

Functional Specification for:

S3ONTHEGO

A secure offsite file backup & accessibility solution

Prepared by: Cormac Redmond
CASE4
53478637

Date: 08/1/2007

Supervisor: Dr. Brian Stone

TABLE OF CONTENTS

TABLE OF CONTENTS	2
1. INTRODUCTION.....	3
1.1 OVERVIEW	3
1.2 BUSINESS CONTEXT	3
1.3 GLOSSARY	3
2. GENERAL DESCRIPTION	5
2.1 PRODUCT/SYSTEM FUNCTIONS	5
2.1 USER CHARACTERISTICS AND OBJECTIVES	6
2.1 OPERATIONAL SCENARIOS	7
2.4 CONSTRAINTS	8
3. FUNCTIONAL REQUIREMENTS	8
3.1 USER AUTHENTICATION.....	9
3.2 SIGNATURE CREATION	9
3.3 FILE UPLOADING	10
3.4 BACKUP ITEM & SNAPSHOT CREATION/MAINTENANCE.....	11
3.5 FILE & SNAPSHOT RESTORATION.....	11
3.6 BACKUP SCHEDULING	12
3.7 BACKUP LOGGING.....	12
4. SYSTEM ARCHITECTURE	13
4.1 S3ONTheGo SERVER	14
4.2 AMAZON S3 SERVER.....	15
4.3 USERS	15
5. HIGH-LEVEL DESIGN	15
5.1 S3ONTheGo WEB SERVICE.....	16
5.1 S3ONTheGo APPLICATION	17
6. PRELIMINARY SCHEDULE.....	19
6.1 GANTT CHART	19
6.2 HARDWARE AND SOFTWARE RESOURCE REQUIREMENTS	20
7. APPENDICES	20
A. AMAZON S3 OVERVIEW	20
A.1. <i>Interacting with Amazon S3</i>	21
A.2. <i>Common Elements</i>	22

1. INTRODUCTION

1.1 Overview

This document describes the requirements for “S3OnTheGo”, a secure off-site backup solution for Microsoft Windows, based on the Amazon S3 Web Service. The aim of S3OnTheGo is to provide an organisation with the ability to allow its employees to backup and synchronise a selection of their files, from any internet-connected location.

The end goal is a backup solution that:

- Is secure to its users and the providing organisation
- Is easily installed and configured
- Requires only one server per S3OnTheGo service-provider to operate
- Gracefully handles failure
- Provides an intuitive and fully functional graphical interface
- Allows the notion of “snapshots” of a backup
- Allows users to schedule backups
- Is cheap and easy to maintain

1.2 Business Context

Current backup solutions generally rely on separate pieces of hardware, such as tapes, hard drives, REV™ drives, CD/DVDs, and NASs (Network-Attached Storage). In the event of a server crash or other disaster, files and system state can be restored from the backup media. These conventional solutions can be costly, always entail the deployment and maintenance of extra hardware and can often over-complicate a simple set-up. These restraints can preclude their use by the “small-time” operator, such as a small business.

S3OnTheGo removes these extra layers of complication to provide a robust and secure backup solution, which also acts as a type of “revision control”, whereby users can rollback to an older “snapshot” of a particular backup, if necessary. On top of that, it provides the ability for users to retrieve and restore backups from any internet-connected location, thus allowing greater flexibility for the travelling user.

The system is intended for use by small-to-medium sized businesses as an alternative or reinforcement to conventional backup systems, and also as a tool to increase accessibility amongst its employees/users.

1.3 Glossary

S3OnTheGo server/service:

Any organisation/entity that holds an Amazon S3 account, and offers the relevant S3OnTheGo web services through an internet-accessible server is an S3OnTheGo service provider.

User:

Anyone who is authorised as a user with an S3OnTheGo service is a user of the software.

SOAP:

SOAP (originally Simple Object Access Protocol) is a protocol for exchanging XML-based messages over computer network, normally using HTTP.

WSDL:

The Web Services Description Language is an XML format published for describing Web services.

XML:

XML (Extensible Mark-up Language) is a W3C initiative that allows information and services to be encoded with meaningful structure and semantics that both computers and humans can understand. XML is ideal for information exchange, and can easily be extended to include user-specified and industry-specified tags.

Amazon S3:

Amazon S3 (Simple Storage Service) provides a web service that can be used to store and retrieve any amount of data, at any time, from anywhere on the web - for a small price. See Appendix A for more details.

Web service:

Web Services are application components that use:

- WSDL - Web Services Description Language, for self-description
- TCP/IP, for transport.
- HTTP, for interaction.
- SOAP - Simple Object Access Protocol, for requesting and granting actions
- XML, for underlying representation

Advanced Encryption Standard (AES):

Advanced Encryption Standard (AES), also known as Rijndael, is a block cipher adopted as an encryption standard by the U.S. government.

Hash function:

A hash function takes a string of any length as input and produces a fixed length string as output, sometimes termed a “message digest” or a “digital fingerprint”. The SHA (Secure Hash Algorithm) hash function, considered to be the successor to MD5 (an earlier, now compromised hash function), will be used in this system.

Salting:

Salting is the process of adding some additional random data to the front of a password before it is hashed, so that in the event that two users happen to have the same password, it isn't obvious by looking at the resulting hash values. The plaintext salt has no value to an attacker (or a malicious system admin) because its function is solely to remove the ability to detect duplicate passwords.

Backup Item:

A Backup Item is an object containing a reference to a chosen directory (for backup), a schedule time to perform the backup (if any), an Item name, and any number of “snapshots”.

Snapshot:

Each time a backup is performed on a Backup Item, a “snapshot” is created. It holds the information identifying what files were backed up at that particular time, and what time the backup occurred, etc.

