CHAPTER 1

ORGANIZATION PROFILE

1.1 INTRODUCTION

Tata Consultancy Services Limited (TCS) is an Indian information technology, consulting, services and business-process outsourcing organization which commenced operations in 1968. It is currently Asia's largest IT services firm with revenues of US $2.28 Billion for Financial Year 2004-2005. As of 2005 it has the largest number of employees among all the Indian IT companies (over 50000). It has posted a net profit of Rs. 2,052 crore for the Financial Year 2004-2005. Ever since its inception, TCS has pioneered the offshore delivery model for IT services. Now, with a presence in 34 countries across 5 continents, and a comprehensive range of services across diverse industries, TCS is one of the leading Information Technology companies. Seven of the Fortune Top 10 companies are among its customers.

TCS is part of one of Asia's largest conglomerates - the TATA Group - which, with its interests in Energy, Telecommunications, Financial Services, Chemicals, Engineering & Materials to name a few. TCS was conferred the best IT employer award by Hewitt Associates and best place to work award by Dataquest. CEO Ramadorai got the Businessman of the Year award from Business India in 2004.

1.2 HISTORY

Tata Consultancy Services was started in 1968. Mr. F.C.Kohli who was overseeing the punch corporate operation of Tisco, a TATA company, realised the potential in the need to provide solutions to its clients as well.

TCS was started as a separate entity to address this market need. TCS has not looked back since then. TCS is credited with bringing the software industry into India.

1.3 THE COMPANY

The company was a privately held entity till June 2004, in July 2004 it went public on the Bombay Stock Exchange and National Stock Exchange of India .The IPO of TCS was a landmark event in the history of Indian economy.

Subramaniam Ramadorai is the current CEO of the company and has taken over from Dr. Fakir Chand Kohli (who is considered to be the *Grand-sire* of Indian IT industry). Ratan Tata is the current chairman.

TCS won the Rajiv Gandhi National Quality Award (RGNQA) for 2003. In 2004, it won the JRD QV award for Business Excellence.

In August 2004, TCS became the world's first organization to achieve an integrated Enterprise wide Maturity Level 5 on both the Capability Maturity Model Integration (CMMI) and the People CMM (PCMM).

TCS has a vision to be among the Top 10 companies by the year 2010. The company's corporate headquarters is located at Air-India building at Nariman Point, Mumbai, India.

1.4 BUSINESS AREAS

TCS serves the following industry and business areas:

1. Banking
2. Life Sciences
3. Insurance
4. Media and Entertainment
5. Manufacturing

6. Telecom
7. Retail and Consumer goods
8. Energy and Utility
9. Transportation

TCS has recently bagged an application maintenance project from ABN Amro worth US$250 million, which is the largest deal signed by an Indian IT Company .TCS has also successfully implemented E-governance projects in various states like Andhra Pradesh.

1.5 PROMINENT BUSINESS UNITS

1.5.1 TATA RESEARCH DEVELOPMENT AND DESIGN CENTER (TRDDC)

The TRDDC unit is the center for R&D work, mostly for internal use of TCS. The products developed are focused on the software development life-cycle, and include Modeling and Code Generation tools.

Active work is going on to integrate the products into the popular Eclipse Integration Framework.

In addition, TRDDC is also involved in the areas of language processing, formal methods, and research on Artificial Intelligence and Decision Support.

On account of the ongoing diversification effort, the other centers situated in India have also begun R&D facilities; the center in Hyderabad carries out work on Security related issues.

1.5.2 INDIAN BRANCHES

TCS has branches in the following Indian cities: Mumbai, Chennai, Kolkata, Delhi, Hyderabad, Bangalore, Thiruvananthapuram, Lucknow, Ahmedabad, Pune and Jamshedpur. As of 2005, it plans to open new offices in Kochi, Bhubaneshwar and Mohali.

1.5.3 GLOBAL UNITS

TCS is present all over the world. It has offices in about 35 countries including USA, UK, Brazil, China, Japan, Hungary, Germany, UAE, South Africa, Uruguay, Portugal, Singapore, Australia, France and many more.

1.6 PRODUCTS

Major products include:

- Quartz - a banking solution
- Master craft - A modeling and code engineering tool
- EX - a financial application

CHAPTER 2

INTRODUCTION

2.1 SYNCML

All applications need data. The information must be available even when the device is off-line. Synchronization of Data between multiple devices and corporate servers is a fundamental component in every Synchronization project. Data is supposed to be consistent when accessed by multiple users who are not always connected to the network. Once a Sync application is developed, Data needs to be transferred to a remote device, over-the-air. Then, it must be kept up-to-date.

SyncML (**Sync**hronization **M**arkup **L**anguage) is a platform-independent information synchronization standard. Existing synchronization solutions have mostly been somewhat vendor, application or operating system specific. The purpose of SyncML is to change this by offering an open standard as a replacement. Several major companies such as Nokia, Sony Ericsson and Siemens AG already support SyncML in their products.

2.2 FILE SYNCHRONIZATION

File Synchronization in computing is the process of making sure that two or more locations contain the same up-to-date information. If you add, change, or delete a file from one location, the synchronization process will add, change, or delete the same file from the other location.

File Synchronization can be one-way or two-way. In one-way sync, files are copied only from a primary location (source) to a secondary location (target) in one direction, but no files are ever copied back to the primary location. In two-way sync, files are copied in both directions, keeping the two locations in sync with each other.

File Synchronization enables to seamlessly and securely access and use your latest data anytime, anywhere - at the office, from home or on the go. Using innovative Secure SyncML Technology that automatically synchronizes all your computers (home PC, office PC, and laptop); File Synchronization ensures that wherever you go, your data will follow.

2.3 ADVANTAGES OF FILE SYNCHRONIZATION

2.3.1 Access your data from home

- Keep your files in sync between your PCs
- Work on your latest data seamlessly
- Save the expense and complexity

2.3.2 Access your data any where

- Use any Web browser to access your data
- Maintain high security and privacy

2.3.3 Share your data with others

- Easily define multiple shared folders
- Efficiently and securely exchange any file.

2.4 JAVA PROGRAMMING LANGUAGE

Java is a reflective, object-oriented programming language developed initially by James Gosling and colleagues at Sun Microsystems. Initially called Oak (named after the oak trees outside Gosling's office), it was intended to replace C++, although the feature set better resembles that of Objective-C. Java should not be confused with JavaScript, which shares only the name and a similar C-like syntax. Sun Microsystems currently maintains and updates Java regularly.

Specifications of the Java language, the Java Virtual Machine (JVM) and the Java API are community-maintained through the Sun-managed Java Community Process. Java was developed in 1991 by James Gosling and other Sun engineers, as part of the Green Project.

After first being made public in 1994, it achieved prominence following the announcement at 1995's Sun World that Netscape would be including support for it in their Navigator browser.

2.4.1 LANGUAGE CHARACTERISTICS

1. Object-oriented programming methodology.
2. Platform-Independence.
3. Built-in support for using computer networks.
4. Designed to execute code from remote sources securely.

2.5 OBJECT ORIENTATION

The first characteristic, object orientation ("OO"), refers to a method of programming and language design. Although there are many interpretations of OO,

One Primary distinguishing idea is to design software so that the various types of data it manipulates are combined together with their relevant operations. Thus, data and code are combined into entities called objects. An object can be thought of as a self-contained bundle of behavior (code) and state (data). The principle is to separate the things that change from the things that stay the same; often, a change to some data structure requires a corresponding change to the code that operates on that data, or vice versa. This separation into coherent objects provides a more stable foundation for a software system's design. The intent is to make large software projects easier to manage, thus improving quality and reducing the number of failed projects.

2.6 PLATFORM INDEPENDENCE

The Look and Feel of a Java GUI is independent of the platform it is running on. The second characteristic, platform independence, means that programs written in the Java language must run similarly on diverse hardware. One should be able to write a program once and run it anywhere.

This is achieved by most compilers by compiling the Java language code "halfway" to byte code-simplified machine instructions specific to the Java platform. The code is then run on a virtual machine (VM), a program written in native code on the host hardware that interprets and executes generic Java byte code. Further, standardized libraries are provided to allow access to features of the host machines (such as graphics, threading and networking) in unified ways.

There are also implementations of Java compilers that compile to native object code, such as GCJ, removing the intermediate byte code stage, but the output of these compilers can only be run on a single architecture.

The first implementations of the language used an interpreted virtual machine to achieve portability. These implementations produced programs that ran more slowly than programs written in C or C++, so the language suffered a reputation for poor performance.

More recent JVM implementations produce programs that run significantly faster than before, using multiple techniques.

Portability is a technically difficult goal to achieve, and Java's success at that goal has been mixed. Although it is indeed possible to write programs for the Java platform that behave consistently across many host platforms, the large number of available platforms with small errors or inconsistencies led some to parody Sun's "Write once, run anywhere" slogan as "Write once, debug everywhere".

Platform-independent Java is, however, very successful with server-side applications, such as web services, servlets, or Enterprise Java.

2.7 JAVA: SWING

The Swing package is used to create graphics in Java. With a Swing application it is possible to create buttons, labels, check boxes, radio buttons, and drop down lists easily. In order to click on the close button (the little x in the top right hand corner of the window) and have the program close, the code has to be written to close it; otherwise the window will close but the program will keep running. It is also possible to create shortcuts that use the keyboard instead of always having to click on the button that was created.

The major drawback to Swing is that it is slower than using native widgets (which can be accomplished using SWT).

There are several different layouts that can be used when creating buttons, for example the Grid Layout can be used to specify the number of columns and rows that you want your buttons to be displayed in, or the Flow Layout can be used to put buttons in a straight horizontal line across the page, or the Box Layout can be used to put buttons in a straight vertical line down the page. Often you will put new layouts within the cells of

other layouts. Some IDEs allow you to draw the GUI graphically and will handle the internals of generating the layout code.

2.8 ECLIPSE (SOFTWARE)

Eclipse is a free software / open source platform-independent software framework for delivering what the project calls "rich-client applications", as opposed to "thin client" browser-based applications. So far this framework has typically been used to develop IDEs (Integrated Development Environments), such as the highly-regarded Java IDE called Java Development Toolkit (JDT) and compiler that comes as part of Eclipse (and which is also used to develop Eclipse itself). However, it can be used for other types of client application as well.

Eclipse was originally developed by Object Technology International (later purchased by IBM), but is now supported by the Eclipse Foundation, an independent not-for-profit consortium of software industry vendors. Many notable software tool vendors have embraced Eclipse as a future framework for their IDEs, among them Borland and IBM Rational.

2.9 XML

The **Extensible Markup Language** (**XML**) is a W3C-recommended general-purpose markup language for creating special-purpose markup languages. It is a simplified subset of SGML, capable of describing many different kinds of data.

Its primary purpose is to facilitate the sharing of data across different systems, particularly systems connected via the Internet.

2.9.1 FEATURES OF XML

XML provides a text-based approach to describe and apply a tree-based structure to information. At its base level, all information manifests as text, interspersed with markup that indicates the information's separation into a hierarchy of *character data*, container-like *elements*, and *attributes* of those elements.

The fundamental unit in XML is the *character*, as defined by the Universal Character Set. Characters are combined in certain allowable combinations to form an XML *document*. The document consists of one or more *entities*, each of which is typically some portion of the document's characters, encoded as a series of bits and stored in a text file.

2.10 CORRECTNESS IN AN XML DOCUMENT

For an XML document to be correct, it must be:

- **Well-formed**. A well-formed document conforms to all of XML's syntax rules. For example, if a non-empty element has an opening tag with no closing tag, it is not *well-formed*. A document that is not well-formed is not considered to be XML; a parser is required to refuse to process it.

- **Valid**. A valid document has data that conforms to a particular set of user-defined content rules that describe correct data values and locations. For example, if an element in a document is required to contain text that can be interpreted as being an integer numeric value, and it instead has the text "hello", is empty, or has other elements in its content, then the document is not *valid*.

2.11 DTD

The oldest schema format for XML is the Document Type Definition (DTD), inherited from SGML. While DTD support is ubiquitous due to its inclusion in the XML 1.0 standard, it is seen as limited for the following reasons:

- It has no support for newer features of XML, most importantly namespaces.
- It lacks expressivity. Certain formal aspects of an XML document cannot be captured in a DTD.

It uses custom non-XML syntax, inherited from SGML, to describe the schema

2.12 PROCESSING XML FILES

SAX and DOM are APIs widely used to process XML data. SAX is used for serial processing whereas DOM is used for random-access processing. Another form of XML Processing API is data binding, where XML data is made available as a strongly typed programming language data structure, in contrast to the DOM.

2.12.1 JAVA API FOR XML PROCESSING

The **Java API for XML Processing**, or **JAXP**, is one of the Java XML programming APIs. It provides the capability of validating and parsing XML documents. The two basic parsing interfaces are:

- the Document Object Model parsing interface or **DOM** interface
- the Simple API for XML parsing interface or **SAX** interface

In addition to the parsing interfaces, the API provides an XSLT interface to provide data and structural transformations on an XML document. The J2SE 1.4 JDK is the first version of a JDK comes with an implementation of JAXP 1.1.

The Simple API for XML defines the events and interfaces used to interact with a SAX-compliant XML parser.

The JDOM project was created by Jason Hunter and Brett McLaughlin. One significant point about SAX and JDOM is that both of them came from the XML developer community, not a standards body. Their wide acceptance is a tribute to the active participation of XML developers worldwide.

CHAPTER 3

SYNCML

3.1 INTRODUCTION

Mobile devices such as PDAs, pagers, mobile phones and laptops are- by nature- not always connected to a network. Yet these devices contain applications which require information obtained from a network in order to be useful. While most PDAs and mobile phones contain applications such as calendars, tasks lists, and address books for storing useful information, this information is far less useful when it is static, only available on the device itself. For example, copies of static information will always be dissimilar when changes are made on one copy or the other. Synchronization offers a device the ability to connect to a network in order to update either the information on the device or the information on the network, such that both sets of information are identical and up-to-date.

Given the proliferation of proprietary mobile devices and protocols, as well as the increasing consumer demand for ubiquitous mobile access of information, leading technology companies saw the need to create a standard, universal language for describing the synchronization actions between devices and applications. They formed a consortium to sponsor the SyncML initiative to create this language.

Currently, the SyncML consortium has been adopted and incorporated into the Open Mobile Alliance, a larger group of over 300 companies which sponsors many collaborative technology projects and protocols.

Software applications must occasionally incorporate application-specific data synchronization in order to mirror changes over time among multiple data sources at a level more granular than File synchronization. An example use of this is the Data Synchronization specification of the Open Mobile Alliance, which continues the work previously done by the SyncML initiative. SyncML was initially proposed to synchronize changes in personal address book and calendar information from computers to mobile phones, but has subsequently been used in applications that synchronize other types of data changes among multiple sources, such as project status changes.

