



BROCK SYSTEM REQUIREMENTS

SYSTEM: BROCK, PORTAL ENGINE
PRODUCT: ENHYDRA ENTERPRISE
VERSION: 4.0



SYSTEM REQUIREMENTS

Published: 1/08/2000
Version: 0.9

TABLE OF CONTENTS

1	DOCUMENT HISTORY	3
2	INTRODUCTION	3
2.1	Overview *	4
2.2	Purpose of the system *	4
2.3	Product perspective	4
2.4	Objectives and success criteria for the system *	4
2.5	Definitions, acronyms, and abbreviations	4
2.6	References	6
2.6.1	User interfaces	7
2.6.2	Software interface	7
2.6.3	Communication interfaces	7
2.7	Product functions	8
2.7.1	Basics	8
2.7.2	Security and Access Control	8
2.7.3	User Profiling	8
2.7.4	Workflow Management	9
2.7.5	Syndication	10
2.7.6	Caching	11
2.7.7	Event Notification	11
2.7.8	Search engine	11
2.7.9	Content management and cataloging	11
2.7.10	Payment	12
2.7.11	Collaboration	12
2.7.12	Reporting and Analysis	12
2.7.13	Data/Message Integration	12
2.8	User characteristics	13
2.9	Constraints	13
2.10	Apportioning of requirements	15
3	PROPOSED SYSTEM	16
3.1	Security and Access Control	16
3.1.1	Authentication	16
3.1.2	Access control rules	16
3.2	Profiling	17
3.3	Workflow management	18
3.3.1	Presentation object identification and access control :	18
3.3.2	Content filtering facility	18
3.3.3	Basic automatic operations	18
3.4	Syndication	19
3.4.1	Support co-branding of the presentation	19
3.5	CFI (Configuration File Interface)	21
3.5.1	Configuration File Interface	21
3.5.2	Configuration utility	21
3.6	API (Application Programming Interface)	21
3.7	CLI (Command Line Interface)	21
3.8	GUI (Graphical User Interface)	21
3.9	Other Interfaces	21
3.10	Performance characteristics	21
3.11	Error handling and extreme conditions	21

3.12	Management.....	22
3.13	Security issues.....	22
3.14	Configuration management.....	22
3.14.1	Installation	22
3.14.2	File list	22
3.14.3	Build system.....	22
3.14.4	Source code	22
3.15	Acceptance criteria.....	22
3.16	Testing.....	22
3.17	Documentation.....	23
3.18	Examples	23
3.19	Internationalization.....	23
4	CAVEATS *	24
4.1	Assumptions (bin them all here) *.....	24
4.2	Exclusions (bin them all here)*.....	24
4.2.1	Planned future extensions	24
4.2.2	Recommendations.....	24
4.2.3	Not planned extensions.....	24
4.3	Dependencies *(bin them all here. important).....	24
4.4	Open Issues and Questions * (bin them all here)	24
APPENDIX A: HOW TO USE THIS TEMPLATE DOCUMENT		25
5	APPENDIX B :OTHER TECHNOLOGIES AVAILABLE	26

1 DOCUMENT HISTORY

This section out-lines the revision history of the document. The revision history will provide a history of changes in the form of a list of the author responsible for the change, the date of change, and a brief description of the change. (Another way to do this in MS-Word is turn on Tools | Track Changes...|Highlight Changes). Last revision appears at the top of the table.

Revision Date	Author	Comments
August 2000	Eric Giguère	Reformatting under Lutris template
May 2000	Eric Giguère	Initial draft

2 INTRODUCTION

The purpose of this document is to unambiguously describe the features and constraints that the Brock system must satisfy, and how this engine will fit in the Enhydra package. This specification of requirements is written in terms that external and internal development teams can understand. It is an essential document to communicate with the groups both inside and outside of the development team on what exactly is being developed. The audience may include Quality Engineering, Publications, Customer Service, Training, Professional Services R&D, Enhydra.org Workgroup mailing lists.

2.1 Overview *

This document provides the requirements for the Brock portal engine throughout the Enhydra Enterprise system but also as an independent engine that may be integrated in other Java application server products. It provides specific requirements for the system but no design nor implementation information whatsoever. When design terms or example are used, its will only be to get a better understanding of the requirements and must not be considered as well established design decisions.

2.2 Purpose of the system *

Project: BROCK, Portal engine

BROCK is an engine that stands to offer Java programmers an API and a code generation facility to produce classes that are used to help to enable portal like functionality to a web site. Many aspects can be included in a portal engine but this project will mainly focus on user profile management and business rule management to automate the customization and personalization of the HTML output. The project will integrate with existing Enhydra components, such as DODS, XMLC, and possibly ROCKS, but will not be limited to the existing Enhydra environment. It will also offer provide a standard module interface that can be used to facilitate reuse between BROCK and other web development frameworks.

BROCK will allow developers easily enable customization in their site, both on the Internet and on its wireless counter part where personalization is a key factor.

