SERVER FUNCTIONALITY DOCUMENT (SFD) DOCUMENTATION PLAN

Prepared by

Mike Hayden

Original Release: Friday, January 12, 2001

TABLE OF CONTENTS

SERVER FUNCTIONALITY DOCUMENT (SFD) DOCUMENTATION PLAN

EXECUTIVE SUMMARY:

Purpose:

The main purpose of the Server Functionality Document (SFD) is to *explain how the server works*, as opposed to how to *use* it. It will cover all components of the baseline server.

The SFD will be the authority for how the server works; it will save time by minimizing or eliminating questions directed to the software engineers. *"It's in the Server Functionality Document."*

Primary Users:

The primary users of the Server Functionality Document will be Core Engineers, QA Team, Service Developers, and Learning Support who need understand *how* the server works.

We estimate that 25-50 people will use the SFD on a regular basis: Development Team (10-15), Documentation Team (5+), and Service Developers (10+).

Main Elements:

The main elements will include:

· A detailed *System Diagram Flowchart*

· At least 12 Chapters as shown in **Table 1 - Estimated Page Counts**, page **9,** and compatible with the *Major Components Diagram*, **Figure 2,** page **6**.

Deadlines: (Preliminary)

Activity/Item	**Result**	**Date**
Project Start		Jan 15, 2001
System Diagram Flowchart	Draft	Jan 22, 2001
4.2 Email Transport	Draft	Jan 29, 2001
9.3 Boolean Logic	Draft	Feb 1, 2001
5.2 Request Types	Draft	Feb 8, 2001
(TBD)		

(Also, see **Manuscript Release Schedule**, page **13** and **Time & Delivery Estimates**, page **14**)

OVERVIEW:

This document is intended to be a *baseline document* that:

- provides a framework for understanding the server’s functionality
- explains how the server works
- explains what the server is, what it does, how it does it, error handling, etc.
- explains the language specification for Geoworks’ GXML tags
- provides program and subroutine calling sequences, as appropriate, for developers
- provides a sanity check on the Java classes
- serves as the basis for adding/modifying current and new software
- defines interactions between server components
- defines high-level constructs or constraints that a component developer must follow
- serves as the basis for designing QA testing

Figure 1. Documentation Hierarchy below shows how the Server Functionality Document and Language Specification fit into the overall documentation hierarchy.

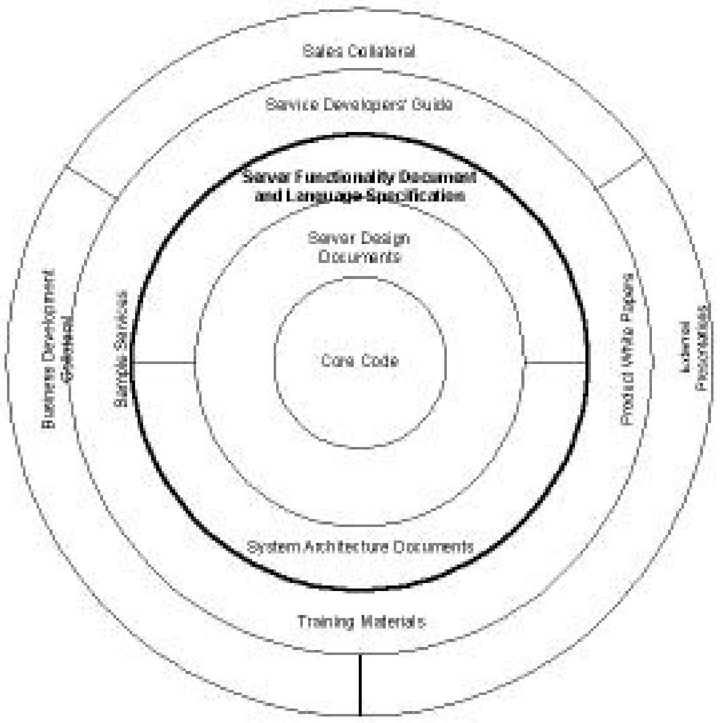
Sales Collateral
Service Developers' Guide
Server Functionality Document and Language Specification
Server Design Documents
Core Code
System Architecture Documents
Training Materials
Business Development Collateral
Sample Services
Product White Papers
External Presentations

Figure 1. Documentation Hierarchy

NOTE: (The Documentation Hierarchy (**Figure 1**) represents general organization only and does not symbolize all documentation. The dark ring separates "inside" Geoworks documentation from "outside" user documentation. "Core Code" represents actual code source. The engineering team will develop the Server Design Documents.)