3.2 WHAT IS SYNCML?

3.2.1 SYNCML

SyncML (Synchronization Markup Language) is the former name (currently referred to as: *Open Mobile Alliance Data Synchronization and Device Management*) for a platform-independent information synchronization standard. Existing synchronization solutions have mostly been somewhat vendor-, application- or operating system specific. The purpose of SyncML is to change this by offering an open standard as a replacement. Several major companies such as Nokia, Sony Ericsson and Siemens AG already support SyncML in their products.

SyncML is most commonly thought of as a method to synchronize contact and calendar information between some type of handheld device and a computer (personal, or network-based service), such as between a Nokia phone and a personal computer.

However, some products are now using SyncML for more general information synchronization purposes. Chirp is a groupware product using SyncML to synchronize project task information across a distributed group of team members, all synchronizing through a central network-based server.

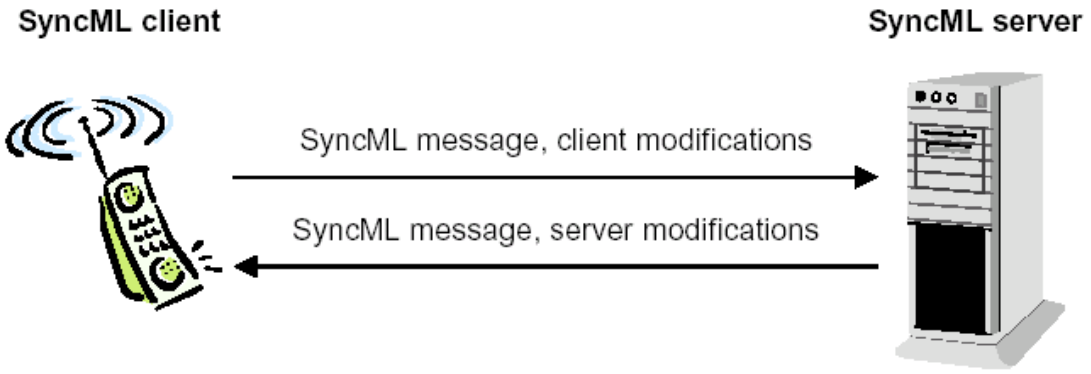


Figure 3.2.1 SyncML message exchanges

The current version of the protocol is 1.1.2. Version 1.2 is a candidate enabler and is available on OMA Release Program and Specifications page.

The SyncML protocol is designed with these goals in mind:

- As a common language, any device should be able to synchronize with any SyncML service (a networked data repository).
- Any service speaking SyncML should be able to synchronize with any SyncML-capable device.
- The protocol must address the limitations of mobile devices, specifically with respect to memory storage.
- It must support a variety of transport protocols such as HTTP, SMTP, Bluetooth and others.
- It must deliver common synchronization commands to all devices.

- It builds upon existing web technologies, specifically XML.
- Support asynchronous communication and error-handling, since the internet has latency.

3.3 SYNCML FUNDAMENTALS

3.3.1 VOCABULARY

Let's begin by defining a vocabulary:

- Client - the mobile device, its application and local database.
- Server - a remote system communicating to the system database or application.
- Modifications - data in fields in a database are changed.
- Sync - The client and server exchange SyncML messages with commands.
- Package - SyncML DTD conformant XML markup describing requests or actions to be taken by either a SyncML client or server. A package is a collection of actions to be performed
- Message - the smallest unit of SyncML markup. Large packages are broken into separate messages.
- Mapping - using an intermediate identifier to tie two pieces of information together.

3.3.2 MESSAGES AND PACKAGES

SyncML messages are requests from either a client or server to perform some action. The action may be to synchronize data, perform some checks on data, update a status, or handle any errors with these actions. Messages are bundled together as packages, as kind of a to-do list. Messages are a laundry list of requests, and they can be pieced together out of order if sufficient mapping information is given to identify to which package the message belongs.

SyncML is designed this way to accommodate for errors and dropped messages. Should one message be dropped, a SyncML client or server will know there is a problem

Because the mapping cannot be completed. It will then issue a request for the information to be resent. Once the data is received, the updates to the information can proceed.

3.3.3 STRUCTURE OF A SYNCML MESSAGE

There are two parts to the SyncML message, a Sync Header <SyncHdr> and Sync Body <SyncBody>. The header contains meta-information about the request, such as the target database <Target> and source database <Source> URIs, Authentication information <Cred>, the session ID <SessionID>, the message ID <MsgID>, and SyncML version declaration <VerDTD>. The body contains the actual requests, alerts and data.

3.3.4 ADDRESSING

Addressing is done through the <source> and <DocURI> tags. A server will have a familiar URI like http://www.chris.syncml.org/sync and a client mobile device will have an IMEA identification number like this 30400495959596904.

3.3.5 MAPPING

SyncML is based on the idea that clients and servers can have their own way of mapping information in their databases. Therefore, clients and servers must each have their own set of unique identifiers.

- Locally Unique Identifiers (LUID) are numbers assigned by the client to a data object in a local database (like a field or a row). They are non-reusable numbers assigned to these objects by the SyncML client.

- Globally Unique Identifiers (GUID) are numbers assigned to a data object for use in a remote database. This identifier is assigned by the server.

LUID and GUID numbers only have to be unique if they are being used in a table between two communicating parties. In other words, these numbers are temporary, used for mapping data to tables and only really exist for the complete duration of transactions between client and server.

The server will create a mapping table to tie the LUID and GUID together.

Client-side data

LUID	Data
5	Green

Server-side data

GUID	Data
5050505	Green

Server Mapping

GUID	LUID
5050505	5

3.3.6 CHANGE LOGS

The Server and Client track of changes made to their databases during synchronization through "change logs". SyncML doesn't define the change logs, instead SyncML does require that the changes and corrections be negotiated between client and -

- Server through messages. Using change logs, the Client and Server know which fields need to be updated. The implementation of change tracking in the application which will use SyncML is not defined.

3.3.7 SYNC ANCHORS

During Synchronization, the Client and Server need to know which fields to update. If a client/server application is checking the fields prior to updating/modifying them, how then does the client/server keep track of the position of current field in the database? The answer is "by using Sync Anchors".

There are two kinds of Anchors: Last and Next. The 'Last' anchor describes which updates occurred during the last synchronization event. The 'Next' anchor describes the current and future synchronization request. These anchors describe the events from the standpoint of the sending device.

Anchors are sent back and forth from client and server to keep track of what is happening to the database fields and what's going on in overall through the lifetime of the sync operation.

By coordinating Sync Anchors and change logs with the type of Sync that is requested, the server application can determine and track (with change logs) which information is the most up-to-date. For example, it is possible to overwrite 'newer' information- that is information for which there is the most recent time-stamp in the change log- with older information. This could be done by choosing a sync in which the

client tells the server to overwrite its information with client data. This is called ‘refresh sync'. The types of syncs are described below.

3.4 SYNC TYPES

There are seven types of Syncs in the SyncML 1.1 language. The following section describes the types of syncs:

1. **Two-way Sync** - The client and server exchange information about modified data. The client sends the modifications first.
2. **Slow sync** - a two-way sync in which all fields in the database are checked on a field-to-field basis. This type of sync is utilized when there is high latency between the client and server.
3. **One-way sync, client only** - the client sends the modified data first. The server accepts and updates the data and does not send its modifications.
4. **Refresh sync from client** - the client sends the entire database to the server. The server does not sync. Rather, the server replaces the target database with the client's database.
5. **One-way sync, server only** - the server sends the modified data first. The client accepts and updates the data and does not send its modifications.
6. **Refresh sync from server** - the server sends all its information from a database to the client, replacing the client's database.
7. **Server alerted sync** - the server remotely commands the client to initiate one of the above sync types with the server. In this way, the server is remotely-controlling the client.

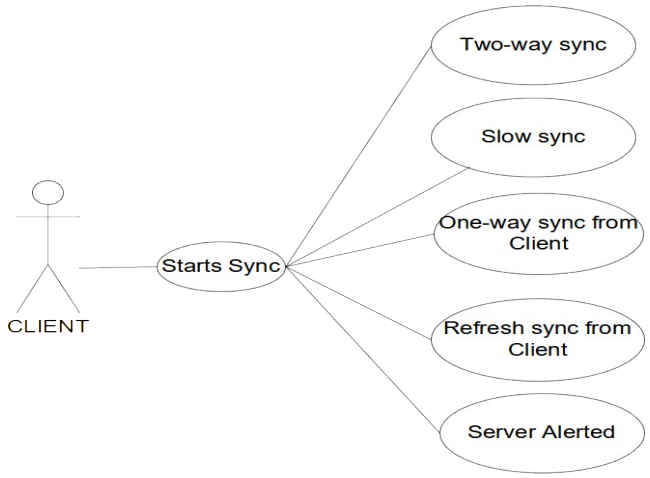


Figure 3.4.1 Sync Types

3.5 SYNC INITIATION

Sync Initiation is the process the client and server must go through prior to an actual Synchronization. The first step is for the client and server to speak the same language, exchanging and revealing each other's capabilities (as defined by device, as in amount of memory, and protocol as defined by DTD). The second step is identification of the databases to be synchronized. Next the two must decide on the type of synchronization. The third and final step is authentication. Once this step is completed successfully, the synchronization activities can begin.

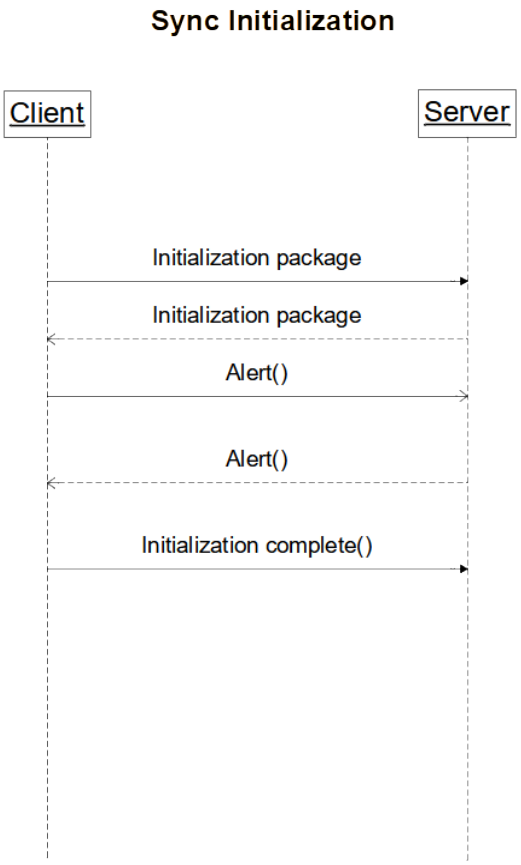


Figure 3.5.1 Sequence Diagram –Sync Initialization

3.6 AUTHENTICATION

The SyncML server can send the client a message containing the <Chal> tag in order to represent an authentication challenge to the information the client is attempting to access. The client must then respond, giving the username and password within the <Cred> tag.

SyncML uses MD5 digest access authentication. The Client and Server exchange credentials during the authentication process, returning error codes if the process breaks

down at some point. The <Cred> tag is used in the <SyncHdr> for holding the credentials to be used for authentication.

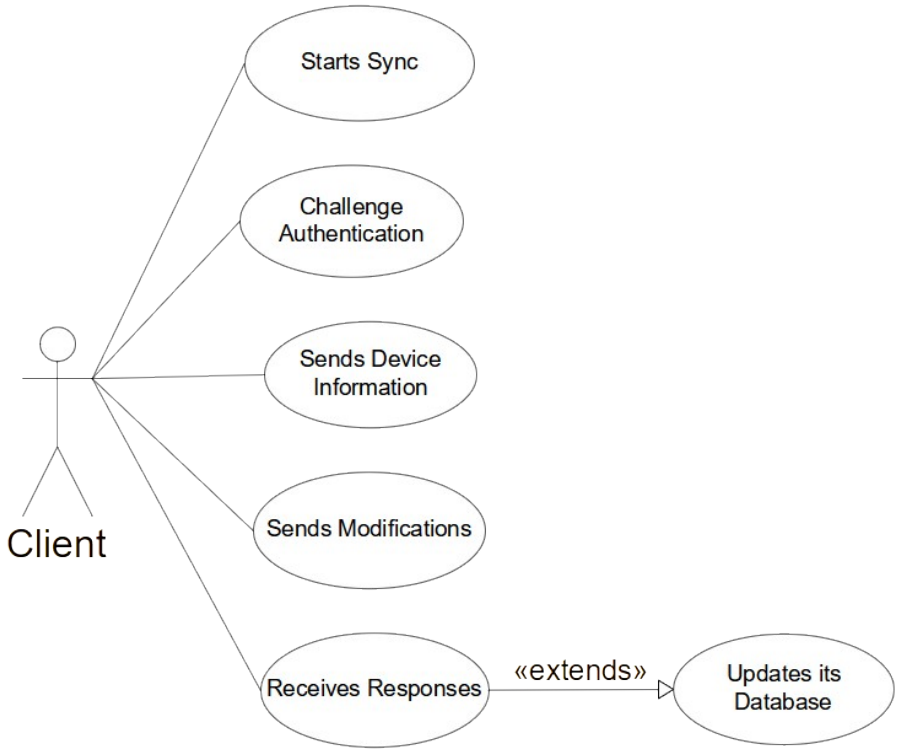


Figure 3.6.1 Sync Client Use Case

3.7 COMMON SYNCML IMPLEMENTATIONS

Nokia was the first company to make a SyncML-enabled phone. It synchronized the calendar database on the phone. SyncML can synchronize to-do lists, calendars, address books, phone-books, and pretty much anything an organizer can do. SyncML is capable of much more. It would be appropriate to use SyncML any time there are two disparate, remote applications which need to share the same data.

3.8 SYNCML SYNTAX

The SyncML programming framework is based on two protocols: SyncML Representation protocol and SyncML Synchronization protocol. The SyncML Representation protocol defines the representation format of SyncML messages (in XML) and details the inner workings of the SyncML framework. The SyncML Synchronization protocol defines the actions between a SyncML client and a SyncML server.

