE SPECIFICATIONS..... | 16 |
| CRAFT/SERV | 16 |
| CRAFT/CLIENT..... | 17 |
| CRAFT/CLIENT INTERFACE..... | 18 |
| USER INTERFACE LOOK AND FEEL | 18 |
| INPUT DESIGN | 18 |
| MENU STRUCTURE..... | 19 |
| OUTPUT DESIGN | 21 |
| SECURITY MODEL | 23 |
| CRAFT/CLIENT..... | 23 |
| <i>System Initialisation</i> | 23 |
| <i>System Recovery</i> | 23 |
| <i>Access Levels</i> | 24 |
| CRAFT/SERV | 25 |
| CRAFT/COM | 27 |

| | |
|---|-----------|
| CRAFT/COM FUNCTIONALITY AND OPERATION | 27 |
| <i>CRAFT/serv prompts</i> | 28 |
| XML VOCABULARY | 29 |
| <i>Initial Handshake and Authentication</i> | 30 |
| <i>Client Upload</i> | 32 |
| <i>Client Download</i> | 35 |
| CRAFT/SERV VOLUME PROJECTIONS..... | 42 |
| <i>Load and Transaction Volumes</i> | 42 |
| <i>Traffic Volumes</i> | 42 |
| CRAFT/CLIENT CODING STANDARDS..... | 44 |
| OBJECT NAMING CONVENTIONS | 44 |
| SUGGESTED PREFIXES FOR DATA ACCESS OBJECTS..... | 44 |
| PREFIXES FOR MENUS | 44 |
| PREFIXES FOR OTHER CONTROLS | 45 |
| STRUCTURED CODING CONVENTIONS | 45 |
| <i>Code Commenting Conventions</i> | 45 |
| <i>Code Formatting</i> | 46 |
| <i>Grouping Constants</i> | 47 |
| <i>Creating Strings for MsgBox, InputBox, and SQL Queries</i> | 47 |
| CONSTANT AND VARIABLE NAMING CONVENTIONS..... | 48 |
| <i>Variable Scope Prefixes</i> | 49 |
| <i>Constants</i> | 49 |
| <i>Variables</i> | 50 |
| <i>Variable Data Types</i> | 50 |
| <i>Descriptive Variable and Procedure Names</i> | 51 |
| <i>User-Defined Types</i> | 51 |
| CRAFT/SERV CODING STANDARDS | 52 |
| OBJECT NAMING CONVENTIONS | 52 |
| CODING METHOD | 52 |
| STRUCTURED CRAFT/SERV CODING CONVENTIONS | 52 |
| <i>Code Commenting Conventions</i> | 52 |
| <i>Revision Control</i> | 53 |
| <i>Code Formatting</i> | 53 |
| CONSTANT AND VARIABLE NAMING CONVENTIONS..... | 54 |
| DESCRIPTIVE VARIABLE AND PROCEDURE NAMES | 54 |
| Y2K COMPLIANCE..... | 55 |
| THE DEVELOPMENT CYCLE..... | 55 |
| CODING STANDARDS | 55 |
| APPENDICES..... | 56 |
| TECHNICAL SPECIFICATION AMENDMENT | 57 |
| PROCESS MODELS | 58 |
| THE DATA MODELS | 59 |
| FUNCTIONAL MODEL | 60 |
| TECHNICAL ENVIRONMENT MODEL..... | 61 |
| SCREENSHOT..... | 62 |
| REPORT MOCKUPS | 63 |
| CRAFT/CLIENT CODING STANDARDS | 64 |

CLASS MODELING..... 65
CRAFT/SERV FUNCTION MAP..... 66

Tables

Table 1 - Functions of Each Module of CRAFT/client..... 14
Table 2 – CRAFT/client Menu Structure..... 21
Table 3 – Data Transfer 43
Table 4 – Menu Control Prefix Examples 45
Table 5 – Section Headings in VB Code 46
Table 6 – Scope of Variables in Visual Basic..... 48
Table 7 – Variable Prefixes to Determine Variable Types 49
Table 8 – Variable Prefixes to Indicate Data Types 51
Table 9 – CRAFT/serv Procedure Header Commenting 53
Table 10 – Columns in the Data Dictionary 59

Code Stubs

Code Stub 1 – Code Formatting..... 47
Code Stub 2 – Underscore Line Continuation 48
Code Stub 3 – Constant Naming 50

THE STATUS OF THIS DOCUMENT

This document is a working document; it will be continuously updated and refined. During the course of development, specific technical elements of the chosen tools will become known to the team, possibly requiring additional tools and/or technique usage. These techniques and/or tools will be detailed in the final version of this document, constituting part of CRAFT/doc.

Configuration management techniques will again be employed to manage this process. Whenever material changes are made, a configuration management form will be completed. An example of this form is included in the appendices to this document.

The *Technical Specification Amendment* form differs from the *Configuration Change Acceptance* form found in the *User Requirement Specification*. The NSRI does not consider the specific technical elements of the solution critical and has entrusted the development team to take superficial technical decisions. Conceptual changes to the CRAFT solution, at a technical level, will be monitored using the *Configuration Change Acceptance* form. Specific technical amendments will be monitored using the internal *Technical Specification Amendment* form.

LOGICAL STRUCTURE OF CRAFT

This section details how the CRAFT Solution is structured. It should be noted that although much of this information can be found in the *User Requirement Specification*, it was necessary to include it here such that the specific elements of each model are noted and are not overlooked when building the system, or reviewing the final product.

CRAFT/client

CRAFT/client is the software package used by the end users at the NSRI. The application will appear to be a “traditional” Windows application. CRAFT/client will use web browser technology to render HTML pages, that are a functional component of the user interface.

Within CRAFT/client, there are various modules. Each module provides access to the various functional areas of the solution.

CRAFT/ops

This is the operations reporting (ops reports) module. Ops reports are completed off-line, using the desktop application. When a user submits the form, it is entered into a queue of commands that will be uploaded to the CRAFT server. This is done whenever the CRAFT/client logs on to the Internet.

Local clients store a year’s worth of historical data on their machines. Historical data can be viewed off-line.

When filing an ops report, CRAFT/ops will provide users with the opportunity to specify the nature of the operation.

CRAFT/gyp

This is the GYPSY book module. It is an automated, online version of the existing “telephone directory” of all numbers pertinent to each station. This includes standard numbers (e.g. station commander, deputy station commander, station committee, etc.) as well as some numbers that may be unique to a given station (e.g. local airforce base, port captain etc).

Stations can modify information relevant to their entry in the GYPSY book. These changes will be queued and uploaded to the CRAFT server. When other stations log on, the CRAFT server will download the updated GYPSY book section to them. In this way, GYPSY book changes are propagated quickly and efficiently.

Each station should still maintain a hardcopy of the GYPSY book, should the electronic copy be unavailable (e.g. during a power failure).

CRAFT/ves

This module is designed primarily to monitor expenditure on vessels (i.e. boats). Other assets, including vehicles and buildings, may also be monitored. Expenditure can be graphed over a given period of time.

This module also has the functionality to request a purchase order, although this will ordinarily take place via telephone. A record of purchase order requests will not be maintained.

CRAFT/log

CRAFT/log is the module for crew members (who are volunteers) to log how much time they spend at sea. CRAFT/log will automatically be updated when an ops report is filed. Much like the other modules in CRAFT/client, master data is stored centrally (on the CRAFT server). In order to modify data, changes are made to local data, and synchronised with the CRAFT server.

CRAFT/spon

CRAFT/spon, as originally specified in the *User Requirement Specification*, has been scoped out of the system. This was at the request of the NSRI, who would prefer to develop a powerful stand-alone sponsors system that is not simply an add-on to the CRAFT Solution.

CRAFT/man

This module provides functionality for monthly management reporting. It will be used by station commanders to submit reports to head office. Most of the

information presented in the monthly management report is generated from data entered in the other modules.

CRAFT/adm

CRAFT/adm is the administration module of the system. This will have the traditional administrative functions of a computer system. These include archiving, backup, forced synchronisation with the CRAFT server and preference setting. It will also have the capability to modify limited reference information in the system.

CRAFT/tech

CRAFT/tech is the server-side component of the system; it can be considered to be the functional back-office of the CRAFT system.

CRAFT/db

This is the master database used by the CRAFT server. Local clients will store a year's worth of historical data on their machines; this will be data pertaining to their station only. This local database is called CRAFT/dbl. Local users will be able to modify certain data, and modifications will be placed in a queue to be run against CRAFT/db when the user logs onto the Internet. This process is known as synchronisation.

The NSRI head office is a special case of the CRAFT/client. It will maintain a copy of the entire database in its CRAFT/dbl. This will be used to generate reports, and make administrative changes to the system. Head office has the ability to make certain changes to any part of the system, with the exception of structural changes that will require a system administrator-level staff member to perform those changes. Modifications will be placed in a queue to be run against CRAFT/db when head office logs onto the Internet.

Head office is required to synchronise with the CRAFT server at least once every 24 hours in order to maintain the accuracy of their data. This will also serve as offsite backup.

Changes are made to records first in, first out basis (FIFO). Records will be locked while they are being changed, so that critical data embraces cannot

occur. A critical data embrace is defined as occurring when two or more CRAFT/client's modify the same piece of data at the same time. When both head office and stations have the ability to change a piece of data, the most recent change is final.

CRAFT/com

CRAFT/com is the communications protocol that will be used to synchronise information between the server and local stations. When stations connect to the CRAFT server, CRAFT/com manages the uploading of queued information from the local station to the server, and the downloading of any queued information from the server to the local station.

Within CRAFT/com is a time synchronisation tool; this is used to set the time on the CRAFT/client's PC with that of the CRAFT server. This is to ensure accurate time stamping of all data queued for the CRAFT server.

CRAFT/serv

This is the server component of the CRAFT Solution. It is envisaged that CRAFT/serv will be hosted at a local internet service provider. When a server is hosted, a hosting agent provides it with network connectivity. This network connectivity to the Internet is crucial for the CRAFT/serv, which must be connected permanently. Connectix is an Internet service provider [ISP] that has sponsored web-hosting services for the NSRI, and has agreed to host the CRAFT/serv.

The CRAFT Solution has been designed to be scalable to the future needs of the NSRI. This has been implemented by writing thorough system documentation, that could be passed to future developers. This means that the system can be expanded and maintained within the context of the CRAFT Solution. Furthermore, the system is being developed using industry standard tools. It is envisaged that further development will not be hindered by a lack of suitable skills.

CRAFT/doc

CRAFT/doc is the documentation package that will accompany the CRAFT Solution. Details of CRAFT/doc are not provided in this document.

ANALYSES

During the course of system design, numerous analysis techniques were undertaken in order to scope and model the CRAFT Solution.

Process Analysis

The major functions within the NSRI have been identified and analysed. This was done in conjunction with Ian Wienburg, the chief executive officer [CEO] of the NSRI.

A context diagram and data-flow diagrams have been included in the appendices to this document. These techniques ensure that accurate modeling of the system has been undertaken.

Data flow diagrams, also known as process models, show the relationship between processes within the system, the information that flows between them and the data stores within the system.

These diagrams have been included in the appendices.

Data Analysis & Modeling

A computerised solution requires and generates large volumes of data. In order to design the CRAFT solution such that it functions effectively and efficiently, it was necessary to undertake advanced data analysis.

Much thought was given to creating a database structure that optimises the use of minimal resources. In technical terms, this would mean designing a relational database in 3NF (3rd normal form). This will reduce the hardware overhead and increase system response times.

A data dictionary and Entity/Relationship Diagram have been included in the appendices to this document.

Functional Analysis

The final analysis level is functional analysis. Two techniques have been employed to ensure accurate system modeling. Both of these models are available in the appendices to this document.

1. A function/entity matrix has been produced. This details all business functions of the system, and the database tables they affect in terms of creating, reading, updating and/or deleting a record. This is included in the appendices to this document.
2. Object orientated analysis has also been performed. Although not strictly functional analysis, object orientated analysis views the system from a business perspective, as opposed to a technical perspective. These object models have been included in the appendices to the *User Requirement Specification*.

The following function table has been drawn up to show the various functions of the logical modules within the CRAFT Solution.

CRAFT/ops

- File an ops report
- Review an ops report
- Edit an ops report

CRAFT/log

- View a crew member's record
- Add/edit/delete a crew member

CRAFT/ves

- Request purchase order
- Add an asset
- Edit/delete an vessel
- Graph expenses over a period of time
- Record expenditure on an item

CRAFT/gyp

- Add a station
- Edit/delete a station
- Download a local version
- View and print the GYPSY book
- Edit a stations' entry
- Add an organisation

CRAFT/adm

- Synchronise the data
- Back-up and archive data
- Add/edit/delete records in the lookup tables:
 - asset_type
 - qualification
 - access_function
 - access
 - insurance_company
- Add/edit/delete records in the data tables...
 - asset
 - organisation
 - station
- Administer security functionality

CRAFT/man

- Draw up a management report
- Edit management reports
- View management reports

Table 1 - Functions of Each Module of CRAFT/client

TECHNICAL ARCHITECTURE

The technical architecture of the system is modeled in the technical environment model that is enclosed in the appendices to this document.

CRAFT/serv

The CRAFT/serv environment is built on top of RedHat Linux, a highly reliable and scalable UNIX-like operating system. The database management system [DBMS] used is the Sybase Adaptive Server Enterprise System 11 for Linux. This server is a well-proven, scalable and reliable enterprise database management system that is freely available under Sybase's Linux promotion strategy.

The server daemon is written in Perl 5, a powerful interpreted cross-platform scripting language. The Perl DBI/DBD modules are used for connectivity to the Sybase DBMS and a Perl-XML module is used to parse the XML used in CRAFT/com.

HTTP services are provided by the Apache webserver, one of the most popular webservers available.

Network connectivity, with specific reference to TCP/IP, is supported by the Linux operating system.

