APPENDIX.D2.4

Ejournal Proposal

eJournal Request Management System

Technical documentation

1. Introduction

The eJournal RMS (request management system) is a system for managing large amounts of requests. The main focus of the system is handeling email to non-personal accounts such as support@company.com or sales@company.com, however, the system is also capable of handeling phone requests, internal todo's and other tasks that needs to be tracked. The system offers several tools, both automatic and interactive, which help meeting the overall goals of the system:

- Minimizing the time spent replying or managing each request.
- Guaranteeing that no request is left unanswered.
- Offering full control over incomming and outgoing communication. This includes:
    - Detailed information for each request, including exact timestamps, detailed logging and complete listing of messages.
    - Flexible reports and statistics, offering an overview of all requests being handled.

This document gives a technical specification of how the eJournal RMS is built, which components it consists of, and which technologies it supports and needs for it's functionality. This document will also explain why certain technologies have been chosen in favour of others.

2. Vocabulary

Througout this document, in order to avoid ambiguity, the following terms will be used to specify the various components of eJournal RMS:

- **Organization.** The organization will be used to describe the company using eJournal RMS.
- **Request.** A request is analogous to a task, a problem, or an action that needs to be performed. Sometimes this is referred to as a 'ticket' in helpdesk terminology. A request will initially be posted, then processed, and finally closed when the problem is considered solved or the task has been performed. A request may later be reopened if needed. A request will contain a unique identifier, the **request id** (sometimes referred to as a 'tracking number'), a log containing a detailed description of all events

regarding the request, and a number of messages, usually representing the dialog between the persons working with the request. Furthermore, a request will contain certain fields, identifying the request's priority, title, category, connected customer, and so forth.

- **User.** A user of the system is an internal person who is using the system to track and solve requests, such as an employee of the support department. A user will be registered in the system with a username, an email address, a password, a priveledge level, and so forth.
- **Customer.** This entity represents an external person, usually a customer or another person who has some relationship with the organization, without being an employee. It could also be an employee of the organization who is not a user of the system, and thus do not have access to process requests. A customer entry contains a unique id, email addresses, a username and password for external access to the system, address and phonenumbers, etc. Furthermore, a customer may be connected to a company.
- **Company.** Several customer entries may be connected to a company. A company entry contains certain fields, such as an internet domain, phone numbers, and so forth.
- **The primary database.** The primary database is the relational database eJournal RMS is using for storage. This database contains tables for requests, messages, request log, users, categories and so forth.
- **The external database.** The system also has the possibility to connect to an external database, such as Oracle, MS Sql or MS Exchange. Relevant information stored in an external database, such as customer information, can be connected to and synchronized with information in the primary database, thus avoiding to maintain duplicate sets of data.

3. Technical overview

The eJournal RMS system is built in a three layer model. The first layer consists of a relational database used for storage. On top of the database, we find the eJournal RMS API (application programmers interface), which offers functionality to performe all tasks towards the logical units of the system, such as requests, users, categories and so forth. Finally, on top of the API, we find the various system applications. These are mainly CGI (common gateway interface) applications, but also include the programs responsible for fetching email into the

system and performing routinely tasks, such as request escalation, request awakening, and automatic logout of users.

3.1 Layer 1 – the primary database

The primary database is a relational database containing virtually all information stored by the system. Some data, such as binary attachments and backup of incomming email are stored directly on disk in order to save time. MySQL has been chosen to be the database used by the system. There are several good reasons for this:

- MySQL is a rather small, fast, lightweight database, still offering all the functionality needed by the system, at a very reasonble price. It is developed by the Sweedish company TCX AB (www.tcx.se), and has rapidly become a very popular database for Internet related applications.
- It's stability, efficiency and scaleability has been proven by numerous applications and users worldwide.
- It requires virtually no management, and is very easy to install.
- MySQL do not perform foreign key verification or cascaded operations. In applications managing more sensitive information, such as financial systems, this verification would be a necessity. In a system like eJournal RMS, however, the verification of data is done only for certain tasks on the API level, thus resulting in faster operation.
- MySQL does not support transactions. In eJournal RMS, transactions are not required, and the benefit is that MySQL is much faster than other transaction oriented databases, such as Oracle or Microsoft Sql. This is especially relevant for Internet applications, which need to be non-session oriented. In contrast to traditional database dependent application, where the application normally connects to the database upon startup and stays connected for the period the application is running, Internet applications need to reconnect to the database repeatedly. The overhead of connecting to a transaction oriented database is rather high, compared to non-transaction oriented database, and consequently using e.g. Oracle or Microsoft Sql as the primary database for eJournal RMS would severly slow down the system. Benchmarks supporting this can be found at www.mysql.com.
- MySQL has been released under the GNU licence, and thus been freely available, including the program source, for non-commercial use since its beginning.