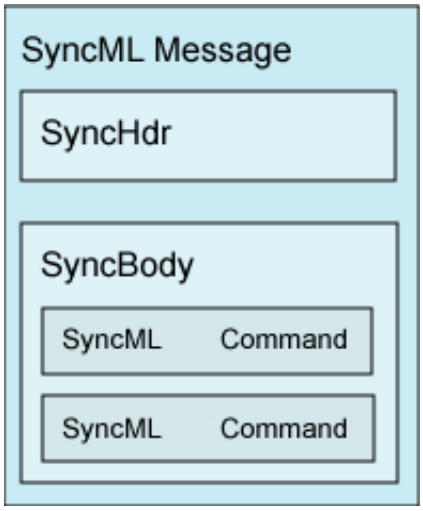


Figure 3.8.1 A Conceptual SyncML message

Representation Protocol: The Representation Protocol defines two DTDs:

- SyncML DTD, which specifies the basic SyncML message format. SyncML specifies various commands; the elements of these commands are specified in this DTD.
- Meta-Information DTD, which contains the data, formats. Different SyncML commands carry data in different formats. Information about this data sits in a meta-element formatted according to this DTD.

Device Management Protocol: SyncML Device Management Protocol specifies the handshake mechanism and the rules to follow for a successful management session, including the:

- State machine for the client and server
- Commands that the server and the client can send
- Method by which the client and server authenticate each other
- Process for checking the integrity of the message exchanged
- Mechanisms for sending objects greater than the transport message size

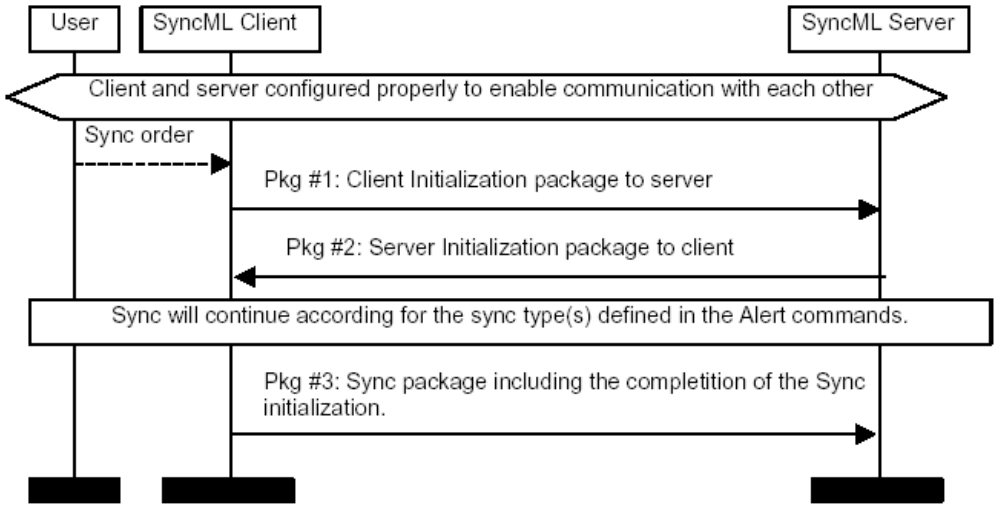


Figure 3.8.2 A SyncML message sequence

3.9 SyncML Example

```
1  <SyncML>
2  <SyncHdr>
3  <VerDTD>1.1</VerDTD>
4  <VerProto>SyncML/1.1</VerProto>
5  <SessionID>104050403</SessionID>
6  <MsgID>5</MsgID>
7  <Cred>...</Cred>
8  </SyncHdr>
9  <SyncBody>
```

```
10  <Status>...</Status>
11  <Sync>
12  <Target>target database URI</Target>
13  <Source>source database URI</Source>
14  <Add>datafield and data</Add>
15  <Replace>an existing data field with some data</Replace>
16  </Sync>
17  </SyncBody>
18  </SyncML>
```

Notice lines {1} and {18} start the SyncML file with the root tags. Next, the SyncHdr is defined by lines {2} and {8}. Further, lines {3,4} define the versioning information, line {5} defines the sessionID to distinguish which unique dialogue is occurring between client and server applications, line {6} shows the MsgID to uniquely identify this set of requests (this entire markup) to be performed by the requested application. Also in the sync Header are credentials, on line {7}.

The Sync Body begins on line {9}. In this part of the SyncML message, device/application status {10}, target/source URIs {12, 13}, and requested actions such as the sync itself between lines {11, 16}, Add and Replace {14, 15} commands are given.

3.10 WBXML AND SYNCML

WAP Binary XML (WBXML) is a form of XML whereby the XML tags are abbreviated in order to shorten the markup for transmission to mobile devices, which commonly have bandwidth and memory limitations. The XML tags are encoded into binary shorthand to save space. Let's take a look at an example so that this will make more sense.

The following is WBXML binary code depicting a SyncML message. Notice in the first line there is a document type definition, represented here in hexadecimal tokens. Can you see what happens to the following string? "//SYNCML//DTD SYNCML 1.1//EN"

Wbxml abbreviations

6D	= "<SyncML>"
6C	= "<SyncHdr>"
71	= "<VerDTD>"

Wbxml abbreviations

C3	= represents the beginning of opaque (xml) data
03	= this represents the length of this opaque data
"1" "." "1"	= The characters "1" followed by "." and "1"
01	= represents "</VerDTD>"

tells the SyncML processor that this is the beginning of opaque (xml) data this represents

The length of this opaque data .the characters "1" followed by "." and "1" represents "</VerDTD>"

All together this WBXML code snippet, *6D6C71C303"1.1"01* represents:

SyncML header snippet

```
1    <SyncML>
2    <SyncHdr>
3    <VerDTD>1.1</VerDTD>
```

So we can see how using WBXML shorthand would be a more compact means of representing XML, saving bandwidth for mobile devices.

CHAPTER 4

HETEROGENOUS DATABASE REPLICATION

4.1 JAVA DATABASE CONNECTIVITY

Java **D**ata**b**ase **C**onnectivity, or **JDBC**, is an API for the Java programming language that defines how a client may access a database. (To be strictly correct, JDBC is not an acronym.) It provides methods for querying and updating data in a database. JDBC is oriented towards relational databases.

The Java 2 Platform, Standard Edition includes the JDBC API together with an ODBC implementation of the API enabling connections to any relational database that supports ODBC. This driver is native code and not Java, and is closed source.

4.2 OVERVIEW

JDBC allows multiple implementations to exist and be used by the same application. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections.

JDBC connections support creating and executing statements. These statements may be update statements such as SQL INSERT, UPDATE and DELETE or they may be query statements using the SELECT statement. Additionally, stored procedures may be invoked through a statement.

Statements are one of the following types:

- Statement - the statement is sent to the database server each and every time.
- Prepared Statement - the statement is compiled on the database server allowing it to be executed multiple times in an efficient manner.
- Callable Statement - used for executing stored procedures on the database.

Update statements such as INSERT, UPDATE and DELETE return an update count that indicates how many rows were affected in the database. These statements do not return any other information.

Query statements return a JDBC row result set. The row result set is used to walk over the result set. Individual columns in a row are retrieved either by name or by column number. There may be any number of rows in the result set. The row result set has metadata that describes the names of the columns and their types.

There is an extension to the basic JDBC API that allows for scrollable result sets and cursor support among other things.

4.3 DRIVERS

4.3.1 TYPES

There are commercial and free drivers available for most relational database servers. These drivers fall into one of the following types:

- Type 1, the JDBC-ODBC bridge
- Type 2, the Native-API driver
- Type 3, the network-protocol driver
- Type 4, the native-protocol drivers
- Internal JDBC driver, driver embedded with JRE in Java-enabled SQL databases. Used for Java stored procedures.

4.3.2 SOURCES

- Data Direct Technologies provides the fastest and most comprehensive suite of Type 4 JDBC drivers for all major databases.
- Sun provides an incomplete list of JDBC drivers and vendors

- SQLSummit.com publishes list of drivers, including JDBC drivers and vendors
- Open Link Software ships JDBC Drivers for a number of target databases, including Bridges to other data access mechanisms (e.g., ODBC, JDBC) which can provide more functionality than the targeted mechanism.

4.3.3 JDBC TYPE 1 DRIVER

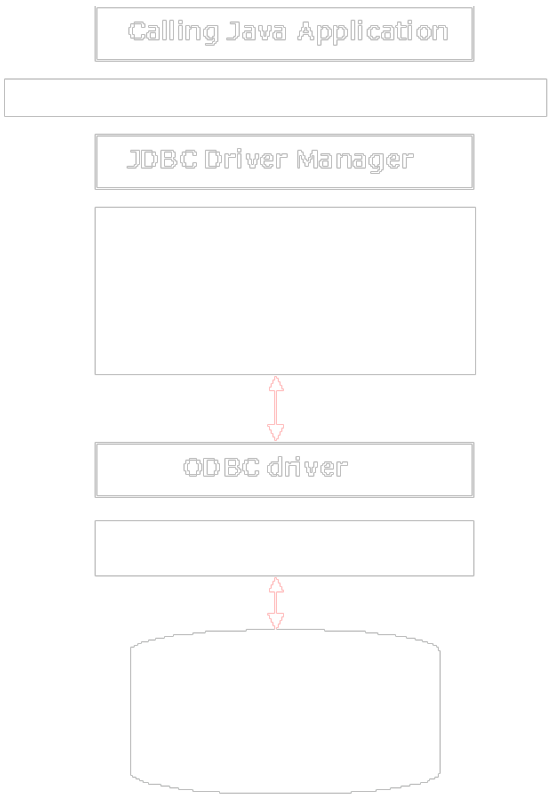


Figure 4.3.3 Schematic of the JDBC-ODBC Bridge

The **JDBC type 1 driver**, also known as the **JDBC-ODBC Bridge** is a database driver implementation that employs the ODBC driver to connect to the database. The driver converts JDBC method calls into ODBC function calls. The bridge is usually used when there is no pure-Java driver available for a particular database.

The driver is implemented in the sun.jdbc.odbc.JdbcOdbcDriver class and comes with the Java 2 SDK, Standard Edition.

The driver is platform-dependent as it makes use of ODBC which in turn depends on native libraries of the operating system. Also, using this driver has got other dependencies such as ODBC must be installed on the computer having the driver and the database which is being connected to must support an ODBC driver. Hence the use of this driver is discouraged if the alternative of a pure-Java driver is available.

Type 1 is the simplest of all but platform specific i.e. only to Microsoft platform.

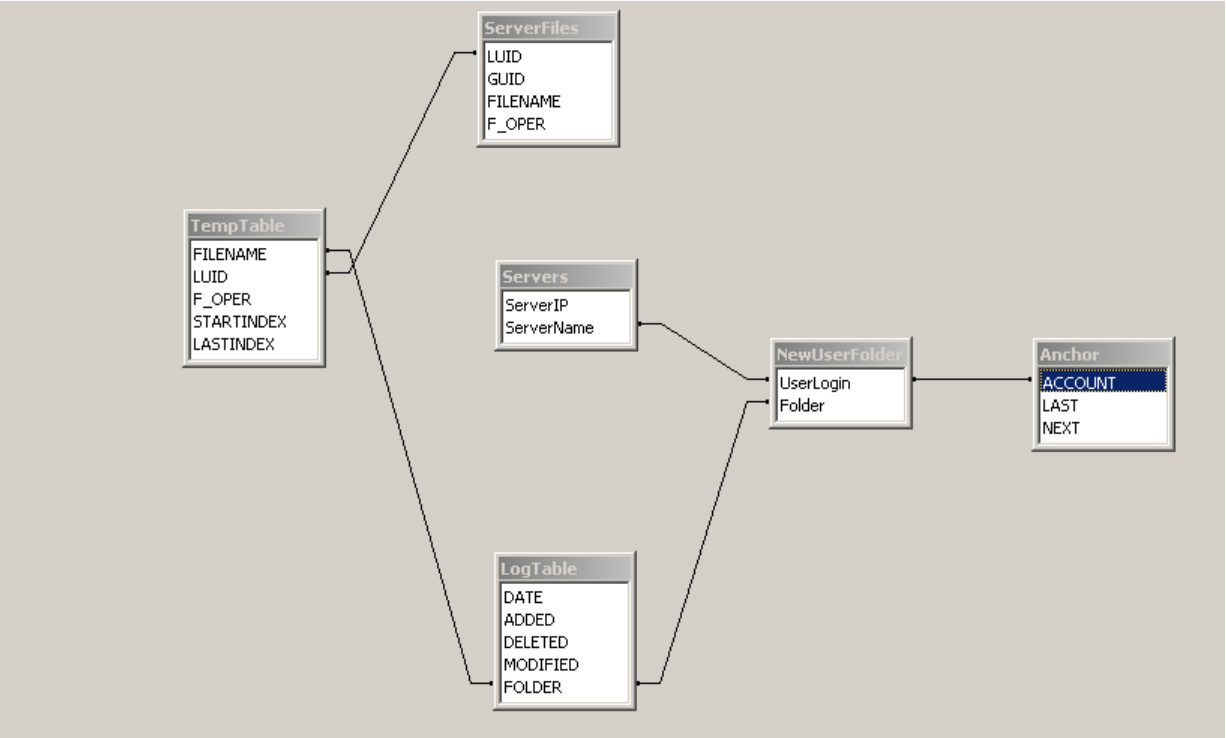


Figure 4.3.4 Sync Client Database Schema

4.4 HETEROGENEOUS DATABASE REPLICATION WITH SYNCML

Availability and performance are primary considerations when we develop distributed applications. But using data stores to address these concerns can result in problems with data synchronization between heterogeneous data stores.

When we design distributed applications, we must consider availability and performance. A common solution is to include a data store on the client system. Typically, the client will require a lightweight data store as a result of limited resources. This approach poses a challenge for data synchronization between heterogeneous data stores. One resolution to this problem is a Java-based approach using JDBC and SyncML standards for heterogeneous database replication.

4.5 REPLICATION OVERVIEW

Replication is the process of duplicating either all or a portion of a database between two environments. To maintain consistency, changes made to the source database are propagated to the replicated database.

Replication may be *one-way* or *two-way*. Two-way replication can be much more difficult because changes made to each database may result in conflicting data. When these changes are propagated between the two databases, a strategy is required to reconcile the differences in order to maintain consistency.

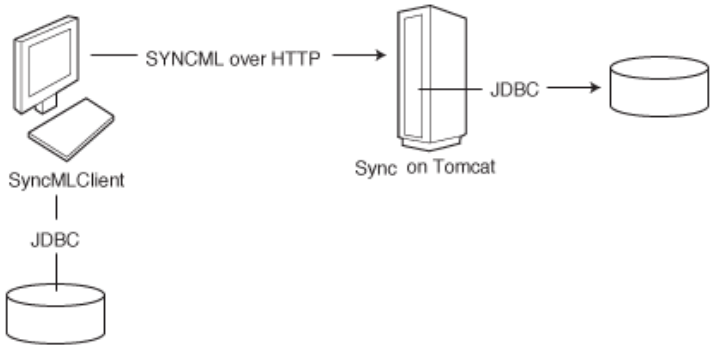


Figure 4.5.1 Heterogeneous Database Replication

4.6 ID HANDLING

As a basic requirement of replication, we need to uniquely identify individual data units to be replicated. For two-way replication, we also need a mapping scheme to identify corresponding units of data between the databases.

