

LE1-1930

Implementation and System Integration

Deliverables D 6.1 & D 6.2 & D 6.3 - public version

**This is the public version of deliverables D6.1,6,2&6.3 (confidential)**

1	INTRODUCTION.....	2
1.1	SCOPE.....	2
1.2	OUTLINE.....	2
1.3	OBJECTIVES	2
1.4	DEMONSTRATORS.....	3
1.5	TECHNICAL WORK	3
1.6	SUMMARY AND CONCLUSIONS	3
2	SYSTEM DESCRIPTION	4
2.1	OVERVIEW OF CAVE TEST SYSTEMS	4
2.2	ARCHITECTURE REVIEW	4
2.3	BASIC SV SYSTEM CAPABILITIES	6
2.4	STANDARDS	7
3	CLIENT/SERVER COMPONENTS OF SYSTEM.....	7
3.1	SPEECHSERVER	7
3.1.1	<i>Basic details</i>	7
3.1.2	<i>Software</i>	7
3.1.3	<i>Speech technology</i>	8
3.2	CAVE NT CLIENT.....	8
3.3	INTEGRATION.....	9
3.3.1	<i>Speech Bus</i>	9
3.3.2	<i>Data Bus</i>	9
4	SS1 FUNCTIONS AND DLL.....	10
4.1	SS1 FUNCTIONS	10
4.1.1	<i>Introduction</i>	10
4.1.2	<i>Description of Individual SS1 Function calls</i>	10
4.2	DLL	16
4.2.1	<i>Introduction</i>	16
5	APPLICATION GENERATOR	16
5.1	ENVIRONMENT.....	16
5.2	BANKING APPLICATION	17
5.2.1	<i>System overview</i>	17
5.2.2	<i>System functionality</i>	17
5.2.3	<i>Extension for CAVE field test</i>	17
5.3	CALLING CARD APPLICATION.....	18
5.3.1	<i>Motivation / Overview</i>	18
5.3.2	<i>System functionality</i>	18
6	APPENDIX A: SS1 PRODUCT DESCRIPTION.....	20
	SPEECH RECOGNITION.....	24

This work has been supported by:

- the Telematics programme of the European Union
- the Office Federal de l'Education et de la Science (in Switzerland)



1 Introduction

The objectives, activities and results of the workpackage 6 efforts have been described in the confidential WP6 deliverable. This document is the public version of this deliverable.

1.1 Scope

This document can be thought of as incorporating the separate deliverables

- D6.1 Real-Time implementation of SV in standard voice boards.
- D6.2 Implementation of the User Interface
- D6.3 Integration of User Interface into Pilot application.

In workpackage 7 deliverables the tests and results including related discussions will be presented.

1.2 Outline

The remainder of this document is structured as follows.

- Firstly an overall management report of the workpackage and the activities performed in it.
- then a condensed description of the basic architecture of the ASR/SV server and telephony-enabled client of the CAVE test systems
- including their integration through TCP/IP and the Dialogic SCBus.
- a complete description of the CAVE API and of the DLL main functions for application developers.
- details of the common software platform on which applications are built
- Appendix A provides a brief technical description of the Speech Server design and component specification.

1.3 Objectives

The objectives of the demonstrator workpackage were to:

- develop a real-time system to demonstrate the technology
- investigate system, hardware, software issues required to implement it
- support a field trial to evaluate speaker verification performance, user interface, customer reaction
- identify capabilities and limitations of speaker verification in real applications
- demonstrate capability of system to business units

Each of these objectives was substantially achieved in the course of the CAVE project.

The performance of the field trial and detailed analysis of results are the subject of workpackage 7. It would be helpful to acoustically check and transcribe the speech data prior to performing further analysis.