2.3 Product perspective

This product will not constitute a self-contained stand-alone package but must rather be designed to be a component in an existing framework. Enhydra will be the first framework on which the engine will hook itself but not the only one. Any Java web or wireless site framework must be able to use the engine as a component in its system.

2.4 Objectives and success criteria for the system *

- TBD

2.5 Definitions, acronyms, and abbreviations

This section holds the different acronyms that are used in this document and their definitions.

Term, Acronym, or TLS	Expansion
B2B	Business to business. The new name for the next phase of the internet, connecting businesses together.
BROCK	Code name for the project. This is the old English nickname for the European Badger, a relative of the Enhydra Lutris, the sea otter.
Enhydra	Java Application Server
HTML	Hypertext Markup Language: The language of the web.

Jdk	Java Developer Kit. All files and programs needed to produce Java programs. Many versions of the jdk are now available, this project is based on Java 2, or jdk version 1.2 minimum.
Jetspeed	An open source syndication engine distributed and maintained on the Apache site.
ICE	Information and Content Exchange: an XML based specification to facilitate business to business communications.
LDAP	Lightweight Directory Access Protocol. Used to query systems specialized in maintaining huge lists, also called directories.
OCF	Open Catalog Format: XML based specification to allow syndication of catalog and their items.
Portal	An entry point for an internet user. Portals can be sites of general interests like yahoo or Netcenter just to name a few or may be on a very specialized subject (vortals). Usually present a customizable interface the meet the needs and tastes of all individual users.
ROCKS	Project on enhydra.org whose goal is to present a framework to abstract presentation APIs and allow application of common presentation logic to many presentation schemes.
SQL	Structured Query Language: A standard language used to retrieve and manipulate data and structure of a relational database.
RSS	RDF Site Summary: An XML based specification to describe syndication channels used by the Netcenter (the Netscape portal).
XML	eXtensible Markup Language. The specification is the universal format for structured documents and data on the Web.
XMLC	XML document compiler and Java code generator of the Enhydra framework.
TCP/IP	Communication protocol of the Internet.
JVM	Java Virtual Machine. The program that executes the byte code produced by a Java compiler
WML	Wireless Markup Language. The equivalent of HTML on portable devices.

2.6 References

This section should list any URL or document that has been referenced in the production of the requirements statement or is a useful reference. This may include white papers, industry standards, architecture documents, related problem statements, references to existing systems, feasibility studies, URLs, etc.

Section

2.6.1 User interfaces

Since we are talking about an engine for the web and wireless development, all user interfaces related in this product will be made of either HTML or WML files. Other standard may emerge in the future but as long as the document format fits in the XML file specification, the interface should be handled by this system without having to adapt large parts of it.

The design it self of the user interfaces is outside the scope of this project. The only physical attribute that one must have to operate with this engine is that the document describing the interface must follow the XML specification in its structure.

On the other hand, some user interfaces may be included in the package that will allow the handling of the configuration files that will be defined and needed by the engine to automate the work of the customization engine in a project.

2.6.2 Software interface

Java system

The software will be totally written in the Java programming language. Version 1.2 of jdk will be necessary to operate it.

Enhydra

The package, as a component, will find integration in Enhydra 3.0 or later version.

Other system

For now, the integration in the Enhydra framework seems to be the wisest choice. But the design must be done with loose coupling with the framework so that others can be added later on.

2.6.3 Communication interfaces

The system, as a component, will need to communicate with various other systems to operate. All communications will happen using TCP/IP protocol, the natural language of the Web.

Another protocol may be necessary to handle and that is the LDAP protocol. LDAP servers can make a very good candidate as storage medium for the information gathered and consulted by the system.

An SQL communication mean is also necessary for this engine to operate since some of its data, probably the personalization information supplied by both the user and his interaction with a BROCK enabled site.

Syndication channels are also an important part of the project. Using the most common syndication protocols, the system will communicate with broadcast engines that provide services using the supported protocols.

2.7 Product functions

This section dresses on overview of the functions that the system must or will provide. This section is separated in sections that can be related to portal needed attributes. Some will have to be done by the system; others by hooking with an external facility that already provide the service. Other sections describe the needs for a broader application framework that enable e-commerce operations. These are included in the specification document to keep in mind the possible enhancement that may be provided in future releases of the product. These sections will be noted as future requirement.

2.7.1 Basics

The system will have to offer the developer some help on many aspects. There are different forms of "personalization". Basically they all fall into a form of capturing user attributes and then using some kind of "customization" engine to deliver content personalized to the user's profile. An elegant implementation of personalization allows for preferences as well as publishing those preferences to a recommendation engine [look at how Amazon.com uses NetPerceptions to suggest books you might like based upon the purchasing behavior of others who've purchased similar items].

2.7.2 Security and Access Control

The core services provided by this component are the establishment of information access control boundaries (managing who can see and do what and when), authentication of remote users, the authorization according to user profile.