Various schemes can be followed based on the replication requirements. For one-way replication, the master database can dictate the ID generation for data units. For two-

Way replication, a mapping scheme must be defined based on the ID generation schemes of the application. The ID generation and mapping schemes can be as simple as mutually exclusive ranges of numbers for IDs used by each database, where IDs are matched exactly. A more complex example may involve an ID generation service or database-specific schemes, where a mapping of IDs must be maintained between databases.

4.7 CHANGE DETECTION

The next step in the replication process is to identify which data units have been changed. In the case of relational databases, you can use additional fields to record the data unit's state and a timestamp of the state change. Alternatively, you can use triggers to automate parts of the change detection process. Insert/update/delete triggers attached to a table can detect changes made to data units and record the changes in a change log table. The change log table can then be used to determine the changes made to data units at any given time.

4.8 REPLICATION

The goal of replication is to produce multiple consistent copies of a common data set. To achieve this goal, the changes detected in each database must be exchanged and reconciled.

Here you are faced with a design challenge. You can choose:

- A data format to represent the changes to be exchanged
- A transport mechanism, such as HTTP, FTP, JMS, and so on, based on the application's requirements

- An exchange protocol between the participating entities
- A conflict detection and resolution mechanism between the data units

SyncML defines a synchronization protocol that uses XML messages for add, remove, and modification of the data units. It also allows for the exchange of security information so that the nodes can perform authentication

CHAPTER 5

SYNCHRONIZATION SOFTWARE – SYNCIT

5.1 INTRODUCTION

SyncIT is the File Synchronization software based on SyncML. Synchronization software is not a new concept; many companies provide software that reconciles changes and resolves changes in data between two distinct sources. However, today's synchronization market consists of multiple proprietary solutions that fail to meet the full spectrum of functionality and device support required by most organizations. Most organizations are not willing to develop multiple conduits and implement several proprietary synching solutions to allow multi device access to a single database for obvious reasons, such as complexity, cost, and maintenance.

Without the ability to access and manipulate accurate data at any time and from any place, mobile devices lose their value. The challenge is to develop mobile applications that are accessible on multiple handheld devices in areas of intermittent wireless coverage, and still allow users to access data with a "change once - update everywhere" paradigm.

5.2 ARCHITECTURE

In *Figure 5.2.1*, Application A represents a networked service that provides synchronization with other client applications (Application B). The server and client are connected over any network transport (HTTP, WSP). The client uses the Sync Client Agent to access the network and send messages to the server via the SyncML Adapter and SyncML Interface (SyncML I/F). The server or Application A, through the Sync Server Agent, receives or sends messages, and manages the entire sync process through the Sync Engine. A SyncML I/F is merely an API to the SyncML Adapter.

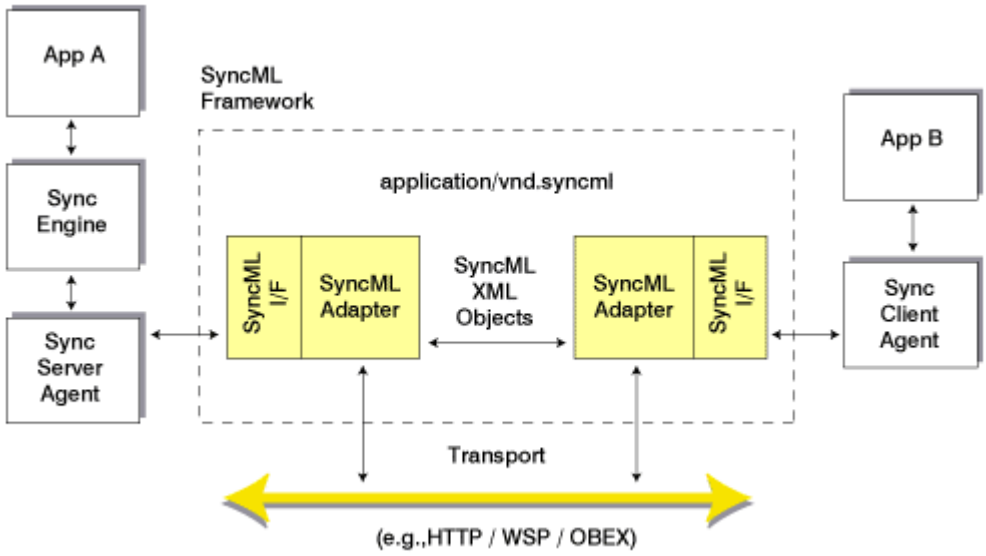


Figure 5.2.1 SyncML Architecture

For example, a laptop acts as the SyncML client, and a server acts as the SyncML server. The SyncML client sends a message to the SyncML server regarding changes to data made on the client. The server then synchronizes the data within the SyncML messages with data stored on the server, and returns modifications back to the SyncML client. The SyncML client contains a sync client agent, and typically has the role of sending modifications first to the server. The client must also be capable of receiving messages back from the server. The SyncML server contains the sync server agent and the sync engine, and usually waits for the client to initiate synchronization, although the server can initiate synchronizations if unsolicited commands are supported on the transport protocol level.

5.3 HARDWARE REQURIMENTS

Processor	Pentium III 2 GHz
Hard Drive	20 GB
RAM	128 MB

5.4 SCENARIO FOR SYNCHRONIZATION

Imagine, for a minute, the business user who owns a mobile phone, a PDA, a laptop computer, and a desktop PC. This would be the best case scenario: Using a single synching solution and standardized format, an update to a contact's phone number stored on the mobile phone would be transmitted to other devices and databases for updating, which could include a contact list on the PDA, a contact list within Outlook on the laptop and desktop PC, a corporate CRM database, and potentially an Internet-based calendar and contact list that the user might occasionally access. The need for keeping this data in sync with the other data sources is an absolute requirement in order for the devices to be of their greatest value, and for the user to work efficiently.

5.5 SYNC SERVER

An application program that accepts connections in order to service requests by Sending back responses. Any given program may be capable of being both an HTTP Client and a server; our use refers only to the role being performed by the Program for a particular connection, rather than to the program's capabilities in general.
Likewise, any server may act as an HTTP origin server, proxy, gateway, or tunnel, Switching behavior based on the nature of each request.

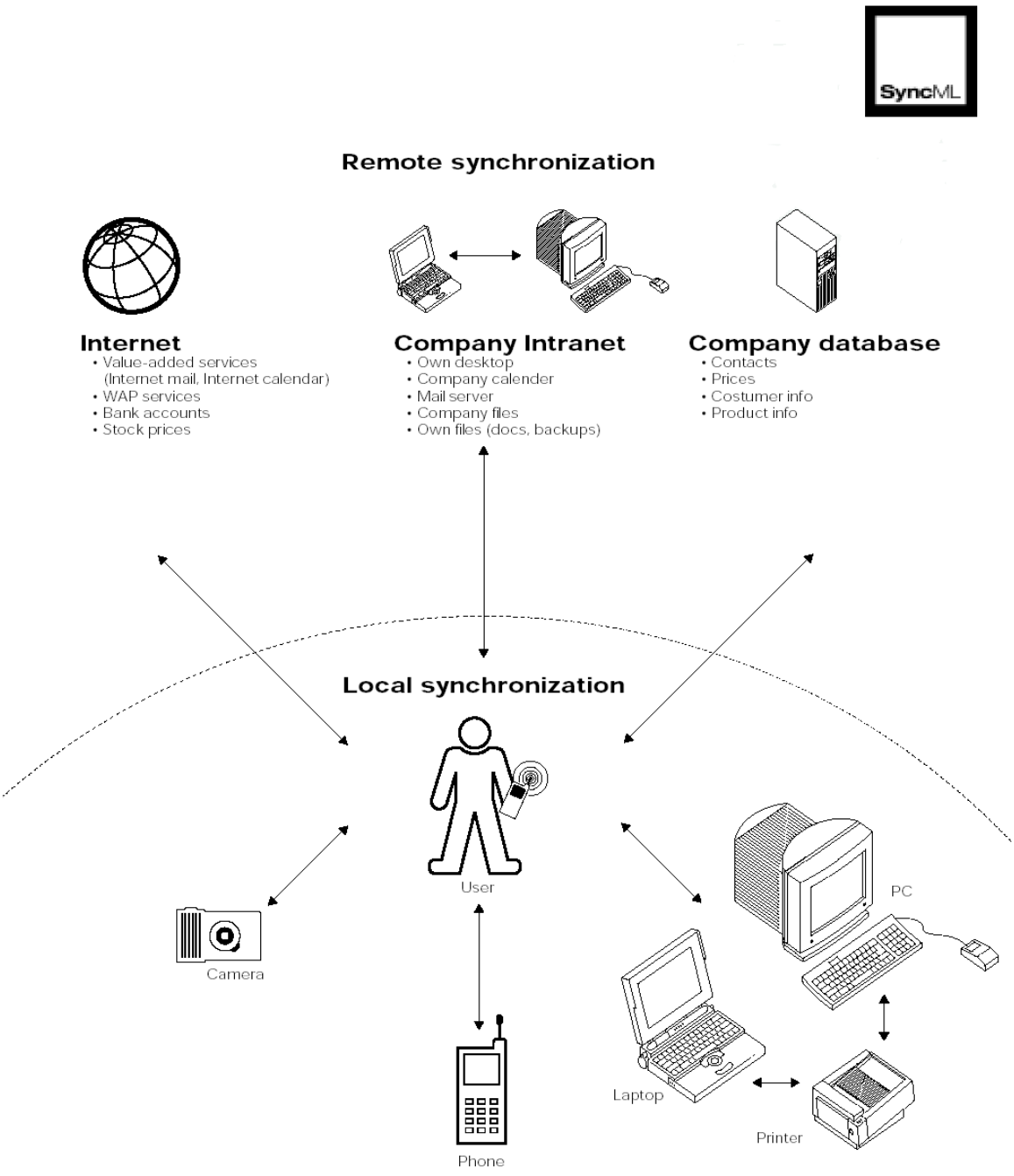


Figure 5.4.1 Scenarios for Synchronization

5.6 ONE WAY SYNC

This is a synchronization type in which the client sends its modifications to the server but the server does not send its modifications back to the client.

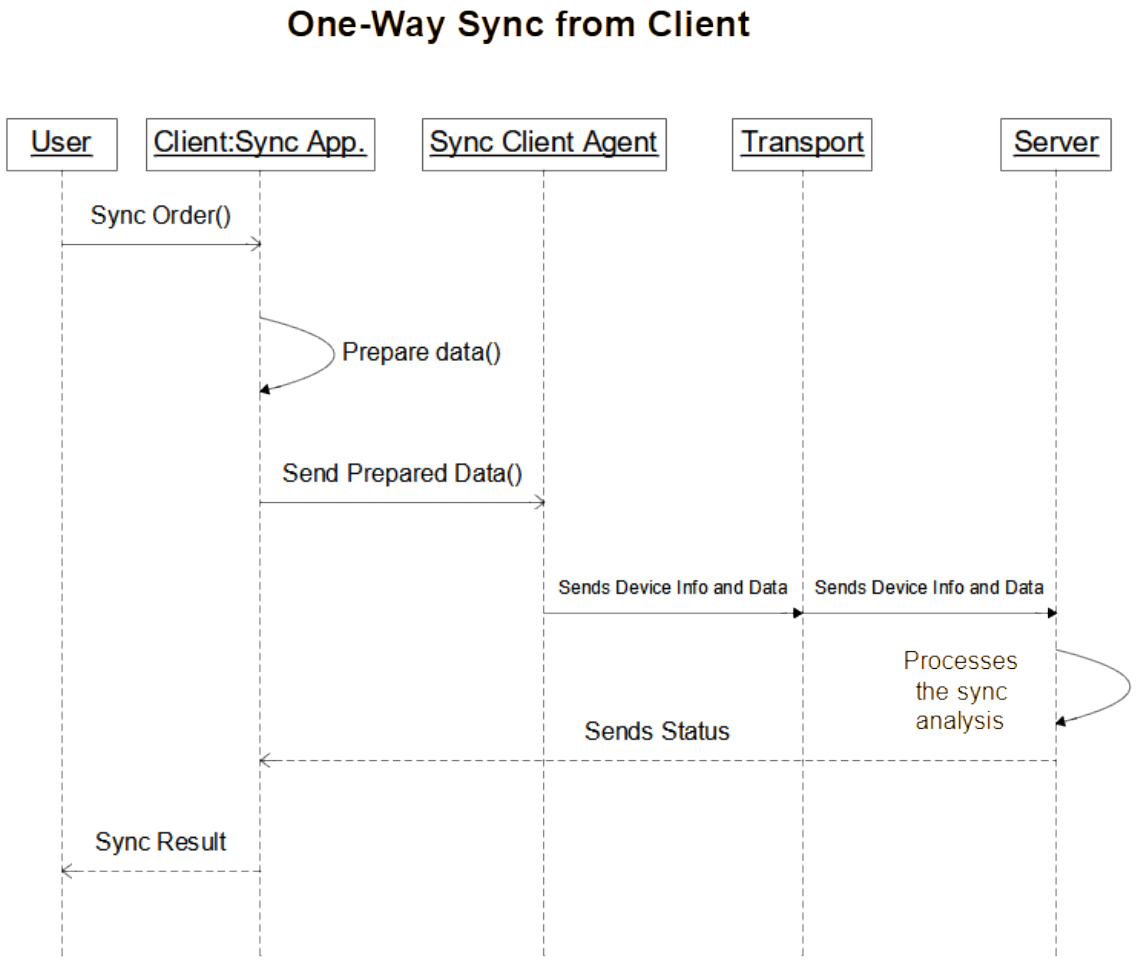


Figure 5.6.1 One Way Sync

5.7 TWO WAY SYNC

Two-way sync is a normal synchronization type in which client and server exchange information about any modifications to the data each contains. The client always initiates this exchange by sending a request to the server. The server processes the synchronization request and the data from the client is compared and unified with the data on the server. After that, the server sends its modified data to the client device, which is then able to update its database with the data from the server. Once it has updated its database, the client sends back the server all the necessary mapping information.