1.4 Demonstrators

Two main applications were selected to demonstrate and investigate use of Speaker Verification for the CAVE project:

- Dutch calling card demonstration
- Swiss German telephone banking service

The project aimed to implement the demonstrators in three stages, in order to:

- provide prototype systems for early feedback into design and service provision
- incorporate more advanced SV algorithms as they were developed in WP4 during the project lifecycle
- incorporate system improvements (functional and technical, including human factors) as a result of experience gained with initial demonstrators
- support later planning and design of systems for in-service integration

Ongoing technical and other developments will permit evaluation of the applications and services in a larger field trial scenario. This decision was argued on following issues:

1. time needed for technical development of the SpeechServer,
2. optimisation of the real-time verification algorithms,
3. integration with the application systems,
4. and partner-specific issues related to approvals needed for interfacing to actual services.

1.5 Technical work

The Technical Specification (workpackage 3) and existing Vocalis ASR was starting point for speechserver preparation. The speechserver (also facilitating the SV modules) has subsequently progressively modified to include the most important WP4 recommendations and important partner feedback. These iterations included the following main items:

- installation and standalone test procedures
- speech recognition improvements
- endpointing algorithm improvements
- Dutch and German whole-word and subword models
- dynamic loading of grammars
- SCxbus hardware and software integration
- speaker verification algorithm enhancements

Application building and the integration of all modules (telephony, application, ASR and SV) was a major task for the userpartners involved in The Netherland and in Switzerland. This process has been intensively supported by Vocalis (provider of the speechserver) and the WP 4 partners (SV algorithms).

1.6 Summary and conclusions

The main objectives of CAVE were substantially achieved during the course of the project.

Both demonstrators were successfully realised in terms of system implementation and integration, and resulted in adequately robust test systems. These finally implemented the main components of the CAVE SV algorithms in realistic real-time applications, and provided a basis for evaluation of the technology in these application domains.

Several issues emerged as a result of this work, which require special attention, and may be expected to be the subject of future research and development on the PICASSO follow-on project.

1. The provision of closely integrated, accurate speech recognition is essential for successful deployment of speaker verification applications. This is not simply a matter of the overall service performance.



2. Recognition approaches and accuracy intrinsically affect speaker verification and impostor algorithms. This includes the detailed procedures for incorporation of silence, world and filler models in the scoring algorithm,
3. Rapid recognition response is needed for good dialogue control, but has to be balanced against accuracy of recognition, especially when lengthy verification utterances are used which may have longer inter-word pause durations
4. Thresholding procedures and algorithms are very important and require continued detailed investigation
5. The SV performance is very dependent upon appropriate quantities and types of training material. This has implications for service provision in terms of human factors and user expectations. This also is an important area for continued future research.
6. The API may require to be redeveloped in future work in the light of emerging ECTF standards.
7. The demonstrators implemented a relatively simple database management system based on the Unix file system. Extension toward realistically sized applications will require significant engineering of the database functions for maintaining customer records, billing control, speaker templates, speech data etc. This is a major task requiring careful specification, and is critically linked to the required number of users/lines, choice of operating systems and available DBMS products.
8. The demonstrators implemented a relatively simple resource manager which mapped one incoming line to a voice resource. This was completely adequate for this stage of the work which was concerned with building a realistic real-time SV service for prototyping and human factors research. A more advanced RM would be needed for a full-specification commercial product or system.

2 System description

2.1 Overview of CAVE test systems

The general structure of the CAVE test systems should be refreshingly familiar to any modern IT professional -- the "client-server" route has been taken. This approach has been used in CAVE as follows; the servers offer speech recognition and speaker verification functionality to any machine networked to them via TCP/IP (with the optional use of the Dialogic SCBus for guaranteed real-time networking of digitised speech data). This approach has a number of technical advantages, and made it possible to effectively partition and distribute the development tasks among the project partners.

Each server offers its resources on some number of independent "channels", thereby allowing for multiple simultaneous connections to the server. The clients to this server are telephony-equipped PCs, each running an application which answers the telephone to incoming callers and maintains the entire dialogue with the user, only securing and using a speech server resource when necessary. The scalability of TCP/IP (and of the Dialogic SCBus if this is deployed) would allow for a simple test system based on a single client and a single server to grow to meet additional demands, so long as the dynamic resource allocation code is written with this in mind. The existing technology of Vocalis has been selected to serve as a suitable platform for the SV algorithms developed on the research platform.

