



**Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom**

TELEPHONE:
INTERNATIONAL:
FAX:
E-MAIL:

**Cambridge (01223) 515010
+44 1223 515010
+44 1223 359779
apm@ansa.co.uk**

ANSA Phase III

Scripts and Agents: Introduction and Work Proposal

Ashley McClenaghan

Abstract

“The Net is the computer” — but how should it be programmed?

Mobile scripting and intelligent agent technology provides one possible answer to this question. It's potential is firing imaginations within the telecoms and computer industries. Giants such as Apple, AT&T, Motorola, Philips and Sony have already invested in this technology by joining the General Magic Alliance. General Magic would like it's Telescript technology to become as ubiquitous in the network as Microsoft's MS-DOS has become on the desktop.

However Telescript is not an open product. The aim of the ANSA B3 Workpackage is to promote the evolution of open scripting and agent technologies for programming the Internet. This document begins by introducing the world of scripts and agents. It then introduces two proposals for ANSA work steps towards the eventual aim of open scripting. The first work step is for prototype a Web server whose functionality can be extended by sending it scripts. The second work step is to develop mobile scripting technology for the in-service software upgrading of electronic devices such as set-top boxes, PDAs, TVs, feature-phones, etc.

APM.1434.00.02

Draft

14th March 1995

Request for Comments (confidential to ANSA consortium for 2 years)

Distribution:

Supersedes:

Superseded by:

Scripts and Agents: Introduction and Work Proposal



Scripts and Agents: Introduction and Work Proposal

Ashley McClenaghan

APM.1434.00.02

14th March 1995

The material in this Report has been developed as part of the ANSA Architecture for Open Distributed Systems. ANSA is a collaborative initiative, managed by Architecture Projects Management Limited on behalf of the companies sponsoring the ANSA Workprogramme.

The ANSA initiative is open to all companies and organisations. Further information on the ANSA Workprogramme, the material in this report, and on other reports can be obtained from the address below.

The authors acknowledge the help and assistance of their colleagues, in sponsoring companies and the ANSA team in Cambridge in the preparation of this report.

Architecture Projects Management Limited

Poseidon House
Castle Park
CAMBRIDGE
CB3 0RD
United Kingdom

TELEPHONE UK
INTERNATIONAL
FAX
E-MAIL

(01223) 515010
+44 1223 515010
+44 1223 359779
apm@ansa.co.uk

Copyright © 1995 Architecture Projects Management Limited
The copyright is held on behalf of the sponsors for the time being of the ANSA Workprogramme.

Architecture Projects Management Limited takes no responsibility for the consequences of errors or omissions in this Report, nor for any damages resulting from the application of the ideas expressed herein.

Contents

1	1	Industrial Interest
2	2	Remote Programming
4	3	Research Areas
4	3.1	Mobile scripts
4	3.2	Agent interaction
4	3.3	Agent intelligence
5	3.4	Our focus
6	4	General Magic
8	5	Telescript Overview
11	6	ANSA B3 Workplan
11	6.1	Technical issues
12	6.2	Work steps
14	7	Script-based, Extensible Web Server
14	7.1	A Web server which supports an extensible set of methods
14	7.2	Why do this?
14	7.3	Methods
15	7.4	Architecture
18	8	Software Upgradable Devices
18	8.1	Software upgradable consumer electronic devices
18	8.2	Downloadable software interfaces
19	8.3	Enabling technology
20	9	Summary

1 Industrial Interest

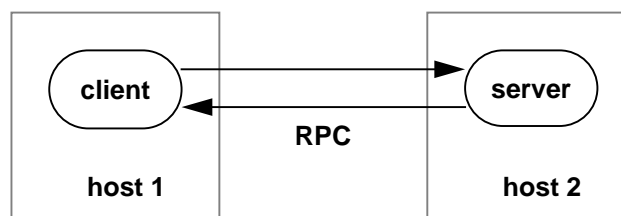
“Intelligent agents”, “mobile scripts”, “softbots”, etc. There seems to be a growing amount of industrial interest in such things, for instance

- General Magic (see §4) are licensing Telescript (see §5) – their mobile agent technology – to telecoms companies to build a new generation of “smart networks” (e.g. AT&T’s Personal Services [Reinhart]) and to consumer electronics companies to build end-user devices (e.g. Motorola’s Envoy PDA).
- Sun Microsystems Labs.[SunLabs] are investing in a project to turn John Outerhout’s [Ousterhouta] Tcl scripting language [Tcl/Tka][Tcl/Tkb] into the scripting language for the Internet.
- The Sunrise project [Sunrise] are doing agent related work involving OpenDoc [OpenDoc] and CORBA architectures.
- The pet project of the Technical Director of BT Labs. is the Ants Project — to foundation a telecoms network on the principles of ant communications networks for robustness, self healing, etc.