Serialization:

Serialization is the process of saving an object onto a storage medium (such as a file, or a memory buffer) or to transmit it across a network connection link, either as a series of bytes or in some human-readable text format such as XML

GUID:

Globally Unique Identifier or GUID is a pseudo-random number used as a unique identifier. While each generated GUID is not guaranteed to be unique, the total number of unique keys (2^{128} or $3.40282366 \times 10^{38}$) is so large that the probability of the same number being generated twice is very small.

2. GENERAL DESCRIPTION

2.1 Product/System Functions

From a high-level perspective, the functions which the system must fulfil are as follows:

Authenticate users:

The system must authenticate users before the application will progress. Upon authentication, the application will proceed. No two users should ever be able to gain access to the same data.

Securely backup files:

The system must backup files in a secure and efficient manner. Files must be encrypted using the highest standards and no unnecessary file transfer must occur.

Retrieve users' backups:

The system must be able to retrieve users' backups from any location. This includes restoring any Snapshot (i.e., older backups of the same Backup Item) from a Backup Item. No unnecessary or unauthorised file transfer must occur.

Allow for scheduling of backups:

The system should handle the scheduling of backups, and perform these operations in the background.

Gracefully handle failure:

The system must not break due to minor failure. Due to the error prone nature of a network connection, the system must handle such errors (within reason).

Maintain logs:

For each user, a log must be maintained which will be updated upon the relevant operations.

Operate securely:

No unnecessary private information should be stored on the user's computer at any time. For instance, the Amazon S3 secret access ID is needed for authorisation with Amazon S3, but the user's machine should never have access to this. Also, S3OnTheGo system administrators should not have access to a user's data.

2.1 User Characteristics and Objectives

A user, as defined above, is anyone who is approved as a user with an S3OnTheGo service. Normally, a company administrator would setup an S3OnTheGo server, and add users to the database. The application should be easily utilised by a novice user. The objectives and requirements of the user-community are expected to be the same as those objectives and requirements that apply to any type of backup software:

Easy and intuitive usage:

Operating the application should be effortless, clear and user-friendly.

Minimal configuration:

Configuring the application must only consist of initially entering an S3OnTheGo server address and, if necessary, a proxy server address. A username and password must be all that is required thereafter.

Error recovery:

Any errors should be detected and handled gracefully in order to minimise the impact on the user.

Logging:

It is expected that the users will require that the system is capable of providing ample statistical information on its operations.

In addition to this the user should expect to be able to perform the following self-explanatory operations:

- Configure application
- Login
- View Backup Items
- Create Backup Item
- Delete Backup Item
- Perform backup
- Perform backup restore

- Schedule a backup
- View application logs

2.1 Operational Scenarios

The following are descriptions of potential end-to-end transactions involving the system for the functions offered to the user:

Initial configuration of application:

- The first time the application is run on a machine, users are prompted to enter server and proxy information
- Once entered, the user is requested to login

Login:

- Upon loading the application, users are prompted to enter a username and password
- Then, after authorisation, the main program will proceed to load
- In the event of authorisation failure, users are informed of the error and requested to repeat the operation

View Backup Items:

- After logging in, a frame presents a list of all the users' Backup Items, regardless of where or when they were created, and regardless of whether or not a backup has been performed on the Item
- The list includes details on the last time of backup and also on scheduling
- Users are able to perform operations on the presented Backup Items, as described below

Backup Item creation:

- The user chooses to create a Backup Item, and is presented with a small window
- They are prompted to select a backup directory, a schedule to perform the backup, and a Backup Item name
- Upon clicking 'OK', the Backup Item is saved, and the main list is refreshed

Backup:

- The user chooses to perform a backup on a Backup Item (as long as the Backup Item was created on that machine)
- Once chosen, a progress window presents itself and the application proceeds to backup the files
- The progress bar updates continuously
- The user is notified upon completion or error

Backup Item deletion:

- The user chooses to delete a Backup Item
- Upon verification, the application proceeds to remove the Backup Item from the Backup Item list, and deletes all backed-up files associated with it

Backup Item restoration:

- The user chooses to perform a restore on a Backup Item

- Next, user is presented with a list of Snapshots to choose from
- After the choosing a Snapshot, the user is presented with a list of all backed-up files associated with that Snapshot
- The user chooses which files they wish to restore
- The application restores the files while providing progress information

Backup Item scheduling:

- The user chooses to create a backup
- A selection of time intervals is presented
- Once chosen and accepted, the application performs the backup in the background, accordingly

Edit configuration:

- The user chooses to change configuration settings
- A window presents itself displaying the following choices:
 - S3OnTheGo server
 - Proxy server
 - Proxy port
- User enters information and selects ‘OK’
- The application restarts so the new configuration is taken into effect

View logs:

- The user chooses to view logs
- A window is presented displaying the logs in a readable fashion

2.4 Constraints

It will not be feasible to test the system with more than two or three machines simultaneously, so questions will remain concerning the scalability of the system.

Also, the Amazon S3 service seems to be temperamental at times, judging by complaints received on the developer forums. This could potentially slow down development and/or have a negative effect on the completed system. Furthermore, there are some file size limitations according to the Amazon S3 release notes: “A load balancer bug causes the connection to close whenever an upload request with content-length between 2 GB and 4 GB is received. Amazon has received a fix for the issue from the load balancer vendor, and is in the process of testing the fix.”

Because S3 is no longer beta, such problems are surprising. However, its developers maintain that they are working quickly to eliminate any residing problems.

3. FUNCTIONAL REQUIREMENTS

In order to understand the following, it is necessary to first understand the Amazon S3 web service, as described in Appendix A.

The following requirements of the system can be identified in ranking order:

3.1 User Authentication

Description:

When a user attempts to run the application, they must first authenticate themselves by providing a username and password. This requirement is concerned with the steps in carrying out the authentication

Criticality:

Obviously this is a key requirement – it is crucial in upholding the security and integrity of the system. Secure authentication is vital in protecting the system's users from malicious intent.