CRAFT/client

The CRAFT/client runs in the Win32 environment. Local database services are provided by a Microsoft Access file database. The CRAFT/client accesses the database using ADO and DAO. The client will use the Microsoft WebBrowser Control to provide HTML parsing facilities. XML parsing services will be provided by an XML COM object.

HARDWARE SPECIFICATIONS

Final system specifications may change slightly as the team evaluates products in more detail to determine their suitability for the CRAFT Solution. It is however envisaged that the final specifications will be not differ greatly from the specifications presented here.

CRAFT/serv

CRAFT/serv requires the following hardware configuration:

- Intel Pentium 133Mhz or greater
- 64Mb RAM
- 4Gb Hard Disk
- 10Mb Network Card

Connectix, the NSRI's website host, has offered the use of a virtual server at their site. This is server a logical server resource, as opposed to a physical server resource. This will be used to backup the database.

CRAFT/serv requires the following software:

- RedHat Linux 6
- Sybase Adaptive Server Enterprise System 11
- Perl 5
- DBI/DBD-Sybase Perl Database Modules
- Perl-XML Module
- Apache HTTP Server
- PerlMod Apache Module

All of these products can be used free of charge under the GNU Public License ([GPL], or "copyleft"), BSD license or their own vendor issued license. The GPL broadly specifies that source code can be modified and distributed, so long as the changes and the original source code are freely available. The BSD license differs from the GPL in that changes to the source code must only be referenced, and not included with any delivered product(s). Vendor issued licenses differ greatly, but usually protect the intellectual property, usage and distribution of their products. The use of these tools requires no expense on behalf of the NSRI or Object Sync.

CRAFT/client

CRAFT/client requires the following hardware:

- Intel Pentium 75Mhz or greater
- 16Mb RAM
- 300Mb or greater disk space

The NSRI have assured Object Sync that each station is equipped with at least a Pentium class computer.

A CRAFT/client machine requires the following software:

- Microsoft Windows 95/98
- Microsoft ADO/DAO
- CRAFT/client Win32 Executable
- Microsoft WebBrowser Control
- XML COM Object
- Microsoft HTML Help

CRAFT/CLIENT INTERFACE

User Interface Look and Feel

The look and feel of the system has been designed to emulate the familiar Internet Web browser interface. CRAFT/client's dynamic content has been scripted, making use of standard Web techniques. All of the functionality sits in ActiveX controls and scripts. ActiveX controls are small compiled programs, designed to be used in Web pages.

Input Design

The input screens of CRAFT/client will be designed to expedite the processing of all transactions. Numerous design techniques have been employed to effect this. The input screen shown in Screenshot (1) shows many of these techniques:

The screenshot displays the 'CRAFT - Station 36' application window. The title bar includes the application name and standard window controls. The menu bar contains 'File', 'Operations', 'GYPSY Book', 'Assets', 'Crew', 'Reports', 'Admin', 'Online Activities', and 'Help'. The toolbar features icons for 'Ups Report', 'Synch', 'Gypsy Book', 'Connect', and 'Hang Up'. A left-hand navigation pane shows a tree view with folders for 'Assets', 'Crew', 'GYPSY Book', 'Help', 'Operations', 'Reports', and 'Outbox'. The main content area is titled 'Add Operation Report' and contains several sections of input fields:

- Caller Details:** Includes a 'Call Time' field with a 'hh:mm' format and a 'Called by' text input field.
- Environment Conditions:** Includes 'Wind Force' and 'Wind Direction' (dropdown), 'Sea State' and 'Visibility' (dropdown), 'Surf Height', and 'Distance offshore' (spinners).
- Rescue Statistics:** Includes 'Vessels used', 'Mobiles Used', 'Persons Rescued', and 'Number of Crew Members' (spinners).
- Description of Operation:** A large text area for entering details.

At the bottom right of the form are buttons for '< Back', 'Next >', and 'Cancel'. A status bar at the bottom contains a warning icon and the text: 'Ops Reports need to be viewed by the Statcom, Ops Manager and fundraising before they are filed.' A 'Next Tip' button is also present.

Screenshot 1 – Add Operation Report

As can be seen on Screenshot (1), all windows will have a standard look and feel. The navigation bar, on the left-hand side of the screen, has a Windows Explorer-type tree. This navigation bar allows access to the various functional modules of the CRAFT/client. This navigation bar is shown whenever the application is running.

The content window, on the right hand side of the screen, is a Web page using familiar controls. Although it is not evident, these Web pages will be rendered using the Web browser control in Visual Basic. The input forms of the system will be designed according to *de facto* Web standards.

Another screenshot has been included in the appendices to this document. On that particular screenshot, the tree menu has been expanded in its entirety.

Hotkeys, accelerator keys, spin-controls and list-boxes will be used wherever possible. Furthermore, a simple design will be used to make the interface optimally functional, while still being easy to learn and use.

Menu Structure

Within CRAFT/client, users will be able to access the functionality using the tree structure (on the left-hand side of the user interface) or through a traditional menu system.

The menu structure, which will be the same for both the window menu and the tree structure, will be as per the following table

| | | |
|--------------------------------|---------------------------|--------|
| <u>F</u> ile | | |
| | <u>L</u> ogin | CTRL-L |
| | <u>R</u> eview Queue | |
| | <u>E</u> xit | ALT-F4 |
| <u>O</u> perations [CRAFT/ops] | | |
| | File <u>O</u> ps Report | CTRL-N |
| | <u>R</u> eview Ops Report | CTRL-O |
| | Request Filed Ops Report | |
| <u>G</u> YPSY Book [CRAFT/gyp] | | |
| | <u>V</u> iew GYPSY Book | CTRL-G |
| | <u>E</u> dit GYPSY Book | |

| Print GYPSY Book

Assets [CRAFT/ves]

| | |
|--|------------------------|
| | Record an Expense |
| | Graph Expenses |
| | Asset Details |
| | Add an Asset |
| | Edit Asset Details |
| | Request Purchase Order |
| | Asset Reports |

Crew [CRAFT/log]

| | | |
|--|--------------------|------------|
| | View Crew Records | CTRL-ALT-L |
| | Add Crew Member | |
| | Edit Crew Member | |
| | Remove Crew Member | |
| | Crew Reports | |

Reports [CRAFT/man]

| | | |
|------|---|------|
| | Operations | |
| | Assets | |
| | Crew | |
| | Management Reports | |
| | <table border="1"> <tr> <td>File</td> </tr> <tr> <td>View</td> </tr> </table> | File |
| File | | |
| View | | |

Admin [CRAFT/adm]

| | | | | | | | |
|----------------|---|------------|------------|--------|------|---------|-------|
| | Change Passwords | | | | | | |
| | Backup | | | | | | |
| | View Sync Log | | | | | | |
| | Modify Lookup Tables | | | | | | |
| | <table border="1"> <tr> <td>Operations</td> </tr> <tr> <td>GYPSY Book</td> </tr> <tr> <td>Assets</td> </tr> <tr> <td>Crew</td> </tr> <tr> <td>Reports</td> </tr> <tr> <td>Admin</td> </tr> </table> | Operations | GYPSY Book | Assets | Crew | Reports | Admin |
| | Operations | | | | | | |
| | GYPSY Book | | | | | | |
| | Assets | | | | | | |
| | Crew | | | | | | |
| | Reports | | | | | | |
| Admin | | | | | | | |
| Online Upgrade | | | | | | | |

Online Activities

| | | |
|--------------------------|--------------------------|------------|
| | Synchronise | CTRL-S |
| <u>H</u> elp [CRAFT/doc] | | |
| | Contents | F1 |
| | Index | SHFT-F1 |
| | How Do I? | |
| <hr/> | | |
| | About the CRAFT Solution | |
| | About CRAFT/client | CTRL-ALT-Z |

Table 2 – CRAFT/client Menu Structure

Navigation using the Windows9x (95 or 98) menu system is as per the standard Windows interface:

- Underlined keys on the first level are accessed using ALT.
- Underlined keys on any sublevels are accessed by pressing the key with the parent menu active.
- Hotkeys are accessible from anywhere within the application.

The logical module of CRAFT/client within which each first level menu item, if applicable, is in square brackets after the menu item.

Output Design

Much like the design of input screens, system output will be designed to be as functional as possible.

All reports will have a standardised look and feel; an example of the standard report template can be found in the figure below. A simple design will again be employed. This is done to highlight important information, and allow for ease of use. Headings will be standardised across the system, and footers will contain additional useful information (e.g. date, page number etc.).

| | |
|--------------------|-------------|
| Module | Date |
| <i>Report Data</i> | |
| Report Title | Page x of x |

Figure 1 – Standard Report Template

Some report mock-ups have been included in the appendices to this document.

Where necessary, the report windows will contain controls to provide ease of access to specific information; these include spin-controls, radio buttons and command buttons.

SECURITY MODEL

CRAFT makes use of an access level security model, which is centrally stored on the server and dynamically distributed to the client.

Local password changes are instantaneous and are reflected on the server on the next synchronisation. By comparing the checksum of an access level and its associated password, changes to the server are transferred to the client machines on the next synchronisation. A checksum is a unique identifier for a piece of data, and is unique to that data.

This allows for password maintenance remotely by the NSRI, and allows for the rapid removal of access rights for compromised accounts.

CRAFT/client

System Initialisation

When a new site requires the installation of the CRAFT client software, the station commander contacts the NSRI headquarters telephonically.

Information about the site and its staff is then, where necessary, captured by the administrative staff, who enters the information into the CRAFT server. When this information has been registered, a unique serial number is generated.

This unique serial number, which is valid for the initial installation of the software and used when synchronising, is used along with the station number to authenticate the site as a valid location, and allows the client software to be loaded. The client software then goes on-line to communicate to CRAFT/serv, to verify the authenticity of the serial number. Generic accounts (with varying access levels) and passwords (where appropriate) are created.

This design has been used to ensure that “rogue” sites do not obtain the CRAFT software and install it. To use the CRAFT software, its unique serial number and knowledge about the generic account details is required to use the system.

System Recovery

If a client site’s installation of CRAFT/client is corrupted or compromised, a similar process occurs. The station commander at the site contacts the NSRI administrative staff at headquarters, who disable the old serial number and supply a new serial number to the site station commander. The software is then installed, and on entry of the unique serial number and connectivity to the server, the software will download the list of account details for that particular site.

Access Levels

The CRAFT system is intended to be as easy to use as possible; as a result, an intrusive security model is not desired. The security of the system needs to be as unobtrusive as possible, without compromising its integrity.

As a solution to this dichotomy, most volunteers of CRAFT/client will not require usernames and passwords to enter data into the system. Only non-volunteers and system commanders will require a password to gain access to the more advanced functions of the product.

There are six defined levels of access to the CRAFT system; they are

- Volunteer
- Station commander
- Fund raising
- Operations manager
- CEO
- System administrator

Volunteer- and station commander-level staff can only view and manipulate data relevant to their station, whereas the operations manager-level staff can view and manipulate data across all stations.

Fund raising-level staff can view and modify financial information across all stations. CEO-level staff have the ability to modify all information other than operations data, which only the operations manager-level staff have modification access to.

System administrator-level staff have access to the entire system, and can modify, add or remove any data.

It is felt that there should only be one operations manager-level staff member, one CEO-level staff member and not more than two system administrator-level staff. This is to ensure data integrity and ensure adequate process accounting.

The specific information that a particular level has access to, and ability to modify are :

Volunteers have the ability to:

- file an ops report
- review an ops report filed from the same site
- view other staff records at the same site
- view, download and print the GYPSY book
- synchronise the data
- record donations

Station commanders can, in addition to volunteers:

- edit ops reports from the same site
- add, edit or delete a crew members
- request purchase orders
- edit a vessel's details
- edit a stations details in the GYPSY book
- request a purchase order

Fund raising staff can:

- review the donations information generated from ops reports
- query, edit, delete and add sponsor information

The **operations manager** has access to all the operational data in the CRAFT system, and can add, modify or delete information. In addition, operations manager-level staff can:

- modify, delete and add asset details
- graph expenses over a period of time
- edit, delete and add station details
- add, edit or delete organisations and other entities to the GYPSY book
- add, edit or delete access security information

CEO-level staff have the ability to view, modify, create or delete all non-operations data in the system. This ensures that operations data in the system is the responsibility of one level of user.

The **System Administrator** access level is intended to be used by maintainers of the system, or any individual or group that must implement low level changes, system administrator-level staff are expected to make use of this access level rarely.

CRAFT/serv

The security of CRAFT/serv is a critical factor in preventing abuse, and the loss or modification of data.

The CRAFT server makes use of the CRAFT/com protocol, and expects the CRAFT/client to respond to certain prompts. This is done by the CRAFT server protocol, which is designed to be simple, and secure.

Access to the server is allowed only to CRAFT/clients that present an active serial number and valid station number pair. New or modified information is then uploaded to the server, and any information or updates that are needed are downloaded.

The server ensures data integrity by comparing the checksum of the information being uploaded to that of the information on the server. If the information received from the CRAFT client is newer, and contains the correct authorisation for the site where the information was generated, the information is modified, added or deleted to the information on the server.

CRAFT/COM

The CRAFT/com synchronisation protocol is a complex part of the CRAFT Solution. This document provides fairly high-level advanced design specifications.

The CRAFT/com synchronisation protocol will be designed to allow batched asynchronous transfer of data between the server and multiple clients.

A Perl daemon (CRAFT/serv) runs out of `inetd(8)` on the CRAFT/tech, listening on TCP port 5555. This port has been selected to allow the daemon to run as a non-privileged user - namely `craftuser`. The `craftuser` has the requisite access rights to the database to allow additions, deletions, updates and stored procedure execution.

The CRAFT/client dials up to an ISP and connects to the server on the above port. A handshake takes place, with all data being transmitted over unencrypted TCP streams. The data, including the handshake, will be structured using XML.

CRAFT/com Functionality and Operation

The CRAFT/com protocol implements the following functionality:

- Data Synchronisation
- Time Synchronisation

The details of these synchronisations can be found in the XML schema below.