2 Remote Programming

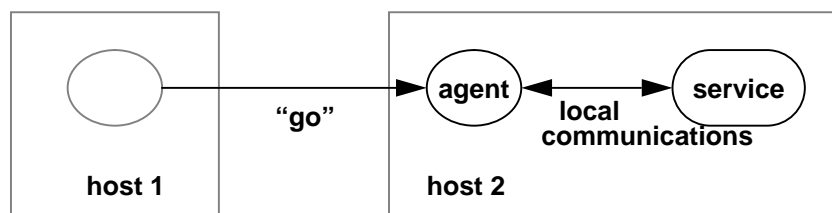
The common programming paradigm of intelligent agents, mobile scripts, softbots, etc. is often called “remote programming”. Figure 2.1 shows the tradition RPC-like programming paradigm which contrasts with figure 2.2 which shows the new remote programming paradigm for distributed systems.

Figure 2.1: The Traditional Client-Server RPC Model



In the RPC client-server model the client communicates with a non-local server. (Social analogy: this is like shouting!)

Figure 2.2: The New Remote Programming Model



In the remote programming model the agent teleports to the server’s host to locally interact with the server. (Social analogy: this is like conversion.)

In the traditional RPC-like programming world, a system is structured into a client-server architecture with statically defined APIs and protocols.

Whereas in the new remote programming world, a system is structured as an interworking set of agents which travel around the network to perform some task. Interfaces and protocols can be introduced or upgraded on-line.

Some agent (mobile script) characteristics:

- AGENT = DATA + PROCEDURES + LIFE
- Agents live and roam around in the network.
- Their called agents since they often act on someone’s or something else’s behalf.

- Network hosts must agree on a language for agents — a universal scripting language. (General Magic propose that this language be Telescript.)

3 Research Areas

The topic scripts and agents means different things to different people. We might identify three areas of relevant research for this topic:

3.1 Mobile scripts

This is about designing the scripting languages themselves.

- Proprietary languages, e.g.
 - Telescript from General Magic (see §4 and §5)
 - Script-X from Kaleida (IBM, Apple, Hatachi and Toshiba).
- Open languages, e.g.
 - (Safe-)Python
 - (Safe-)Tcl [Tcl/Tka][Tcl/Tkb][Safe-Tcl][MIME/Safe-Tcl][Email/Safe-Tcl][Email/Safe-Tclb]. Tcl is already widely used by the Internet community. It is quite stable, free, available for a number of platforms, and comes with a number of useful extension packages.

3.2 Agent interaction

This is about figuring out how autonomous agents will interact with one another and find services within the Net.

- Interaction models, e.g. based on
 - OpenDoc [OpenDoc]
 - Microsoft's COM OLE2
 - OMG's CORBA
- Interaction languages, e.g.
 - Knowledge Query and Manipulation Language [KQML]
 - Cyc [Cyc]

3.3 Agent intelligence

This is about making agents smart.

E.g.

- SIM_AGENT [SIM_AGENT].

3.4 Our focus

Our proposed focus under ANSA Workpackage B3 is on the first of these areas: mobile scripts. Agent interaction is related to ANSA's work on Trading and Matchmaking [Deschrevel], and Meta-data [Madsen95].

4 General Magic

Before we look specifically at the ANSA B3 work proposals for scripts and agents we introduce the business context by looking at the most prominent business player in this field: General Magic.