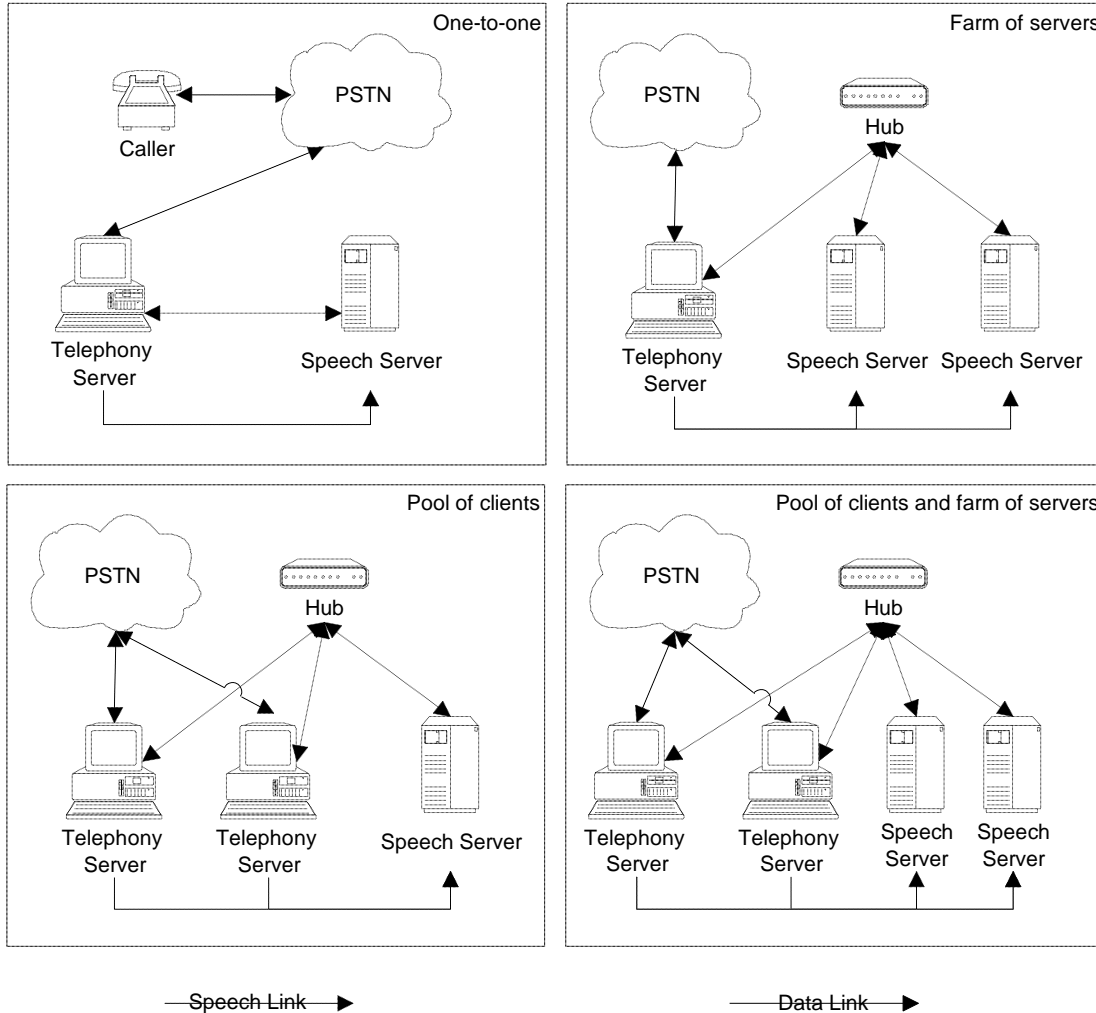
2.2 Architecture review

Based on the client/server model, the Speech Server architecture distributes the basic voice processing tasks among specialised systems to achieve better overall performance and resource utilization. This architecture can also be extended to enable one or more servers to act as Intelligent Peripherals in telephone network applications.