Data synchronisation and message transfer are facilitated by an XML vocabulary. The XML vocabulary is relatively simple and therefore does not include a *Document Type Definition* (DTD).

Time synchronisation is facilitated by the standard prompts presented by the CRAFT/serv.

A typical connection session describes the standard method of operation of the protocol. Details of some dialogue sessions can be found in the enclosed XML schema. A session is defined as a single instance of a connection between the CRAFT/client and CRAFT/serv. A transaction is defined as a dialogue between a CRAFT/client and CRAFT/serv. A session may contain many transactions.

An overview of a CRAFT/com session is provided in the figure below.

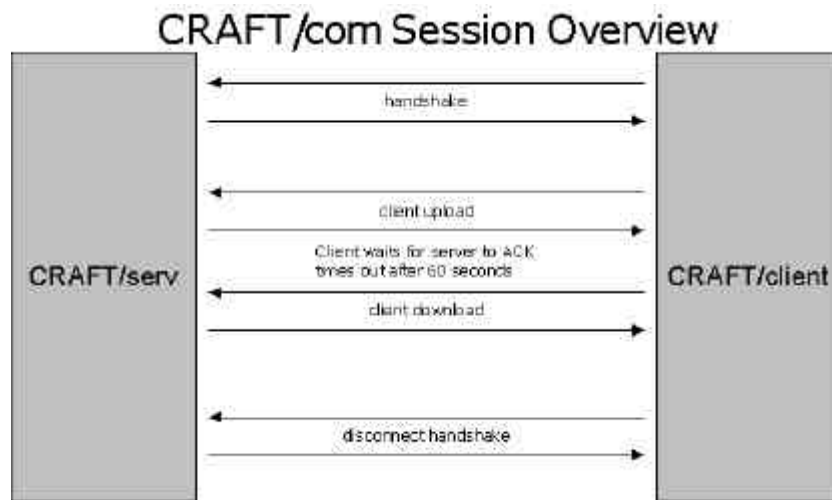


Figure 2 – CRAFT/com Session Overview

CRAFT/serv is the serverside implementation of CRAFT/com. CRAFT/serv provides the following functionality:

- Transaction control (including record locking strategies and integrity maintenance).
- Data manipulation
- Security Administration
- Server load management

This functionality and its implementation is fully detailed in the XML vocabulary section.

CRAFT/serv prompts

These prompts are returned by the daemon on client connection or may be presented after any client request to the server when the CRAFT/serv is not in a READY state. The prompts allow the CRAFT/client to determine the state of the CRAFT/serv and the current time.

CRAFT/serv ready to serve:

```
220 CRAFT/serv 15:12 24 April 1999
220 +Ready
```

Temporarily unavailable due to server load:

```
440 CRAFT/serv 15:12 24 April 1999
440 +HighLoad
```

Permanently unavailable due to fatal error:

```
550 CRAFT/serv 15:12 24 April 1999
550 +Error
```

Upon receiving an error status, the CRAFT/client can act accordingly. It is suggested that the CRAFT/client passes these errors on to the end-user to ensure transparency of all synchronisation failures.

XML Vocabulary

All data (except for time synchronisation) is structured in XML. The standard XML schema that CRAFT/com is built upon is found below. It is important to note that some comment fields are RCS control fields.

```
<?xml version="1.0" standalone="yes"?>
<!--
  ** NOTE: Do not use this file - it is a template - make a copy and remove this line **

  $Revision$
  $Date$
  $Author$
  $Locker$
  $state$

  $RCSfile$

  CRAFT/com: STANDARD_TEMPLATE
  Object Sync
  CRAFT Solution
  Copyright 1999 University of Cape Town

  $Log$
-->
<!--
  XML Schema Structure:

  1. First element is the XML Document Element Name
  2. Second Element is a control Area
  3. Third element is a data area
  4. Data area contains unparsed CDATA section for RAW SQL or Unicode Text
-->

<STANDARD_TEMPLATE>
  <CNTROLAREA>
    <PROTOCOL>
```

```

<NAME>CRAFT/com</NAME>
<VER>1</VER>
</PROTOCOL>
<BSR>
  <VERB></VERB>          <!-- Action to be taken -->
  <NOUN></NOUN>          <!-- Take action on object -->
</BSR>
<SENDER>
  <STATIONID></STATIONID> <!-- StationID: Integer -->
  <COMPONENT></COMPONENT> <!-- CRAFT Component this was generated by -->
  <REFERENCEID></REFERENCEID> <!-- Reference Format: YYYYMMDDXXXX -->
</SENDER>
<DATETIME>
  <YEAR></YEAR>          <!-- Year Format: YYYY -->
  <MONTH></MONTH>        <!-- Month Format: MM -->
  <DAY></DAY>            <!-- Day Format: DD -->
  <HOUR></HOUR>          <!-- Hour Format: HH 24hour -->
  <MINUTE></MINUTE>      <!-- Minute Format: MM -->
  <SECOND></SECOND>     <!-- Second Format: SS -->
  <TIMEZONE></TIMEZONE> <!-- Timezone Format: [+_]DDDD -->
</DATETIME>
</CONTROLAREA>
<DATAAREA>
  <![CDATA[

    <!--
      Embed SQL or Unicode Text in this CDATA section
    -->

  ]]>
</DATAAREA>
</STANDARD_TEMPLATE>

```

Function specific XML structures built from this schema template can be found below. For brevity's sake, not all XML schema have been included. The schema has been designed to be as flexible as possible to allow for possible future system additions.

Initial Handshake and Authentication

A session always begins with a handshake and authentication. This dialogue is detailed below:

- CRAFT/client connects to CRAFT/serv and waits for prompt [timeout if not connected within 30 seconds].
- CRAFT/serv issues prompt.
- CRAFT/client interprets prompt. Non-fatal error prompts result in a 30s timer.
- CRAFT/client sets time from prompt if CRAFT/serv returns *READY* prompt.
- Authentication dialogue begins by CRAFT/client issuing an authentication stream as detailed below:

```

<?xml version="1.0" standalone="yes"?>
<!--
$Revision: 1.1 $
$Date: 1999/06/12 09:57:59 $

```

```

$Author: danielc $
$Locker: danielc $
$state$

$RCSfile: client_auth.xml,v $

CRAFT/com: CLIENT AUTH
Object Sync
CRAFT Solution
Copyright 1999 University of Cape Town

$Log: client_auth.xml,v $
Revision 1.1 1999/06/12 09:57:59 danielc
Initial revision

-->
<!--
XML Schema Structure:

1. First element is the XML Document Element Name
2. Second Element is a control Area
3. Third element is a data area
4. Data area contains unparsed CDATA section for RAW SQL or Unicode Text

-->

<CLIENT_AUTH>
  <CNTROLAREA>
    <PROTOCOL>
      <NAME>CRAFT/com</NAME>
      <VER>1</VER>
    </PROTOCOL>
    <BSR>
      <VERB>AUTH</VERB>           <!-- Action to be taken -->
      <NOUN>CLIENT</NOUN>       <!-- Take action on object -->
    </BSR>
    <SENDER>
      <STATIONID></STATIONID> <!-- StationID: Integer -->
      <COMPONENT>CRAFT/client</COMPONENT>   <!-- CRAFT Component this was generated by -->
      <REFERENCEID></REFERENCEID> <!-- Reference Format: YYYYMMDDXXXX -->
    </SENDER>
    <DATETIME>
      <YEAR></YEAR>           <!-- Year Format: YYYY -->
      <MONTH></MONTH>         <!-- Month Format: MM -->
      <DAY></DAY>             <!-- Day Format: DD -->
      <HOUR></HOUR>           <!-- Hour Format: HH 24hour -->
      <MINUTE></MINUTE>       <!-- Minute Format: MM -->
      <SECOND></SECOND>       <!-- Second Format: SS -->
      <TIMEZONE></TIMEZONE>   <!-- Timezone Format: [+_]DDDD -->
    </DATETIME>
  </CNTROLAREA>
  <DATAAREA>
    <STATSER></STATSER>      <!-- Enter Station Serial Number Here -->
  </DATAAREA>
</CLIENT_AUTH>

```

The CRAFT/serv responds with an authentication ACK of either AUTH *granted* or *denied*:

```

<?xml version="1.0" standalone="yes"?>
<!--
$Revision: 1.2 $
$Date: 1999/06/12 10:21:51 $
$Author: danielc $
$Locker: $
$state$

$RCSfile: server_auth_ack.xml,v $

CRAFT/com: SERVER AUTH ACK
Object Sync
CRAFT Solution

```

```
Copyright 1999 University of Cape Town

$Log: server_auth_ack.xml,v $
Revision 1.2 1999/06/12 10:21:51 danielc
changed spec

Revision 1.1 1999/06/12 10:19:42 danielc
Initial revision

-->
<!--

XML Schema Structure:

1. First element is the XML Document Element Name
2. Second Element is a control Area
3. Third element is a data area
4. Data area contains unparsed CDATA section for RAW SQL or Unicode Text

-->

<SERVER_AUTH_ACK>
  <CNTRLAREA>
    <PROTOCOL>
      <NAME>CRAFT/com</NAME>
      <VER>1</VER>
    </PROTOCOL>
    <BSR>
      <VERB>AUTH</VERB>          <!-- Action to be taken -->
      <NOUN></NOUN>              <!-- Noun is GRANTED or DENY -->
    </BSR>
    <SENDER>
      <STATIONID></STATIONID>    <!-- StationID: Integer -->
      <COMPONENT>CRAFT/serv</COMPONENT>  <!-- CRAFT Component this was generated by -->
      <REFERENCEID></REFERENCEID> <!-- Reference Format: YYYYMMDDXXXX -->
    </SENDER>
    <DATETIME>
      <YEAR></YEAR>              <!-- Year Format: YYYY -->
      <MONTH></MONTH>            <!-- Month Format: MM -->
      <DAY></DAY>                <!-- Day Format: DD -->
      <HOUR></HOUR>              <!-- Hour Format: HH 24hour -->
      <MINUTE></MINUTE>          <!-- Minute Format: MM -->
      <SECOND></SECOND>          <!-- Second Format: SS -->
      <TIMEZONE></TIMEZONE>     <!-- Timezone Format: [+_]DDDD -->
    </DATETIME>
  </CNTRLAREA>
</SERVER_AUTH_ACK>
```

Client Upload

Upon receiving the AUTH ACK from the CRAFT/serv, the CRAFT/client uploads new data to the CRAFT/serv.

```
<?xml version="1.0" standalone="yes"?>
<!--

$Revision: 1.1 $
$Date: 1999/06/12 14:12:13 $
$Author: danielc $
$Locker: $
$state$

$RCSfile: client_data_up.xml,v $

CRAFT/com: CLIENT DATA UP
Object Sync
CRAFT Solution
Copyright 1999 University of Cape Town
```

```

$Log: client_data_up.xml,v $
Revision 1.1 1999/06/12 14:12:13 danielc
Initial revision

-->

<!--

XML Schema Structure:

1. First element is the XML Document Element Name
2. Second Element is a control Area
3. Third element is a data area
4. Data area contains unparsed CDATA section for RAW SQL or Unicode Text

-->

<CLIENT_DATA_UP>
  <CONTROLAREA>
    <PROTOCOL>
      <NAME>CRAFT/com</NAME>
      <VER>1</VER>
    </PROTOCOL>
    <BSR>
      <VERB>UPLOAD</VERB>          <!-- Action to be taken -->
      <NOUN>DB</NOUN>             <!-- Take action on object -->
    </BSR>
    <SENDER>
      <STATIONID></STATIONID>      <!-- StationID: Integer -->
      <COMPONENT></COMPONENT>     <!-- CRAFT Component this was generated by -->
      <REFERENCEID></REFERENCEID> <!-- Reference Format: YYYYMMDDXXXX -->
    </SENDER>
    <DATETIME>
      <YEAR></YEAR>                <!-- Year Format: YYYY -->
      <MONTH></MONTH>              <!-- Month Format: MM -->
      <DAY></DAY>                  <!-- Day Format: DD -->
      <HOUR></HOUR>                <!-- Hour Format: HH 24hour -->
      <MINUTE></MINUTE>            <!-- Minute Format: MM -->
      <SECOND></SECOND>           <!-- Second Format: SS -->
      <TIMEZONE></TIMEZONE>       <!-- Timezone Format: [+_]DDDD -->
    </DATETIME>
  </CONTROLAREA>
  <DATAAREA>
    <![CDATA[

      <!--
      Embed SQL or Unicode Text in this CDATA section
      -->

    ]]>
  </DATAAREA>
</STANDARD_TEMPLATE>

```

Each record update is a transaction and every transaction is ACK'ed with either a "success" or "fail" reply from the CRAFT/serv. If the transaction fails, a reason is included in the reply. CRAFT/client will time out after 30 seconds if no reply is forthcoming from the server. An attempt will be made to reissue the transaction. If the client has only partially submitted a transaction and the server does not receive more data for 30 seconds, the server will rollback any changes made for that particular transaction and wait for further data from the client.

```
<?xml version="1.0" standalone="yes"?>
```

```
<!--
```

```
$Revision: 1.1 $
```

```

$Date: 1999/06/12 14:30:19 $
$Author: danielc $
$Locker: $
$state$

$RCSfile: server_up_ack.xml,v $

CRAFT/com: SERVER ACK/NACK UPLOAD
Object Sync
CRAFT Solution
Copyright 1999 University of Cape Town

$Log: server_up_ack.xml,v $
Revision 1.1 1999/06/12 14:30:19 danielc
Initial revision

-->
<!--

XML Schema Structure:

1. First element is the XML Document Element Name
2. Second Element is a control Area
3. Third element is a data area
4. Data area contains unparsed CDATA section for RAW SQL or Unicode Text

-->

<SERVER_UP_RESP>
  <CNTROLAREA>
    <PROTOCOL>
      <NAME>CRAFT/com</NAME>
      <VER>1</VER>
    </PROTOCOL>
    <BSR>
      <VERB>RESPONSE</VERB>          <!-- Action to be taken -->
      <NOUN>UPLOAD</NOUN>           <!-- Take action on object -->
    </BSR>
    <SENDER>
      <STATIONID></STATIONID>        <!-- StationID: Integer -->
      <COMPONENT></COMPONENT>        <!-- CRAFT Component this was generated by -->
      <REFERENCEID></REFERENCEID>    <!-- Reference Format: YYYYMMDDXXXX -->
    </SENDER>
    <DATETIME>
      <YEAR></YEAR>                  <!-- Year Format: YYYY -->
      <MONTH></MONTH>                <!-- Month Format: MM -->
      <DAY></DAY>                    <!-- Day Format: DD -->
      <HOUR></HOUR>                  <!-- Hour Format: HH 24hour -->
      <MINUTE></MINUTE>              <!-- Minute Format: MM -->
      <SECOND></SECOND>              <!-- Second Format: SS -->
      <TIMEZONE></TIMEZONE>          <!-- Timezone Format: [+_]DDDD -->
    </DATETIME>
  </CNTROLAREA>
  <DATAAREA>
    <RESPONSE>
    <RESPONSE>
    <REASON>
  </REASON>
  </DATAAREA>
</SERVER_UP_RESP>

```

The CRAFT/com uses an *atomic locking* strategy to ensure data integrity. All transactions and database tuples are serialised using a serial number in the following format: YYYYMMDDXXXX. When a transaction is run against the database, the CRAFT/serv determines whether the serial number and primary key combination for that tuple is newer than the one it has been provided with. If it is, the transaction fails and the failure is reported to the CRAFT/client.