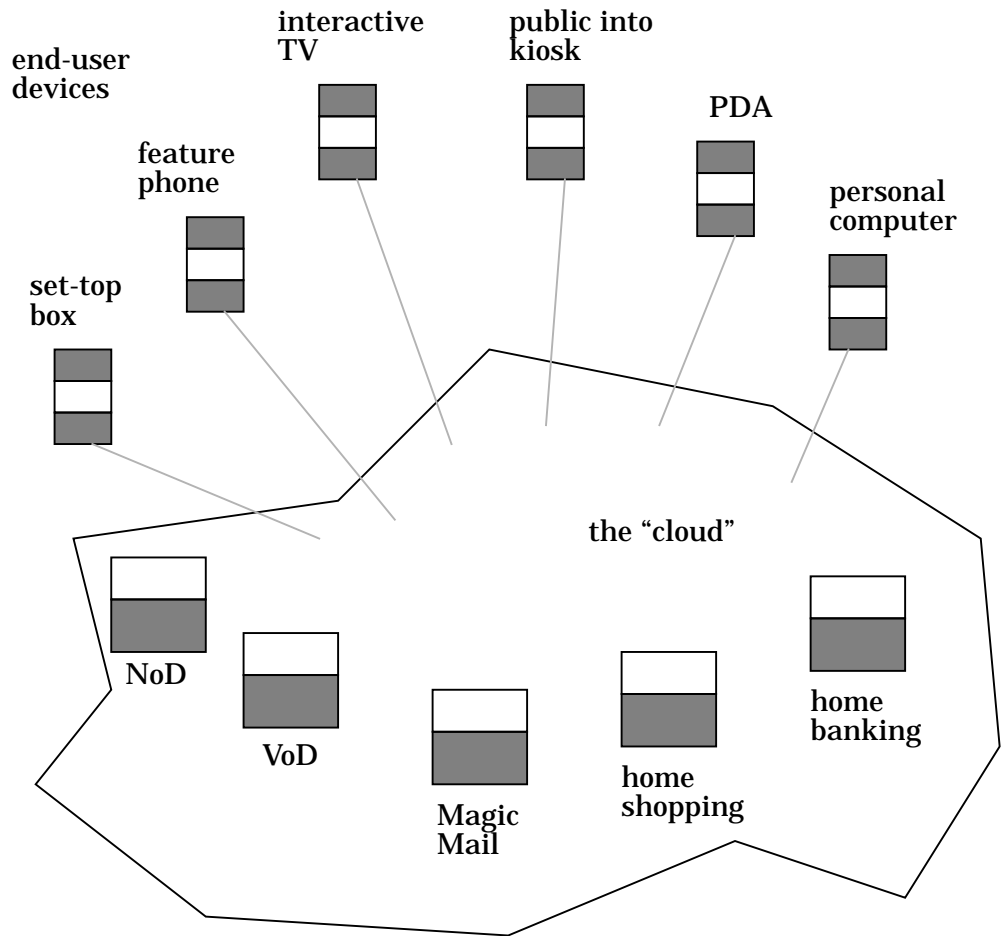
- 1990: General Magic Inc. spun-off from Apple.
- 1993: General Magic Alliance announced. The Alliance includes: Apple, AT&T, Motorola, Philips and Sony.
- Aim: Establish a new class of personal communication products and services.
- Belief: The Net is the computer.
- Approach: Make Telescript ubiquitous — establish Telescript as the global standard for intelligent messaging. Key technologies: Telescript, Magic Cap [GM-ICF93][GM-TeleLess93][GM-TeleProg93]

General Magic are targeting telecoms companies in an effort to have them use Telescript as the programming infrastructure in their networks — e.g. AT&T have used Telescript to implement it's new Personal Link Service [Reinhart]. General Magic are also targeting consumer electronics companies in an effort to have them use Telescript and Magic Cap (their Telescript enabled operating system/graphical user interface) in end-user products — e.g. Sony and Motorola have both launched Telescript/Magic Cap based PDAs onto the market.




Figure 4.1 depicts how General Magic see the electronic marketplace:

- Telescript will form the homogenous distributed systems platform — it will be present in end-user devices, networks and service provider computers.
- The “cloud” is the value added network — the telecoms net supporting third party services like Magic Mail (email), video-on-demand, news-on-demand, home shopping, home banking, etc.
- End-user devices includes devices such as PDAs, set-top boxes, interactive TVs, personal computers, feature phones, public information kiosks, etc.
- End-users will rent resources in the cloud — e.g. mail-boxes in which their incoming email will be stored, CPU resources for processing, media content when viewing VoD or reading an electronic newspaper, etc.

Figure 4.1: General Magic's Electronic Marketplace



key:

-  Magic Cap
-  the application
-  Telescript

5 Telescript Overview

Here is a short summary of Telescript's main features [GM-TeleLess93][GM-TeleProg93]:

- The Telescript language is a pure (everything is an object) object oriented language which looks a bit Smalltalk-ish.
- Telescript is an interpreted language. To run a Telescript program you need a Telescript Engine. Various Engines are available for a number of different platforms. The Magic Cap software running on Sony and Motorola PDAs consists of a Telescript Engine plus a GUI plus communications mechanisms.
- There are two language levels to Telescript: High Telescript and Low Telescript. High Telescript has an object oriented syntax and is compiled to Low Telescript. Low Telescript has a postfix syntax for stack-based interpretation.
- The basic network configuration is to run a Telescript Engine on each node (computer) in the network. The network of interworking Telescript Engines provides an abstract homogeneous environment in which to build distributed systems.
- The most important class in the Telescript language is the Process. Telescript supports pre-emptive, prioritized multi-tasking of Process objects. A Process instance can be thought of as an object with a life of its own.
- Place and Agent are the two important subclasses of the Telescript Process. A Place object represents a virtual space in which other objects can interwork (through local communication). Each Telescript Engine can support a number of places.
- An Agent object is a Process object which can migrate between Places. An agent may move between Places on the same Engine, or between Places which exist on different Engines. Hence the Telescript notion of a distributed system is a number of distinctly located places and a number of Agents which move between these Places.
- Places provide meeting locations for Agents. At a Place, Agents can exchange information and perform computation. Places also route travelling Agents.
- Telescript supports persistent objects — Telescript Engines implicitly save and recover object state information. This is as a failure recovery mechanism where objects are automatically recovered to the state previous to a system failure. Persistency is also transparently supported when objects migrate. When an Agent transports itself (using the “go” method) from one Place to another Place on a different Engine: its execution is suspended, its state information encoded and transferred

through the communications medium, then decoded, and finally the Agent's execution is resumed at its new location.