The system should have a login model that supports role-based security, so that, for example, customer service reps can do certain things and the senior management others. Basically this boils down to supporting the concept of groups as well as individual users.

A user of the system will always be a member of at least one group. Authorization rules can be assigned either on groups or directly on a user. A user gains authorization rules from his membership in a user group and from direct assignation. Groups can also be organized as trees so that user right management can be simplified. A group based on another group will have all the rules of its parent plus its own, a behavior very similar to inheritance in object oriented systems.

2.7.3 User Profiling

This component is employed by multiple applications and allows the sharing of individual user profiles across application boundaries. It supports profile management, personalization, organizational hierarchy (in a B2B scenario, the profile may not be attached to individual persons, but to their corporate roles) and behavior tracking. The access control and personalization services are often very closely connected to each other.

The behavior tracking in particular is very important in the wireless environment for a portal developer. The system must offer to developers using it a way to simplify and manage what content gets sent to the portable device. If not, users will feel as though they are being spammed and will not use the service.

Both profiling and behavior tracking information can generate useful events for the portal user (described in the Event section).

And with behavior tracking also comes the need to have dynamic user attribute creation. Every profiling system will create its own user profile attributes for its use in generating personalized content. We can also have some rules executed by a profiling that can generate and update user attributes.

The user may also do himself the gathering of profiling information. The need for personalization may be true on the Internet but also in Intranets where people will want it to a very high degree. For example, the people working in a the same marketing department may want to be notified on client demands based on certain conditions or apply personal filters on the content on requests from clients.

User attributes can be captured in different ways, including asking the user (some kind of "preferences" form), monitoring user activity (see "<http://www.opensesame.com>") or aggregating their transactional activity.

The "customization" can be done via some kind of rules-based engine, or by using collaborative filtering or some other technology (there are several of these <http://www.netperceptions.com/> is one example).

The use of a Java engine (not open source) called Jess (<http://herzberg.ca.sandia.gov/jess/>) has been discussed. This product, although it may need licensing, offers a great deal of power if personalization must be pushed very far on a particular portal. Jess offers a framework to build expert systems. The use of such a product, for first version at least, as not been adopted but future enhancement to the personalization engine may require such capability.

2.7.4 Workflow Management

A workflow component has three modules: the workflow definition module, the business rules definition module and the workflow engine. The component supports workflow for Buying Process Automation, Customer Relationship Management, Rule and Role Based Content Routing.

The system must also be able to filter the rules based on user security privileges. Based on his rights, an authorization engine must filter the available operations based on the context and from pool of available operations from this context. This way, users in the system sees only what they are allowed to see.

All the authorization process must take place before showing the available operations to a user so that he can't have access to the operations he doesn't have the right to use. If he doesn't have to right to do something, he must not be given the choice to do it. So the complete set of operations offered by the system must be cataloged in some storage medium. This way, the filtering can be done using the context information and the user set of authorization rules.

2.7.5 Syndication

When we mention portals, we also must include some kind of syndication. Syndication in this case refers to presenting content in such a way that it is reusable by others. Dotcoms being online organizations are typical type of web service providers that can use this aspect of the project to present many different content sold or offered by content providers.

The BROCK engine should fill syndication needs of portals in two different ways.

One is to make a BROCK application easily connected to an external syndication engine using some of the most widely spread technologies. At least two of those three different standards can be implemented in the first version:

RSS: Helper site: <http://www.oasis-open.org/cover/rss.html>
 Netscape site: <http://my.netscape.com/publish/help/mnn20/>
OCF: <http://www.martsoft.com/ocp/>
ICE: <http://www.icestandard.org/>

By integrating the act of syndicating content in the framework, application developers will have an efficient way to integrate external content in their portal.

The other is that the engine should make it incredibly easy to then syndicate the content of a portal site to other portals using the chosen syndication technologies. Other technologies may also apply but integrating the most widely spread is a good start.

The wireless industry (to come) will also benefit a great deal in a syndication engine but, it cannot be approach in the same way that we do with web content, mostly because of the important differences between the two network infrastructures.

In the wireless environment, the network is typically slow and the link to Internet depends on the operators' investment on the bandwidth. In related experiences in dealing with operators, it has been observed that they prefer to have the services hosted in their network instead on the Internet. There are two main reasons for this: faster response and less operation cost in reducing Internet usage.

However, to get the same kind of explosive growth as Internet, the creativity of Internet companies is necessary. In order to get their content into operators' system, not only the content has to be syndicated, but also the programs logic.

Portals get information from various sources and different source may require different presentation of their information. For example, a portal may gets news information from newspaper, TV and radio. The later two are become popular in broadband portal. Also, weather and traffic info may come in tabulated format.

The content gets syndicated to the portal and the portal would choose to present the data in its own style with an acknowledge of the content source. There are many ways to accomplish that. Presently, two different approaches can be seen on the market:

1. Define a set of formats by the portal and require content source to deliver the content in such formats.
2. Define a superset model of all the formats by the portal and all content fits into this format.