In a typical Speech Server configuration, Telephony Servers are client systems that hold telephony hardware and run the application itself. They are networked to Speech Servers, server systems equipped with speech recognition and/or SV hardware and software (globally called "recognisers" or "resources").



There can be a pool of clients and only one server, or a farm of servers and one client, or even a pool of clients and a farm of servers. Such details are totally transparent to the application. Simple configurations with one Telephony Server linked to a single Speech Server are also supported.



Clients don't necessarily have to be networked together but need to be linked to every single server using a TCP/IP-UDP/IP LAN or WAN.

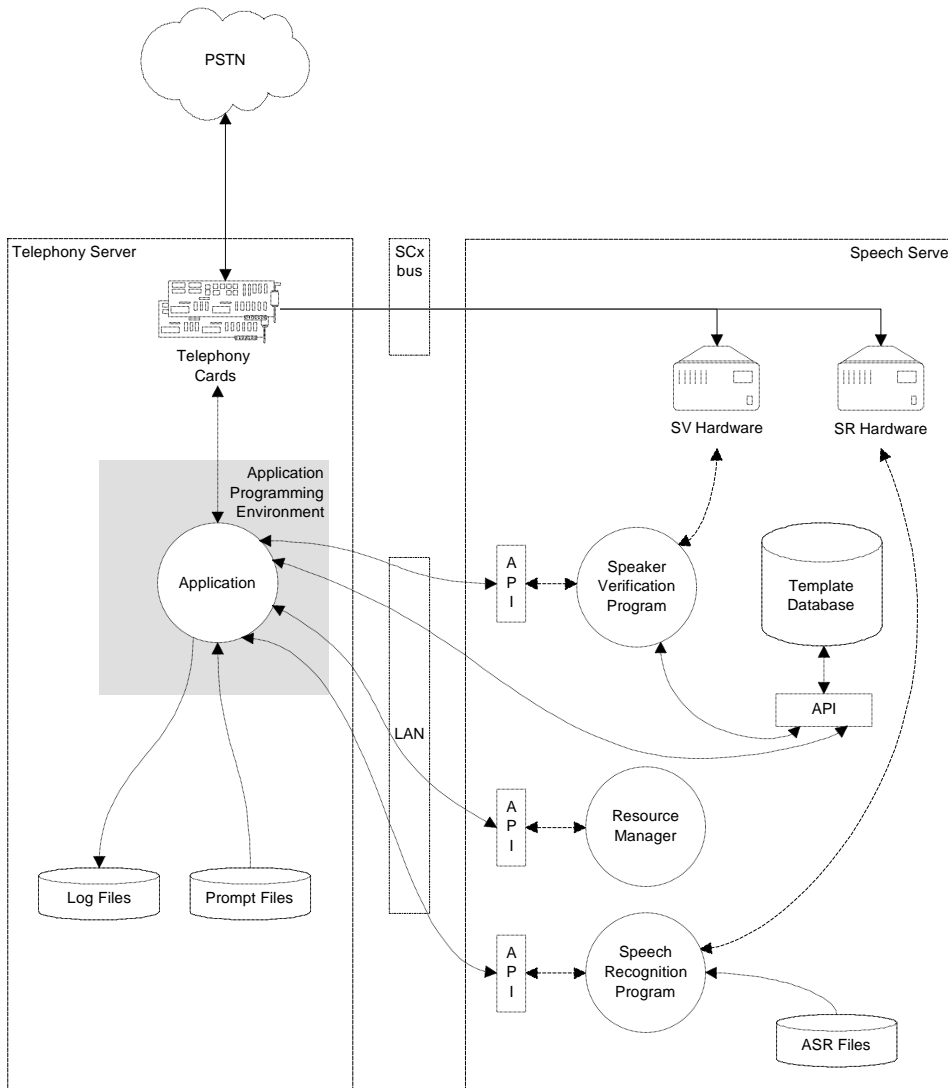
Although data and commands are transmitted through the LAN, the recogniser gets its speech data for recognition purposes from an external speech bus: the SCxbus.

As a result, clients and servers are doubly linked: by a data-only TCP/IP-UDP/IP network and also by an external SCxbus speech bus.