- Agents have “attributes” such as “identify” and “owning authority” which uniquely identify the Agent and the entity responsible for it. These attributes may be used for authentication. Telescript objects also have a “permit” attribute which may be used to limit the amount of resources which they may consume (e.g. a Place may ask an Agent to pay it 30 “Teleclicks” before granting it access to some resource). A secure “permits” feature is crucial to stop Agents from creating a crash-limited number of clones of themselves, exhausting resources, or other such anti-social behaviour.

(Apparently you can't define a legal Telescript Place which holds visiting Agents to ransom unless you can circumvent security features and hack the Interpreter code!)

- The Telescript world is divided into “regions”. Each Engine uses a “regions” database to route migrating Agents. Places and Agents are identified using “Telenames”:

```
Telename( Locally-Unique-Name, Region-Name )
```

As I understand it, the “regions” database provides only simple routing hints. More sophisticated trading services (e.g. the ANSA Matchmaker) are too be supplied as third party services.

I think that each Telescript Place provides a trading service to local Agents. This enables an Agent to request a “meeting” with another Agent of a “conformant” class (using Telescript's inheritance rules).

- Inter-process communication — Telescript Agents can only interact when they are co-located within the same Place. (There's no RPC-like notion in the Telescript world). An Agent (the meeting initiator) can request to meet with another Agent (e.g. a specific Agent instance, or any instance of some (sub)class). Assuming that the target Agent is available, the Place provides the meeting initiator with an 'object reference' to the target Agent. The meeting initiator can then invoke the methods defined in the target Agent. The “meeting” procedure is asymmetric — the target Agent is not passed an object reference to the meeting initiator.
- Roughly speaking, an object may have 3 types of method:
 - “system” — invoked by the Engine.
 - “private” — invoked only by itself.
 - “public” — invoked by anyone.
- Important methods of the Place class include:
 - “initialize”— invoked at instance creation-time. Initialization maybe “escalated” to its superclass(es)
 - “entering” — invoked when an object (an Agent or even another nested Place) enters the Place or is created within the Place.
 - “exiting”— invoked when an object leaves the Place.
 - “transferredIn” — invoked when an Agent tries to leave but fails for some reason (communications failure, invalid address, etc.) .
- Important methods of the Agent class include:

- “live” — the ‘main’ code of the Agent. A codification of the Agent’s destiny!
- “go”— e.g.:

```
self.go(ticket(telename(localName,regionName),...))
```

6 ANSA B3 Workplan

Telescript is a well thought out product but, for us, it has three disadvantages:

1. proprietary — its use is licensed by General Magic
2. private networks — it has been developed for use on private networks (e.g. AT&T's Personal Link Service network) but not the public Internet
3. low-level — it lacks higher level abstractions for script and agent programming.

In contrast, the aim of the ANSA B3 Workpackage [ANSA][ReesEdwards] is to:

1. evolve an open mobile scripting language
2. for programming the Internet
3. with high level abstractions appropriate for building open systems structured as distributed, interacting, mobile agents.

6.1 Technical issues

Some of the important technical issues to be tackled are:

- Scripting language
What programming language, what abstractions, etc.
I want to be as pragmatic so I intend to use explore existing scripting languages (like Safe-Tcl — which seems the most promising) and not invent a new language.
- Safe interpretation — creating a containment structure in which scripts are safely interpreted to prevent them from accessing resources in authorized or uncontrolled ways. The Safe-Tcl interpreter will provide the basic restricted environment in which to execute agents.
- Security
Authenticating scripts, script senders, and interpreters.
I may look at using Philip Zimmermann's Pretty Good Privacy (PGP) [Zimmermann] for authentication and encryption.
- Transport
Moving scripts from between interpreters.
It would be interesting to explore how scripts might be packaged and transported in three different contexts: the Web, email, and some direct interpreter-to-interpreter transport protocol.
- Content formats -
Moving data between hosts.

MIME [MIME/Safe-Tcl] provides a way to encapsulate different data blobs for transport to different platforms. (This seems analogous to General Magic's Interplatform Content Format [GM-ICF93].)

- Resource negotiation

The sender of a script may have to negotiate with the receiver of the script for CPU and memory usage, access to local resources such as files, databases, transport mechanisms, the availability of libraries, payment for resource usage, etc.