However, we can do better using Java and XML and this is kind of like building channel for each content source.

2.7.6 Caching

[TBD, maybe in version 1]

Portals with hi personalization capability or hi traffic need the use of some type of caching engine to speed up the user response.

2.7.7 Event Notification

[Future requirement]

This component is capable of notifying users of important occurrences such as a contract award, winning an online auction, or a request for purchase authorization. The following is supported: Application to User; Message Broadcast; Message to Email, Fax, Pager, SMS, Message Boards, News Groups.

Event notifications can also be transmitted with a client side application that would let the receiver respond immediately to it. In the bid example, if a user receives a notification about a bid that has been beaten, he could be provided a way to directly bid again without having to go through the site.

Events could result in the evaluation of a business rule based on behavior capture data. For example, a frequent buyer of some product is a perfect target for a sale event on this product or any other related product.

2.7.8 Search engine

[Future requirement]

This component enables users to easily locate the information they want in product catalogs, support knowledge bases and information/content repositories by using methods like text based search, parametric based search, role and rule based search.

2.7.9 Content management and cataloguing

[Future requirement]

Content management involves dynamic content generation from a data repository. Catalog management involves pricing calculation, export/import, and catalog generation. Essential considerations are how the user benefits from optimal content distribution and content organization, how an e-commerce application takes advantage of the component to dynamically transform content from large data resources into useful information for the user and how content/application/system managers define and organize criteria and rules.

2.7.10 Payment

[Future requirement]

This component should provide the following facilities: Shopping Cart; Payment Method Support; and Payment Verification/Funds Capture. This involves using or connecting to external services since only financial institutions like banks or credit card providers are allowed or capable of accepting electronic money transfers.

2.7.11 Collaboration

[Future or external system requirement]

This component lets users confidentially share information and collaborate in real time. It supports Mediation; Negotiation; Bidding/Auctioning; Collaborative Buying Process (like Letsbuyit.com); Online Community.

2.7.12 Reporting and Analysis

[Future or external system requirement]

This component puts together an overall picture of how the e-commerce application is being utilized, how it is impacting the target audience, how trading partners are performing, how it's success is measured and how it is increasing overall efficiency. It does this by supporting the definition and creation of concise, intelligent, user-defined reports sorted for high-level analysis on a real-time basis. It includes traditional online analytical processing (OLAP) functionalities, has the ability to define and generate event-driven reports, conduct multi-dimensional data mining and establish priority management. The following type of reporting is supported: Statistical; Historic; Trends; Log; Error tracking.

The system could also have an API for user management that worked in terms of XML documents for reporting as opposed to a full blown HTML interface. This would enable the reports to be easily integrated into the look and feel of existing sites through XSL.

2.7.13 Data/Message Integration

[Future or external system requirement]

This component supports integration with various types of back-end: Client/Server; ERP/MRP; Legacy; EDI, XML, cXML, text. It does this by allowing developers to define rules and criteria for constructing and parsing the appropriate message and data types for the back-ends.

In J2EE parlance, Data/Message Integration would correspond somewhat to the currently non-existent Connector API. It's worth noting that IBM WebSphere already has working Connectors for back-ends like SAP, Lotus Notes etc. The Connector API will probably draw heavily upon IBM's work.

Collectively, the above components provide the next generic layer of web application platforms that could be developed on an open source basis.

2.8 User characteristics

The users of this system will be programmers that create sites using the Java language. Basically, these people will use the engine to control both the workflow but also the content of their on-line service.

The users must know the Java programming language to use this product. They must also work with server side type of environment that interacts with the service customers using an Internet connection. Since it also has a need for an SQL engine, the user of this system will need to have one accessible both when developing his solutions and when deploying it.

The users of BROCK must also understand the different technologies involved in its operations and some of the standards that the engine implements in order to properly use it. No graphical user interfaces are yet planned for the first phase so configuration files, no matter in which form they will be written in, will be the mean for the developer to get the engine working for him. This kind of configuration involves more knowledge of the system but also offers greater control over system options.

2.9 Constraints

a) Hardware limitations

Hardware should be abstracted in this project so that it runs on all platforms that have a JVM. Therefore, the project should hold no hardware specific code at all to achieve true portability.