Technical issues:

The S3OnTheGo server will contain a database containing a list of valid users. For each user, the database should hold the username, a salted hashed digest of the user's password using a SHA256/SHA512 hash function, and the unique salt value itself.

Therefore, when a user provides a username and password, it is expected that the application will retrieve the salt value from the S3OnTheGo server (through a web service), and create the SHA256/SHA512 digest of the password and salt value. Then, this digest should be sent back to the S3OnTheGo server, which compares its value with that of the digest in the database. This secure method has the advantage that a user's password is never sent 'over-the-wire'.

Dependencies with other requirements:

None

3.2 Signature Creation

Description:

The application exchanges information with S3 through the SOAP protocol and every SOAP request must contain a signature, as described in Appendix A. The signature provides verification that the principal possesses the AWS Secret Access Key corresponding to the specified Access Key ID. Each signature is only functional for a short amount of time (less than 15 minutes).

Criticality:

This is a completely necessary process, as all operations regarding S3 require signature creation.

Technical issues:

The signature element contains the HMAC-SHA1 digest of the values "AmazonS3" + OPERATION + Timestamp, where OPERATION is a SOAP operation, and Timestamp is a current time object, as described in Appendix A. The AWS Secret Access Key is used as a key to create the digest.

In order to keep the AWS Secret Access Key secure, it is envisaged that the application will request a signature from the S3OnTheGo server, through a web

service. This has the additional benefit that the application/user will never have access to the AWS Secret Access Key.

Dependencies with other requirements:

Authentication would be required before this process could take place.

3.3 File Uploading

Description:

Uploading files is an essential part of this system. All the users' backup files must be uploaded, as well as information on their Backup Items, and Snapshots, etc. Files must also be encrypted before uploading, and no files should be transferred unnecessarily. Also, users should have no control over how files are encrypted.

Criticality:

This step is completely necessary as a core part of the system. The system performance could be partly quantified based on the reliability and efficiency of file uploading.

Technical issues:

As mentioned earlier, Amazon has a 5 gigabyte file size limit, and uploading files between 2 to 4 gigabytes often causes problems. To overcome this, the application will segment large files (over one megabyte, for example) into multiple parts before uploading. This should allow for easy error-handling; e.g., if a network error occurs, the application can re-commence uploading from the point where the error occurred. Also, due to the nature of SOAP, it is necessary to employ such an approach in order to determine the progress of a file upload.

As stated in Appendix A, Amazon S3 is used to store objects. Objects are the closest representation to a normal file. Each object has four parts: value, key, metadata, and an access control policy. Each object is contained a bucket. This system will only have one bucket, i.e. S3OnTheGoBucket, or similar. The access control policy has no relevance, as all objects will be private by default.

Before uploading a file, a key will be made so that the Backup Item to which the file belongs can be identified. Also necessary, will be a way to determine the 'version' of the file. This will probably be done by creating a hash digest of the file path amalgamated with the last modification time of the file. Finally, when files are stored in multiple parts, a part number should be created and added. Utilising such a naming convention will also provide the means to determine where no file upload is necessary – i.e., where the file is already uploaded (from previous backups, etc).

With regards to encryption, all files should be encrypted using the Advanced Encryption Standard (see Glossary). They will be encrypted using the user's hashed password as an encryption key, thus eliminating the ability of an unauthorised person to decrypt a user's files.

Dependencies with other requirements:

This requirement depends on the Authentication and Signature requirements.

3.4 Backup Item & Snapshot Creation/Maintenance

Description:

When a user creates a Backup Item, or performs a backup, the application must take this into effect. Backup Items and Snapshot information must be available from any location, and also employ some method to show and maintain the relationship between two.

Criticality:

This step is essential in the organisation and operation of user-chosen backups and file restoration.

Technical issues:

Each Backup Item should contain a reference to a directory, a scheduling time interval, and number of references to Snapshots. Each Snapshot should contain a reference to its parent Backup Item, and a list of the files which were backed up. These objects should have a consistent naming convention in order to interact with the rest of the system.

When a user creates a Backup Item, the system should store this information at S3, so it is accessible from any location. Also, when a backup is performed, a Snapshot is created, and uploaded too. This will probably be carried out by serializing the Backup Item and Snapshot objects, and then uploading them like any other file.

Each Backup Item should have a unique identifier, such as GUID. Each Snapshot should also have a unique identifier but also a way of determining to which Backup Item it belongs. Therefore, when storing these objects on S3, these unique identifiers could be used as a way to determine the relationship between a Backup Item and its Snapshots, through the use of metadata, or some filename prefix, etc. This will in turn allow for the application to distinguish between Backup Items, and a Backup Item's Snapshots, etc.

Dependencies with other requirements:

This requirement depends on the Authentication, Signature Creation, and File Uploading requirements.

3.5 File & Snapshot Restoration

Description:

Restoring files might take place for a few reasons: The application may possibly be requesting information on a user's Backup Item, or the Backup Item's Snapshots. Or, the user may be requesting to restore their files from a Backup Item.

Criticality:

File restoration is, in essence, the whole initiative of a backup system, and so this requirement is critical. The system performance could be partly quantified based on the reliability and efficiency of file restoration.

Technical issues:

Restoring a user's Backup Item and Snapshot information should simply involve sending a SOAP request and then decrypting the received files. However, when a user chooses to restore files from a Snapshot, more steps will be required. As mentioned earlier, a Backup Item can have many Snapshots. A Snapshot is a record of a previous backup, which maintains a record of file hashes, etc. Thus, by making use of the naming convention described in 3.3, the application should be able to determine exactly what files should be restored for each Snapshot and also, by checking the local file system, prevent any unnecessary downloading.