The SFD is not intended as:

- a training guide
- detailed implementation-level information
- concept explanations for an external audience

Goals:

The user of the SFD should be able to:

- navigate the SFD to locate desired information
- determine the purpose, logic and data flows for all server components
- determine how a given server component works, what data it requires and what data is generated
- determine how the server will respond to a given set of input

DESIGN PLAN:

Audience:

Core Engineers, QA Team, and Service Developers, developer support and new employees that need to know how the server works (however, the SFD is not intended as an introductory document, more as a reference).

Primary Tasks:

The primary task of the users of the Server Functionality Document will typically be one or more of the following:

- to understand how the server works for the purpose of adding or modifying server features
- to determine all server features and how to test them
- to determine (as a service developer) how to use a server feature

Subject Matter Knowledge:

The typical user of the SFD will have the following qualifications:

- Service Developer: 2+ years of html and scripting (Perl, shell, etc.)
- Core Developer: 2+ years of Java programming
- Technical Writer: 2+ years writing about programming and operating systems

Media:

The documentation will be delivered as easy-to-maintain “on-the-fly,” under revision control, in the following format:

- html
- hardcopy (for reviews only)

DEVELOPMENT PLAN

First, the Server Architect (Paul Canavese) and I will develop a detailed *System Diagram Flowchart* (expanded from the *Major Components Diagram*, **Figure 2** below). This *System Diagram Flowchart* will show all the major components with basic logic and data flows.

Next, we will document one of the easier components so we can develop a model for the rest of the components (proof of concept).

As each component is developed, it will be released in draft form for technical review. This will ensure quality and accuracy as-we-go and prevents a massive, time-consuming review at the end.

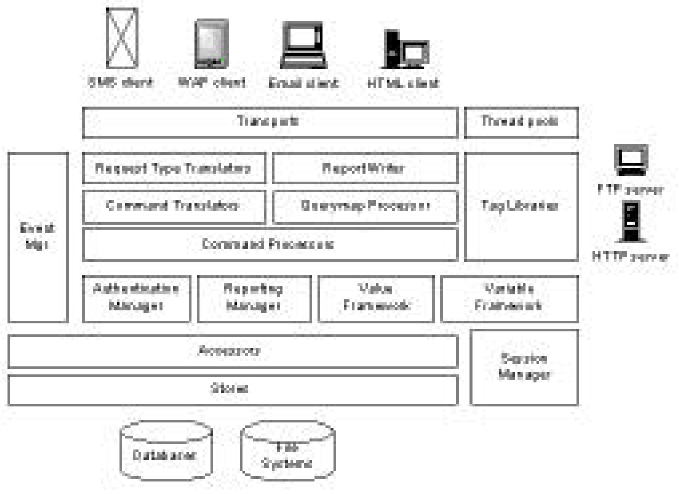

Figure 2. Major Components Diagram

It may prove valuable to start with GXML because it is stable and not being redesigned. Then, we can document the easier “stand alone” components such as:

- Transports
- Accessors
- Stores

Then, we can document the more challenging components, such as:

- Value and Variable frameworks and how they interact, their expectations, etc.
- Request Translation (logic)

- Querymaps and Templates

Sub-components must be documented, as well.

I will conduct interviews with engineers, as required to obtain the necessary details. Sometimes it will be necessary to evaluate Java code, engineering outlines, etc.; this requirement will differ for each component or module.

Design Considerations

Most technical/user manuals have major sources of unclarity, such as, redundancies, weak verbs, abstract nouns, showy writing, improper subordination, misused passives, etc. These faults hinder the manual's effectiveness by giving misleading or unnecessary information (called "noise" in communications theory).

Worse, technical/user manuals typically fail to separate **procedure** from **data**, making them practically useless as tutorials and quick references.

These problems will be "designed out" of the Server Functionality.