On the other hand, a certain power will be required by the server to run a BROCK application smoothly so this kind of application does not require to give high performance on low power server machines.

b) Interfaces to other applications

TBD

c) Parallel operation

The system is designed to help web developers create on-line service and portal sites. By definition, this kind of application always has to serve many concurrent connections with a certain number of users. So, all transactions controlled by the framework must be thread safe and allow others to execute similar transactions even if some are already in progress. This applies to all external system querying:

- databases for user credentials and profiling information
- html template recovery and production engine (xmlc or other)
- live content providers (syndication)

d) Audit functions

All operations done under the control of the framework must be “auditable” as described in the profiling section of the project. From a point of view, profiling and auditing plays the exact same role except that profiling information relates to users as auditing may be more relevant to the operation themselves. We don’t want to know what a particular use does but rather who does what. The same engine must be used to provide the information.

e) Control functions

TBD

f) Reliability requirements;

BROCK is more a framework than a working application but some sections of the project will hold live code that will be executed as it is (without any user modifications). This code must be solid and must not crash under even the worst conditions. Some inner error handling mechanism must protect the engine from unexpected and uncontrolled crashes that may lead to a service shut down. But, since crashed cannot be totally eliminated, the system must still be able to automatically be restarted either by the Web server or any human or machine command.

g) Criticality of the application

This attribute is in direct relation with the kind of application that is build with the framework. Most of them, since they will be on-line services, must be very reliable to avoid down time. Too many shortages can hurt badly an ASP. The BROCK framework will help, by providing a framework, can elevate the quality of the produced application. For this to happen, BROCK code must itself be highly reliable and fault tolerant, since it will operate in a very distributed environment.

For example, a shortage at an on-line content provider must not hang the server when it tries to refresh the content of its page from another provider.

h) Safety and security considerations.

The only section of BROCK that will accumulate is the one that covers the user information and profiling. This kind of data is the most sensible information imaginable. It not only contains information about the habits of a particular user but also it credentials, status in the system and may also contain financial information (depending on the application).

It may be outside of this project scope to handle the security measures for this information but we must provide in the product documentation some training on where and how to setup the user information database. A product that allows outsiders to peek in its internal data will not live long in the wild of the net.

So all the user related info stored by the system being highly confidential must be treated as so. All information transfers must be made under certain agreement of both parties implied in the transaction to limit the possibility of unauthorized access to the content.

2.10 Apportioning of requirements

All the subsections identified as either [Future requirement] or [Future or external system requirement] in the product functions section will not be part of the first version of this system.

3 PROPOSED SYSTEM

This section holds all the needs that the system must solve. Each requirement is describe in enough details so that the designers can use this text as base work. But, no design instruction are given in these descriptions or if some are used, its only to help the understanding the of the text and must not be considered as design decisions.

3.1 Security and Access Control

3.1.1 Authentication

The framework must hold a package of classes and interfaces that will fill the authentication needs. A user must be identified before any personalized behavior is enabled on the site. If a user of the application is not identified, then default behavior must be easily provided.

A default SQL database must be included in the framework to provide all the user information. Personalization data of the user will also be included in this database (described later). Programmers of the system may change the database if they want to by inheriting the API classes and providing their new tables to support the added attributes

An abstraction layer must be included in the package to handle user-related data, especially the engine that does the actual authentication. So, user names and passwords, the most common authentication scheme, may be recovered from any DB or LDAP server.

The user abstraction that results from authentication should be powerful enough to support a generalized way of writing the rules. In other words, the subject on whom rules are applied must not be reserved for standard user/password authentication but also include identification from a card, or it could even be setup to accept other systems, etc.

3.1.2 Access control rules

Based on the identification, rules stored in some kind of XML file must be available for the application programmer to consult and filter the content of the site accordingly. The rule-recovering engine must be totally encapsulated in classes so that the programmer will be able to extract the content of the file without having to know its structure in programming.

The same rule engine must offer some basic kind of query language to allow quick recovery of the information based on some selection criteria. The workflow manager that guides the navigation in the site could supply the selection criteria (elaborated further in this document).

Provided an unauthorized access, the framework should provide the programmer an easy mean to catch the error, provide an error page along with redirections to the last authorized page. The error page must be customizable in the engine so that contextual layout may be presented in the application.

This section of the project must be JAAS compliant at the later in the second release of the engine. This framework will probably be widely use since it's included as part of the Java 1.3 version of the SDK. Since this version of the SDK is not yet widely spread, the first release will not have to include it.

There should be a means to 'script' the rules via an external file. This rule file should be secure and maintained and allow non-technical users to modify them. (A GUI rule-builder would be nice but not essential for v1).

Some existing technologies can be used to create simple rule engines. Using JavaCC and JJTree (Java Compiler-compiler).

TBD

3.2 Profiling

Formerly called user profiling, the section of the project as been renamed for it will not only cover the handling of user related data but also the profiling information related client devices. This mechanism that will allow devices to identify themselves to a server is very important in the wireless net since all device are not made equal.

Profiling, for the first version, must be implemented to provide the basic services:

- User preference retrieval from the storage
- Configurable user preference gathering for the operations of the application that are under BROCK framework
- Device information exchange with portable devices

The W3C has published a note for the RDF standard that presents the designs for the basic operation needed of a profiling engine for web (and wireless) enabled applications. This note can be found at <http://www.w3.org/TR/NOTE-CCPP/>. It is an effort to standardize how client preference and device profile to be transfer to the server side. This note must be studied closely and the final design of BROCK should be inspired from it since it holds some good designs that have already been discussed and revised by groups of interest in the subject.