An incoming script might be delivered in an envelope which contains details such as the script's owner, the resources required by the script (e.g. CPU time, memory size, access permissions to local services), how much the owner is prepared to pay, etc. Only after the local script interpreter has authenticated the script and agreed to the requested resource access, it will bring the script to life.

It is likely that resource negotiation protocols will be designed for specific services as they come on-line.

- Interaction

How autonomous scripts interact and find services.

On a local scale we may require a mechanism which allows an agent to request to meet with another local agent. (This is akin to a cross between an ANSA Trader service and a Binding service). If a suitable target agent is available, then the meeting requester is given a reference to this target agent that may be used when invoking method of the target agent.

In the Telescript model one agent may invoke the methods of a co-located agent. (Co-located meaning that the two agents are "in" the same Place.)

We could introduce a new Safe-Tcl primitive which parcels interaction, e.g.

```
SafeTcl_invoke(targetAgentRef, method, args...)
```

This could be introduced into Safe-Tcl via the "declareharmless" command. And could be implemented on top of "send" for Tcl/tk, or an RPC call for Tcl-DP.

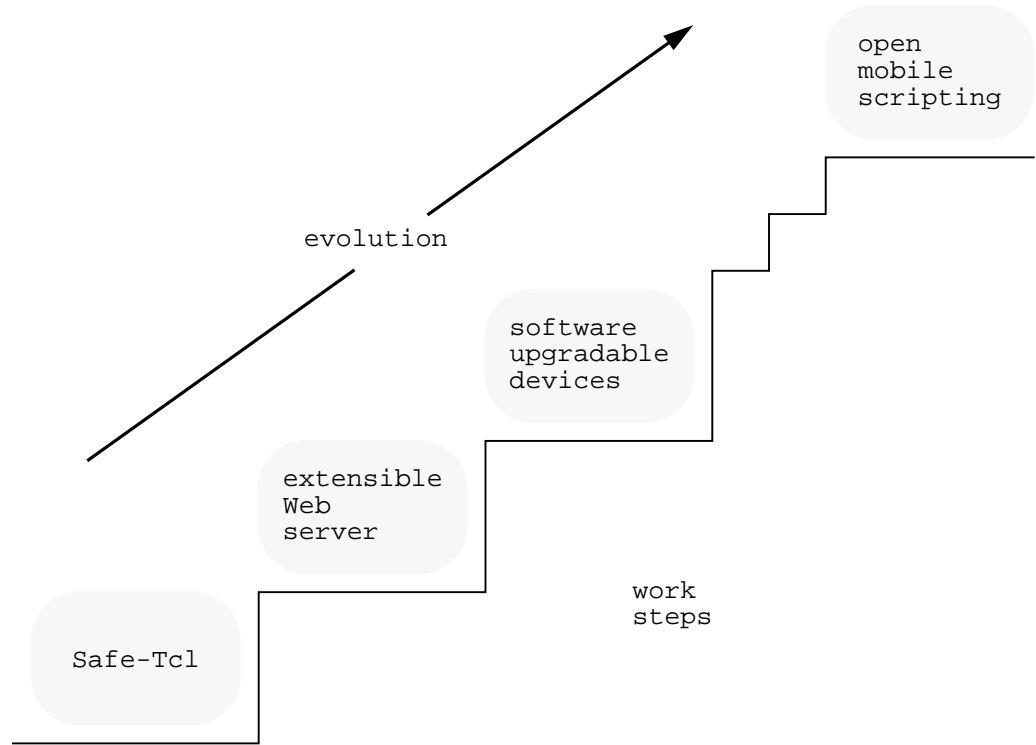
More global scale trading and trading for higher-level services will be implemented by third parties.

6.2 Work steps

Figure 6.1 shows two proposed work steps in the ANSA B3 Workpackage. The initial work steps are:

- Prototyping a script-based, extensible Web server.
- Exploring software upgradable devices.

Figure 6.1: Proposed ANSA B3 Work Steps



7 Script-based, Extensible Web Server

- Proposal: To prototype a script-based Web server which supports an extensible set of methods.

This is my short term work proposal (i.e. for about three months from this date).

7.1 A Web server which supports an extensible set of methods

The HyperText Transfer Protocol (HTTP/1.0) defines how Web clients and servers communicate. HTTP uses the RPC request-response model where:

1. a Web client sends a request to a Web server
2. the request invokes a specified method on the server
3. the result of the method is sent back to the client as a response.

HTTP is designed to support an extendable set of methods. At present most Web servers recognise only a basic set of methods (e.g. GET, HEAD, POST, etc.), and provide no support for extending this method set. The proposed work would prototype a Web server which supports an extensible set of methods.

7.2 Why do this?

To learn about:

- Tcl and Safe-Tcl
- safe interpretation
- on-the-fly (in-service, on-line) software modification
- using encryption technology (e.g. PGP) for authentication and other security
- using the Web as a delivery and execution context for scripts.