The Speech Server API supports the latest recogniser hardware/software available at the time of commencement of the project. Recognition resources are controlled remotely, from a voice processing application running on a Telephony Server, by calling Speech Server API functions. Calls are issued from the Telephony Server to the appropriate Speech Server system where the function's actual code is executed.

Function results are returned across the data

Speech Servers can in principle be integrated with available third party speech recognition resources and might well coexist with them.



2.3 Basic SV System Capabilities

The Speech Server capabilities are outlined below:

- support for speech recognition
- support for text dependent SV
- support for text independent SV
- support for text prompted SV is not required by CAVE

The initial systems were based on existing technology as far as possible to ensure robust, real-time operation, with resources devoted to integration of the SV components and basic system support functions. Once this was realised, a platform was available for development of the more sophisticated features of SV and ASR ultimately recognised as being needed to meet the project goals.



2.4 Standards

The Speech Server architecture is based on open industry standards: UNIX operating system, TCP/IP-UDP/IP protocols, resilient industrial PC platform, Ethernet data networking, SCxbus speech bus (Dialogic SCSA hardware), C language software.

This ensures that it meets the requirements for a large class of industrial applications.

SCSA (ECTF) had been specified in the Technical Annex as the intended standard to which CAVE project developments should adhere if possible. At present the bus and hardware standards have been well-defined and are adopted by a number of CTI vendors and manufacturers, including Dialogic who provide a wide range of industry standard SCSA-compatible boards suitable for project use and approved for installation in many countries. These were adopted in the system development and integration work.

3 Client/Server components of system

3.1 SpeechServer

3.1.1 Basic details

The adoption of a client-server architecture allows the efficient sharing of expensive speech recognition and speaker verification resources between clients in multiple different configurations. However, the relatively low call volumes of the CAVE field test systems (that is, hundreds of calls per day rather than thousands), made it only necessary to deploy a basic one-client-one-server configuration..

The ASR/SV server component of the CAVE test system is simply known as "Speech Server One" (hereafter abbreviated to SS1) and is based on a standard PC chassis with a 200MHz Pentium Pro processor at its heart.

At each site it was found unnecessary to follow the initial plan of isolating the CAVE test system onto its own LAN segment. System upgrades proved easier to deliver by Internet e-mail.

Each SS1 was based on SCO Openserver 5, a variant of the UNIX operating system (OS). Openserver 5 is a robust OS and a proven technology for robust industrial applications; it is one of the only commercial UNIX variants for Intel PCs.

3.1.2 Software

The server provides a speech recognition and speaker verification engine as well as speaker verification enrolment database engine.

The CAVE server engine can be started up or shut down from the UNIX command line. Command-line operands control the language for which speaker-independent models are loaded and the level to which server-side logging information will be output. This helps the application developer to debug client-side applications. The server was shipped with a diagnostic application which, through the use of TCP loopback, allowed the testing of all important API functions. The source code also served as the basis for the implementation of the client-side "wrapper functions" which are described in section 5.

A proprietary "CAVE API" encapsulates the functionality of the SS1 and offers it to clients on the same TCP/IP network. The use of the Dialogic SCBus for real-time speech data networking is optional but recommended; API functions are available which allow speech to be downloaded to the server over the TCP/IP network.



3.1.3 Speech technology

The Vocalis speech server uses state-of-the-art HMM speech technology and conventional approaches for recogniser engine.

The server software mainly consists of three parts,

- front-end processing,
- speech recognition and
- speaker verification.

For several reasons we have to omit on full descriptions here.

3.1.3.1 Speaker Verification

The CAVE verification techniques have been integrated within the SS1 speech server.

The implementation has compromised three elements, suggestions from CAVE WP4 off-line experiments, real-time requirements and the existing recogniser implementation.

The verification algorithm is based on digits vocabulary and each word is treated as a unit for speaker modelling.

The speaker template consists of a number of speaker-dependent digit models. The server provides a database to store speaker templates. The application can access the speaker template according to a given user account. The server also provides a database utility for enrolment speech data. A multi-session enrolment can be easily implemented at the application level within system-defined limits

The verification procedure uses speech recognition digit models as the world model for score normalisation. There are two sets of digit model for speech recognition, male and female. For each speaker verification the server activates both male and female models.