The RDF standard seems a real good way to exchange the profiling information between different systems. Since some of the information gathered by a profiling engine could be published (everything that is not user related), RDF seems a sensible choice to fill this requirement. Geographical profiling is a good example non-sensible information that can be offered to external system in order to allow them to present filtered information based on their current location.

Even the user related profiling information could be exchanged using RDF files but only between partners. A privacy concern here drives the importance of separating the two all the way through the design of the engine to minimize risks of releasing non-public information by mistake, or have it stolen.

3.3 Workflow management

An effective and programmer friendly framework must be build to help programmers integrate the authorization engine in the workflow control of the site he builds.

This section of the project may collide with some implementations that another Enhydra project covers (ROCKS). Some more reading will be done before including any specific requirements for an elaborate system to control navigation and presentation of the site. But, no matter what ROCKS will offer, some basic functions needs to be handled for the system to be complete and operative on its own.

3.3.1 Presentation object identification and access control :

Presentation objects are, with user authentication, the building blocks of the authorization engine. Operations in a web environment can be resume to the request for a certain page. Even if that page is never shown on screen (like a commit page that redirects), these are still requests handled by the web server and, in the end, in a presentation object. The system must be able to identify the requested page and automatically recover the rules for it, if any.

3.3.2 Content filtering facility

A page may present many different operations to its users. Based on his credentials and a context, the user is presented a certain list of operations he is allowed to use. This querying for a filtered list of pages must be handled by the framework and presented to the programmer for its use.

The context information is the minimum information the engine needs to take content manipulation or selection actions. The smallest context must at least contain user and his location in the application. With these, an engine can do a lookup its it rule database to see if some applies to him where he is. This context must be available to the programmer.

3.3.3 Basic automatic operations

The engine could do the workflow control on a site, or at least part of it, totally automatically if instructed to do so by some configuration utility. Using markups in a source page or any other kind of placeholders, the content of the page could be managed automatically by the engine.

Special kind of operations can be defined in the system to take different automatic actions. It comes to specify a type in a placeholder so that generic objects built in BROCK could handle some requests. Since BROCK is a portal engine, popular syndication technologies will be included in it. As part of the content selection engine

(workflow), syndication data recipients can be created to replace the placeholders defined in or for a presentation object. So, by inserting a tag in a template HTML page, an application will be filled with externally supplied and pre-formatted data. This topic is elaborated in the next section.

All the objects manipulated by the automatic operations must be made available to the programmer using an API provided by the framework.

3.4 Syndication

The syndication engine must provide a template driven approach to allow programmers to manipulate and redirect content provided by external parties in the pages of the portal site he builds. There could be entire pages or fragments of it. This capability must apply to all XML derived specification for presentation documents such as HTML (pages successfully compiled by XSLC or other XML compliant parsers) and WML.

Each syndication target will have its own control class in the project. The linking of the control classes with the different sections in the presentation document must be accomplished using tags in the source page. These tags could be replaced by a class supplied an engine that will allow

- Recovering a compiled Enhydra presentation object
- Live compilation of a template file to get a new Java object
- Recovery of a serialized Java object
- Recovery from a caching engine

We must also provide a platform for developing syndicated applications to facilitate not only content syndication but also the programming logic. And with the platform comes a set of standard API to develop such programs.

As we develop standard APIs and interfaces for creating portal modules, we could also have standard access mechanisms that would generate RSS or OCF feeds of the modules' content. Thus, the engine will be usable to both the use of syndicated content but also to build syndication mechanisms for the applications built with it.

3.4.1 Support co-branding of the presentation

Regarding to the co-branding/syndication/presentation, basic requirement is all content sources have to be syndicated to the portal and each content source requires different presentation.

The engine must offers a simple way to programmers to hook up syndicated content with different presentation objects. A good example will be to syndicate some content to both a web and to a portable device. Both won't have the same presentation. The basic operations of a syndication engine can be summarized in three distinct encoding and interpretation phases:

1. Each content source defines the most suitable format for its own data and syndicated to the portal with that format. The format is preferably in XML but can be in other format and the portal engine will convert them into XML. To accomplish this, the ICE standard seems to be a good choice but other standard may also apply.

2. Once the content arrives at the portal, it's stored into a XML database. But since XML DB are not yet widely spread and that the technology is not yet ready, we must be able to use a standard DB to store the XML document (using a blob field for example) with various field indexed for fast query. But as the XML DB progress, this will certainly change and the document could then be stored in its native form.

3. Define each content source as a channel and develop a set of unique presentations for its contents. The business and data logic also developed.

To achieve all that with the minimum work involved from the programmer, the framework should provide:

1. A XML storage, probably from the content management component
2. An API to develop channel
3. A framework to deploy the channel quickly (name space and etc). This includes the mapping of the information supplied by the content provider to the presentation object that will publish it.

Once the framework support these, the portal will be able to quickly bring in new content partners. The work for each channel is localized and can be rapidly developed and deployed.