> Consequently, it is being rapidly developed and fixed by numerous users among the Internet community, similarly in fashion to the development of the Linux operating system. Furthermore, the number of supporting applications is growing very fast.

It is important to note, however, that the eJournal RMS is built in a very modularized fashion, and support for other databases is easy to implement. Support for other well known databases is expected when these products will support faster connections, e.g. using connection caching.

3.2 Layer 2 – the eJournal RMS API

On top of the database lies the eJournal RMS API, which is a C++ library offering a vast range of operations towards the logical entities of the system. The API ensures the integrity of the system, and also provides a uniform interface for the front end applications. This API also makes it easy to develop new applications, offering special functionality.

3.3 Layer 3 – the front ends

The eJournal RMS system is mainly accessed through a set of CGI applications running on the server, interacting with a standard web server (normally Apache or Microsoft Information Server), accessed by the users through a standard web browser (normally Netscape communicator, Opera, or Microsoft Internet Explorer). For reasons of speed and stability, Java is not used. Javascript is used for some operations, but is not required. There are two domains for the web interface; internally and externally.

The internal domain is accessed by the users of the system and, dependent of their privelege level, they will have access to perform all relevant operations on requests, such as listing, reassigning, replying, closing, postponing, and so forth.
The other domain is the external web front-end, which is accessible to the customers. The visual design of this front-end may easily be altered through the use of HTML templates in order to become an integrated part of the organization's web site.
The functionality of both the internal and external front-end is explained more in detail in section 4. Finally, the scheduled processes are also running in layer 3, using the API for operations performed on the system. This includes fetching email into the system, sending out emails summarizing status of active requests, and escalating unhandled requests.

4. System functionality

In order to achieve the goals stated in the introduction, eJournal RMS offers a wide range of tools. This section will explain in detail the functionality of these tools.

4.1 Automatic request processing

One of the fundamental tasks of the system is to fetch mail from defined POP3 mailboxes and either create new requests or attach these messages to existing requests. In order to connect messages to an existing request, the incomming email is scanned for a special key together with a request number. All email sent out from the system will contain this key in the subject line, and consequently a reply to this email will be connected to the same request. If the key is not found, the system will look for message id's in the header of the email, and match them with id's stored in the database. Again, if a match is found, the message is connected to the existing request. This will normally be the case where a user has replied to an incomming request, possibly to inquire more information from the customer. When the customer replies to the email, the reply will be connected to the request, which then, will list all communication between the user and the customer for this specific case. If a message is not connected to an existing request, a new request is created with the incomming email as the first message. A user is assigned the request, based on several possibilities for delegation, the request is marked as active, and priority and category is chosen. Finally, if a new request is posted, a definable email is immediately sent back to the customer, including the request id and the address to the web interface where the customer may observe and interact with the processing of the request. Note that one may precisely define which parts of a request should be made visible to the user.

There are several ways to control the actions being performed when an email is recieved. The easiest method is to set up several email adresses, such as support@company.com, sales@company.com, etc, and then specify which category these requests shall be stored in. Categories can here be thought of as mailboxes. Based on the chosen category, a user who is a member of the category, will be assigned the request. Users may easily set their own status to 'unavailable' in order to not have any requests automatically assigned to them. It is also possible to set up a schedule with teams of users working at specified hours. If nothing else is specified, the request will be assigned to one of the category members, based on a defined

model. The priority of the request will also be set by the definition of the mailbox, but can be overridden by the setup of the customer (recognized by the email address) or the company (recognized by the domain in the email adress). Although simple, this email handling will prove sufficient for most cases.

A far more complex way to set up email handling is by setting up searchpatterns for email recieved by certain mailboxes. These searchpatterns may be regular expressions or normal substring searches, and may completely define the actions being performed, such as:

- Automatically opening or closing the request.
- Overriding assigned category, user, customer information (name, email, etc) and request information.
- Specifying or blocking the automatic reply to the customer, possibly including a reference to a FAQ entry (the FAQ module will be explained further down).
- Trigger emails sent to arbitrary adresses.
- Not importing the email, but rather storing it in a temporary storage, where it may be examined and then imported or deleted.

Finally, eJournal RMS also includes functionality which will prevent email loops, which may occur when another automatic system, such as a vacation notice program, will reply to messages originating from the eJournal RMS. A definable treshold will make the system block any automatic emails from being sent to an address if a certain number of emails already have been sent to the same address within a specified period of time. Blocking may of course also be defined by the searchpattern method described above, e.g. to discard email comming from 'mailer-daemon' or similar services.