3.1.3.2 Real-time integration of CAVE WP4 SV algorithms

The developments within the 'algorithm' workpackage 4 were progressively integrated within the SS1 as far as possible. However there were some constraints upon what could be achieved within the lifetime of the project. These were dictated by

- a) Hardware: computational processing power and memory size limits
- b) Software: existing real-time recognition and modelling software design features within Vocalis products
- c) Human factors: real-time performance requirements for rapid verification and recognition
- d) Resources available to re-implement major design differences within the project, especially considering the unforeseen effort to complete technical developments for the integrated systems

In the later stages of the project, experiments performed within workpackage 4 indicated that the implemented parameterisation resulted in an overall verification performance loss of a factor of approximately three when the same data was tested using workpackage 4 HTK software tools.

3.2 CAVE NT Client

The additional PC chassis in the CAVE test system can either be thought of as a client to the ASR/SV server, or as a "telephony server" in its own right, with human callers as the ultimate clients. For the purposes of this discussion a "telephony server" will be referred to here as the "client".

The CAVE client chassis is another PC -based on an Intel Pentium Pro processor and runs the Microsoft Windows NT 4.0 operating system. This was the most appropriate operating system for a number of reasons:



- the use of cross-platform networking standards such as TCP/IP and the SCBus did not constrain the use of the same operating system for client and server
- NT is fast becoming the "platform of choice" for computer-telephony systems of all kinds on account of its graphical user-interface and stability -- as such, the range of CT development environments and other applications is already large and increasing rapidly.
- The ease of porting Unix software to NT meant that drivers for the Dialogic hardware we would need to install in the client were already available.

The Dialogic D300SC-E1 ISDN Primary Rate board has been used both in The Netherlands and in Switzerland.

The ISDN system software was shipped with a number of configuration files corresponding to various European and worldwide ISDN protocols. For the Swiss installation it was necessary to modify the EuroISDN protocol file in order for the ISDN board to accept calls

3.3 Integration

3.3.1 Speech Bus

The Dialogic SCBus was used to facilitate communication of speech between the client and the server.

The SCBus is Dialogic's real-time time-division multiplexed (TDM) digital speech bus and is supported by the vast majority of its current range of products. It enables full-duplex speech data communication between all the SCBus-compliant Dialogic resources in a system. Each Dialogic resource is allocated, at system start-up, a fixed SCBus time-slot to which the transmission channel of the resource is connected. The reception channel of the resource is connected to the same time-slot so the default behaviour of the resource is loop-back. To enable half-duplex communication between two resources A and B, it is only necessary to connect the reception channel of resource A to the transmission channel of resource B. (Full-duplex communication can then be enabled by connecting the reception channel of B to the transmission channel of A).

The SCBus solution was deployed into the CAVE test systems by installing SCx/160 cards into both the client and the server and then adding a Dialogic D41/ESC card into the server; this extra card was necessary as there is no way to read "raw" information off the SCX/160 card.

The only modification necessary to the system software was to add an API call to allow the client process to tell the server to start listening to speech on a particular SCBus time slot rather than to start downloading speech from a previously-opened TCP socket.

Since the assignment of SCBus time-slots to channels on the telephony server is always the same, then the identity of the time-slot can always be obtained so long as the appgen provides a way of obtaining the number of the Dialogic resource used every time an incoming call is answered. No specific support for SCBus functionality was required from the appgen.

3.3.2 Data Bus

Remote Procedure Calls (RPCs) over TCP/IP or UDP/IP are used to trigger server-side functions from the client. Due to pressure of time and the need to privatise the upgrade of the server's speaker verification algorithms, this transport mechanism was not eventually upgraded to full RPC for the mimic and field tests. However it is likely that an upgraded version of the SS1 for the forthcoming CAVE "sequel" project, PICASSO, will also feature RPC.