3.5 CFI (Configuration File Interface)

3.5.1 Configuration File Interface

If applicable, this section describes the format of the file used to configure and/or extend this system. What kind of file settings will be exposed? A comprehensive list of file settings and their possible parameters would be ideal. Include default behavior where applicable.

3.5.2 Configuration utility

If applicable, describe any customized utility or tool (besides an editor) that will be provided to configure this system.

3.6 API (Application Programming Interface)

This section describes the API requirements for the AVS and general Enhydra Security.

3.7 CLI (Command Line Interface)

If applicable, this section describes the CLI requirements. What kind of command line will the system provide? A comprehensive list of command line options and switches and their meaning and options would be ideal. Include default behavior where applicable.

3.8 GUI (Graphical User Interface)

If applicable, this section describes the user interface requirements. What kind of interface should the system provide, what is the level of expertise of the target user. The detailed requirements of the actual contents of the screens presented to the user are included here as screen shots of prototyped UI.

3.9 Other Interfaces

If applicable, this section describes other interfaces this system will have with other systems, such as file-level interfaces, protocols, etc.

3.10 Performance characteristics

This section should provide the high level description of the performance related requirements. How responsive should the system be? How many concurrent users should it support? What is an extreme load that should be supported?

3.11 Error handling and extreme conditions

This section should provide the high level description of requirements related to error/exception handling. Which exceptions should be the system handle? What form should these exceptions be captured and presented to the user?

3.12 Management

This section should provide the high level description of the management related requirements.

3.13 Security issues

This section should provide the high level description of the security related requirements. To what level should the system be protected against external intrusion or malicious users? What level of non-repudiation, authentication, authorization, and encryption should be provided throughout the system?

3.14 Configuration management

3.14.1 Installation

Specify how the system will be delivered. What installation program, if any, will be used to install the system?

3.14.2 File list

List all the .jar and any other filenames that are to be delivered which contain the functionality of this system.

3.14.3 Build system

Specify any special tools or unique requirements for building this system (that are truly unique to this system).

3.14.4 Source code

List the names of packages that implement the system.

3.15 Acceptance criteria

The acceptance criteria may be to pass a particular 'standards-based' (i.e. J2EE) test suite, or meet a particular industry standard specification (or subset of the standard).

3.16 Testing

Test Hooks

What hooks will be built into the system to make it testable at a unit and system level? Will diagnostic or println messages be logged, will asserts be used?

Fault Model

The fault model identifies how a component may fail. It is used to incorporate fault mitigation into the component design and to direct test approaches. Discuss the types of failures and any fault propagation from or to other components, and any design features adopted to detect, diagnose, or resolve faults. Identify any extrinsic systems entrusted with system invariants. You may reference existing exception designs or API documentation rather than replicating them here.

Testware & Unit Testing

List the unit or other tests to be created for this system. Tests should comply with the testware requirements of the Lutris test harness, e.g., signaling when particular system requirements are verified. If the tests will not, please explain why.

3.17 Documentation

This section describes how the documentation requirements for the system. Will there be online man pages, online help, readme text files, html, printed doc, - usage switch option? Will Javadoc comments be completed (usually required)? Will there be a tutorial? A sample?

3.18 Examples

Define what, if any, example code will be supplied (checked-in) to illustrate the use of the system.

3.19 Internationalization

Define how the system is designed for a global audience. How the system will:

1. Represent data internally in Unicode. (if not in Java)
2. Localizable (translated to foreign languages). Resourcing mechanism of strings (no hardcode messages).
3. Locale sensitive (handle different locale date, time, numeral, currency, formats)
4. Locale sensitive collation
5. Keyboard entry in foreign language input method editors and keyboards
6. Read and write to local file systems and databases that are encoded in non-Unicode, non-ASCII format.
7. If there is UI, specify the conventions (i.e. layouts such as GridBagLayout) that will enable the UI to be translated into foreign languages and dynamically resize, without requiring any changes to the source code (excepting the resource files, of course).

4 CAVEATS *

4.1 Assumptions (bin them all here) *

This section includes all assumptions that impact the scope of the development of the system

4.2 Exclusions (bin them all here)*

This section includes all requirements for the system that are explicitly *not* going to be achieved for this release. This section should provide the high level description of what the anticipated scope of the future changes need to be supported in the current system design.

4.2.1 Planned future extensions

Future extensions we will definitely want to include in some future release.

4.2.2 Recommendations

Recommendations (what you think “would be nice” but not required) go here. Dream here.

4.2.3 Not planned extensions

Extensions we definitely do not want to do ever (true exc lusions). Get real here.

4.3 Dependencies *(bin them all here. important)

This section includes all dependencies on the entire system (Enhydra Enterprise) and any other existing or to-be-created subsystems. Include:

1. All other systems that this system is dependent on and what the dependency is.
2. All other systems that are (or may be) dependent on this system.
3. What are the Java system dependencies? (For example: JDK v1.2.2 VM and JNDI v1.2.1 are required). Explicitly list all .jar filenames that are not implicitly included in standard JVM or J2EE deliveries from Sun or any other third party.
4. What are the Operating System dependencies, if any?
5. What are the external hardware dependencies, if any?