Also:

- The HTTP/1.0 definition allows for a server with an extendable set of methods but no-one has implemented such a server yet (to the best of my knowledge).
- This work relates to ANSA's Web-CORBA work [ReesEdwards][Edwards], and Web meta-data work [Madsen95].

7.3 Methods

The basic idea is that there will be three sets of methods:

- Kernel management methods

These are ever present methods.

- INFO

An INFO method request will allow a client to ask the server to return details about the environment in which a method script will be interpreted. The server will respond with a list of installed methods, a list of usable script procedures, and a list of global variables accessible to executing method scripts.

- INSTALL

An INSTALL method request will allow a client to ask the server to install the given script as a HTTP callable method. The given method script may use procedures in other already installed method scripts and, in-kind, the procedures in the given script will be made available for other method scripts to use.

- REMOVE

A REMOVE method request will allow a client to ask the server to removed an installed method.

- Standard library methods

These methods are automatically loaded.

- GET

- HEAD

- POST

- etc.

- User supplied methods

The idea is that when the prototype server starts-up for the first time that it will support only the three basic management methods. The standard library methods like GET, HEAD, POST, etc. will then be loaded automatically. Methods installed using INSTALL will persist across server process reincarnations until removed by a REMOVE request or a re-configuration of the server.

7.4 Architecture

Figure 7.1 shows the structure of the prototype server.

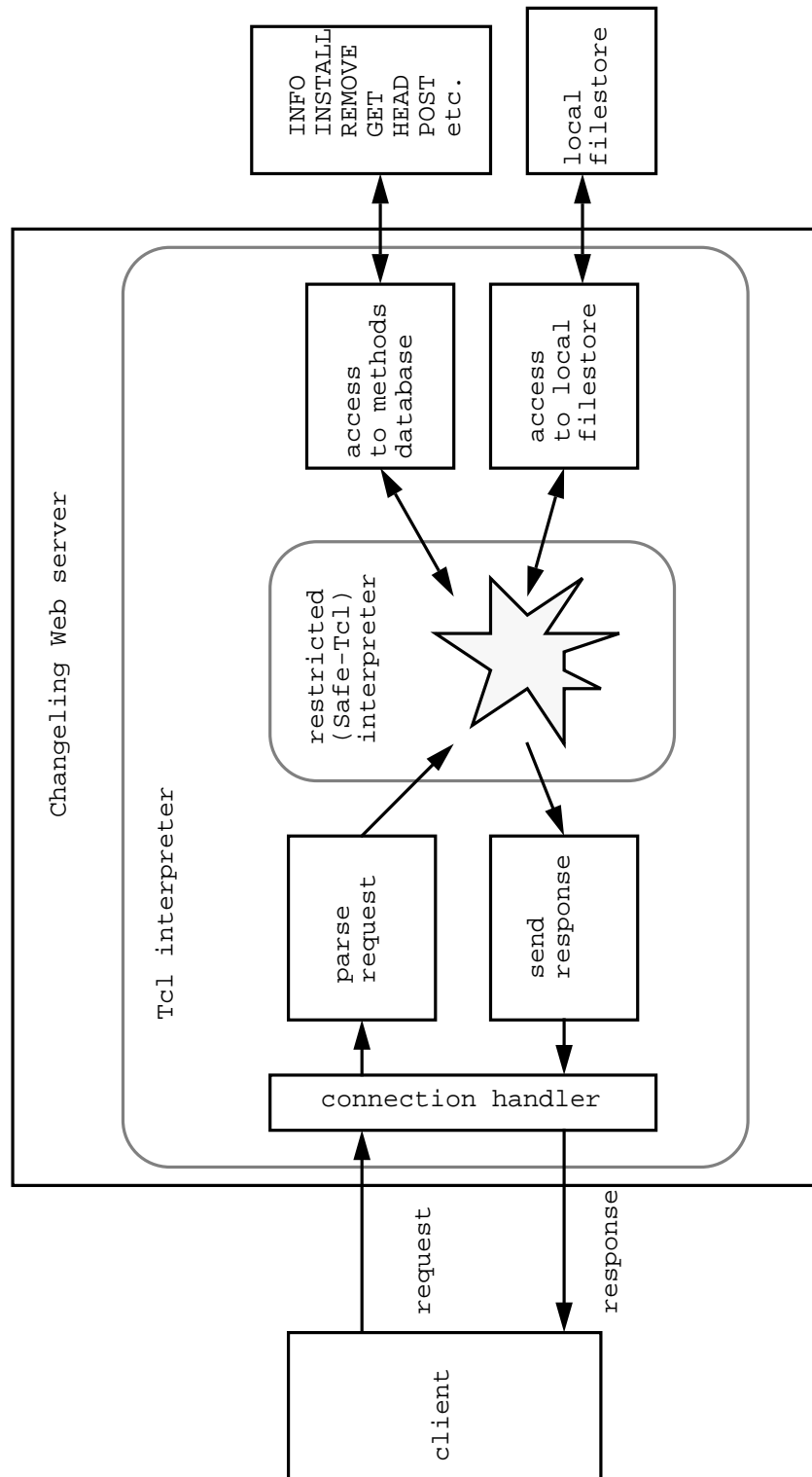
- The connection handler

The main loop of the server is a Tcl script (with Tcl-DP, TclX and Safe-Tcl extensions) which listens for new HTTP connections at the server port.