4 SS1 Functions and DLL

4.1 SS1 Functions

4.1.1 Introduction

The SS1 provides a resource of speech recognition and speaker verification engine as well as speaker verification enrolment database engine. It is delivered with a set of "C" function calls on client side. This allows the system developers to fully access the server resources from the client side. The functions are considered as very basic blocks for building recognition and verification tasks. They provide a basic layer for the system developers to build further level of API on client side with other client side software.

4.1.2 Description of Individual SS1 Function calls

FUNCTION PROTOTYPE:

```
int sv_open_channel(char *server_name, short server_port_number);
```

Parameters:

char *server_name -- Dotted IP address of server in either numerical or alphabetical format.

short server_port_number -- TCP port on server to which to connect.

Description:

This function opens a connection to the specified SS1 server on the specified port number, thereby "reserving" a recognition/verification resource, and returns a socket ID used for all further communication with the server on this port. This function uses standard Winsock TCP/IP function calls. The current version of the SS1 server software provides n multiple recognition and verification resources simultaneously by the simple expedient of running n distinct, self-contained server processes, each one listening out on a different TCP port on the server. Consequently, this places the burden of resource management on the client. It is assumed that when sv_open_channel is called, the port being connected to is known to be available. See later for more details of the client-side server resource management routine implemented in the CAVE field test systems.

Return value:

Success (0), or any one of a number of non-zero values indicating that an error has occurred. These can then be acted upon by the calling process.

FUNCTION PROTOTYPE:

```
int sv_close_channel(int cd);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call.

Description:

This function closes communication with the server on the current socket. In fact, this function is identical to close(cd).

Return value:

Success (0), or any one of a number of non-zero values indicating that an error has occurred. These can then be acted upon by the calling process.

**FUNCTION PROTOTYPE:**

```
int sv_open_recogniser(int cd);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call.

Description:

This function allocates necessary resources on the server for current recogniser channel.

Return value:

0 for success or -1 to indicate that a TCP error occurred.

FUNCTION PROTOTYPE:

```
int sv_close_recogniser(int cd);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call.

Description:

This function frees the server resources which has been allocated for this recogniser channel by previous sv_open_recogniser call.

Return value:

0 for success or -1 to indicate that a TCP error occurred.

FUNCTION PROTOTYPE:

```
int sv_download_speech(int cd, char *speech, int nbyte);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call.

char *speech -- pointer to buffer of speech held in memory.

int nbyte -- number of bytes to download

Description:

This function downloads speech to the server via TCP. It is not used in either CAVE field test system.

Return value:

0 for success or -1 to indicate that a TCP error occurred.

FUNCTION PROTOTYPE:

```
int sv_download_grammar(int cd, FSGRAM *fsg);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call.

FSGRAM *fsg -- pointer to data structure for storing grammar.

Description:

This function downloads a grammar, pre-loaded into memory, to the server. The grammar is stored in one of the available "slots" where it remains until it is overwritten or the server is shut down. Due to the time-lag associated with this function, it is for the most part used only to download grammars to the server at boot time.

Return value:

0 for success or -1 to indicate that a TCP error occurred.

**FUNCTION PROTOTYPE:**

```
int sv_verify_speech(int cd, VERIFY_REQ *req);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call.
VERIFY_REQ *req -- pointer to verification request data structure.

Description:

This function instigates a server-side verification process. All the necessary parameters for this verification are contained in the VERIFY_REQ structure. The function returns immediately; the server can subsequently be polled on the same TCP socket using the sv_status API function until its status indicates that the verification has been performed; the result of the verification can then be received using the sv_retrieve_verify_result API call.

Return value:

0 for success or -1 to indicate that a TCP error occurred.

FUNCTION PROTOTYPE:

```
int sv_recognise_speech(int cd, RECOGNISE_REQ *req);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call.
RECOGNISE_REQ *req -- pointer to recognition request data structure.

Description:

This function instigates a server-side recognition process. As for sv_verify_speech, all the required parameters are contained in the accompanying data structure, and the server can be polled at intervals until completion is indicated.

Return value:

0 for success or -1 to indicate that a TCP error occurred.

FUNCTION PROTOTYPE:

```
int sv_record_speech(int cd, RECORD_REQ *req);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call.
RECORD_REQ *req -- pointer to record request data structure.