One of the biggest barriers to learning a new subject is its nomenclature (the terms used to describe the things the subject deals with).

A subject must have accurate nomenclature with exact meanings before it can be understood and communicated. Otherwise, a "student" starts to study the subject, then meets difficulties. Why? Because the student must learn many new principles, new methods, AND a whole new language!

Typically, students give up a study or become confused and unable to learn **because they have gone PAST a misunderstood word**. The confusion or inability to learn occurs **after** the undefined or misunderstood word(s)! Unless students understand this, they won't get very far in the field of study.

Thus, technical terms and jargon must be defined for the reader in a glossary or, better yet, *when used.* However, it is assumed that the reader will be familiar with the everyday terms associated with his/her job, therefore, emphasis will be placed on defining new terms in software, special applications, electronics, supplier's equipment, etc.

The SFD will be:

designed to be **easy to read**, using:

- writing style that is direct, clear, and concise
- text that involves the reader with short, easy, learning steps
- plain English with clear explanations and illustrations
- explanations of jargon and technical terms

designed to be **easy to use**, with fast access, using:

- simple formatting with low overhead maintenance
- hyperlinked documents and cross referencing

designed to be **easy to update**, using:

- modularized, logical sections

- consistent layout of text, illustrations, etc.
- consistent file names

The SFD will be a controlled document. We will develop a practical method for Engineers to update it; this may include a review process that includes developer support.

Document Contents

In addition to the Table of Contents and Index, the SFD will contain at least 12 chapters covering the following topics (topics and order subject to change), compatible with the *System Diagram Flowchart* (to be created). See **Table 1 - Estimated Page Counts** below.

The **Release 4.x** column shows the total estimated number of pages for all sections. Those entries in the **Release 4.0** column that have no entry in the **Release 3.x** column must be written/updated from Release 3.x.

The **Release3.x** column shows the total estimated number of pages of Release 3.x functionality that won't be change significantly in Release 4.0 and beyond.

Table 1 - Estimated Page Counts

Configuration	**Release 4.x**	**Release3.x**
1.0 Overview	2	
1.1 Configuration manager	4	
1.2 Dynamic configuration	5	
1.3 Validation	5	
Managers		
2.0 Overview	1	
2.1 Facade model	5	
Values and Variables		
3.0 Overview	1	1
3.1 Value types	10	10
3.2 Value casting	2	2
3.3 Variable framework	5	5
3.4 Permanent variables	3	3
3.5 System variables	2	2
Transports		
4.0 Overview	5	5
4.1 HTTP	15	15
4.2 Email	10	10
4.3 SMS	10	10
4.4 WAP push	5	5
Request processing		
5.0 Overview	2	2
5.1 General request variables	3	3
5.2 Request types	6	6
5.3 Commands	10	10
5.4 Authentication	5	5
5.5 Session management	10	10

Timed events		
6.0 Overview	5	
6.1 Table design	3	
6.2 Registration and cancellation	2	
6.3 Event triggering and execution	5	
6.4 Event accessor	2	
Accessors and stores		
7.0 Overview	1	
7.1 Stores	10	
7.2 Accessors	15	
Document processing		
8.0 Overview	1	1
8.1 Document syntax	10	10
8.2 Document types	5	5
Tags		
9.0 Overview	1	1
9.1 Document tags	2	2
9.2 References	3	3
9.3 Boolean logic	4	4
9.4 Control flow	8	8
9.5 Comparisons	5	5
9.6 Control flow	10	10
9.7 Type manipulation	15	15
9.8 Queries	8	8
9.9 Programmatic access	3	3
9.10 Other	2	2
Logging		
10.0 Overview	1	
10.1 Request logging	3	
10.2 Session logging	3	
10.3 Dispatch logging	3	
10.4 Error logging	5	
Supporting functionality		
11.0 Overview	1	
11.1 Thread pools	3	
11.2 Reporting mechanism	10	
Tools		
12.0 Overview	2	
12.1 Debugger	20	
12.2 Administration	15	
Total:	**312**	**181**

DOCUMENT DESIGN

Major Components Diagram

The **Figure 3. Major Components Diagram** below represents server system architecture; it is expected to remain generally the same for Release 4.0.