4.4 Open Issues and Questions * (bin them all here)

This section outlines open issues related to the completion of the specification. Bin them all here, no matter how insignificant they may seem. If possible, name the person responsible for resolution of the issue, and give a deadline if known. If applicable, a list of major open bugs is also appropriate (including bug numbers from the bug tracking system).

Section

APPENDIX A: HOW TO USE THIS TEMPLATE DOCUMENT

This document is a template for system requirement documents used in Lutris Enhydra product development. Please don't make copies of these, but rather come back often to get the latest revision of the template. That way you can benefit from frequent updates and revisions of content. Note that the actual section headings are offered as a content guide only - modify them for the project at hand.

Each document uses a common style to achieve a consistent and professional look across all documentation produced by Enhydra Product Development.

When using the templates remember to:

- Spell and grammar check the document!
- Change the title page of the document from <Template> to Document to indicate that the doc is no longer a template. Save the file with a filename that no longer contains "template".
- Update the *Version* number of the document. This occurs both on the front cover and in the header of the body pages.
- Update the *Product Name* and *System Name* throughout the document. These occur both on the front cover and in the header of the body pages.
- Update the Publish Date of the document. This occurs both on the front cover and in the footer of the body pages. Right click on the date and select *Update Field* if the date shown is incorrect.
- Owing to a bug in word it is sometimes necessary to update the "Y" of "Page X of Y" in the footer. Again right click on "Y" and select *Update Field*.
- Update the *Product Name* and *System Name* in the footer on the body pages, next to the confidential note.
- Update the table of contents. To perform this, right click on the TOC and select *Update Field*. When presented with the option to update line numbers only or entire table, select *Entire Table*.
- The word document name should be named in a similar fashion to the templates. I.e. the <ProjectAcronym>_<DocName>_<VersionNumber>. E.g. memo_Architecture_Specification_v1_1.doc
- A couple of other notes:
 - If there are multiple *Application Design Specification* documents, then remember to add a document subtitle under the "Application Design Specification" title on the cover page.
 - Please maintain this document in Word .doc format – which is Lutris corporate standard. Publishing in .pdf, .html, etc is optional, but the base document must remain in this format.
 - Please don't modify the styles - the idea is to create a unified and professional feel across all projects.
 - Please send all suggestions, comments etc regarding the format and contents of this template to scott.harrison@lutris.com. Please help me make this a more useful (and less burdensome) document.

Additional Instructions:

1. For each subsection, *not* completed, use these conventions so we know what areas have been addressed.
 - a. If it does not apply, insert a [N/A] at the top of the subsection.
 - b. If it does apply, but will be filled in later, insert a [TBD] at the top of the subsection
2. When ready for posting, convert to .pdf file format (see John Marco for details how to do this) and post it on the appropriate Enhydra.org workgroup (ask Greg Schwarzer to post for you)

5 APPENDIX B : OTHER TECHNOLOGIES AVAILABLE

This document is filled with links to sites that presents either specifications, existing products or other Open Source projects that can either be used or studied for the design of BROCK. The table below summarizes these links by presenting the item accompanied with a quick description and the link to get more information on it.

Name	Description	URL
Castor	Castor is the shortest path between Java[tm] objects, XML documents, SQL tables and LDAP directories. It provides Java to XML binding, Java to SQL/LDAP persistence, and then some more.	http://castor.exolab.org/index.html
wftk	Open Source workflow management toolkit	http://www.vivtek.com/wftk
Argo UML	Open source case tool using UML notation.	http://argouml.tigris.org/index.html
OpenDav	DAV (Distributed Authoring and Versioning) server that can be used in this project.	http://opendav.exolab.org/
JetSpeed	Open Source implementation of an Enterprise Information Portal.	http://java.apache.org/jetspeed/site/overview.html
Ozone	XML based store Ozone	www.ozone-db.org
JAAS	Java Authentication and Authorization Service	http://java.sun.com/products/jaas/
WebXL Internet Caching	Specialized hardware to do handle the caching of web pages.	http://www.quantex.com/webxl/index.asp
BEA Personalization server	BEA commercial personalization engine. Part of the BEA E-commerce suite	http://www.bea.com/products/weblogic/personalization/announcing_20.html
W3C Note for CC/PP	Note for content negotiation standard	http://www.w3.org/TR/NOTE-CCPP/
Net Perceptions	Commercial personalization engine	http://www.netperceptions.com/home/
Dynamo personalization server	Commercial personalization engine part of a suite of tools for e-commerce applications.	http://www.atg.com/products/dps/dps_main.html
JavaCC	Java compiler compiler. It is very widely used parser generator.	http://www.metamata.com/JavaCC/
JJTree	A tree building preprocessor that ships with JavaCC	http://www.metamata.com/JavaCC/