Client Download

CRAFT/client requests CRAFT/serv to supply an MD5 hash to determine whether any datastores need to be updated. This hash is the product of running a concatenated string of all primary keys and serial numbers in a datastore through an MD5 hashing algorithm. If the hash on the server side does not correspond to the hash on the client side, the client requests the server for a list of all primary keys and serial numbers in that datastore. A comparison is done on the client side and individual data records are requested.

An MD5 hash request would be implemented as follows:

```
<?xml version="1.0" standalone="yes"?>
<!--
$Revision: 1.1 $
$date: 1999/06/13 11:31:45 $
$Author: danielc $
$Locker: danielc $
$state$

$RCSfile: client_hash_request.xml,v $

CRAFT/com: CLIENT HASH REQUEST
Object Sync
CRAFT Solution
Copyright 1999 University of Cape Town

$Log: client_hash_request.xml,v $
Revision 1.1 1999/06/13 11:31:45 danielc
Initial revision

-->
<!--
XML Schema Structure:

1. First element is the XML Document Element Name
2. Second Element is a control Area
3. Third element is a data area
4. Data area contains unparsed CDATA section for RAW SQL or Unicode Text

-->
<CLIENT_HASH_REQUEST>
  <CONTROLAREA>
    <PROTOCOL>
      <NAME>CRAFT/com</NAME>
      <VER>1</VER>
    </PROTOCOL>
    <BSR>
      <VERB>REQUEST</VERB>          <!-- Action to be taken -->
      <NOUN>HASH</NOUN>            <!-- Take action on object -->
    </BSR>
    <SENDER>
      <STATIONID></STATIONID>      <!-- StationID: Integer -->
      <COMPONENT>CRAFT/client</COMPONENT>  <!-- CRAFT Component this was generated by -->
      <REFERENCEID></REFERENCEID> <!-- Reference Format: YYYYMMDDXXXX -->
    </SENDER>
    <DATETIME>
```

```

<YEAR></YEAR>                <!-- Year Format: YYYY -->
<MONTH></MONTH>              <!-- Month Format: MM -->
<DAY></DAY>                  <!-- Day Format: DD -->
<HOUR></HOUR>                <!-- Hour Format: HH 24hour -->
<MINUTE></MINUTE>           <!-- Minute Format: MM -->
<SECOND></SECOND>           <!-- Second Format: SS -->
<TIMEZONE></TIMEZONE>       <!-- Timezone Format: [+-]DDDD -->
</DATETIME>
</CONTROLAREA>
<DATAAREA>
  <![CDATA[
    <!--
      Embed SQL or Unicode Text in this CDATA section
    -->

    ]]>
  </DATAAREA>
</CLIENT_HASH_REQUEST>

```

The server returns the MD5 hash to the client. A 60-second timer is attached to this transaction set. If the server does not respond, the client may attempt to request the hash again.

```

<?xml version="1.0" standalone="yes"?>
<!--
  $Revision: 1.1 $
  $Date: 1999/06/13 11:40:26 $
  $Author: danielc $
  $Locker: danielc $
  $state$

  $RCSfile: server_hash_send.xml,v $

  CRAFT/com: SERVER HASH SEND
  Object Sync
  CRAFT Solution
  Copyright 1999 University of Cape Town

  $Log: server_hash_send.xml,v $
  Revision 1.1 1999/06/13 11:40:26 danielc
  Initial revision

-->
<!--
  XML Schema Structure:

  1. First element is the XML Document Element Name
  2. Second Element is a control Area
  3. Third element is a data area
  4. Data area contains unparsed CDATA section for RAW SQL or Unicode Text

-->
<SERVER_HASH_SEND>
  <CONTROLAREA>
    <PROTOCOL>
      <NAME>CRAFT/com</NAME>
      <VER>1</VER>
    </PROTOCOL>
    <BSR>
      <VERB>SEND</VERB>          <!-- Action to be taken -->
      <NOUN>HASH</NOUN>         <!-- Take action on object -->
    </BSR>
    <SENDER>

```

```

<STATIONID></STATIONID>      <!-- StationID: Integer -->
<COMPONENT></COMPONENT>     <!-- CRAFT Component this was generated by -->
<REFERENCEID></REFERENCEID> <!-- Reference Format: YYYYMMDDXXXX -->
</SENDER>
<DATETIME>
  <YEAR></YEAR>              <!-- Year Format: YYYY -->
  <MONTH></MONTH>            <!-- Month Format: MM -->
  <DAY></DAY>                <!-- Day Format: DD -->
  <HOUR></HOUR>              <!-- Hour Format: HH 24hour -->
  <MINUTE></MINUTE>         <!-- Minute Format: MM -->
  <SECOND></SECOND>         <!-- Second Format: SS -->
  <TIMEZONE></TIMEZONE>     <!-- Timezone Format: [+_]DDDD -->
</DATETIME>
</CNTROLAREA>
<DATAAREA>
  <HASH>34d814f9f807ae215b8b155c439acb3c</HASH>
</DATAAREA>
</SERVER_HASH_SEND>

```

If the server hash matches that of the hash generated on the client, no further activity is needed. The CRAFT/client disconnects by sending a simple "quit" [lower or uppercase is permitted] to the CRAFT/serv.

If the hashes do not match, the CRAFT/client requests a full primary key and serial number transfer.

```

<?xml version="1.0" standalone="yes"?>
<!--
  $Revision: 1.1 $
  $Date: 1999/06/13 11:48:03 $
  $Author: danielc $
  $Locker: danielc $
  $state$

  $RCSfile: client_key_ser_request.xml,v $

  CRAFT/com: CLIENT KEY SER REQUEST
  Object Sync
  CRAFT Solution
  Copyright 1999 University of Cape Town

  $Log: client_key_ser_request.xml,v $
  Revision 1.1 1999/06/13 11:48:03 danielc
  Initial revision

-->
<!--
  XML Schema Structure:

  1. First element is the XML Document Element Name
  2. Second Element is a control Area
  3. Third element is a data area
  4. Data area contains unparsed CDATA section for RAW SQL or Unicode Text

-->
<CLIENT_KEY_SER_REQUEST>
  <CNTROLAREA>
    <PROTOCOL>
      <NAME>CRAFT/com</NAME>
      <VER>1</VER>
    </PROTOCOL>

```

```

<BSR>
  <VERB>REQUEST_KEYS</VERB>          <!-- Action to be taken -->
  <NOUN>[datastore]</NOUN>          <!-- Take action on object -->
</BSR>
<SENDER>
  <STATIONID></STATIONID>          <!-- StationID: Integer -->
  <COMPONENT></COMPONENT>          <!-- CRAFT Component this was generated by -->
  <REFERENCEID></REFERENCEID> <!-- Reference Format: YYYYMMDDXXXX -->
</SENDER>
<DATETIME>
  <YEAR></YEAR>                    <!-- Year Format: YYYY -->
  <MONTH></MONTH>                  <!-- Month Format: MM -->
  <DAY></DAY>                      <!-- Day Format: DD -->
  <HOUR></HOUR>                   <!-- Hour Format: HH 24hour -->
  <MINUTE></MINUTE>               <!-- Minute Format: MM -->
  <SECOND></SECOND>               <!-- Second Format: SS -->
  <TIMEZONE></TIMEZONE>          <!-- Timezone Format: [+_]DDDD -->
</DATETIME>
</CNTROLAREA>
<DATAAREA>
  <![CDATA[

    <!--
      Embed SQL or Unicode Text in this CDATA section
    -->

  ]]>
</DATAAREA>
</CLIENT_KEY_SER_REQUEST>

```

The CRAFT/serv responds with the set of keys and serial numbers. Timeout occurs after 60 seconds.

```

<?xml version="1.0" standalone="yes"?>

<!--

$Revision: 1.1 $
$date: 1999/06/13 11:54:26 $
$Author: danielc $
$Locker: danielc $
$state$

$RCSfile: server_key_send.xml,v $

CRAFT/com: SERVER KEY SEND
Object Sync
CRAFT Solution
Copyright 1999 University of Cape Town

$Log: server_key_send.xml,v $
Revision 1.1 1999/06/13 11:54:26 danielc
Initial revision

-->

<!--

XML Schema Structure:

1. First element is the XML Document Element Name
2. Second Element is a control Area
3. Third element is a data area
4. Data area contains unparsed CDATA section for RAW SQL or Unicode Text

-->

<SERVER_KEY_SEND>
  <CNTROLAREA>
  <PROTOCOL>

```

```

<NAME>CRAFT/com</NAME>
<VER>1</VER>
</PROTOCOL>
<BSR>
  <VERB>SEND_KEYS</VERB>          <!-- Action to be taken -->
  <NOUN>[datastore]</NOUN>        <!-- Take action on object -->
</BSR>
<SENDER>
  <STATIONID></STATIONID>          <!-- StationID: Integer -->
  <COMPONENT></COMPONENT>         <!-- CRAFT Component this was generated by -->
  <REFERENCEID></REFERENCEID>     <!-- Reference Format: YYYYMMDDXXXX -->
</SENDER>
<DATETIME>
  <YEAR></YEAR>                   <!-- Year Format: YYYY -->
  <MONTH></MONTH>                 <!-- Month Format: MM -->
  <DAY></DAY>                     <!-- Day Format: DD -->
  <HOUR></HOUR>                   <!-- Hour Format: HH 24hour -->
  <MINUTE></MINUTE>               <!-- Minute Format: MM -->
  <SECOND></SECOND>               <!-- Second Format: SS -->
  <TIMEZONE></TIMEZONE>           <!-- Timezone Format: [+_]DDDD -->
</DATETIME>
</CNTROLAREA>
<DATAAREA>
  <ROW>                            <!-- Repeat row element as often as is necessary -->
    <KEY></KEY>
    <SERIAL></SERIAL>
  </ROW>
</DATAAREA>
</SERVER_KEY_SEND>

```

The CRAFT/client then determines which serial numbers are newer and requests the CRAFT/serv to send SQL update statements to update the client-side database.

```

<?xml version="1.0" standalone="yes"?>
<!--
$Revision: 1.1 $
$date: 1999/06/13 12:02:22 $
$Author: danielc $
$Locker: danielc $
$state$

$RCSfile: client_update_req.xml,v $

CRAFT/com: CLIENT UPDATE REQUEST
Object Sync
CRAFT Solution
Copyright 1999 University of Cape Town

$Log: client_update_req.xml,v $
Revision 1.1 1999/06/13 12:02:22 danielc
Initial revision

-->
<!--
XML Schema Structure:

1. First element is the XML Document Element Name
2. Second Element is a control Area
3. Third element is a data area
4. Data area contains unparsed CDATA section for RAW SQL or Unicode Text

-->
<CLIENT_UPDATE_REQ>
  <CNTROLAREA>
    <PROTOCOL>

```

```

<NAME>CRAFT/com</NAME>
<VER>1</VER>
</PROTOCOL>
<BSR>
  <VERB>REQ_UPDATE</VERB>          <!-- Action to be taken -->
  <NOUN>[datastore]</NOUN>         <!-- Take action on object -->
</BSR>
<SENDER>
  <STATIONID></STATIONID>          <!-- StationID: Integer -->
  <COMPONENT>CRAFT/client</COMPONENT> <!-- CRAFT Component this was generated by -->
  <REFERENCEID></REFERENCEID>      <!-- Reference Format: YYYYMMDDXXXX -->
</SENDER>
<DATETIME>
  <YEAR></YEAR>                   <!-- Year Format: YYYY -->
  <MONTH></MONTH>                 <!-- Month Format: MM -->
  <DAY></DAY>                     <!-- Day Format: DD -->
  <HOUR></HOUR>                   <!-- Hour Format: HH 24hour -->
  <MINUTE></MINUTE>               <!-- Minute Format: MM -->
  <SECOND></SECOND>               <!-- Second Format: SS -->
  <TIMEZONE></TIMEZONE>          <!-- Timezone Format: [+_]DDDD -->
</DATETIME>
</CNTROLAREA>
<DATAAREA>
  <KEY></KEY>                     <!-- Repeat keys as many times as necessary -->
</DATAAREA>
</CLIENT_UPDATE_REQ>