We will document how each component works and how each component interfaces with other components.

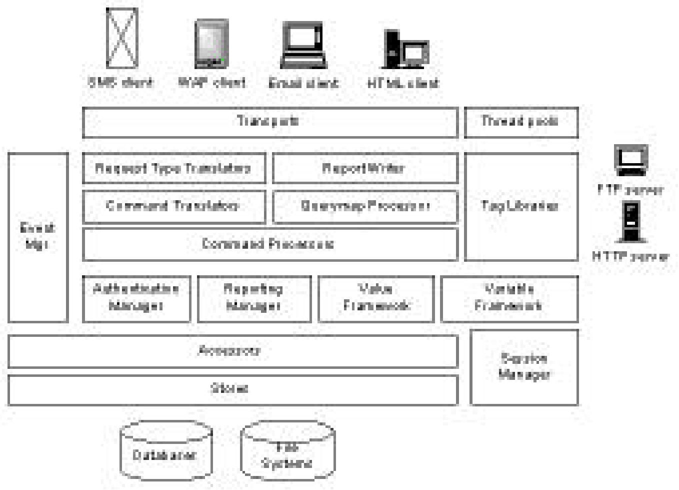

Figure 3. Major Components Diagram (repeated)

Expansion of *Major Components Diagram* to the *System Diagram Flowchart*

The *Major Components Diagram* will be expanded from the *System Diagram Flowchart* to include both **process flow** and **data flow** between processes.

- The resulting *System Diagram Flowchart* will serve as a map to navigate the SFD.
- The *System Diagram Flowchart*, showing the server's logic, will help clarify the associated descriptions.
- The process of flowcharting may help the redesign when some components are complicated.

Assumptions

The Server Architect and the Service Developers will maintain the SFD; when they change something, they must immediately change the SFD. Thus, *we will need to develop a practical method for updating the Server Functionality Document* that engineers can maintain and that the documentation group can use as a base.

For the 4.0 release, engineers are responsible for Server Design Documents and Systems Engineering for System Architecture Documents.

Problems to Solve:

- Current documentation (Jellyfish internal web page) was written for developers; at this point, it's neither complete nor maintained and will be replaced by the SFD.
- Previously, we haven't had a standard process for releasing documentation for each release; new policies and procedures are correcting this.
- Some presumptions are not documented (and should be).

To Do:

Design/writing of new 4.0 documentation is being dispatched to engineering team members. As Server Architect, Paul will review this documentation.

- Call engineers in UK to find out, from their experience, what would be most important – UK has both core and service engineers.
- Audit current code – identify things we need to cover.
- Survey the code to verity that every configuration added is reflected in the documentation and how the server actually works.
- Document http transport (not documented now).

At this point, I'll be focussing on components that have little or no change.

Questions / Issues to Resolve:

- What are other engineers going to be looking at first (re 4.0), so I know what to work on?
- What is the best way to provide the engineers the ongoing update capability?
- Do we need to go through the tags (Self documenting? Javadoc?)
- How do we integrate in Javadocs for Java APIs?
- In what order does Geoworks want to teach things? What is best order for teaching and presenting materials?

Other Considerations:

- Server 3.1 last major release Toshiba Servlet 1.0 – is an interim solution to scalability.

- Engineering currently planning how to implement Release 4.0 – significant re-architecture – moving toward a more modular design with 4.0, 4.5, and 5.0.

Format: (also see Additional Notes, page 15)

- Html is format of choice for internal documents (see diagram), so engineers can update.
- Html format desired due to Linux users.
- Under revision control – engineers can "diff" between different versions of the document (also see **How will we implement "Revision Control?"**, page **16**).

MANUSCRIPT RELEASE SCHEDULE

I will provide manuscripts in MS Word format for review by Stakeholders according to **Table 2 - Manuscript Release Schedule** below. The manuscripts will be converted to html after they are reviewed and approved.

Table 2 - Manuscript Release Schedule

Activity/Item	**Result**	**Date**
Project Start		Jan 15, 2001
System Diagram Flowchart	Draft	Jan 22, 2001
4.2 Email Transport	Draft	Jan 29, 2001
9.3 Boolean Logic	Draft	Feb 1, 2001
5.2 Request Types	Draft	Feb 8, 2001
(TBD)		

The **Table 2 - Manuscript Release Schedule** above shows delivery of three selected sections from different chapters to demonstrate proof-of-concept. After that, we will schedule sections for future delivery at 3-5 week intervals until done.