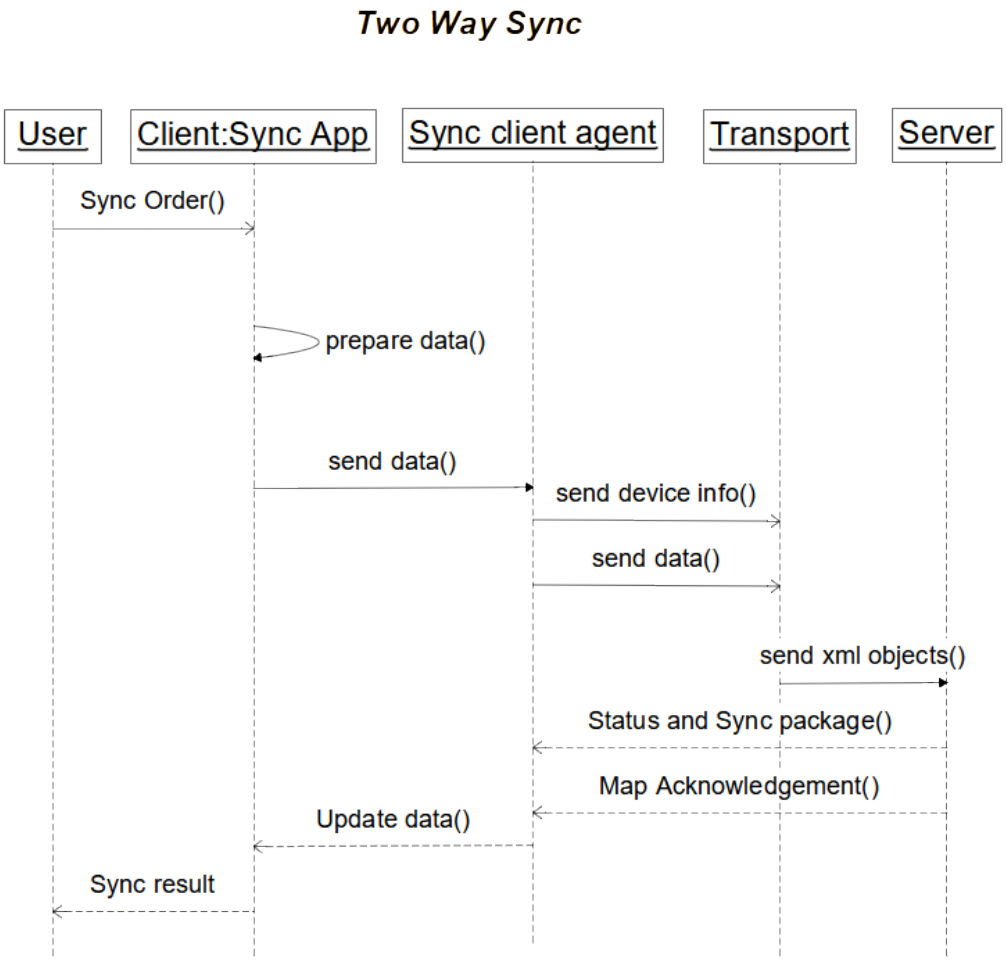


Figure 5.7.1 Two Way Sync

5.8 SLOW SYNC

The slow sync is a form of the two-way synchronization in which all the items in the client databases are compared with all the items in the server databases on a field-by-field basis. A slow sync can be requested if the client and server sync anchors are mismatched or if the client or server loses its change log information. In practice, the slow sync means that the client sends all its data to the server and the server does a field-by-field analysis, comparing its own data with that sent by the client. After the sync analysis, the server returns all the modification information to the client. In turn, the client returns the mapping information for all data items added by the server.

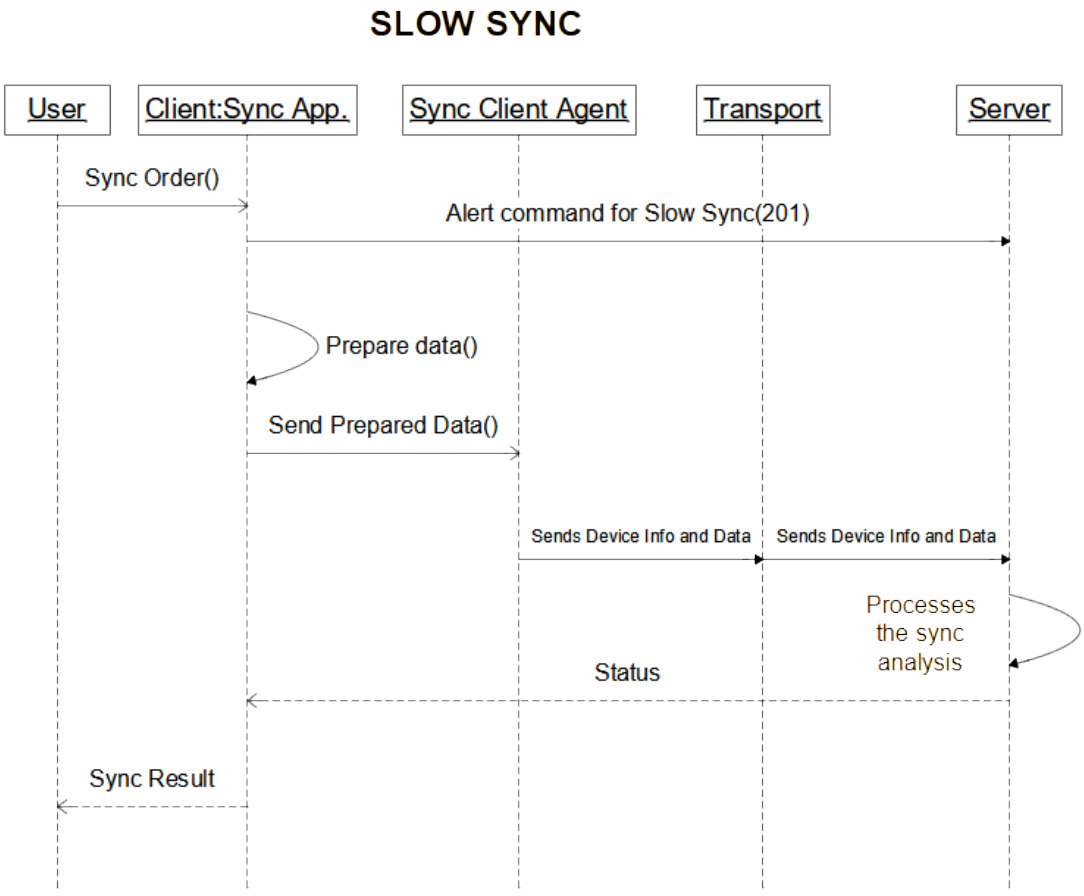


Figure 5.8.1 Slow Sync

5.9 REFRESH SYNC FROM CLIENT ONLY

Here the client exports all its data from a database to the server. The server is expected to replace all data in the target database with the data sent by the client.

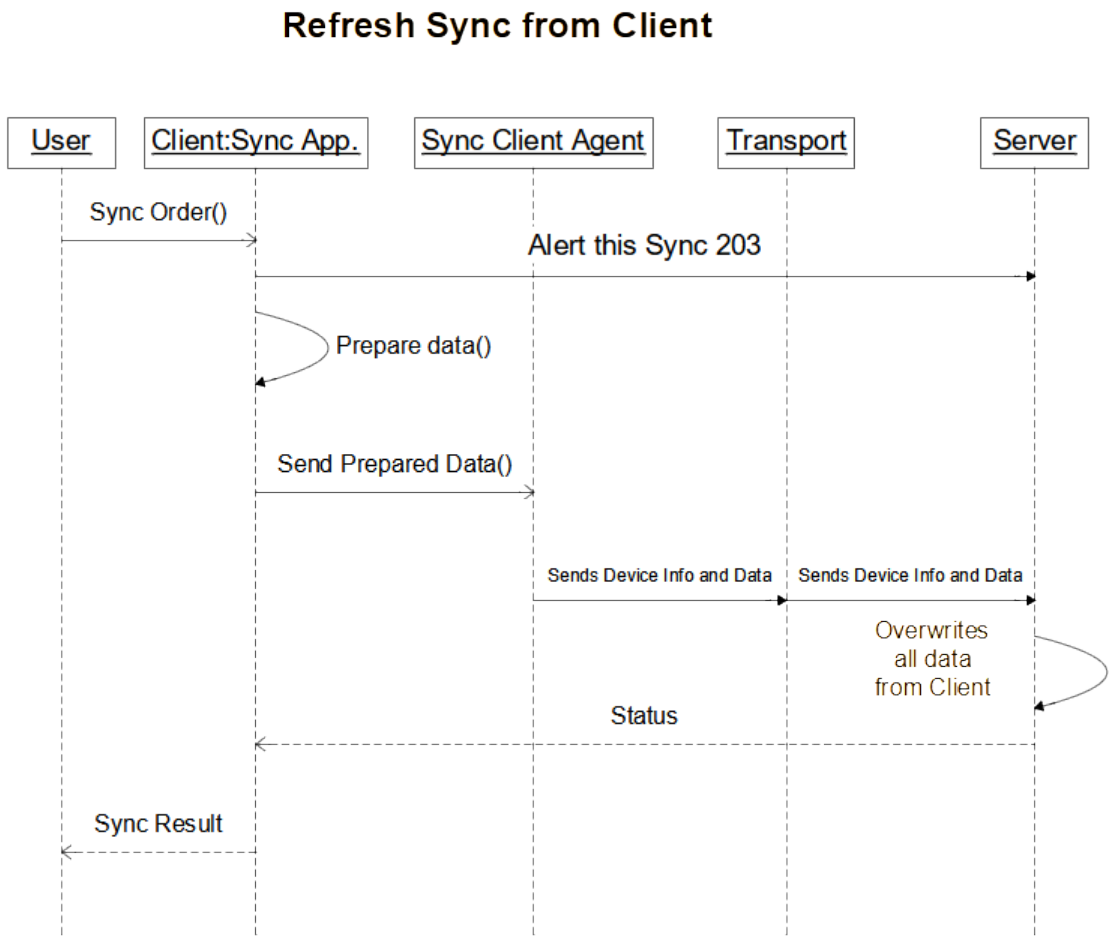


Figure 5.9.1 Refresh Sync from Client

5.10 CONCLUSIONS

SyncML is certainly the protocol of the future, as interoperability between devices, transport protocols, and network databases becomes a bigger priority for businesses and consumers. However, SyncML does face some challenges along the way. First and foremost is that SyncML is a standard, not an application or software. It allows device and server manufacturers to move to one clearly defined standard in which their products can be interoperable; unfortunately, it does not guarantee compliance. This leads to the next challenge -- partial implementation of the standard by vendors.

In a perfect world, our syncing devices will be entirely plug-and-play with any data store on any network, enabling a device to sync with any database, in any location, at any time. Reality says that this level of cooperation among vendors will be challenging, to say the least, exhibited by the fact that many vendors have claimed full support of SyncML, but have yet to implement the standard into their products while continuing with their proprietary syncing efforts. A third challenge is the fact that the standard is in its infancy, and with that comes certain assumptions within the industry. Similar to the sentiment surrounding young standards such as WAP, certain vendors or businesses may be loathe to implement a "young" standard for fear that the standard is still evolving at such a rate that their implementation will be at risk in a year's time.

Despite the challenges, the upside offered by one synchronization protocol is too great for vendors to resist. Ultimately, **SyncIT** creates value for the user by providing ease-of-use and greater access to larger sets of data, thereby generating user satisfaction and creating demand for additional services, devices, and applications