```

The CRAFT/serv responds accordingly. The SQL statement generated by the CRAFT/serv includes all requested changes for that datastore - the client must attempt to commit the whole statement or roll the transaction back and raise a notification.

```

<?xml version="1.0" standalone="yes"?>
<!--
  $Revision: 1.1 $
  $Date: 1999/06/13 12:41:26 $
  $Author: danielc $
  $Locker: danielc $
  $state$

  $RCSfile: server_update_send.xml,v $

  CRAFT/com: SERVER UPDATE SEND
  Object Sync
  CRAFT Solution
  Copyright 1999 University of Cape Town

  $Log: server_update_send.xml,v $
  Revision 1.1 1999/06/13 12:41:26 danielc
  Initial revision

-->
<!--
  XML Schema Structure:

  1. First element is the XML Document Element Name
  2. Second Element is a control Area
  3. Third element is a data area
  4. Data area contains unparsed CDATA section for RAW SQL or Unicode Text

-->
<SERVER_UPDATE_SEND>
  <CNTROLAREA>
    <PROTOCOL>

```

```

<NAME>CRAFT/com</NAME>
<VER>1</VER>
</PROTOCOL>
<BSR>
  <VERB>SEND_UPDATE</VERB>          <!-- Action to be taken -->
  <NOUN>[datastore]</NOUN>          <!-- Take action on object -->
</BSR>
<SENDER>
  <STATIONID></STATIONID>          <!-- StationID: Integer -->
  <COMPONENT>CRAFT/client</COMPONENT> <!-- CRAFT Component this was generated by -->
  <REFERENCEID></REFERENCEID> <!-- Reference Format: YYYYMMDDXXXX -->
</SENDER>
<DATETIME>
  <YEAR></YEAR>                    <!-- Year Format: YYYY -->
  <MONTH></MONTH>                  <!-- Month Format: MM -->
  <DAY></DAY>                      <!-- Day Format: DD -->
  <HOUR></HOUR>                   <!-- Hour Format: HH 24hour -->
  <MINUTE></MINUTE>               <!-- Minute Format: MM -->
  <SECOND></SECOND>               <!-- Second Format: SS -->
  <TIMEZONE></TIMEZONE>          <!-- Timezone Format: [+_]DDDD -->
</DATETIME>
</CONTROLAREA>
<DATAAREA>
  <![CDATA[

      <!--
      Embed SQL or Unicode Text in this CDATA section
      -->

  ]]>
</DATAAREA>
</SERVER_UPDATE_SEND>

```

As has been explained, once CRAFT/client completes the synchronisation, CRAFT/client disconnects by sending a simple "quit" [lower or uppercase is permitted] to CRAFT/serv. If CRAFT/client does not communicate with CRAFT/serv for a period of 60 seconds, CRAFT/serv will close the connection.

CRAFT/SERV VOLUME PROJECTIONS

Load and Transaction Volumes

Should all the CRAFT/clients connect to the CRAFT/serv at the same time, the projected server load would look as follows:

| | |
|------------------|-------|
| Stations | 26 |
| Regional Offices | 3 |
| Head Quarters | 1 |
| | <hr/> |
| | 30 |

Thus the maximum number of concurrent connections to the server would be 30. This is highly unlikely to occur.

The CRAFT/com daemon is to be run out of Linux's inetd, or internet daemon. The daemon can therefore be spawned as many times as needed. For each spawned, or child process, a shared memory copy of Perl is used, minimising resident memory usage. The risk of this scheme is that should a single spawned process die, all other processes will die. Should stability be a problem, CRAFT/serv would have enough physical and virtual memory to allow for a separate protected copy of Perl to be spawned for each incoming connection.

Transaction volumes on the database would vary between 58 per second as an upper limit and less than one per hour at the lower limit.

Traffic Volumes

CRAFT/client to CRAFT/serv traffic would exhibit the following characteristics and rules:

The largest single database synchronisation is the sponsor datastore synchronisation. The sponsor datastore needs to be held locally at all stations and can grow to many thousands of people. Many would assume that operational volumes would be the largest; however, they are unlikely to exceed 365 incidents per station per year.

To synchronise the sponsor datastore on CRAFT/client, CRAFT/serv must first determine whether an update is necessary, using the MD5 checksum comparison as described in the CRAFT/com section.

If the checksums do not match, a full comparison of all primary keys and their serial numbers is done. At an upper limit of 10 000 sponsors, the data transfer from CRAFT/serv to CRAFT/client would look as shown in Table (3):

| Data | Column | Type | Size |
|---------------|---------------|-------------|-------------|
| Primary Key | IntSponsorNo | Integer | 4 bytes |
| Serial Number | IntSerial | Integer | 4 bytes |
| | | Total | 8 bytes |

Table 3 – Data Transfer

At 10 000 records and 8 bytes per record, a total of 78KB is transferred from CRAFT/serv to CRAFT/client. At 28.8Kb, a modem standard, this would take 21 seconds to transfer.

Operational transfers (i.e. operations reports) would be under 10KB per report per station per day.

CRAFT/CLIENT CODING STANDARDS

CRAFT/client will be coded according to the coding standards in this document. This is to ensure uniformity between various members of the team, and allows for easy upgrading of the system by a different development team at a later stage.

The coding standards to be employed during Visual Basic [VB] development are an amalgamation of the standards suggested by Microsoft as well as the experience of the coding team.

Object Naming Conventions

Objects will be named with a consistent prefix that makes it easy to identify the type of object. A table detailing the control prefixes is included in the appendices to this document.

Suggested Prefixes for Data Access Objects

The Object Sync team will use standard prefixes to denote Data Access Objects [DAO]. A table of these prefixes is included in the appendices to this document.

An example of how this would be used is included below:

```
Dim dbBiblio As Database
Dim recPubsInNY As Recordset, strSQLStmt As String
Const DB_READONLY = 4 ' Set constant.

'Open database.
Set dbBiblio = OpenDatabase("BIBLIO.MDB") ' Set text for the SQL
statement.
strSQLStmt = "SELECT * FROM Publishers WHERE _
    State = 'NY'"

' Create the new Recordset object.
Set recPubsInNY = db.OpenRecordset(strSQLStmt, _
    dbReadOnly)
```

Prefixes for Menus

Menu control prefixes will be extended beyond the initial "mnu" label by adding an additional prefix for each level of nesting, with the final menu caption at the end of the name string. The following table lists some examples.

| Menu caption sequence | Menu handler name |
|-----------------------|--------------------|
| File Open | mnuFileOpen |
| File Send Email | mnuFileSendEmail |
| File Send Fax | mnuFileSendFax |
| Format Character | mnuFormatCharacter |
| Help Contents | mnuHelpContents |

Table 4 – Menu Control Prefix Examples

When this naming convention is used, all members of a particular menu group are listed next to each other in VB's Properties window. In addition, the menu control names clearly document the menu items to which they are attached.

Prefixes for Other Controls

For controls not listed above, the development team will standardise on a unique two or three character prefix for consistency. More than three characters will only be employed if necessary for clarity.

For derived or modified controls, for example, one would extend the prefixes above so that there is no confusion over which control is really being used. For third-party controls, a lower-case abbreviation for the manufacturer could be added to the prefix. For example, a control instance created from the Visual Basic Professional 3D frame could use a prefix of `fra3d` to avoid confusion over which control is being used.

Structured Coding Conventions

In addition to naming conventions, structured coding conventions, such as code commenting and consistent indenting, can greatly improve code readability.

Code Commenting Conventions

All procedures and functions will begin with a brief comment describing the functional characteristics of the procedure (i.e. what the function does). This description should not describe the implementation details (i.e. how the function does what it does) because these often change over time, resulting in unnecessary comment maintenance work, or worse yet, erroneous comments. The code itself and any necessary inline comments will describe the implementation.

Arguments passed to a procedure should be described when their functions are not obvious and when the procedure expects the arguments to be in a specific range. Function return values and global variables that are changed by the procedure, especially through reference arguments, must also be described at the beginning of each procedure.

Procedure header comment blocks should include the section headings as per the following table.

| Section heading | Comment description |
|-----------------|--|
| Purpose | What the procedure does (not how). |
| Assumptions | List of each external variable, control, open file, or other element that is not obvious. |
| Effects | List of each affected external variable, control, or file and the effect it has (only if this is not obvious). |
| Inputs | Each argument that may not be obvious. Arguments are on a separate line with inline comments. |
| Returns | Explanation of the values returned by functions. |

Table 5 – Section Headings in VB Code

The following commenting conventions will be used:

- Every important variable declaration will have an inline comment describing the use of the variable being declared.
- Variables, controls, and procedures will be named clearly enough such that inline commenting is only needed for complex implementation details.

An example of this can be seen in the code stub in the following section.

At the start of the .bas module that contains the project's Visual Basic generic constant declarations, developers will include an overview that describes the application, enumerating primary data objects, procedures, algorithms, dialogs, databases, and system dependencies. Sometimes a piece of pseudocode describing the algorithm will also be included.

Code Formatting

The code formatting will use the following conventions:

- Standard, tab-based, nested blocks will be indented four spaces (a de facto standard).
- The functional overview comment of a procedure will be indented one space. The highest level statements that follow the overview comment should be indented one tab, with each nested block indented an additional tab. An example is included below:

```

'*****
' Purpose:   Locates the first occurrence of a
'           specified user in the UserList array.
' Inputs:
'   strUserList():  the list of users to be searched.
'   strTargetUser:  the name of the user to search for.
' Returns:   The index of the first occurrence of the
'           rsTargetUser in the rasUserList array.
'           If target user is not found, return -1.
'*****

Function intFindUser (strUserList() As String, strTargetUser As _
    String)As Integer
    Dim i As Integer          ' Loop counter.
    Dim blnFound As Integer   ' Target found flag.
    intFindUser = -1
    i = 0
    While i <= Ubound(strUserList) and Not blnFound
        If strUserList(i) = strTargetUser Then
            blnFound = True
            intFindUser = i
        End If
    Wend
End Function

```

Code Stub 1 – Code Formatting

Grouping Constants

Variables and defined constants should be grouped by function rather than split into isolated areas or special files. Visual Basic generic constants should be grouped in a single module to separate them from application-specific declarations.

Creating Strings for MsgBox, InputBox, and SQL Queries

When creating a long string, the underscore line-continuation character will be used to create multiple lines of code for easier reading and debugging. This technique is particularly useful when displaying a message box (MsgBox) or input box (InputBox) or when creating an SQL string. An example of this is in the code stub below.

```

Dim Msg As String
Msg = "This is a paragraph that will be " _
& "in a message box. The text is" _
& " broken into several lines of code" _

```

```

& " in the source code, making it easier" _
& " for the programmer to read and debug."
MsgBox Msg

Dim QRY As String
QRY = "SELECT *" _
& " FROM Titles" _
& " WHERE [Year Published] > 1988"
TitlesQry.SQL = QRY
    
```

Code Stub 2 – Underscore Line Continuation

Constant and Variable Naming Conventions

In addition to objects, constants and variables also require well-formed naming conventions. Variables will always be defined with the smallest scope possible. Global (public) variables can create enormously complex state machines and make the logic of an application extremely difficult to understand and will therefore be avoided if at all possible.

The scope of variables within Visual Basic is listed in the table below:

| Scope | Declaration | Visible in |
|-----------------|---|--|
| Procedure-level | 'Private' in procedure, sub, or function | The procedure in which it is declared |
| Module-level | 'Private' in the declarations section of a form or code module (.frm, .bas) | Every procedure in the form or code module |
| Global | 'Public' in the declarations section of a code module (.bas) | Everywhere in the application |

Table 6 – Scope of Variables in Visual Basic

In CRAFT/client, global variables will be used only when there is no other convenient way to share data between forms. When global variables must be used, it is good practice to declare them all in a single module, grouped by function.

Give the module a meaningful name that indicates its purpose, such as Public.bas.

It is good coding practice to write modular code wherever possible. For example, if CRAFT/client displays a dialog box, all the controls and code required to perform the dialog's task will be placed in a single Visual Basic form. This helps to keep the

application's code organized into useful components and minimizes its run-time overhead.

With the exception of global variables (which should not be passed), procedures and functions should operate only on objects passed to them. Global variables that are used in procedures should be identified in the declaration section at the beginning of the procedure. In addition, Object Sync will pass arguments to subs and functions using ByVal, unless there is an explicit need to change the value of the passed argument.

Variable Scope Prefixes

As project size grows, so does the value of recognizing variable scope quickly. A one-letter scope prefix preceding the type prefix provides this, without greatly increasing the size of variable names. This will be implemented in order to provide additional scalability to the CRAFT Solution.

The table below indicates how variables will be prefixed to indicate their scope.

| Scope | Prefix | Example |
|--------------------|---------------|--------------------|
| Global | g | gstrUserName |
| Module-level | m | mblnCalcInProgress |
| Local to procedure | None | dblVelocity |

Table 7 – Variable Prefixes to Determine Variable Types

A variable has global scope if it is declared Public in a standard module or a form module. A variable has *module-level* scope if declared Private in a standard module or form module, respectively.

It is important to note that consistency is crucial to productive use of this technique; the syntax checker in Visual Basic will not catch module-level variables that begin with "p".

Constants

The body of constant names will be mixed case with capitals initiating each word. Although standard Visual Basic constants do not include data type and scope information, prefixes like i, s, g, and m can be very useful in understanding the value

and scope of a constant. For constant names, the same rules as variables will be implemented.

An example of this is presented in the following code stub:

```

mintUserListMax      'Max entry limit for User list
                    '(integer value,local to module)
gstrNewLine          'New Line character
                    '(string, global to application)
    
```

Code Stub 3 – Constant Naming

Variables

Declaring all variables saves programming time by reducing the number of bugs caused by typographical errors (for example, aUserNameTmp versus sUserNameTmp vs. sUserNameTemp). On the Editor tab of the Options dialog, check the Require Variable Declaration option. The Option Explicit statement requires that you declare all the variables in your Visual Basic program.

Variables will be prefixed to indicate their data type. Optionally, especially for large programs, the prefix could be extended to indicate the scope of the variable.