For maximum efficiency, Stakeholders should review and return said manuscripts to writer **within 72 hours**, unless otherwise agreed at the time the manuscript is released. Additional Manuscript Releases will be made available, if requested; additional time and materials may be necessary to produce.

NOTE: Review manuscripts will be double-spaced to facilitate easy editing between lines; final text will be single-spaced.

Time & Delivery Estimates

Estimates of time required for this project are shown below based on minimum, nominal, and maximum number of pages. **NOTE**. These estimates are based on data from similar projects of 150-300 pages, plus available information **prior** to task commencement.

Table 3. Time Estimates Based on Page Count

Start Date January 15, 2001

	Minimum	Nominal	Maximum
Pages	150	250	300
Hours	357	595	714
	893	893	1071
Weeks	9	15	18
	22	22	27
Months	2.2	3.7	4.5
	5.6	5.6	6.7
Complete	March 18, 2001	April 29, 2001	May 20, 2001
	June 20, 2001	June 20, 2001	July 21, 2001

SME HOURS			
Consult	25	42	50
Review	13	21	25
Product Mgt.	5	8	10

Table 4. Delivery Dates based on Page Count & Efficiency

Pages	Efficiency Level	Delivery Date
150	pages @ max efficiency	March 18, 2001
250	”	April 29, 2001
300	”	May 20, 2001
150	pages @ min efficiency	June 20, 2001
250	”	June 20, 2001
300	”	July 21, 2001

Budgetary estimates do not include delays due to possible “bugs” in equipment and/or software or for research and documentation thereof. Please supply any “electronic files” applicable to the development of the SFD.

“Efficiency” is a measure, based on experience, of how fast information is made available, how fast reviews are accomplished, how many “surprises” along the way, etc.

ADDITIONAL NOTES

Why html is the preferred format

Html is the format of choice for the creation and maintenance of the internal engineering documentation because:

- html is a text-based format that is easy to edit across all platforms
- html is easy to view on all platforms
- it’s easy to hyperlink to documents
- it’s easily to track differences between versions

NOTE: While html pages can be lengthy, our page count estimates are based on 8 1/2 x 11 pages.

Html limitations

Html editors don’t have as many features as modern word-processing software and we recognize certain limitations:

- using html, chapter and section numbering is a manual process (however, list numbering is automatic)
- manual numbering is a trade off for engineering’s ease of use and update
- since we’re mainly concerned with *specifying functionality*, we’re not as concerned with the order of presentation (any new functionality will be added to the end of the original Table of Contents minimizing renumbering problems)

The SFD will be "lightly formatted," but by using a simple standard format, we can use style sheets for additional formatting, if desired.

Symantec's Visual Page, a GUI word processor, can be used as an html editor; it offers WYSWYG with search and replace, bullets, numbered lists, etc. Also, we can check availability of other html editing tools.

Because of its powerful word-processing features, I will use MS Word. After review and approval, I will convert the file to html using Dreamweaver (a powerful tool for managing html and the site).

If necessary, we can clean up converted html code with the W3C Html Tidy program or similar tool.

How will we implement "Revision Control?"

We will use Perforce. Perforce provides the features to control and flag changes, the same as with software changes. Developers can modify the text then resubmit under Perforce, causing an email to automatically go out to all who are watching for changes.

We'll write-up a specific policy guide and exact procedure for that process.

Only one person will be working on a section at a time, with Paul as the reviewer.

The current files are not under revision control, but Paul is the only person with *write permission* – these files will be installed under Perforce if more than one person is going to be working on them.

Can we search the entire document, once completed?

Yes. We will be able to search individual chapter files or we can merge all chapters into one file and search it. Similarly, we will be able to print out individual chapters or merge and print the whole document.

Can we automatically generate key-word indexing?

Perhaps, it depends. We could write a script, but we're aiming for simplicity. We will "shop" for indexing software. However, complicated tools are sometimes not compatible across platforms and we want to avoid compatibility problems.