APPENDIX 1

SCREEN SHOTS

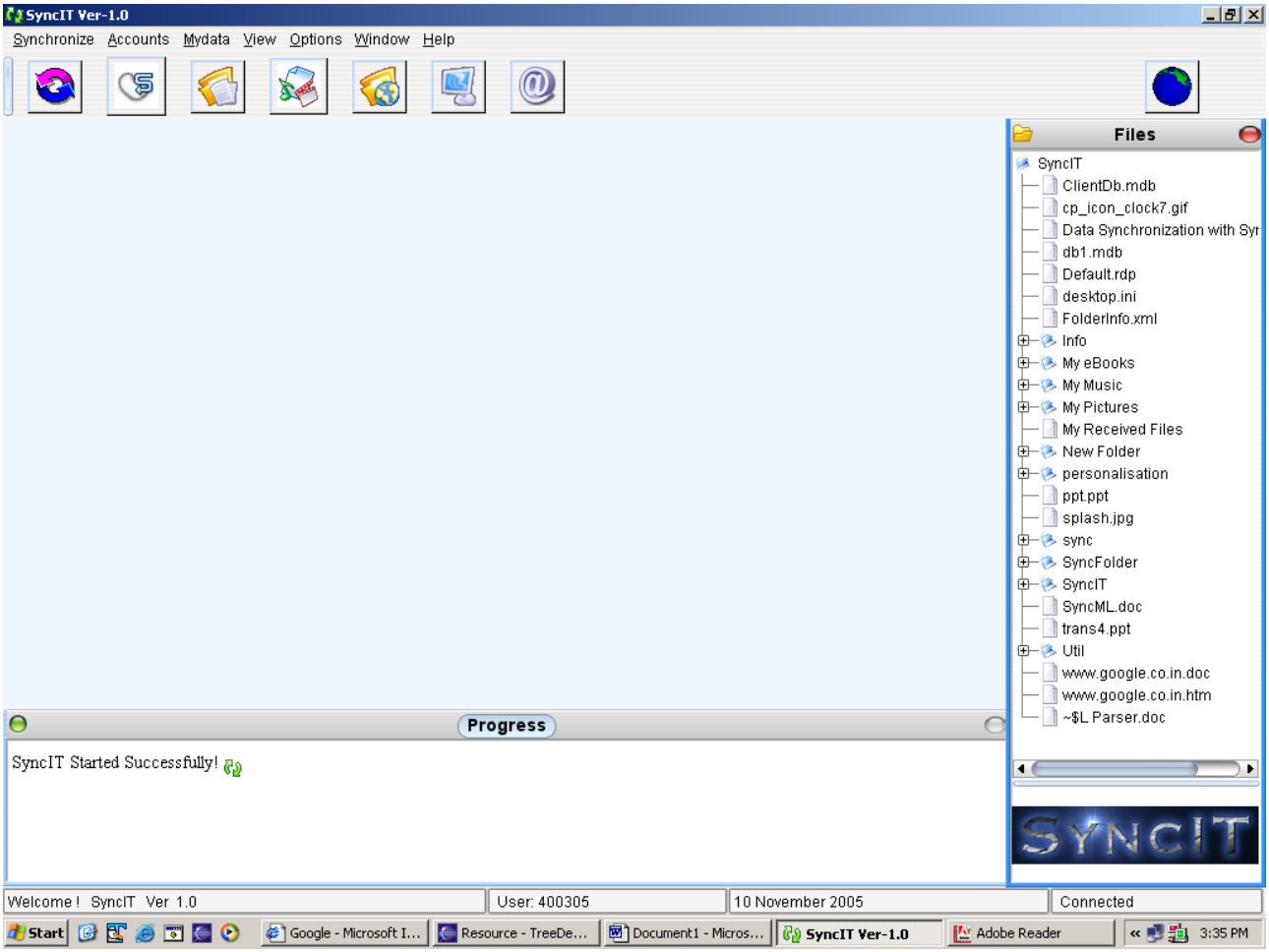


Figure A1.1 'SyncIT' Main Window

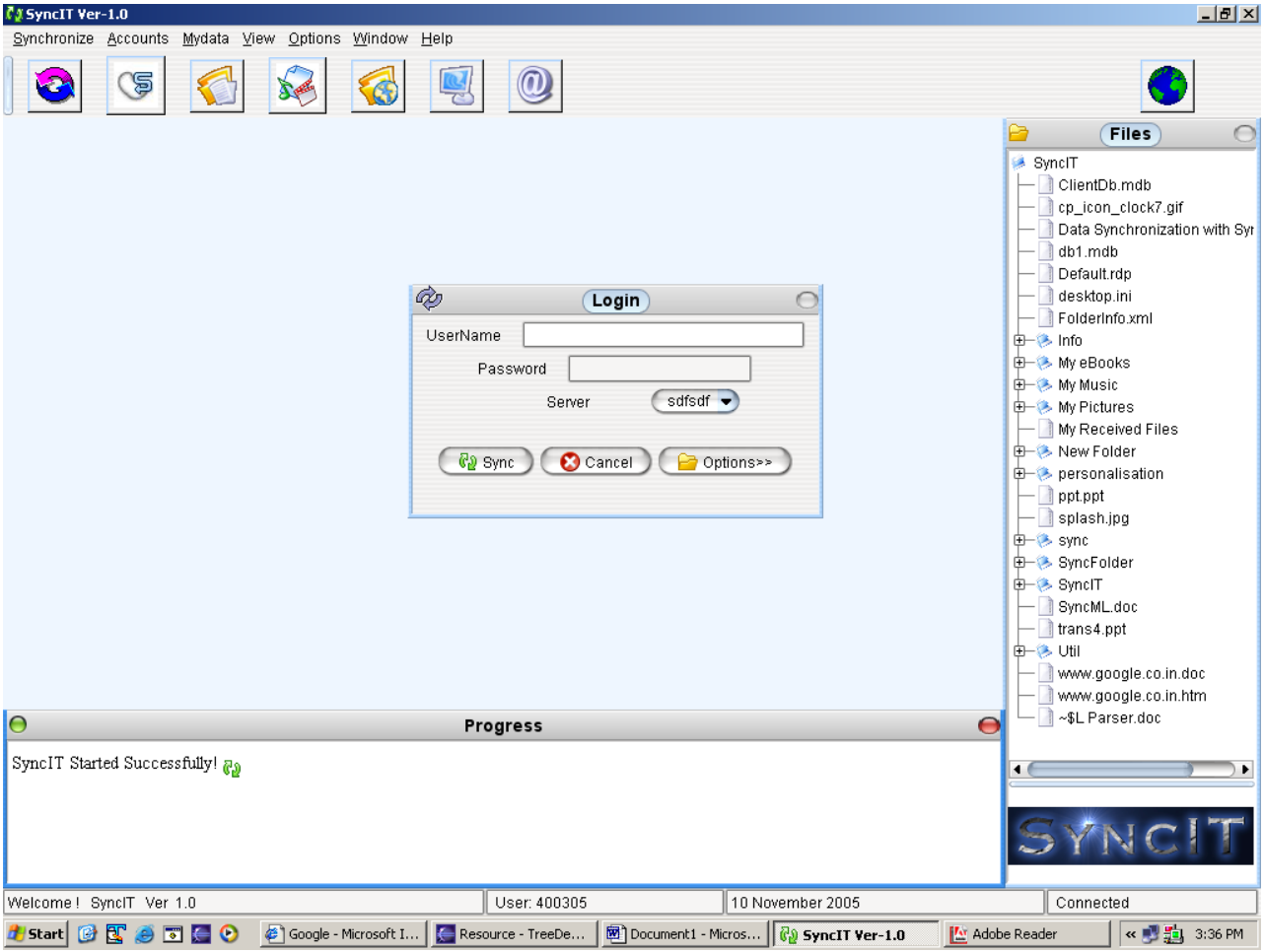


Figure A1.2 'SyncIT' Login Window

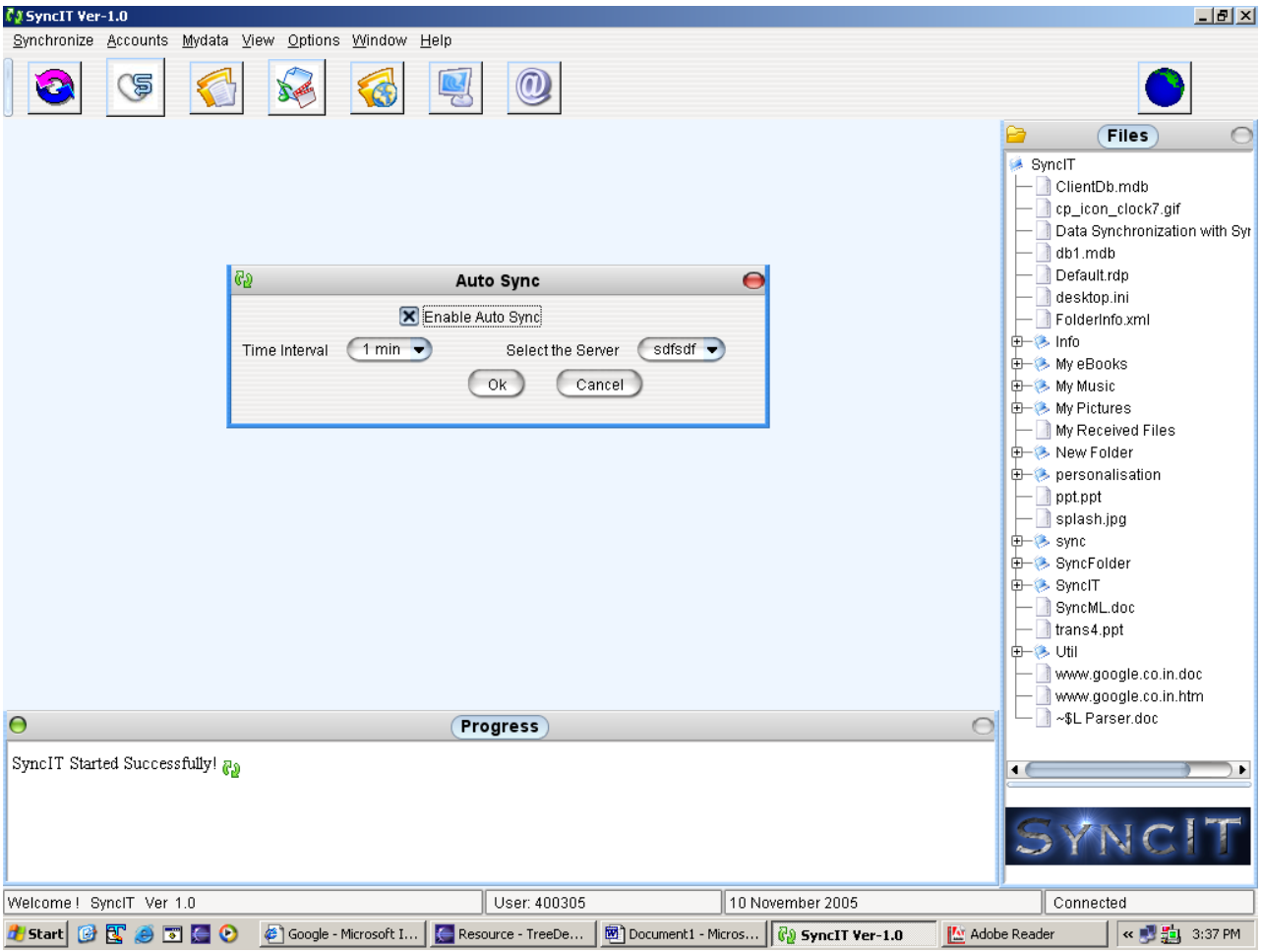


Figure A1.3 'SyncIT' Auto Sync Window

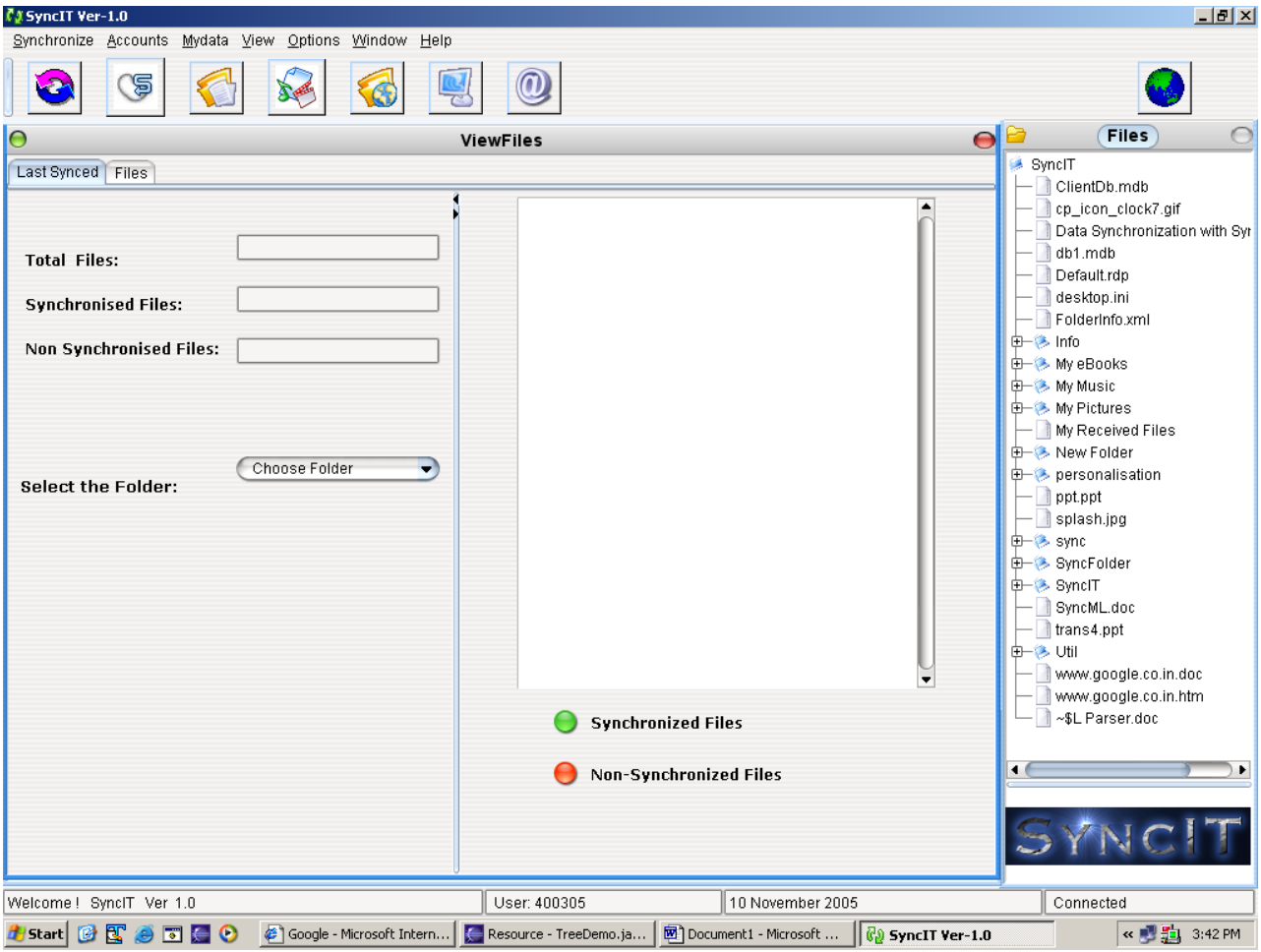


Figure A1.4 'SyncIT' View Files Window –Last Synced Tab

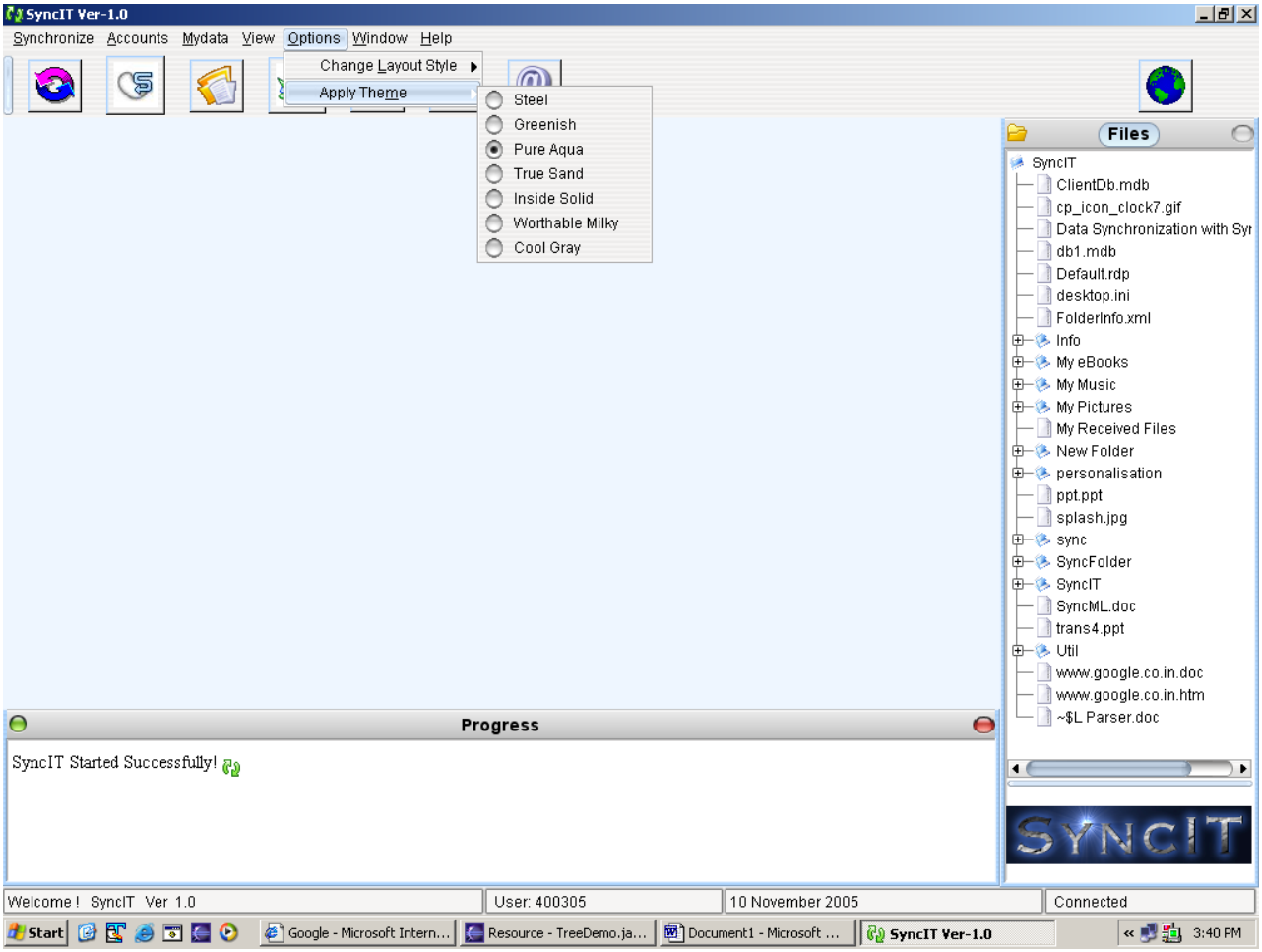


Figure A1.5 'SyncIT' Apply Theme Sub-Menu

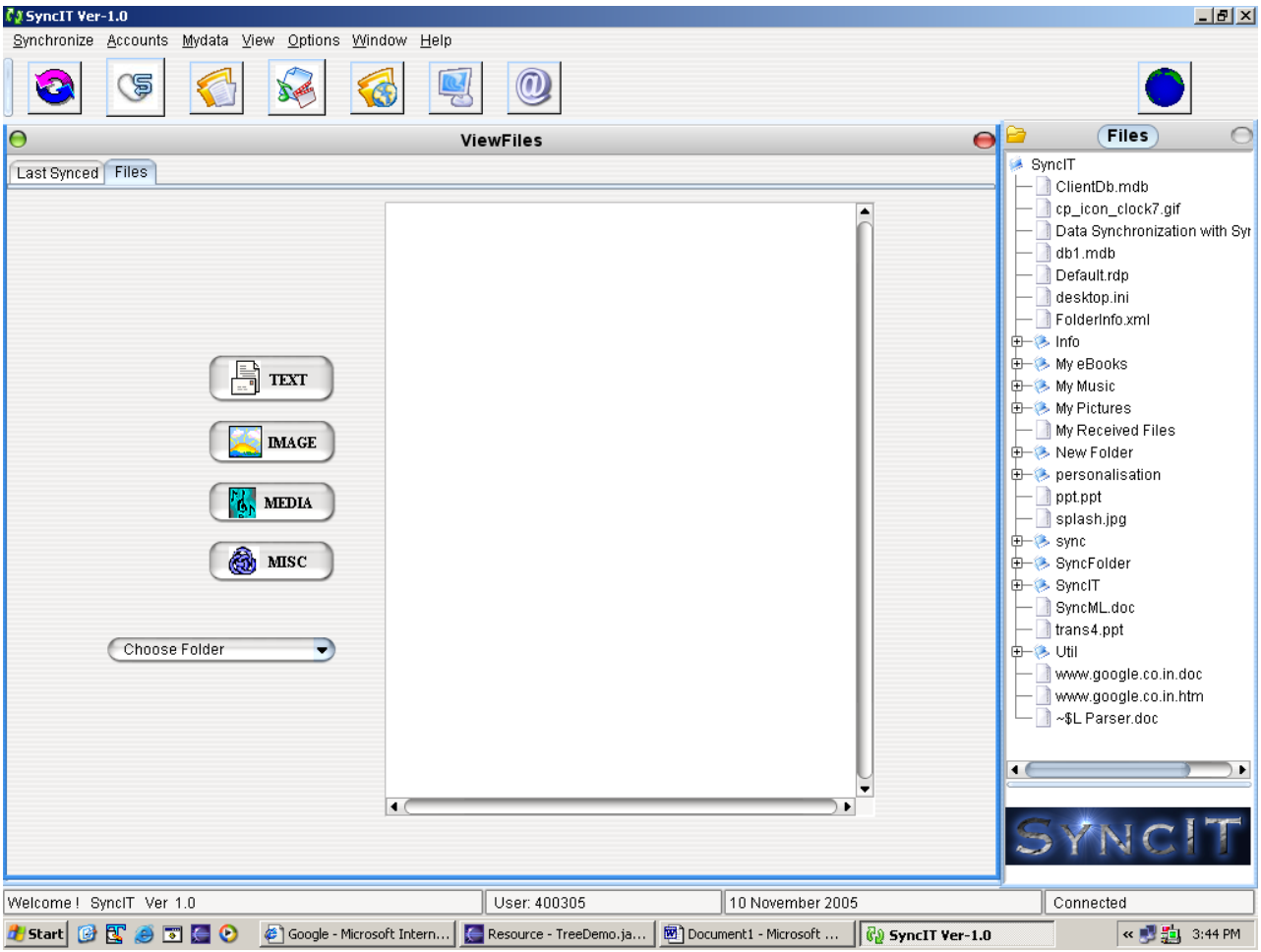


Figure A1.6 'SyncIT' View Files Window –Files Tab

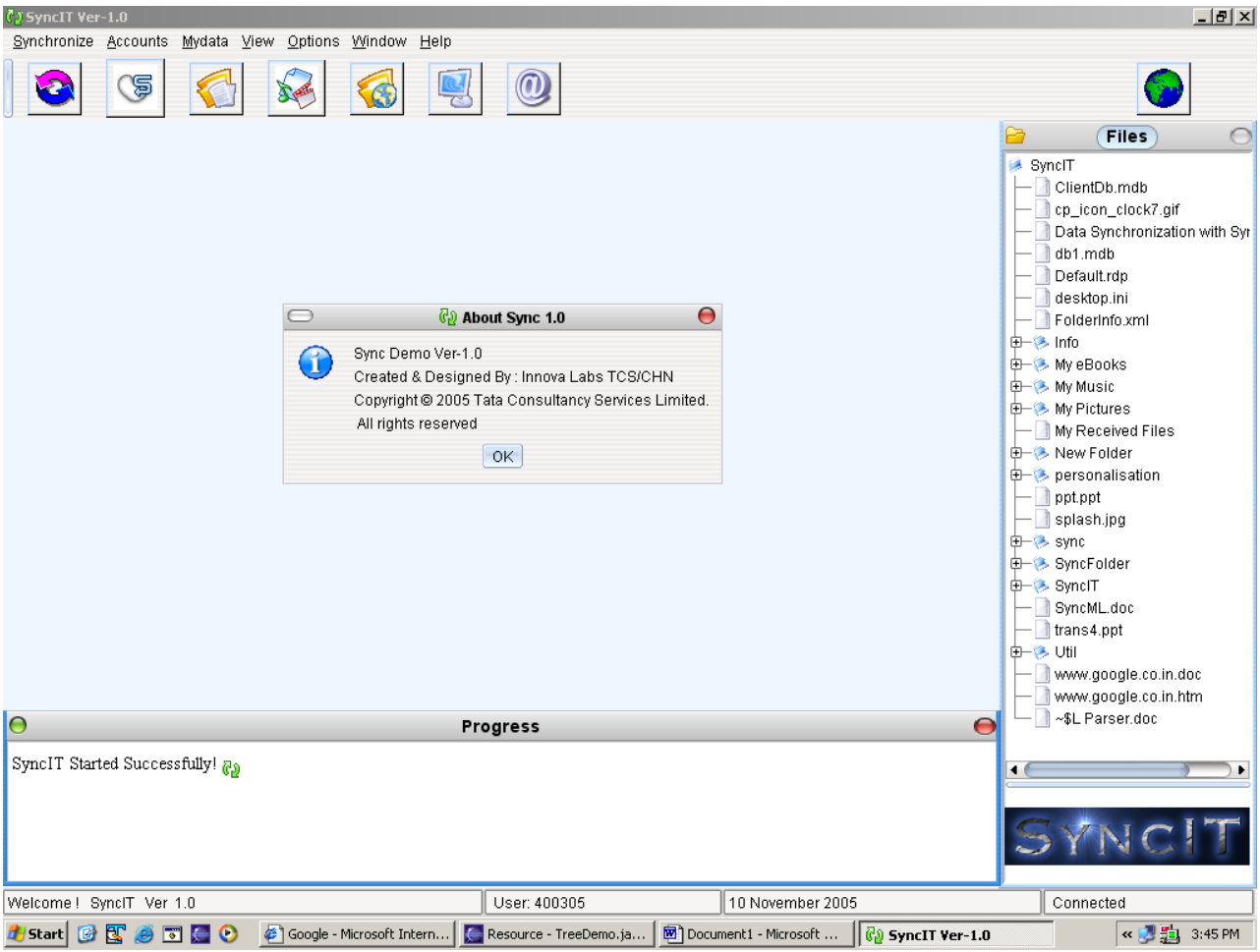


Figure A1.7 'SyncIT' About SyncIT Dialog

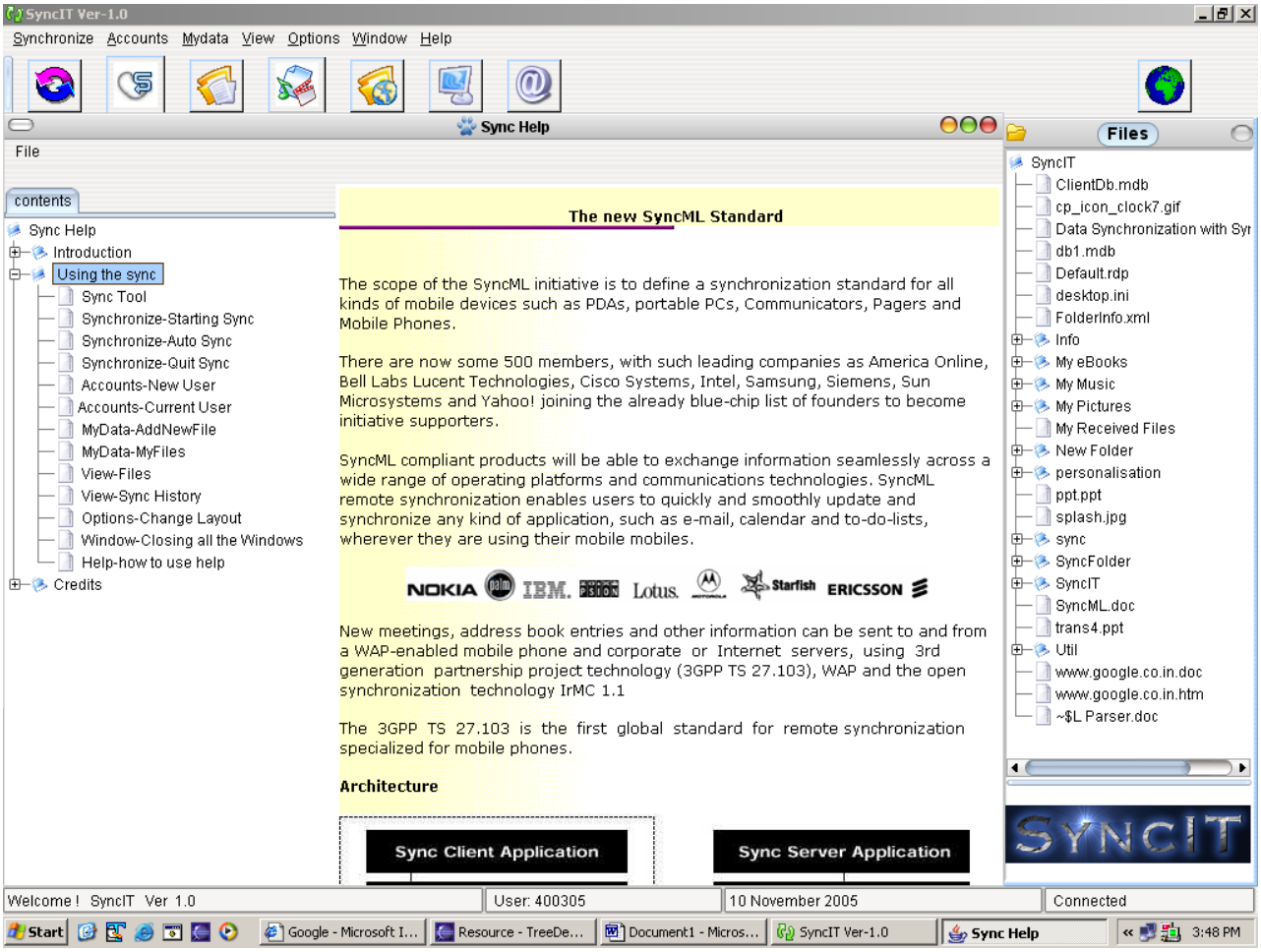