Variable Data Types

Prefixes to indicate a variable's data type will be used as per the table below.

| Data type | Prefix | Example |
|-------------------|---------------|----------------|
| Boolean | bln | BlnFound |
| Byte | byt | BytRasterData |
| Collection object | col | ColWidgets |
| Currency | cur | CurRevenue |
| Date (Time) | dtm | DtmStart |
| Double | dbl | DblTolerance |
| Error | err | ErrOrderNum |
| Integer | int | IntQuantity |
| Long | lng | LngDistance |
| Object | obj | ObjCurrent |
| Single | sng | SngAverage |
| String | str | StrFName |

| | | |
|-------------------|-----|-------------|
| User-defined type | udt | udtEmployee |
| Variant | vnt | vntCheckSum |

Table 8 – Variable Prefixes to Indicate Data Types

Descriptive Variable and Procedure Names

The body of a variable or procedure name will use mixed case and will be as long as necessary to describe its purpose. In addition, function names will begin with a verb, such as InitNameArray or CloseDialog.

For frequently used or long terms, standard abbreviations will be used to keep name lengths reasonable. When using abbreviations, the development team will make sure they are consistent throughout CRAFT/client. Randomly switching between Cnt and Count within a project will lead to unnecessary confusion.

User-Defined Types

In a large project with many user-defined types, it is often useful to give each user-defined type a three-character prefix of its own. If these prefixes begin with "u", they will still be easy to recognize quickly when you are working with a user-defined type. For example, "ucli" could be used as the prefix for variables of a user-defined Client type. This technique will be adopted during the development of CRAFT/client.

CRAFT/SERV CODING STANDARDS

CRAFT/serv will primarily be coded using Perl. Perl is an interpreted structured C-like programming language with very strong string manipulation capabilities. Perl supports modules (loadable programme packages) and objects with multiple inheritance and virtual methods.

Object Naming Conventions

The CRAFT/serv will not use many Perl object structures apart from those interfaces provided by the DBI/DBD and XML modules. It is not necessary to use full object orientated design for this project as code reuse and modularity can be achieved more effectively with the use of functions.

The object naming conventions used will therefore be those supplied by the module provider. At this stage of the design, final Perl module selection has not been finalised. These naming conventions will be included in the final documentation in order to provide for supplementary modification.

Coding Method

All Perl scripts must be written using the *strict* convention. This forces the programmer to always declare all variables and also disallows accidental symbolic dereferencing.

Structured CRAFT/serv Coding Conventions

Code Commenting Conventions

Code commenting will be implemented using the same techniques that will be employed when coding CRAFT/client.

Procedure header comment blocks should include headings as detailed in the table below:

| Section heading | Comment description |
|------------------------|----------------------------|
|------------------------|----------------------------|

| | |
|-------------|--|
| Purpose | What the procedure does (not how). |
| Assumptions | List of each external variable, control, open file, or other element that is not obvious. |
| Effects | List of each affected external variable, control, or file and the effect it has (only if this is not obvious). |
| Inputs | Each argument that may not be obvious. Arguments are on a separate line with inline comments. |
| Returns | Explanation of the values returned by functions. |

Table 9 – CRAFT/serv Procedure Header Commenting

The following commenting conventions will be used:

- Every important variable declaration will have an inline comment describing the use of the variable being declared.
- Variables and procedures will be named clearly enough such that inline commenting is only needed for complex implementation details.

Arguments passed to a procedure should be described when their functions are not obvious and when the procedure expects the arguments to be in a specific range. Function return values and global variables that are changed by the procedure, especially through reference arguments, must also be described at the beginning of each procedure.

Revision Control

All Perl functions will be placed under a revision control system [RCS]. Change logs and locking will be controlled via this mechanism.

Code Formatting

The code formatting will use the follow the following conventions:

- Standard, tab-based, nested blocks should be indented four spaces (a defacto standard).
- The functional overview comment of a procedure should be indented one space. The highest level statements that follow the overview comment should be indented one tab, with each nested block indented an additional tab.

Constant and Variable Naming Conventions

Perl has three data structures: scalars, arrays of scalars, and associative arrays of scalars, known as "hashes". Normal arrays are indexed by number, starting with 0. (Negative subscripts count from the end.) Hash arrays are indexed by string.

Perl is typeless, that is, data structures are not explicitly string, integer, long etc. The method that one uses to address the structure determines how it behaves. As such, specific naming conventions for variables are not necessary. The different structures are already differentiated by the syntax of Perl.

Variables will always be defined with the smallest scope possible. Global (Public) variables can create enormously complex state machines and make the logic of an application extremely difficult to understand and will therefore be avoided if at all possible.

Descriptive Variable and Procedure Names

Descriptive variables and procedures will be named using the same techniques that will be employed when coding CRAFT/client.

Y2K COMPLIANCE

A complete Year 2000 Compliance statement is available in the *User Requirement Specification*. In this section one will find details relevant to the development of the CRAFT Solution.

The Development Cycle

During the course of the development cycle, Object Sync will be monitoring vendor statements with respect to the compliance of their products. Wherever possible, patches and upgrades will be downloaded and installed across the development network. This applies to various desktop and development tools.

The development network is based on a non-date compliant protocol, TCP/IP, and requires no specialised effort to ensure year 2000 compliance.

Coding Standards

Although more detail on coding standards can be found elsewhere in this document, it is important to note that all dates will be coded in four digit format.

APPENDICES

Appendices that have not been referenced during the *System Specifications* have been included at the end of the appendices.

All models are numbered using a system similar to domain name server [DNS] zone file serial numbering for certificate of authority [SOA] records: YYYY-MM-DD-xxx where:

YYYY = year

MM = month

DD = day

XXX = model revision as per the day

Technical Specification Amendment

The *Technical Specification Amendment* form differs from the *Configuration Change Acceptance* form found in the *User Requirement Specification*. The NSRI does not consider the specific technical elements of the solution critical and has entrusted the development team to take superficial technical decisions. Conceptual changes to the CRAFT solution, at a technical level, will be monitored using the *Configuration Change Acceptance* form. Specific technical amendments will be monitored using the internal *Technical Specification Amendment* form.

Technical Specification Amendment

Date _____

Change to CRAFT/_____

Please detail the nature of the change...

Please detail the intention behind this change...

This change has been accepted by

PLEASE SIGN YOUR NAME HERE

Daniel Chalef / Jason du Preez / Khetan Gajjar /David Reinhardt / Eden
Ridgway on behalf of Object Sync.

Process Models

Process models (data flow diagrams) show the relationship between processes within the system, the information that flows between them and the stores within the system.

These process models employ the Gane and Sarson Notation.

The data flow diagrams were developed using Visio Professional 5.

The enclosed data flow diagrams include

- Context Diagram [Model 1999-04-28-010]
 - High Level DFD [Model 1999-04-28-010]
 - Personnel Records DFD [Model 1999-04-28-010]
 - GYPSY Book DFD [Model 1999-04-28-010]
 - Operations DFD [Model 1999-04-28-010]
 - Reports DFD [Model 1999-04-28-010]
 - Maintenance DFD [Model 1999-04-28-010]
 - System Admin DFD [Model 1999-04-28-010]

The Data Models

The data dictionary lists all tables within the database and the fields contained in the tables. The data dictionary also includes estimations of data sizes.

The table below details the columns in the data dictionary:

| | |
|-------------|---|
| Column | The field name. |
| Type | The data type. |
| Default | The default value. |
| Allow Null | Whether the field is allowed to contain a null value. |
| Constraints | Whether any constraints have been placed on the field (e.g. if the data value could be Yes or No). |
| Storage | The amount of space each record uses (in bytes) |
| Total Each | The total number of bytes per record for the entire table. |
| No Local | Estimated number of records that the table will hold in CRAFT/db1 |
| No Total | Estimated number of records in CRAFT/db |
| Size Local | Size that CRAFT/db1 will use in kilobytes |
| Size Total | Size that CRAFT/db will use in kilobytes |

Table 10 – Columns in the Data Dictionary

The data dictionary has been developed using Microsoft Excel '97.

The data dictionary model number is Data Dictionary 1999-06-13-000.

The Entity Relationship Diagram [ERD] models the relationships between tables within the database.

The ERD employs traditional crow's foot ERD notation.

The ERD has been developed using Visio Professional 5.

The ERD is model number 1999-06-13-000

| Operation | | | | | Storage | Total | Each | No Local | No Total | Size Local | Size Total |
|---------------------|-----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|
| Column | Type | Default | Allow Null | Constraints | | | | | | | |
| IngOpNo | long | autoincrement | N | | 4 | | | | | | |
| IngReportFiledBy | long | (none) | N | | 4 | | | | | | |
| IngStationNo | long | (none) | N | | 4 | | | | | | |
| dteCallDate | Date | Today | Y | | 8 | | | | | | |
| tmeCallTime | Time | Now | Y | | 8 | | | | | | |
| tmsReportTime | TimeStamp | Now | Y | | 8 | | | | | | |
| strCalledBy | char(255) | (none) | Y | | 265 | | | | | | |
| chrExOperation | Byte | O | Y | E/O | 1 | | | | | | |
| chrWriteforDonation | Byte | Y | Y | Y/N | 1 | | | | | | |
| strWFDRReason | char(255) | (none) | Y | | 265 | | | | | | |
| strStatComRemarks | char(255) | (none) | Y | | 265 | | | | | | |
| chrPressNotified | Byte | N | Y | Y/N | 1 | | | | | | |
| chrPublished | Byte | N | Y | Y/N | 1 | | | | | | |
| chrSeenAwardsCom | Byte | N | Y | Y/N | 1 | | | | | | |
| chrSeenOpsHead | Byte | N | Y | Y/N | 1 | | | | | | |
| chrSeenStatCom | Byte | N | Y | Y/N | 1 | | | | | | |
| chrSeenFundRaising | Byte | N | Y | Y/N | 1 | | | | | | |
| intBoatsTowed | Integer | (none) | Y | | 2 | | | | | | |
| intBoatsHelped | Integer | (none) | Y | | 2 | | | | | | |
| intLifesSaved | Integer | (none) | Y | | 2 | | | | | | |
| intPersonsAssisted | Integer | (none) | Y | | 2 | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | |
| dbiSerial | long | | | ddmmyyy999 | 4 | | | | | | |
| | | | | | | 852 | 60 | 6000 | | 540 | 54000 |

| Person_Rescued | | | | | Storage | Total | Each | No Local | No Total | Size Local | Size Total |
|-----------------|-----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|
| Column | Type | Default | Allow Null | Constraints | | | | | | | |
| IngPersRecNo | long | autoincrement | N | | 4 | | | | | | |
| IngOpNo | long | (none) | N | | 4 | | | | | | |
| IngContactNo | long | (none) | N | | 4 | | | | | | |
| strLastName | char(40) | (none) | Y | | 50 | | | | | | |
| strFirstNames | char(40) | (none) | Y | | 50 | | | | | | |
| tmsReportTime | TimeStamp | Now | Y | | 8 | | | | | | |
| strStreet | char(30) | (none) | Y | | 40 | | | | | | |
| strSuburb | char(30) | (none) | Y | | 40 | | | | | | |
| strCity | char(30) | (none) | Y | | 40 | | | | | | |
| strProvince | char(30) | (none) | Y | | 40 | | | | | | |
| strPostCode | char(20) | (none) | Y | | 30 | | | | | | |
| strCountry | char(30) | ZA | Y | | 40 | | | | | | |
| chrLifeJacket | Byte | (none) | Y | Y/N | 1 | | | | | | |
| chrMedicalTreat | Byte | (none) | Y | Y/N | 1 | | | | | | |
| strComment | char(255) | (none) | Y | Y/N | 265 | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | |
| dbiSerial | double | | N | ddmmyyy999 | 8 | | | | | | |
| | | | | | | | 120 | 12000 | | 840 | 84000 |

| Asset | | | | | Storage | Total | Each | No Local | No Total | Size Local | Size Total |
|----------------|-----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|
| Column | Type | Default | Allow Null | Constraints | | | | | | | |
| IngAssetNo | long | autoincrement | N | | 4 | | | | | | |
| IngAssetTypeNo | long | (none) | N | | 4 | | | | | | |
| IngStationNo | long | (none) | N | | 4 | | | | | | |
| strDescription | char(100) | (none) | Y | | 110 | | | | | | |
| strName | char(30) | (none) | Y | | 40 | | | | | | |
| strRegNo | char(30) | (none) | Y | | 40 | | | | | | |
| strSource | char(50) | (none) | Y | | 60 | | | | | | |
| dbiUsage | double | (none) | Y | | 8 | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | |
| dbiSerial | double | | N | ddmmyyy999 | 8 | | | | | | |
| | | | | | | | 20 | 300 | | 60 | 900 |

| Asset_Type | | | | | Storage | Total | Each | No Local | No Total | Size Local | Size Total |
|----------------|-----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|
| Column | Type | Default | Allow Null | Constraints | | | | | | | |
| IngAssetTypeNo | long | autoincrement | N | | 4 | | | | | | |
| strDescription | char(100) | (none) | N | | 110 | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | |
| dbiSerial | double | | N | ddmmyyy999 | 8 | | | | | | |
| | | | | | | | 20 | 20 | | 40 | 40 |

| Asset_Maintenance | | | | | Storage | Total | Each | No Local | No Total | Size Local | Size Total |
|-------------------|-----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|
| Column | Type | Default | Allow Null | Constraints | | | | | | | |
| IngAssetMaintNo | long | autoincrement | N | | 4 | | | | | | |
| IngAssetNo | long | (none) | N | | 4 | | | | | | |
| chrPurchaseNo | char(30) | (none) | N | | 40 | | | | | | |
| dbiAmount | double | (none) | Y | | 8 | | | | | | |
| strReason | char(255) | (none) | Y | | 265 | | | | | | |
| strSupplier | char(30) | (none) | Y | | 40 | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | |
| dbiUsage | double | (none) | Y | | 8 | | | | | | |
| dbiSerial | double | | N | ddmmyyy999 | 8 | | | | | | |
| | | | | | | 400 | 6000 | | | 1600 | 24000 |