A new Tcl interpreter is spawned to handle each connection¹.

1. Which means that in this initial design stateful client-server interactions will have to be implemented using the local filestore to store state information between HTTP connections. Later designs might spawn and maintain a Tcl interpreter for each stateful interaction.

Figure 7.1: Architecture of the Changeling Web Server



- Parsing the request

The spawned interpreter handling an HTTP connection will parse the header information in the incoming request. The results of the parse will be placed into a data structure which will later be made available to the requested, executing method.

- **The restricted interpreter**

A restricted (Safe-Tcl) interpreter is created to execute the requested method script. The restricted interpreter executes method scripts inside a leak-proof environment which prevents the method scripts from directly accessing any outside resources. The requested method script is loaded into the restricted interpreter and executed (interpreted).
- **Access to local resources**

A method script (executing within the restricted interpreter) may indirectly access outside resources using a set of well controlled resource access procedures made available by the Tcl interpreter. These procedures will provide facilities for:

 - write/read access to a limited, private filestore (this could, for example, be used implement state across a number of HTTP requests, where the files persist for the duration of some transaction-ID or a timeout occurs)
 - reading and writing resources such as HTML documents
 - access to the information obtained from parsing the request headers
 - constructing and sending an HTTP response.
- **Authentication and access privileges**

The server will have to be able to authenticate users when they install, remove or invoke a method. This may be based on PGP public-key security. For example, a user installing an method may want to specify that only users from a given group will have privilege to access the method. Also, a potential method user may first want to authenticate the owner of a method before invoking it.

8 Software Upgradable Devices

- Business context

HARDWARE = £FEW

SOFTWARE = £MANY

Therefore value is added through software

- The coming of the ubiquitous Net

The IPing Internet standard will provide massive connectivity — potentially almost all electronic devices could be connected to the Net. Target devices include: set-top boxes, printers, point-of-sales devices, feature phones, PDAs, photocopiers, TVs, VCRs, hi-is, software interfaces, etc.

The added value of software and the possibilities of future connectivity opens up a massive potential market for in-service software upgrades and on-line maintenance of electronic devices.

8.1 Software upgradable consumer electronic devices

Software is playing an ever increasing role in the world of consumer electronics. Much of the added value in products like TVs, hi-if systems, VCRs, printers and photocopiers comes from sophisticated software. Products are often upgraded by upgrading the software alone. For example, a TV range will have the same basic hardware chassis with value and diversity in the range added through software. In a TV many of the functions along the audio/video signal processing path are performed in software, as are the teletext and on-screen display functions. For, say TVs, some manufactures imagine scenarios where they download upgrading software via either teletext broadcasts or telephone connections. And due to the diversity of processors used within their TV range, the downloaded software needs to be an interpretable script rather than an executable binary.

8.2 Downloadable software interfaces

Mobile script technology could also be used to implement downloadable software interfaces to new services (such as VoD, NoD, home banking, home shopping, etc.) as they come on-line. For example, when a new VoD service comes on-line, a set-top box user could download the video-browser interface for the new service in script form. We might even imagine the exciting scenario where the set-top box user could construct a complete virtual high-street of interactive services. Where the virtual store-front to each service will be realised by a downloaded interpretable script.

8.3 Enabling technology

The enabling technology in these scenarios is mobile scripting. Mobile scripting technology will allow us to:

- operate over a diverse range of hardware platforms
- download software
- perform in-service upgrades
- provide safe, high-level programming primitives.

9 Summary

This document is an introduction to the ANSA B3 Workpackage on Scripts and Agents. We introduced scripts and agents, and outlined the interest that research organizations and industry has in them. In particular we looked at General Magic – the most prominent present player in the field – and its Telescript technology. However Telescript is proprietary and low-level which brought us to the ANSA B3 Workpackage. ANSA's aim is to evolve open mobile scripting technology. We proposed two work steps towards this goal: a script-based, extensible Web server, and an exploration of software upgradable devices.

References

[Ousterhouta]

John Ousterhout, *Personal Web page*, <URL:http://playground.Sun.COM:80/~ouster/>, 1995,, Sun Microsystems Labs.

