This page last changed on Apr 21, 2010 by ghostar.

Hera Proxied Email Manager Component Requirement Specification

1 Overview

The proxied email service is a basic email server facade that will allow for the actual email addresses of consumers to be anonymous when Hera partners wish to send emails. Consumers come to the site and fill out qualification forms to connect with Hera partners for a number of services. The by-product of this interaction is called a Lead. The client wishes to protect consumer privacy by masking consumer email addresses to partners but without disrupting normal email usage paradigms. Partners will need to communicate with consumers and vice versa. However the client wants to centrally route the email through a common gateway. Long term this central routing would allow features such as: charge partners per communication or to communicate at all and detect partners that violate privacy agreements by selling contact information to 3rd parties.
This component will provide methods for generation of proxy addresses, CRUD methods for access of real/proxy addresses and database logging service. This component encapsulates all access to the database.

1.1 Logic Requirements

The HeraProxiedEmailManager, IProxyEmailGenerator, HashBasedProxyEmailGenerator and EmailLog classes shown in the Hera Proxied Email Main Class Diagram need to be implemented. The following requirements are needed:

1.1.1 CreateProxyEmailAddress:

- It will use the HashBasedProxyEmailGenerator implementation by default although the implementation to be used should be configurable.
- The HashBasedProxyEmailGenerator will use the userId and emailAddress to generate a proxy address. The proxy address will be generated in the following format:

<userId><underscore><email address hash code>@<domain>

- Note that the local part of the address (the part that comes before @) can be a maximum of 64 characters. In this case it will contain only numbers because userId is a long and email address hash code (use object.HashCode) is an integer.
- If addToDatabase is true, then this will save the userId, emailAddress and generatedAddress to the UserEmailMap table.
- The generated address will be returned.

1.1.2 IsProxyEmailExist and IsRealEmailExist:

These methods will check in the UserEmailMap if the real/proxy email address exists. If the address exists, but is blocked then it returns false.

1.1.3 BlockUsersByRealAddress and BlockByRealAddress:

BlockUsersByRealAddress adds entries to the Blocking map table while BlockByRealAddress sets the Blocked flag in UserEmailMap table. The former is used for user-to-user blocking while the latter is used for global blocking.

1.1.4 LogEmailToDatabase:

This will save the parameters into the EmailLog and the EmailRecipients and EmailBccRecipients table. The body parameter is the entire message including the attachments as a byte array.

1.1.5 SearchEmailLogs:

This will search the email logs using search parameters.
The columnNames can be:

- FromUser
- Subject
- RecipientId
- BccRecipientId
- TimeStamp

The columnValues will be searched using direct equality. The string to DateTime conversion will be done using the default locale. SQL AND will be used for combining the search parameters. SQL IN will be used for searching the recipients.
Any improvements to this method (without adding significant complexity) are welcome.
All other methods of this class are simple database operations and are thus not described in any more detail.

1.1.6 Thread Safety, Concurrency and Transactions:

It is required to make the WCF service use Multiple Concurrency mode for the service. Hence the service operation will need to be thread-safe.
Access to database must be synchronized and default row-level locking should be enough to handle the database concurrency issues.
All database write operations must be protected by a transaction.

1.1.7 Configuration:

Configuration API should be used for providing configuration to all classes.

1.1.8 Exception Handling:

The Base Exception component must be used as the base component for all exceptions defined in this class.

1.1.9 Logging:

Logging Wrapper component must be used for logging. Logging should be supported at ERROR level at least. Method entry, exit and parameters may be logged.

1.1.10 Data Access:

All data access must be through (existing, modified or new) stored procedures

1.2 Required Algorithms

None

1.3 Example of the Software Usage

This service will provide an API to an application which can create and retrieve email addresses and email logs.

1.4 Future Component Direction

- None as of now

2 Interface Requirements

2.1 Graphical User Interface Requirements

None

2.2 External Interfaces

See TCUML Hera Proxied Email Main Class Diagram

Architecture: Hera_Proxied_Email_Architecture.zip

2.3 Environment Requirements

Development Language: C# 3.5
Compile Target: C# 3.5

2.4 Package Structure

Hera.ProxiedEmail

3 Administration Requirements

3.1 What elements of the application need to be configurable?

- The instance of IProxyEmailGenerator to be used for generating proxy addresses.
- The domain name to use when generating proxy address.
- The logger to use and whether to use logging.
- The database connection properties

3.2 Technical Constraints

Are there particular frameworks or standards that are required?

C#, SQL

3.2.1 TopCoder Software Component Dependencies:

- Configuration API 1.0
- Logging Wrapper 3.0
- Exception Manager 2.0
- Connection Factory 1.0 (usage is upto designer)
- Data Access Interface 2.1.1 (usage is upto designer)
- Object Factory 1.2 (usage is upto designer)

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.2 Third Party Component, Library, or Product Dependencies:

None

3.2.3 QA Environment:

Windows Server 2003 or 2008
SQL Server 2005

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.1 Help / User Documentation

- Design documents must clearly define intended component usage in the 'Documentation' tab of the TCUML Tool.