The eJournal RMS also includes functionality for escalating requests with certain priorities, if they have not been dealt with within defined periods. For instance, it is possible to define a customer or a company to have a predefined priority. If a request is posted by this customer, it will be set to a specified priority, and be placed at the beginning of a chain of events, each with a definable timeout. If a certain period of time elapses since the request was posted, and it has not yet been dealt with by the initially assigned user, it may change ownership to another user. If action is not taken within a new period of time, an email may be sent to the

person in charge, or possibly to a mobile phone. Using this functionality, it is possible to guarantee important customers response within e.g. two hours during regular business hours.

4.2 User functionality

Besides the functionality meant to automate as much as possible, eJournal RMS also includes several functions assisting the users of the system. Common operations such as redelegating requests, changing priorities, replying and postponing, are easily accessible. It is also possible to search among all the requests in the system, and perform batch operations on a set of requests. Requests will be connected to customers, and thus all communication with this customer, or all customers in a company, is readily available. Requests that are not currently solvable can be postponed to a given date, upon which they will be automatically awakened and a notice will be sent to the user. The system can be set up with several commonly used reply templates, e.g. guidelines or pricelists, which then easily can be used in the reply to the customer. A user will also easily see which requests are active, and requests where the customer has replied with new information will be identified. An administrator will also be able to see how many active requests there are in the system, for a specific category, or for a single user.

A significant part of the system is the statistics module. Using this module, reports and graphs may easily be created, displaying a vast range of information, such as average responstime pr. weekday, user or category. It is also possible to export the cummulative data for these statistics to a file accessible by e.g. Microsoft Excel. Based on these reports, it is easy to identify whether the level of support or response from the organization is within its goals. Since all information is broken down and stored in an SQL database, it is possible to extract virtually all kinds of statistics.

Another highly important module of the system is the FAQ (frequently asked questions) module. This module allows the users to create and maintain a hierarchly organized knowledgebase, accessible to both external and internal users. This knowledgebase consists of frequently asked questions with answer, binary attachments and WWW references. Requests can easily be modified and published to the FAQ database, and thus it is possible for the users to continously maintain the knowledgebase with new entries.The knowledgebase is keyword searchable, and entries may be included in replies to customers, either manually or automatically.

5. Integration

eJournal RMS is designed to be integratable with other existing solutions and databases, in order to avoid having to maintain duplicates of e.g. customer and company data. This solution has been designed to meet the following important criteria:

- **Redundancy.** eJournal RMS needs to be fully operational even if the external database is temporarily unavailable.
- **Efficieny.** The system cannot base itself on realtime connections to other databases, since this may slow down the system.
- **Multiplicity.** The system needs to be able to contain entries for customers who are not present in the external database. Furthermore, for customers who are present in the external database, additional information may be needed, such as email adresses, usernames or passwords. In these cases, information will be stored both in the primary database and in the external database.
- **Singularity.** Information about one customer should only need to be maintained through one interface.

In order to achieve these goals, the eJournal RMS uses a caching technique for operations where speed is important, such as listing several requests. Operations where speed is not an issue, such as identifying customers based on their email address when importing email, realtime connections to the external database is used. This becomes a one-way synchronization, which can be updated either trigged by events, such as listing a customer view, or by routinely tasks, such as nightly batches. Entries that exist in the external database are stored in the primary database, but marked as uneditable, and connected to the external database with a database identificator and foreign key. A timestamp also ensures that the information is not updated more often than necessary. Furthermore, timestamps and flags will show wether the external database is unavailable, and thus ensure that this will not slow down the system. It is also possible to integrate the eJournal RMS with several databases, which then will be accessed by defined priorities. Supported database access methods include ODBC, OCI and LDAP.

6. Requirements

The eJournal RMS has been designed to be a highly flexible system. The server will run on Microsoft NT, Linux, or any other modern Unix-flavour. Since the front-end is web-based, clients can run from any platform with a regular browser supporting frames, such as Windows 95/98/2000, Unix, Mac, Be OS or even handheld devices. Furthermore, the clients can access the system from any place with a Internet connection, depending on wether the server is accessible outside the organization's internal network. Following this model, administrating the system is straightforward. For upgrading, it is only neccesary to upgrade the server-side software, and no administration of the clients is necessary. Consequently, backup is also an easy task to administrate. A common solution is to mount the drive in a Windows Network, using SAMBA for Unix servers, and use existing backup routines. The system does not require any major hardware ressources, and a Pentium II 200Mhz or higher for the server should be sufficient. It is also quite possible to run the server software on an existing server with available ressources. Disk usage is limited to storing the messages, the attachments and the database files, but it will of course grow continously. We recomend starting with 2-3GB of space, which should be sufficient for a long period of time, depending on the usage of the system.

The eJournal RMS uses only internet standard protocols for communicating. Email is sent through a standard SMTP gateway, and recieved through POP3 mailboxes which will be scanned at definable intervals. The frontend is using standard HTML with no special features, through a regular HTTP service. For increased security, the system will also work transparently with a SSL enabled web server (HTTPS).