Similarly, as described in 3.3, by handling large files in small parts, error handling is improved and the retrieval is more efficient.

The next step includes the amalgamation and appropriate renaming of files and file parts.

Dependencies with other requirements:

This requirement depends on the Authentication, Signature Creation and Backup Item & Snapshot Creation/Maintenance requirements.

3.6 Backup Scheduling

Description:

Through interacting with a Backup Item, a user can choose to schedule a backup at certain time intervals. The application must be aware of this and carry out the backup in the background, as necessary.

Criticality:

Backup scheduling is only a feature of the system and is not essential for core functionality. However, such a feature would be expected by the user-community, and is therefore quite necessary.

Technical issues:

This requirement could be implemented in a number of ways. The most obvious way would be to implement some form a timer that regularly performs a check and begins a backup at the given time.

Dependencies with other requirements:

In carrying out a schedule, this requirement depends on all the above requirements.

3.7 Backup Logging

Description:

When any operations are performed, such as restoring a backup, or creating a Backup Item, the system should record this information and allow for a user to view it. This information will be based on the user's machine, and not on the S3 server. It should only be concerned with recent activity.

Criticality:

This step is only an added feature of the system and not necessary to the core functionality. It is a requirement of the user community and insuring that the logs are informative and easy-to-read, is important.

Technical issues:

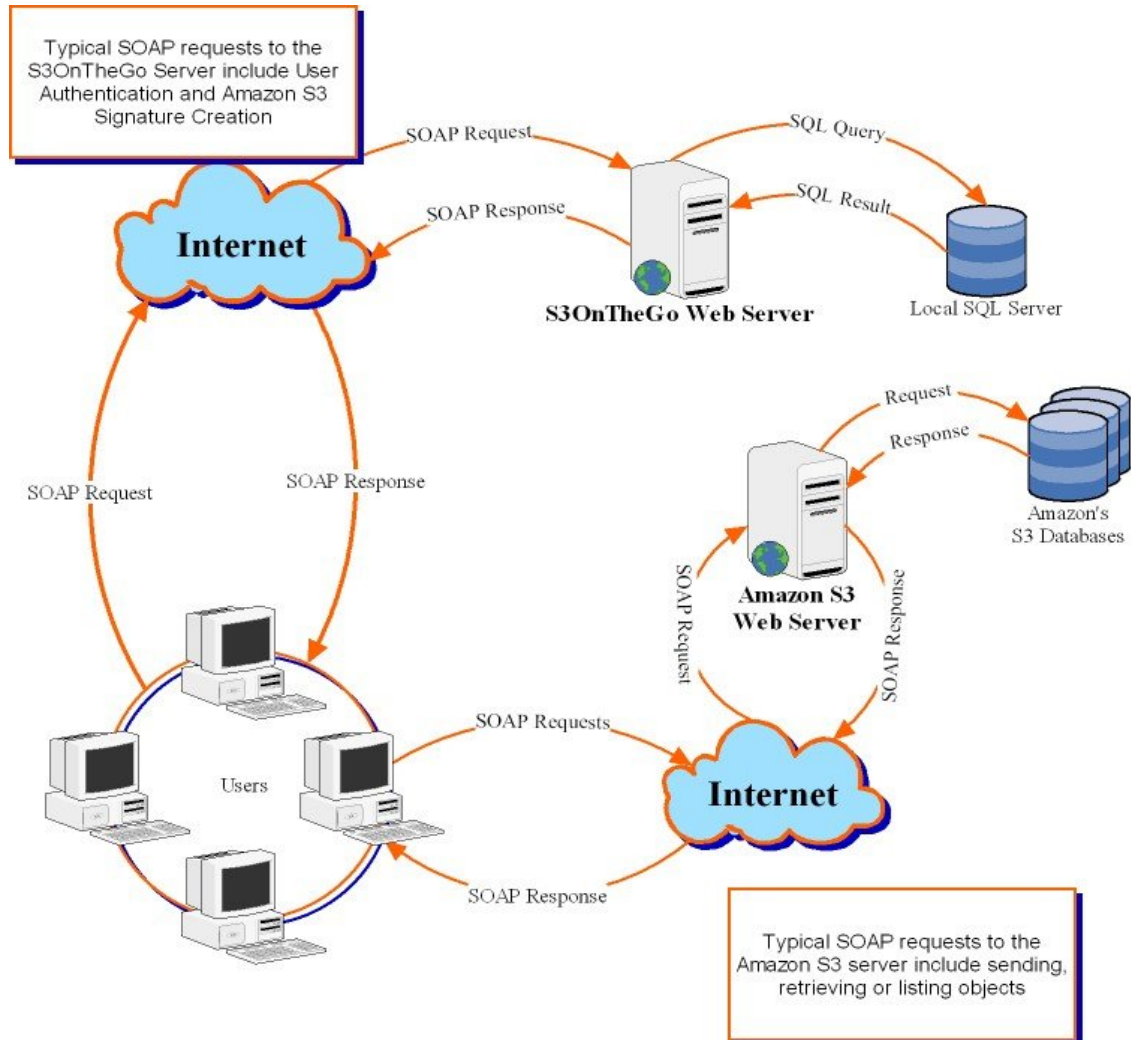
The easiest way to store and read such information would be through using an XML file. Each time a relevant action is carried out, the function handling the action should write to the relevant information to the log file.

Dependencies with other requirements:

This requirement depends on most other requirements for the generation of the logs.

4. SYSTEM ARCHITECTURE

This section describes the system's context and architecture in relation to the below diagram:



As evident from the diagram, the architecture of the system can be broken down into three components:

- An S3OnTheGo Server
- The Amazon S3 Server
- The Users

4.1 S3OnTheGo Server

The S3OnTheGo server is responsible for the User Authentication and Signature Creation requirements. This will be possible through providing a web service

operating on the SOAP protocol, allowing users to authenticate themselves, and request signatures.