Figure A1.8 'SyncIT' Sync Help Window

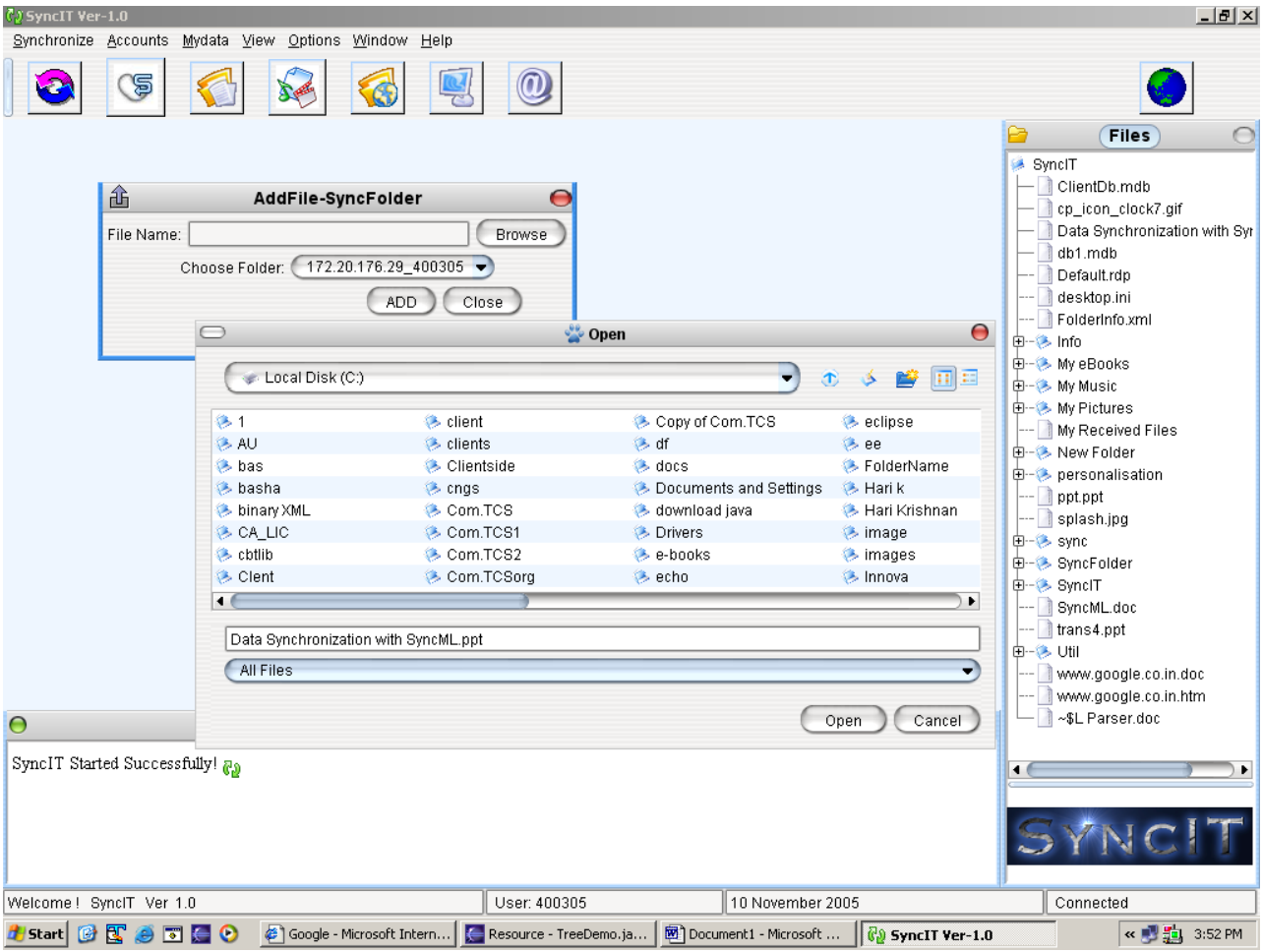


Figure A1.9 'SyncIT' Add File Window

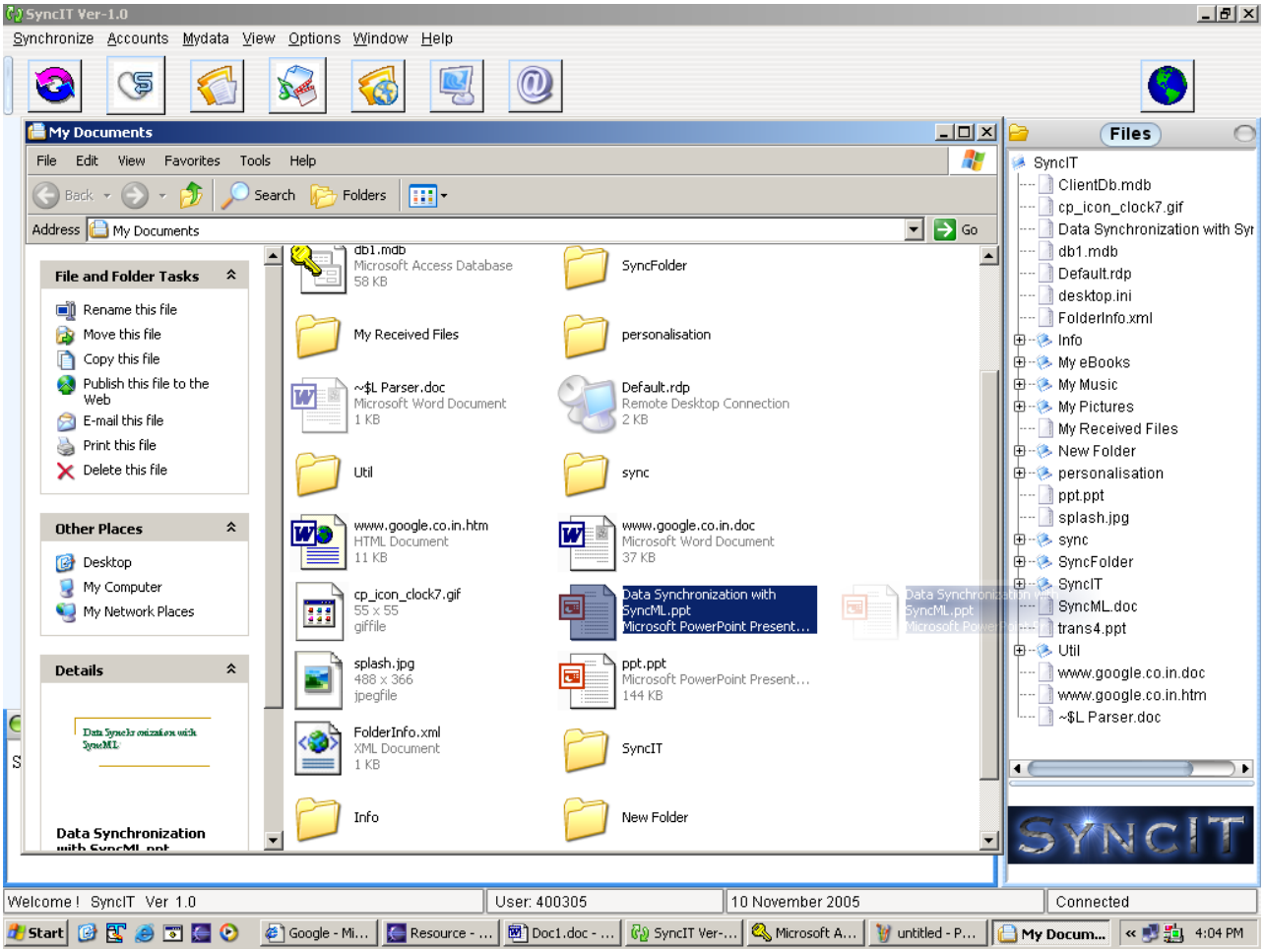


Figure A1.10 'SyncIT' Drag and Drop in Progress

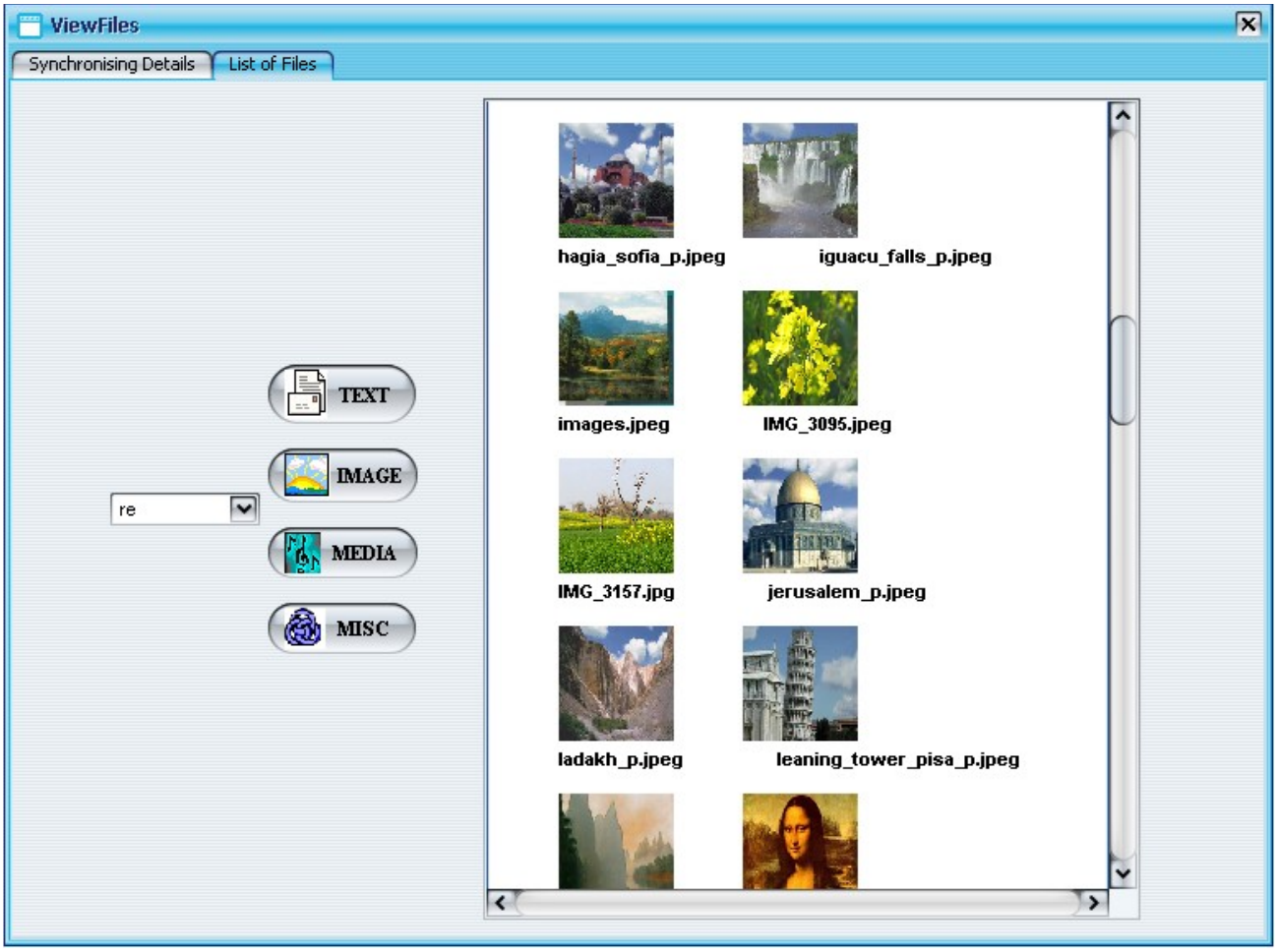
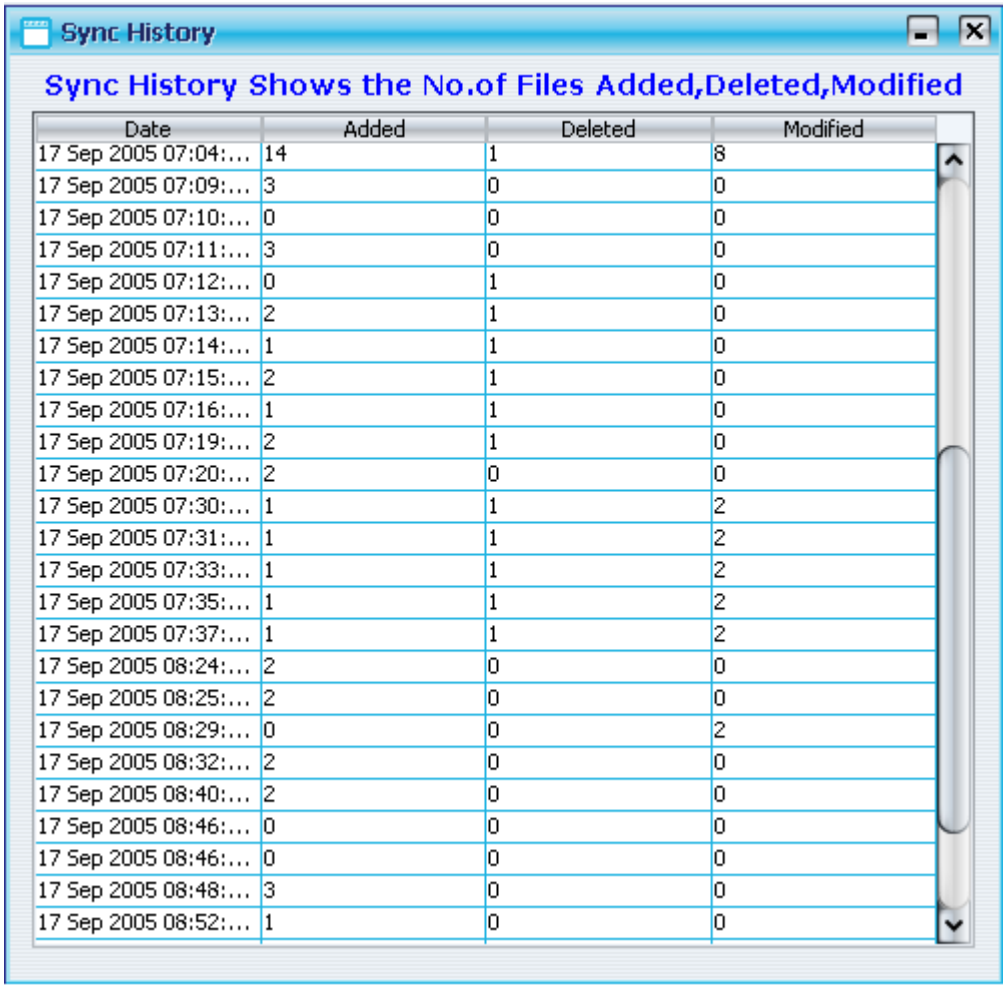


Figure A1.11 'SyncIT' View Files Window-Thumbnail Display



Date	Added	Deleted	Modified
17 Sep 2005 07:04:...	14	1	8
17 Sep 2005 07:09:...	3	0	0
17 Sep 2005 07:10:...	0	0	0
17 Sep 2005 07:11:...	3	0	0
17 Sep 2005 07:12:...	0	1	0
17 Sep 2005 07:13:...	2	1	0
17 Sep 2005 07:14:...	1	1	0
17 Sep 2005 07:15:...	2	1	0
17 Sep 2005 07:16:...	1	1	0
17 Sep 2005 07:19:...	2	1	0
17 Sep 2005 07:20:...	2	0	0
17 Sep 2005 07:30:...	1	1	2
17 Sep 2005 07:31:...	1	1	2
17 Sep 2005 07:33:...	1	1	2
17 Sep 2005 07:35:...	1	1	2
17 Sep 2005 07:37:...	1	1	2
17 Sep 2005 08:24:...	2	0	0
17 Sep 2005 08:25:...	2	0	0
17 Sep 2005 08:29:...	0	0	2
17 Sep 2005 08:32:...	2	0	0
17 Sep 2005 08:40:...	2	0	0
17 Sep 2005 08:46:...	0	0	0
17 Sep 2005 08:46:...	0	0	0
17 Sep 2005 08:48:...	3	0	0
17 Sep 2005 08:52:...	1	0	0

Figure A1.12 ‘SyncIT’ Sync History Window

REFERENCES

[1] Online free encyclopedia **www.en.wikipedia.org**

[2] Open mobile alliance website **www.oma.org**

[3] Sun website **www.java.sun.com**

[4] Java programming tutorials at **www.java2s.com**