| Operation_Asset | | | | | Storage | Total | Each | No Local | No Total | Size Local | Size Total |
|------------------|-----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|
| Column | Type | Default | Allow Null | Constraints | | | | | | | |
| IngOpAssetNo | long | autoincrement | N | | 4 | | | | | | |
| IngAssetNo | long | (none) | N | | 4 | | | | | | |
| IngCoxonPersonNo | long | (none) | N | | 4 | | | | | | |
| IngOpNo | long | (none) | N | | 4 | | | | | | |
| dbiHours | double | (none) | Y | | 8 | | | | | | |
| strDamage | char(255) | (none) | Y | | 265 | | | | | | |
| strActionTaken | char(255) | (none) | Y | | 265 | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | |

| | | | | | | | | | | | | |
|-----------|--------|---|-------------|---|-----|-------|-----|-------|--|--|--|--|
| dblSerial | double | N | ddmmyyyy999 | 8 | | | | | | | | |
| | | | | | 120 | 12000 | 720 | 72000 | | | | |

| Operation_Person | | | | | | | | | | | | |
|------------------|--------|---------------|-------------|-------------|---------|-------|------|----------|----------|------------|------------|--|
| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total | |
| IngOpPersNo | long | autoincrement | N | | 4 | | | | | | | |
| IngOpAssetNo | long | (none) | N | | 4 | | | | | | | |
| IngPersNo | long | (none) | N | | 4 | | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | | |
| dblSerial | double | N | ddmmyyyy999 | 8 | | | | | | | | |
| | | | | | 300 | 30000 | 300 | 30000 | | | | |

| Person | | | | | | | | | | | | |
|-----------------|----------|---------------|-------------|-------------|---------|-------|------|----------|----------|------------|------------|--|
| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total | |
| IngPersNo | long | autoincrement | N | | 4 | | | | | | | |
| IngStationNo | long | (none) | N | | 4 | | | | | | | |
| IngContactNo | long | (none) | N | | 4 | | | | | | | |
| strLastName | char(40) | (none) | Y | | 50 | | | | | | | |
| strFirstNames | char(40) | (none) | Y | | 50 | | | | | | | |
| strIDNo | char(40) | (none) | Y | | 50 | | | | | | | |
| strStreet | char(30) | (none) | Y | | 40 | | | | | | | |
| strSuburb | char(30) | (none) | Y | | 40 | | | | | | | |
| strCity | char(30) | (none) | Y | | 40 | | | | | | | |
| strProvince | char(30) | (none) | Y | | 40 | | | | | | | |
| strPostCode | char(20) | (none) | Y | | 30 | | | | | | | |
| strCountry | char(30) | ZA | Y | | 40 | | | | | | | |
| intInitialHours | integer | (none) | Y | | 2 | | | | | | | |
| chrActive | Byte | Y | Y | Y/N | 1 | | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | | |
| dblSerial | double | N | ddmmyyyy999 | 8 | | | | | | | | |
| | | | | | 404 | 100 | 1500 | 500 | 7500 | | | |

| Person_Qualification | | | | | | | | | | | | |
|----------------------|----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|--|
| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total | |
| IngPersQualNo | long | autoincrement | N | | 4 | | | | | | | |
| IngPersNo | long | (none) | N | | 4 | | | | | | | |
| IngQualTypeNo | long | (none) | N | | 4 | | | | | | | |
| dteDateRec | date | (none) | Y | | 8 | | | | | | | |
| dteDateExp | date | (none) | Y | | 8 | | | | | | | |
| strAuthority | char(30) | (none) | Y | | 40 | | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | | |
| dblSerial | double | (none) | N | ddmmyyyy999 | 8 | | | | | | | |
| | | | | | 500 | 8000 | 500 | 8000 | | | | |

| Qualification_Type | | | | | | | | | | | | |
|--------------------|-----------|---------------|-------------|-------------|---------|-------|------|----------|----------|------------|------------|--|
| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total | |
| IngQualTypeNo | long | autoincrement | N | | 4 | | | | | | | |
| strDescription | char(255) | (none) | N | | 265 | | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | | |
| dblSerial | double | N | ddmmyyyy999 | 8 | | | | | | | | |
| | | | | | 20 | 20 | 60 | 60 | | | | |

| Vessel_Rescued | | | | | | | | | | | | |
|--------------------|-----------|---------------|-------------|-------------|---------|-------|------|----------|----------|------------|------------|--|
| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total | |
| IngVslRescNo | long | autoincrement | N | | 4 | | | | | | | |
| IngOpNo | long | (none) | N | | 4 | | | | | | | |
| IngSkipperPersonNo | long | (none) | N | | 4 | | | | | | | |
| strDescription | char(255) | (none) | Y | | 265 | | | | | | | |
| chrSkipperCert | Byte | (none) | Y | Y/N | 1 | | | | | | | |
| strCertAuthority | char(30) | (none) | Y | | 40 | | | | | | | |
| chrFlares | Byte | (none) | Y | Y/N | 1 | | | | | | | |
| strRegNo | char(30) | (none) | Y | | 40 | | | | | | | |
| strName | char(50) | (none) | Y | | 60 | | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | | |
| dblSerial | double | N | ddmmyyyy999 | 8 | | | | | | | | |
| | | | | | 30 | 3000 | 150 | 15000 | | | | |

| Operation_Equipment | | | | | | | | | | | | |
|---------------------|-----------|---------------|-------------|-------------|---------|-------|------|----------|----------|------------|------------|--|
| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total | |
| IngOpEquipNo | long | autoincrement | N | | 4 | | | | | | | |
| IngOpNo | long | (none) | N | | 4 | | | | | | | |
| strDescription | char(255) | (none) | Y | | 265 | | | | | | | |
| chrDamage | Byte | N | Y | Y/N | 1 | | | | | | | |
| strComment | char(30) | (none) | Y | | 40 | | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | | |
| dblSerial | double | N | ddmmyyyy999 | 8 | | | | | | | | |
| | | | | | 180 | 18000 | 720 | 72000 | | | | |

| Station | | | | | | | | | | | | |
|--------------|----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|--|
| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total | |
| IngStationNo | long | autoincrement | N | | 4 | | | | | | | |
| IngContactNo | long | (none) | N | | 4 | | | | | | | |
| strName | char(30) | (none) | Y | | 40 | | | | | | | |
| strStreet | char(30) | (none) | Y | | 40 | | | | | | | |
| strSuburb | char(30) | (none) | Y | | 40 | | | | | | | |
| strCity | char(30) | (none) | Y | | 40 | | | | | | | |
| strProvince | char(30) | (none) | Y | | 40 | | | | | | | |
| strPostCode | char(20) | (none) | Y | | 30 | | | | | | | |
| strCountry | char(30) | ZA | Y | | 40 | | | | | | | |
| strRegion | char(30) | (none) | Y | | 40 | | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | | |
| dblSerial | double | (none) | N | ddmmyyyy999 | 8 | | | | | | | |
| | | | | | 3 | 500 | 12 | 2000 | | | | |

| Organisation | | | | | | | | | | | | | |
|----------------|-----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|----|------|
| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total | | |
| IngOrgNo | long | autoincrement | N | | 4 | | | | | | | | |
| IngStatNo | long | (none) | Y | | 4 | | | | | | | | |
| IngContactNo | long | (none) | N | | 4 | | | | | | | | |
| IngOrgTypeNo | long | (none) | N | | 4 | | | | | | | | |
| strName | char(50) | (none) | Y | | 60 | | | | | | | | |
| strContactName | char(40) | (none) | Y | | 50 | | | | | | | | |
| strStreet | char(30) | (none) | Y | | 40 | | | | | | | | |
| strSuburb | char(30) | (none) | Y | | 40 | | | | | | | | |
| strCity | char(30) | (none) | Y | | 40 | | | | | | | | |
| strProvince | char(30) | (none) | Y | | 40 | | | | | | | | |
| strPostCode | char(20) | (none) | Y | | 30 | | | | | | | | |
| strCountry | char(30) | ZA | Y | | 40 | | | | | | | | |
| strEmail | char(40) | (none) | Y | | 50 | | | | | | | | |
| strComments | char(255) | (none) | Y | | 265 | | | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | | | |
| dblSerial | double | (none) | N | ddmmyyyy999 | 8 | | | | | | | | |
| | | | | | | | | | | 10 | 1000 | 70 | 7000 |

| Audit | | | | | | | | | | | | | |
|----------------|-----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|-------|--------|
| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total | | |
| IngAuditNo | long | autoincrement | N | | 4 | | | | | | | | |
| strDescription | char(30) | (none) | N | | 40 | | | | | | | | |
| IngStationNo | long | (none) | N | | 4 | | | | | | | | |
| IngRefNo | long | (none) | N | | 4 | | | | | | | | |
| strAction | char(20) | (none) | Y | | 30 | | | | | | | | |
| tmsTimeStamp | TimeStamp | (none) | Y | | 8 | | | | | | | | |
| dblSerial | double | (none) | N | ddmmyyyy999 | 8 | | | | | | | | |
| | | | | | | | | | | 5000 | 50000 | 10000 | 100000 |

| Insurance | | | | | | | | | | | | | |
|----------------|-----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|----|------|
| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total | | |
| IngInsuranceNo | long | autoincrement | N | | 4 | | | | | | | | |
| strName | char(50) | (none) | Y | | 60 | | | | | | | | |
| strContactName | char(40) | (none) | Y | | 50 | | | | | | | | |
| strStreet | char(255) | (none) | Y | | 265 | | | | | | | | |
| strSuburb | Byte | (none) | Y | | 1 | | | | | | | | |
| strCity | Byte | (none) | Y | | 1 | | | | | | | | |
| strProvince | char(30) | (none) | Y | | 40 | | | | | | | | |
| strPostCode | char(255) | (none) | Y | | 265 | | | | | | | | |
| strCountry | char(255) | ZA | Y | | 265 | | | | | | | | |
| strTelephone | char(30) | (none) | Y | | 40 | | | | | | | | |
| strFax | char(30) | (none) | Y | | 40 | | | | | | | | |
| strEmail | char(40) | (none) | Y | | 50 | | | | | | | | |
| strComments | char(255) | (none) | Y | | 265 | | | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | | | |
| dblSerial | double | (none) | N | ddmmyyyy999 | 8 | | | | | | | | |
| | | | | | | | | | | 5 | 100 | 70 | 1400 |

| StatCom_Report | | | | | | | | | | | | | |
|-----------------|-----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|-----|-------|
| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total | | |
| IngStatComRepNo | long | autoincrement | N | | 4 | | | | | | | | |
| IngStationNo | long | (none) | N | | 4 | | | | | | | | |
| dteDate | date | Today | Y | | 8 | | | | | | | | |
| dblFuel | double | (none) | Y | | 8 | | | | | | | | |
| dblOil | double | (none) | Y | | 8 | | | | | | | | |
| strCallsNotOps | integer | (none) | Y | | 2 | | | | | | | | |
| strComment | char(255) | (none) | Y | | 265 | | | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | | | |
| dblSerial | double | (none) | N | ddmmyyyy999 | 8 | | | | | | | | |
| | | | | | | | | | | 60 | 6000 | 240 | 24000 |

| Contact_Reference | | | | | | | | | | | | | |
|-------------------|----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|-----|-------|
| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total | | |
| IngContactRefNo | long | autoincrement | N | | 4 | | | | | | | | |
| IngContactTypeNo | long | (none) | N | | 4 | | | | | | | | |
| IngReferenceNo | long | (none) | Y | | 4 | | | | | | | | |
| IngReferenceType | long | (none) | Y | | 4 | | | | | | | | |
| strCountryCode | char(10) | (none) | Y | | 20 | | | | | | | | |
| strCode | char(10) | (none) | Y | | 20 | | | | | | | | |
| strNumber | char(30) | (none) | Y | | 40 | | | | | | | | |
| dblSerial | double | (none) | N | ddmmyyyy999 | 8 | | | | | | | | |
| | | | | | | | | | | 300 | 15000 | 600 | 30000 |

| Organisation_Type | | | | | | | | | | | | | |
|-------------------|----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|----|----|
| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total | | |
| IngOrgTypeNo | long | autoincrement | N | | 4 | | | | | | | | |
| strDescription | char(30) | (none) | Y | | 40 | | | | | | | | |
| dblSerial | double | (none) | N | ddmmyyyy999 | 8 | | | | | | | | |
| | | | | | | | | | | 30 | 30 | 30 | 30 |

| Contact_Type | | | | | | | | | | | | | |
|------------------|----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|----|----|
| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total | | |
| IngContactTypeNo | long | autoincrement | N | | 4 | | | | | | | | |
| strDescription | char(30) | (none) | Y | | 40 | | | | | | | | |
| dblSerial | double | (none) | N | ddmmyyyy999 | 8 | | | | | | | | |
| | | | | | | | | | | 30 | 30 | 30 | 30 |

Province

| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total |
|-------------|----------|---------|------------|-------------|---------|-------|------|----------|----------|------------|------------|
| strProvince | char(30) | (none) | N | | 40 | | | | | | |
| dblSerial | double | (none) | N | ddmmyyyy999 | 8 | | | | | | |
| | | | | | | | | 30 | 30 | 30 | 30 |

User

| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total |
|--------------|----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|
| lngUserNo | long | autoincrement | N | | 4 | | | | | | |
| lngStationNo | long | integer | N | | 4 | | | | | | |
| strUserName | char(30) | (none) | Y | | 40 | | | | | | |
| strPassword | char(30) | (none) | Y | | 40 | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | |
| dblSerial | double | (none) | N | ddmmyyyy999 | 8 | | | | | | |
| | | | | | | | | 5 | 200 | 10 | 400 |

Function_List

| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total |
|----------------|----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|
| lngFunctionNo | long | autoincrement | N | | 4 | | | | | | |
| strDescription | char(30) | (none) | Y | | 40 | | | | | | |
| dblSerial | double | (none) | N | ddmmyyyy999 | 8 | | | | | | |
| | | | | | | | | 300 | 300 | 300 | 300 |