Typically, a company implementing S3OnTheGo will setup this server, create an Amazon S3 account, and install the provided web service. This will all be possible through a simple S3OnTheGo administration utility. An SQL Server must be installed in order to hold user information, such as the username, salt value, and password digest, etc. Once complete, users must inform the application of the S3OnTheGo server hostname, which then allows it to carry out authentication, signature creation, etc.

4.2 Amazon S3 Server

The Amazon S3 server provides the SOAP Web Service, allowing the application to request operations on buckets and objects. Each SOAP request requires a signature, which the S3OnTheGo server can create on-the-fly, as needed.

4.3 Users

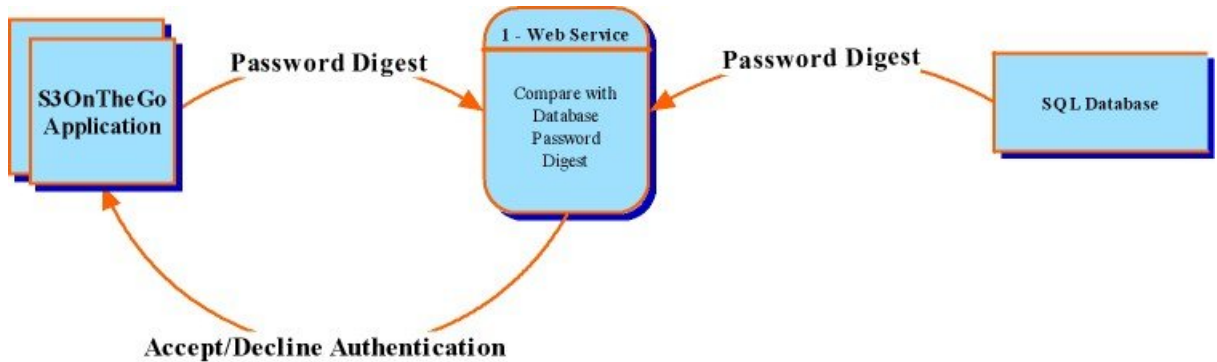
The users are a group of people who are logged on to the S3OnTheGo service. The user application will communicate between the S3OnTheGo server and the Amazon S3 server.

5. HIGH-LEVEL DESIGN

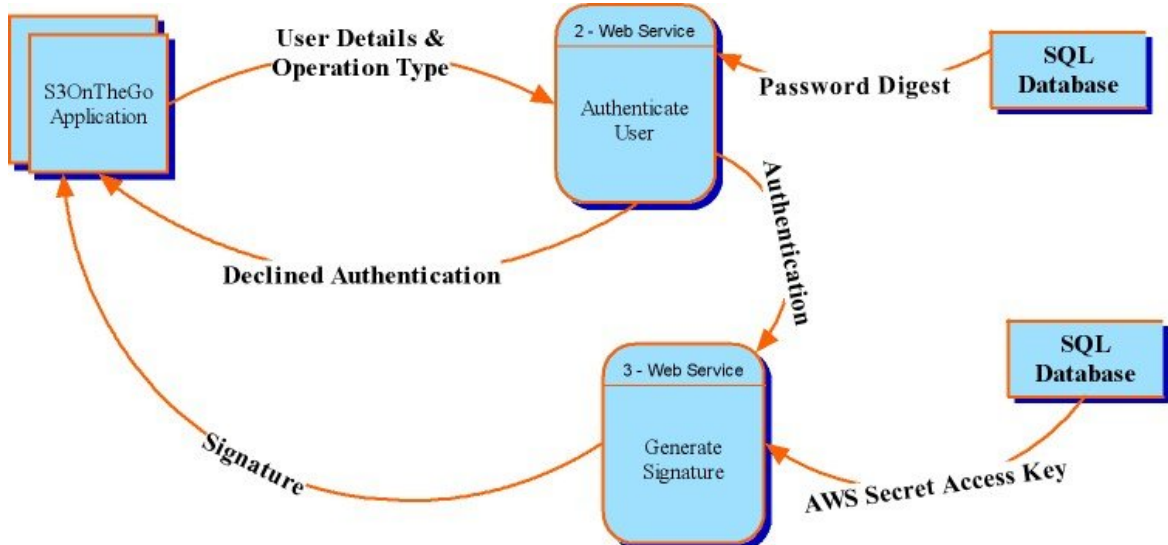
The following sections show at a high-level how the system should behave under different circumstances:

5.1 S3OnTheGo Web Service

DFD describing the behaviour of the web service in dealing with “Initial User Authentication”:

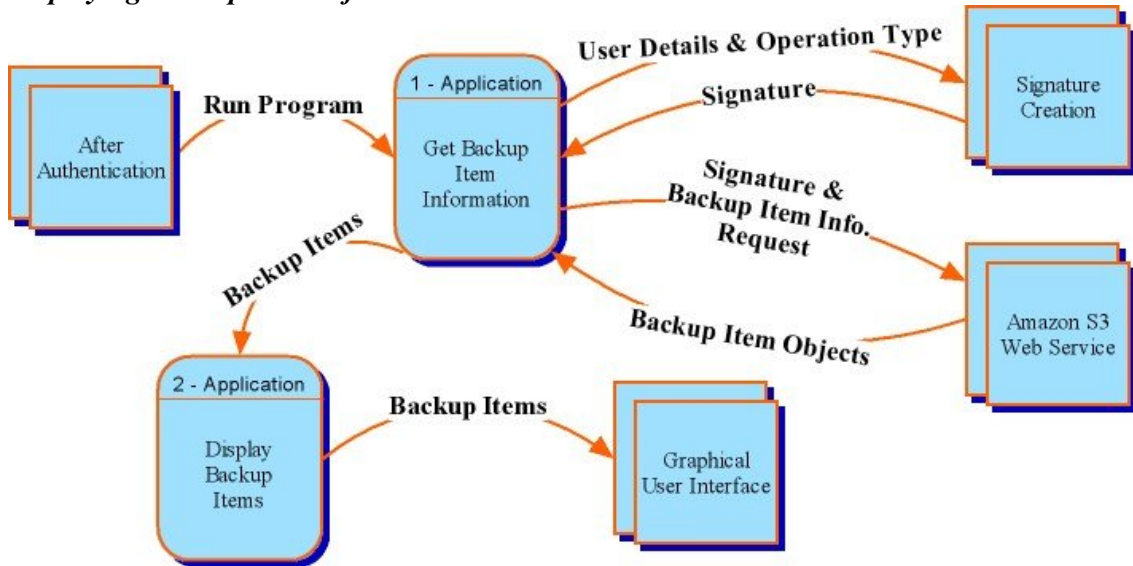


DFD describing the behaviour of the web service in dealing with “Signature Creation”:

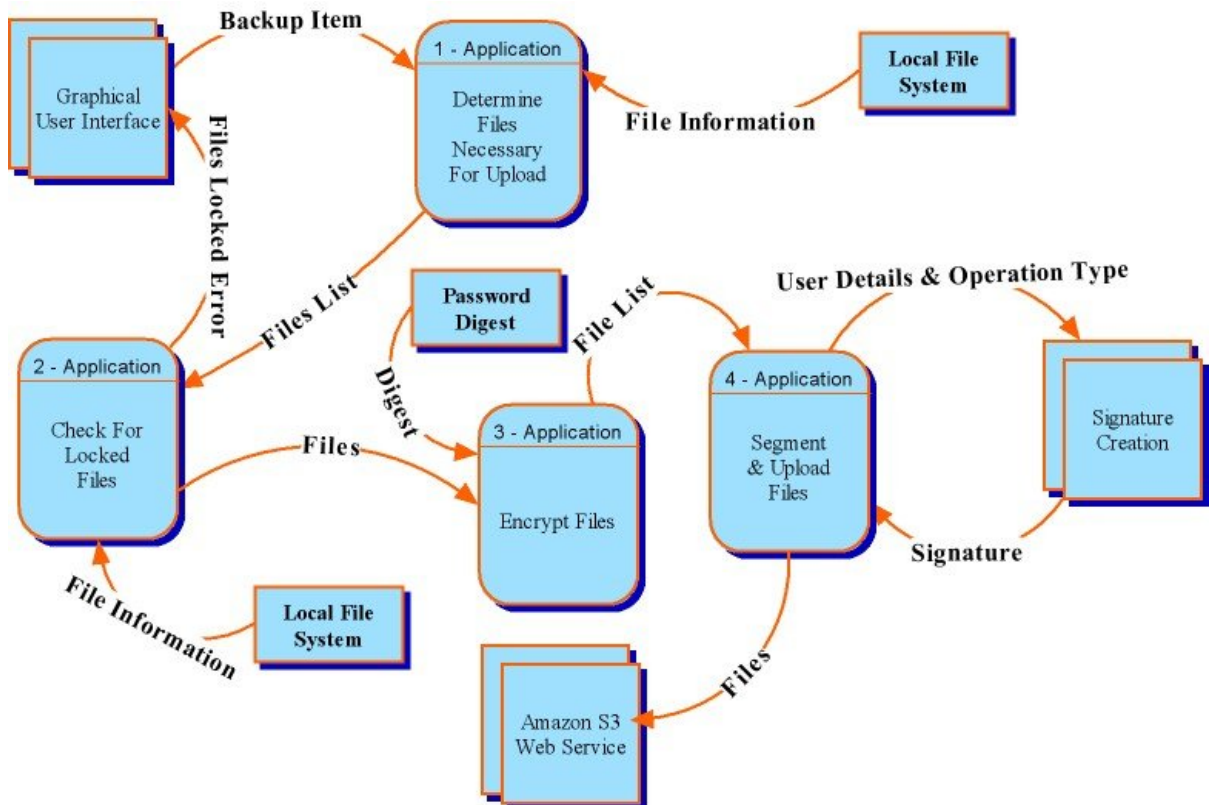


5.1 S3OnTheGo Application

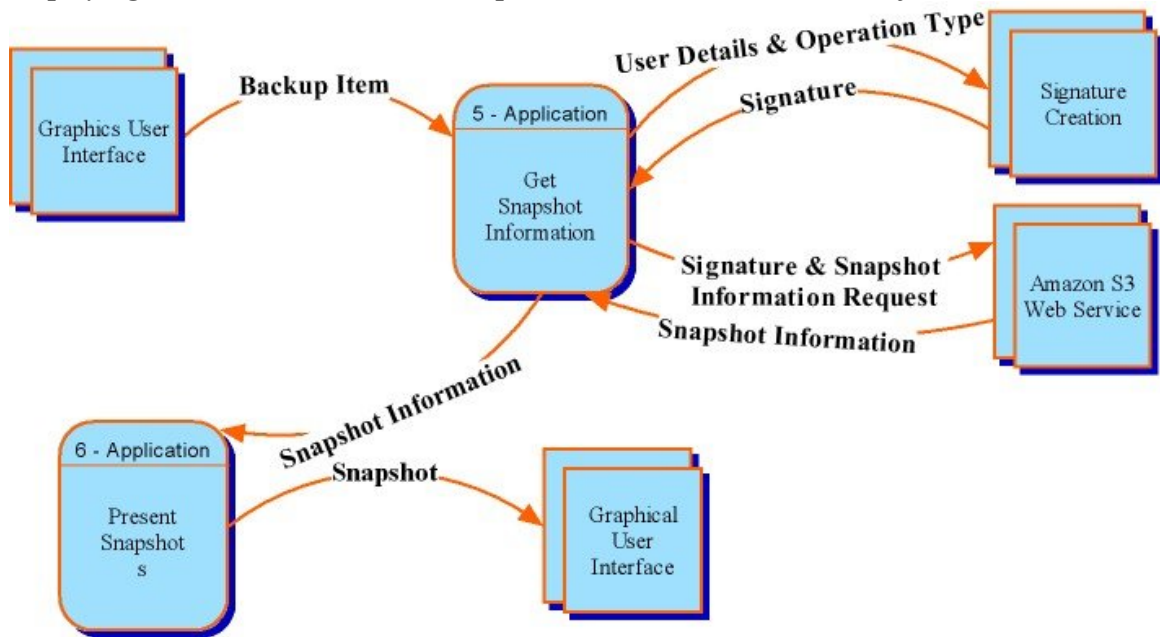
DFD describing the behaviour of the application in dealing with “Retrieving & Displaying Backup Item Information”:



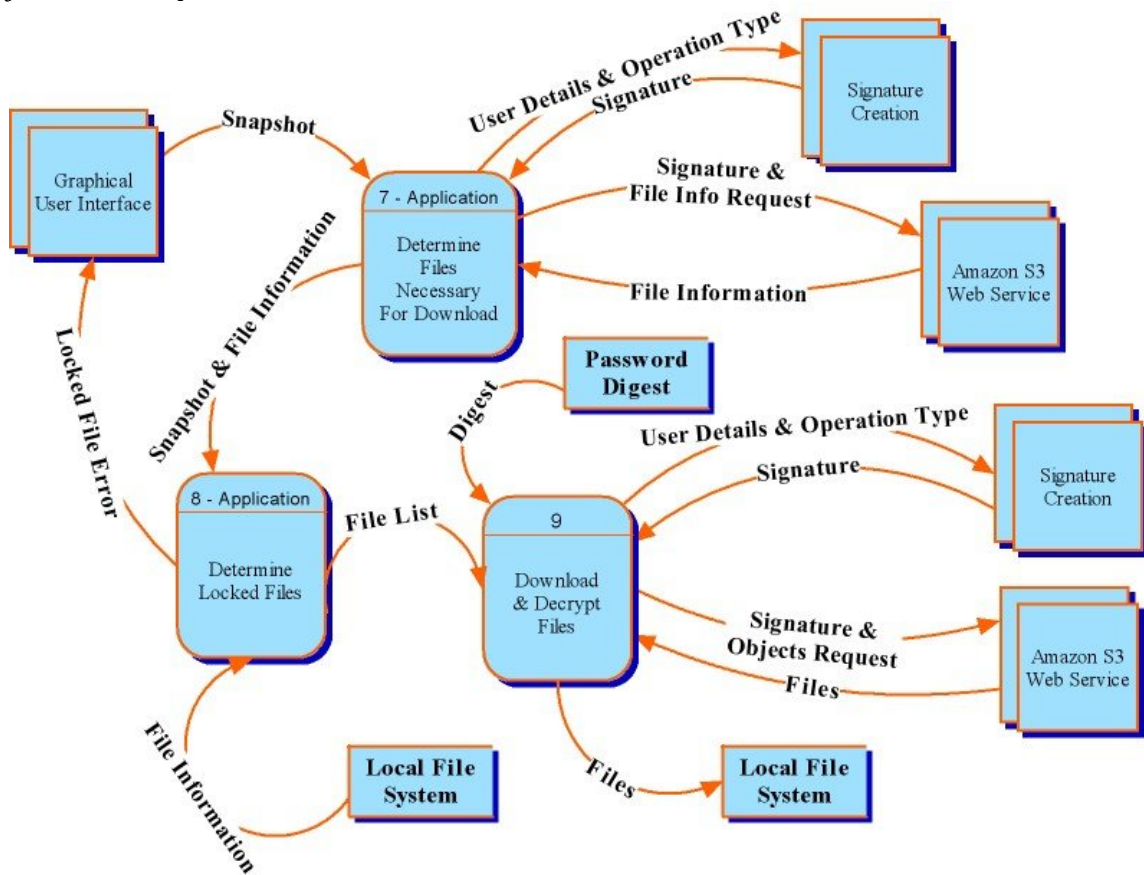
DFD describing the behaviour of the application in dealing with “Performing a Backup on a Backup Item”:



DFD describing the behaviour of the application in dealing with “Retrieving & Displaying Snapshot”:



DFD describing the behaviour of the application in dealing with “Restoring Files from a Backup Item”:



6. PRELIMINARY SCHEDULE

6.1 GANTT Chart

ID	Task Name	Start	Finish	Duration	Feb 2007				Mar 2007				Apr 2007			
					4/2	11/2	18/2	25/2	4/3	11/3	18/3	25/3	1/4	8/4	15/4	22/4
1	Amazon S3 Research	01/02/2007	07/02/2007	1w												
2	Basic File Functionality	01/02/2007	12/02/2007	1w 3d												
3	Efficient File Upload Functionality	12/02/2007	14/02/2007	3d												
4	Efficient File Restore Functionality	14/02/2007	15/02/2007	2d												
5	Backup Item & Snapshot Functionality	15/02/2007	19/02/2007	3d												
6	S3OnTheGo Server SQL Database	19/02/2007	21/02/2007	3d												
7	S3OnTheGo Server Web Service	21/02/2007	23/02/2007	3d												
8	User Authentication Functionality	21/02/2007	27/02/2007	1w												
9	Windows Forms Research	27/02/2007	06/03/2007	1w 1d												
10	Basic Graphical Interface	02/03/2007	14/03/2007	1w 4d												
11	Advanced Graphical Interface	14/03/2007	03/04/2007	3w												
12	Activity Logging	03/04/2007	09/04/2007	1w												
13	Backup Scheduling	09/04/2007	12/04/2007	4d												
14	Testing	12/04/2007	23/04/2007	1w 3d												
15	Documentation	23/04/2007	30/04/2007	1w 1d												

Notes:

Basic File Functionality refers to the uploading and downloading of files from the Amazon S3 Web Server, without any error handling or efficiency issues taken into consideration.