[Ousterhoutb]

John Ousterhout, *Tcl: A Universal Scripting Language*, USENIX Symp. on Very High Languages, October 26, 1994, Sun Microsystems Labs.

[Ousterhoutc]

John Ousterhout, *Scripts and Agents: The New Software High Ground*, Invited talk, USENIX Conf., January 19, 1995, Sun Microsystems Labs.,

[SunLabs]

Sun Microsystems Labs., *Web page of plans*, <URL:http://playground.Sun.COM:80/~ouster/sunplans.html>, 1995

[Sunrise]

Sunrise Project, *Web page*, <URL:http://www.acl.lanl.gov/sunrise/DistComp/intro.html>, 1995

[Reinhart]

Andy Reinhart, *The Network with Smarts*, BTYE, October 1994, Pages 51-64.

[OpenDoc]

OpenDoc, *Web pages giving overviews on OpenDoc*, <URL:http://www.acl.lanl.gov/sunrise/DistComp/OpenDoc/overview.html>, <URL:ftp://cil.org/pub/cilabs/general/CILabs_Bckgrdr_9409/CILabs_Bckgrdr_9409.html>, 1995

[KQML]

Tim Finin, Rich Fritzson, Don McKay, Robin McEntire, (Knowledge Query & Manipulation Language) KQML - A Language and Protocol for Knowledge Exchange, <URL:http://retriever.cs.umbc.edu:80/kqml/>

[Cyc]

R V Guha, Douglas B Lenat, *Enabling Agents to Work Together*, Comms. of the ACM, Vol.37, No.7, July, 1994

[SIM_AGENT]

Aaron Sloman, Riccardo Poli, (*SIM_AGENT*) *Playing God: A toolkit for building agents*, draft, Uni. of Birmingham, Dec. 1994

[GM-ICF93]

Barbara Nelson, *Interplatform Content Formats*, General Magic Inc., 2465 Latham Street, Suite 100, Mountain View, CA 94040, USA, October 1993, Magic 315

[GM-TeleLess93]

Scott Kronick, *Telescript Language Lessons - The Fun Book*, General Magic Inc., 2465 Latham Street, Suite 100, Mountain View, CA 94040, USA, June 1993, Magic 270

[GM-TeleProg93]

Susan Rayl, *Telescript Programming*, General Magic Inc., 2465 Latham Street, Suite 100, Mountain View, CA 94040, USA, April 1993

[Hanck94]

John Hanckmann, *Telescript: the emerging standard for intelligent messaging*, Philips Telecommunication Review, Vol. 52, No. 1, Pages 15-19, March 1994

[Tcl/Tka]

John S. Ousterhout, *Tcl and the Tk Toolkit*, Addison-Wesley, 1994

[Tcl/Tkb]

Brent Welch, *Practical Programming in Tcl and Tk*, Draft, Xerox-PARC, 3333 Coyote Hill Road, Palo Alto CA, 94304, USA, August 1994

[MIME/Safe-Tcl]

Nathaniel Borenstein, Marshall T. Rose, *MIME Extensions for Mail-Enabled Applications: application/Safe-Tcl and multipart/enabled-mail*, November 1993

[Safe-Tcl]

Nathaniel Borenstein, Marshall T. Rose, *(Safe-Tcl) swish man page*, October 1993

[Email/Safe-Tcla]

Nathaniel Borenstein, *EMail With A Mind of Its Own: The Safe-Tcl Language for Enabled Mail*, First Virtual Holdings, Inc., 25 Washington Avenue, Morristown, NJ 07960, USA

[Email/Safe-Tclb]

Nathaniel Borenstein, Marshall T. Rose, *A Model for Enabled Mail*, July 1994

[MailingLista]

Software Agents Mailing List, agents@eng.sun.com

[MailingListb]

Safe-Tcl Mailing List, safe-tcl@CS.UTK.EDU

[Madsen95]

Mark Madsen, *Meta-Information Management Overview*, **1414.01**, APM Ltd., Cambridge UK, 1995

[Edwards]

Nigel Edwards, *A CORBA IDL Compiler for the World Wide Web*, **1419.01**, APM Ltd., Cambridge UK, 1995

[ANSA]

ANSA, 1994-1996 ANSA Workplan, 1275.02, APM Ltd., Cambridge UK, September 1994

[ReesEdwards]

Owen Rees, Nigel Edwards, An Overview of the Information Services Framework, 1306.00.09, APM Ltd., Cambridge UK, February 1995

[Deschrevel]

Jean-Pierre Deschrevel, The NASA Model for Trading and Federation, 1005.01, APM Ltd., Cambridge UK

[Zimmermann]

Philip Zimmermann, PGP User's Guide, Volume 1: Essential Topics, Boulder Software Engineering, 3021 Eleventh Street, Boulder, Colorado 80304, USA, October 1994