User_Function

| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total |
|---------------|--------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|
| lngFunctionNo | long | autoincrement | N | | 4 | | | | | | |
| lngUserNo | long | (none) | N | | 4 | | | | | | |
| dblSerial | double | (none) | N | ddmmyyyy999 | 8 | | | | | | |
| | | | | | | | | 1500 | 60000 | 1500 | 60000 |

Group

| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total |
|------------|--------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|
| lngGroupNo | long | autoincrement | N | | 4 | | | | | | |
| lngUserNo | long | (none) | N | | 4 | | | | | | |
| dblSerial | double | (none) | N | ddmmyyyy999 | 8 | | | | | | |
| | | | | | | | | 5 | 20 | 5 | 20 |

Station_Registration

| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total |
|--------------|------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|
| lngStationNo | long | autoincrement | N | | 4 | | | | | | |
| intRegKey | long | (none) | Y | | 4 | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | |
| | | | | | | | | 1 | 50 | 1 | 50 |

Station_Table_Synch

| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total |
|------------------|----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|
| lngStationNo | long | autoincrement | N | | 4 | | | | | | |
| lngRegKey | long | (none) | Y | | 4 | | | | | | |
| strQualification | char(40) | (none) | Y | | 50 | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | |
| dblSerial | double | (none) | N | ddmmyyyy999 | 8 | | | | | | |
| | | | | | | | | 1 | 50 | 1 | 50 |

System_Message

| Column | Type | Default | Allow Null | Constraints | Storage | Total | Each | No Local | No Total | Size Local | Size Total |
|--------------|-----------|---------------|------------|-------------|---------|-------|------|----------|----------|------------|------------|
| intSysMessNo | long | autoincrement | N | | 4 | | | | | | |
| lngStatNo | long | (none) | Y | | 4 | | | | | | |
| strMessage | char(255) | (none) | Y | | 265 | | | | | | |
| chrRead | Byte | N | Y | | 1 | | | | | | |
| chrArchived | Byte | N | Y | Y/N | 1 | | | | | | |
| dblSerial | double | (none) | N | ddmmyyyy999 | 8 | | | | | | |
| | | | | | | | | 100 | 1000 | 300 | 3000 |

| |
|--------------|
| Key |
| Primary Key |
| Foreign Keys |
| Columns |
| Serial |

18927 595810 k
18.927 595.81 MB

Functional Model

This matrix lists all business functionality and specifies how, if at all, the function affects tables in the database. This matrix is also known as a CRUD diagram: records can be **created**, **read**, **updated** or **deleted**.

The CRUD Diagram has been developed using Microsoft Excel 97.

The CRUD diagram is CRUD 1999-06-13-000

| Function | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | | | | |
|----------------------------|-------------------|---------------------|----------------|----------------|-----------|-----------------|------------------|--------|-------|------------|----------------------|---------------|-------------------|---------|----------------|--------------|---------------------|----------------------|----------------|-------|----------------|-----------|------|-----------------|-------|----|----|----|----|----|----|----|----|----|----|----|--|--|--|--|
| | Donation_Received | Operation_Equipment | Person_Rescued | Vessel_Rescued | Operation | Operation_Asset | Operation_Person | Person | Asset | Asset_Type | Person_Qualification | Qualification | Asset_Maintenance | Station | Deputy_StatCom | Organisation | Station_Table_Synch | Station_Registration | System_Message | Audit | StatCom_Report | Insurance | User | Access_Function | Group | | | | | | | | | | | | | | | |
| Synchronise Data | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | CR | UD | | | | |
| Add Person | | | | | | | | C | | | C | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| View Person Records | | | | | | | R | R | | | R | R | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Edit Person | | | | | | | | U | | | U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Archive Person | | | | | | | | U | | | U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Add Operation | C | C | C | C | C | C | C | R | RU | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Edit Operation | UD | UD | UD | UD | U | UD | UD | R | U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Review Ops Report | R | R | R | R | RU | R | R | R | R | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Archive Operation | U | U | U | U | RU | U | U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Add Asset Maint Record | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Edit Asset Maint Record | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Archive Asset Maint Record | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Add Asset | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Edit Asset | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Archive Asset | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Add Asset Type | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Archive Asset Type | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Add Station | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Edit Station | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Archive Station | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Add Station Sponsor | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Edit Station Sponsor | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Archive Station Sponsor | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Add Donation Received | C | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Edit Donation Received | U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Archive Donation Received | U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Add Qualification | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Edit Qualification | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Archive Qualification | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Add Organisation | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Edit Organisation | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Archive Organisation | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Add User | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Edit User | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Delete User | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Add Access Function | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Edit Access Function | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Delete Access Function | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Add Group | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Edit Group | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Delete Group | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Add System Message | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Edit System Message | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Archive System Message | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Add Insurance | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Edit Insurance | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Archive Insurance | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Add StatCom Report | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Edit StatCom Report | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Archive StatCom Report | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Archive Audit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Delete Audit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Technical Environment Model

This model details the technical architecture of the CRAFT solution.

This model was developed using Visio Professional 5.

The technical environment is model number 1999-04-28-000

Screenshot

This is a mockup of the CRAFT/client user interface with the tree menu fully expanded.

The screenshot was developed using Paint Shop Pro 5.03.

CRAFT - Station 36

File Operations GYPSY Book Assets Crew Reports Admin Online Activities Help

Ops Report Synch Gypsy Book Connect Hang Up

CRAFT Station 36

- Assets
 - Add an Asset
 - Asset Details
 - Asset Reports
 - Edit Asset Details
 - Graph Expenses
 - Record an Expense
 - Request Purchase Order
- Crew
 - Add Crew Member
 - Crew Reports
 - Edit Crew Member
 - Remove Crew Member
 - View Crew Records
- GYPSY Book
 - Edit GYPSY Book
 - Print GYPSY Book
 - View GYPSY Book
- Help
 - Contents
 - How Do I?
 - Index
- Operations
 - File Ops Report
 - Request Filed Ops Report
 - Review Ops Report
- Reports
 - Assets
 - Crew
 - Management
 - File Management Report
 - View Management Report
 - Operations
- Outbox

Add Operation Report

Caller Details:

Call Time:

Called by:

Environment Conditions:

Wind Force: Wind Direction:

Sea State: Visibility:

Surf Height: Distance offshore:


Rescue Statistics:

Vessels used: Mobiles Used:

Persons Rescued: Number of Crew Members:

Description of Operation:

< Back Next > Cancel

 Ops Reports need to be viewed by the Statcom, Ops Manager and fundraising before they are filed. Next Tip

Report Mockups

These are mockups of some basic reports that the system will be able to generate. The mockups are using fictional data.

These mockups were generated using Microsoft Word 97.

The following mockups have been included:

- Basic Report Template [Report Template 1999-06-12-000]
 - Management Report [1999-06-13-001]
 - Crew Report [1999-06-13-000]

Module

Date

Data

Report Title

Page x of x

Granger Bay – Station 16

Statcom: F. Potgieter
Deputy Statcom: L. Hendricks

Operations Summary

| | |
|-------------|----|
| Rescues | 7 |
| Exercises | 12 |
| Shows | 0 |
| Assists | 6 |
| Tow-Ins | 1 |
| Lives Saved | 2 |

Equipment Summary

| | |
|-------------------|------------|
| Boat Repairs | R12 000.00 |
| Equipment Repairs | R626.54 |
| Fuel | 1 234/ |
| Oil | 123/ |

Granger Bay – Station 16
Crew Log – Johannes Pretorius

Statcom: F. Potgieter
Deputy Statcom: L. Hendricks

Hours served

| | |
|------------|------|
| This month | 18 |
| Total | 2145 |

Activity Hours

| | <u>This Month</u> | <u>Total</u> |
|------------|-------------------|--------------|
| Exercises | 6 | 1700 |
| Shows | 1 | 5 |
| Operations | 11 | 440 |

Certifications

| | |
|---------------|------------------------------------|
| June 1997 | Life saving level 2 |
| August 1998 | Life saving level 3 |
| January 1999 | Scuba diving trainer certification |
| February 1999 | Life saving level 4 |

CRAFT/client Coding Standards

- Object Coding Prefixes
- Prefixes for DAO Objects

| Control type | Prefix | Example |
|--|--------|------------------|
| 3D Panel | pnl | pnlGroup |
| ADO Data | ado | adoBiblio |
| Animated button | ani | aniMailBox |
| Check box | chk | chkReadOnly |
| Combo box, drop-down list box | cbo | cboEnglish |
| Command button | cmd | cmdExit |
| Common dialog | dlg | dlgFileOpen |
| Communications | com | comFax |
| Control (used within procedures when the specific type is unknown) | ctr | ctrCurrent |
| Data | dat | datBiblio |
| Data-bound combo box | dbcbo | dbcboLanguage |
| Data-bound grid | dbgrd | dbgrdQueryResult |
| Data-bound list box | dblst | dblstJobType |
| Data combo | dbc | dbcAuthor |
| Data grid | dgd | dgdTitles |
| Data list | dbl | dblPublisher |
| Data repeater | drp | drpLocation |
| Date picker | dtp | dtpPublished |
| Directory list box | dir | dirSource |
| Drive list box | drv | drvTarget |
| File list box | fil | filSource |
| Flat scroll bar | fsb | fsbMove |
| Form | frm | frmEntry |
| Frame | fra | fraLanguage |
| Gauge | gau | gauStatus |
| Graph | gra | graRevenue |
| Grid | grd | grdPrices |
| Hierarchical flexgrid | flex | flexOrders |
| Horizontal scroll bar | hsb | hsbVolume |
| Image | img | imgIcon |
| Image combo | imgcbo | imgcboProduct |
| ImageList | ils | ilsAllIcons |
| Label | lbl | lblHelpMessage |
| Lightweight check box | lwchk | lwchkArchive |
| Lightweight combo box | lwcbo | lwcboGerman |
| Lightweight command button | lwcmd | lwcmdRemove |
| Lightweight frame | lwfra | lwfraSaveOptions |
| Lightweight horizontal scroll bar | lwhsb | lwhsbVolume |
| Lightweight list box | lwlst | lwlstCostCenters |
| Lightweight option button | lwopt | lwoptIncomeLevel |
| Lightweight text box | lwtxt | lwoptStreet |
| Lightweight vertical scroll bar | lwvsb | lwvsbYear |
| Line | lin | linVertical |
| List box | lst | lstPolicyCodes |
| ListView | lvw | lvwHeadings |
| MAPI message | mpm | mpmSentMessage |
| MAPI session | mps | mpsSession |

Object Coding Prefixes

| | | |
|---------------------|-----|-----------------|
| MCI | mci | mciVideo |
| Menu | mnu | mnuFileOpen |
| Month view | mvw | mvwPeriod |
| MS Chart | ch | chSalesbyRegion |
| MS Flex grid | msg | msgClients |
| MS Tab | mst | mstFirst |
| OLE container | ole | oleWorksheet |
| Option button | opt | optGender |
| Picture box | pic | picVGA |
| Picture clip | clp | clpToolbar |
| ProgressBar | prg | prgLoadFile |
| Remote Data | rd | rdTitles |
| RichTextBox | rtf | rtfReport |
| Shape | shp | shpCircle |
| Slider | sld | sldScale |
| Spin | spn | spnPages |
| StatusBar | sta | staDateTime |
| SysInfo | sys | sysMonitor |
| TabStrip | tab | tabOptions |
| Text box | txt | txtLastName |
| Timer | tmr | tmrAlarm |
| Toolbar | tlb | tlbActions |
| TreeView | tre | treOrganization |
| UpDown | upd | updDirection |
| Vertical scroll bar | vsb | vsbRate |

Prefixes for DAO Objects

| Database object | Prefix | Example |
|-----------------|--------|------------------|
| Container | con | ConReports |
| Database | db | DbAccounts |
| DBEngine | dbe | DbeJet |
| Document | doc | DocSalesReport |
| Field | fld | FldAddress |
| Group | grp | GrpFinance |
| Index | ix | IdxAge |
| Parameter | prm | PrmJobCode |
| QueryDef | qry | qrySalesByRegion |
| Recordset | rec | recForecast |
| Relation | rel | relEmployeeDept |
| TableDef | tbd | tbdCustomers |
| User | usr | usrNew |
| Workspace | wsp | wspMine |

Class Modeling

The object models details the class structure to be used within CRAFT/client.

The object model was developed using Microsoft Visual Modeler Version 2.0.

The object model is Class Model 1999-06-13-000

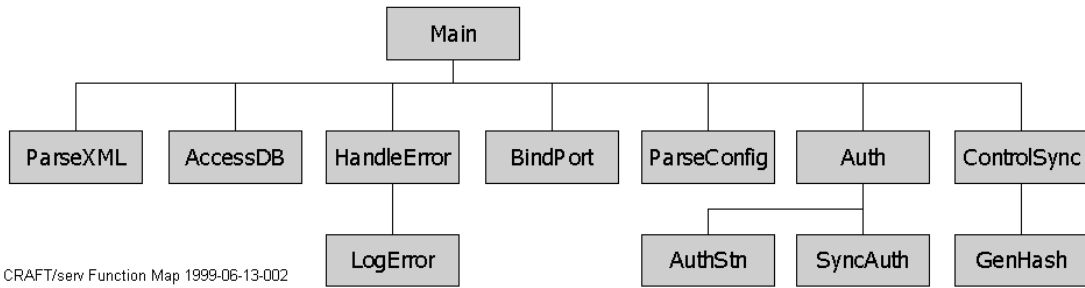
CRAFT/serv Function Map

The CRAFT/serv function map was developed with Paint Shop Pro 5.03.

The CRAFT/sev function map is CRAFT/sev function map model 1999-06-13-002

CRAFT/serv Function Map

This diagram outlines the functions that will be used within the Perl CRAFT/serv daemon. It does not necessarily denote the manner in which they are called or their physical library or module location.



CRAFT/serv Function Map 1999-06-13-002