S3OnTheGo Server SQL Database refers to installing the database, creating a database schema, and then developing the relevant stored procedures.

S3OnTheGo Server Web Service refers to the creation and testing of an ASP.NET Web Service, which will be used for user authentication, and signature creation.

Advanced Graphical Interface refers to enhancing the Basic Graphics Interface to a professional standard, and also the implementation of extra features, as described throughout this document.

6.2 Hardware and Software Resource Requirements

The following technologies will be used to develop the system:

- C#.NET (Microsoft .NET Framework 2.0)
- ASP.NET (Microsoft .NET Framework 2.0)
- SQL Server 2000/SQL Server 2005 Express
- Amazon Simple Storage Service (Amazon S3 web service)

Two or three computers will be necessary in both the development and demonstration of the system. One will act as the S3OnTheGo server, which will contain an SQL Server database, and an ASP.NET Web Service. This will be on a Microsoft Windows platform. The other machines will act as users of the system and contain the S3OnTheGo user application.

Both the S3OnTheGo server and user machines must have the Microsoft .NET 2.0 Framework installed. Alternatively, users may install the user application on a Linux platform running a later version of Mono 1.2. All machines must be connected to the Internet.

Also, it is anticipated that some third party libraries could be employed in the building of the graphics user interface, to provide a more professional look and feel.

7. INSTALLING & CONFIGURING THE S3ONTHEGO SYSTEM

8. APPENDICES

A. Amazon S3 overview

It is crucial to understand how the Amazon S3 web service works in order to understand the major topics in this document:

Amazon S3 has a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It is intentionally built with a minimal feature set:

- Write, read, and delete objects containing from 1 byte to 5 gigabytes of data each, with accompanying metadata. The number of objects which can be stored is unlimited
- A straightforward flat object store model, where each object is stored and retrieved via a unique developer-assigned key.
- Authentication mechanisms are provided to ensure that data is kept secure from unauthorized access. Objects can be made private or public, rights granted to specific users.

- Standards-based REST and SOAP interfaces designed to work with any Internet-development toolkit.

Bucket:

A bucket is simply a container for objects stored in Amazon S3. Each S3 account holder can create up to 100 buckets. Every object is contained within a bucket.

Object:

Amazon S3 is used to store objects. An object has four parts: value, key, metadata, and an access control policy.

There is no notion of a file system, and it is left up to the developer to define their own by using keys and metadata:

Key:

The key is the handle that is assigned to an object so that it can be fetched later. No two objects in a bucket may have the same key. A key is a sequence of Unicode characters whose UTF-8 encoding is at most 1024 bytes long.

Keys can be listed by prefix. By choosing a common prefix for the names of related keys, and marking these keys with a special character that delimit hierarchy, you can use the list operation to select and browse keys hierarchically. This idea is similar to how groups of files are organized into directories in a file system.

Keys are often given a suffix that indicates something about the type of data in the object, though this is not required. For example, often keys will use a suffix of ".jpg" to indicate that an object is an image.

Metadata:

An Amazon S3 object's metadata is a set of key-value pairs associated with the object. The idea is to store information about the object in its metadata. There are two kinds of metadata: system metadata, and user metadata.

- System metadata is utilized by Amazon S3, and is sometimes computed by Amazon S3
- User metadata entries are specified by the developer, the user of Amazon S3.
- Amazon S3 does not interpret this metadata - it simply stores it, and then passes it back when it is requested
- Metadata keys and values can be any length, but must conform to UTF-8

A.1. Interacting with Amazon S3

Standards-based REST and SOAP interfaces are designed to work with any Internet-development toolkit. This system will interact with S3 using SOAP 1.1 over HTTP. The S3 WSDL, which describes the S3 API in a machine-readable way, is available at: <http://s3.amazonaws.com/doc/2006-03-01/AmazonS3.wsdl>.

SOAP Endpoint:

This system will send S3 SOAP messages to an SSL secured endpoint. Note that authenticated SOAP requests are only accepted over SSL. The available S3 SOAP endpoints are <http://s3.amazonaws.com/soap> and <https://s3.amazonaws.com/soap> (SSL).

A.2. Common Elements

The account holder is assigned an AWS Access Key ID and Secret Access Key when they register for an AWS account at <http://aws.amazon.com>. The following are authorization-related elements necessary with *every* SOAP request:

AWSAccessKeyId:

The account holder's AWS Access Key ID.

Timestamp:

This must be a dateTime (<http://www.w3.org/TR/xmlschema-2/#dateTime>) in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as 2005-01-31T23:59:59.183Z. Authorization will fail if this timestamp is more than 15 minutes away from the clock on Amazon S3 servers.

Signature:

The signature is a RFC 2104 HMAC-SHA1 digest (described in glossary or at <http://www.ietf.org/rfc/rfc2104.txt>) of the concatenation of "AmazonS3" + OPERATION + Timestamp, using your AWS Secret Access Key as the key. For example, in the following CreateBucket sample request, the signature element would contain the HMAC-SHA1 digest of the value "AmazonS3CreateBucket2005-01-31T23:59:59.183Z":

```
<CreateBucket xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Bucket>quotes</Bucket>
  <Acl>private</Acl>
  <AWSAccessKeyId>1D9FVRAYCP1VJS767E02</AWSAccessKeyId>
  <Timestamp>2005-01-31T23:59:59.183Z</Timestamp>
  <Signature>SZf1CHmQ/nrZbsrC13hCZS061yws</Signature>
</CreateBucket>
```