Description:

This function works identically to sv_recognise_speech except that it instigates a general purpose "record" function on the server. See the earlier description of the RECORD_REQ structure for more details about this function.

Return value:

0 for success or -1 to indicate that a TCP error occurred.

**FUNCTION PROTOTYPE:**

```
int sv_build_template(int cd, MODEL_REQ *req);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call.
MODEL_REQ *req -- pointer to modelling-request data structure.

Description:

This function instigates a server-side enrolment process, in which a speaker-dependent model is created for the speaker whose user_id number is indicated in the MODEL_REQ data structure. As for the above functions, the call returns immediately and must eventually be followed up by a call to sv_retrieve_model_result.

Return value:

0 for success or -1 for a TCP error.

FUNCTION PROTOTYPE:

```
int sv_status(int cd);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call

Description:

This function, which can be called at any time, returns the status of a recognition/verification resource on the server. This is particularly crucial in the CAVE field test systems where it is not possible to issue a blocking (synchronous) TCP call to retrieve a result from the server immediately after the original request. The server returns a code of SV_BUSY only when it is performing speech recognition or speaker verification on speech obtained from a server-side file, from the TCP network or over the SCBus, and returns SV_IDLE at all other times.

Return value:

SV_BUSY, SV_IDLE.

FUNCTION PROTOTYPE:

```
int sv_retrieve_verify_result(int cd, VERIFY_RES *res);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call
VERIFY_RES *res -- pointer to structure for verification result.

Description:

This function is used to retrieve the result of a verification process on the server. The function can only be called after a sv_verify_speech API call and the behaviour of the server is undefined if it is called at any other time. A non-zero value in status indicates that an error occurred during verification. This error should be acknowledged by the parent process.

Return value:

0 for success or -1 for a TCP error.

**FUNCTION PROTOTYPE:**

```
int sv_retrieve_recognise_result(int cd, RECOGNISE_RES *res);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call
RECOGNITION_RES *res -- pointer to structure for recognition result.

Description:

This function is used to retrieve the result of a recognition process on the server. As for the previous case, the function can only be called after a corresponding sv_recognise_speech call, and a non-zero value in status indicates that a recognition error occurred.

Return value:

0 for success or -1 for a TCP error.

FUNCTION PROTOTYPE:

```
int sv_retrieve_record_result(int cd, RECORD_RES *res);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call
RECORD_RES *res -- pointer to structure for recording result.

Description:

This function is used to retrieve the result of a recording process on the server. It works identically to the previous functions.

Return value:

0 for success or -1 for a TCP error.

FUNCTION PROTOTYPE

```
int sv_retrieve_model_result(int cd, MODEL_RES *res);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call
MODEL_RES *res -- pointer to structure for modelling result.

Description:

This function is used to retrieve the result of a speaker modelling process on the server and can only be called after an sv_build_template call. It works identically to the previous functions.

Return value:

0 for success or -1 for a TCP error.

FUNCTION PROTOTYPE:

```
int sv_add_user(int cd, USER_INFO *user);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call
USER_INFO *user -- pointer to structure containing new user details.

Description:

This function adds a new user to the server-side user database.

Return value:

SV_USER_CREATED, indicating that the function completed successfully, or
SV_USER_ALREADY_EXIST, indicating that the server-side user database
already contains this user.



FUNCTION PROTOTYPE:

```
int sv_delete_user(int cd, USER_INFO *user);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call
USER_INFO *user -- pointer to structure containing user details.

Description:

This function deletes an existing user from the server-side user database.
This function is currently not used by the CAVE field test systems.

Return value:

SV_USER_DELETED, indicating that the function completed successfully, or
SV_USER_NOT_FOUND, indicating that the server-side user database did not
contain this user.

FUNCTION PROTOTYPE:

```
int sv_rename_user_file(int cd, USER_FILE_INFO *newfile, USER_FILE_INFO  
*oldfile);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call
USER_FILE_INFO *newfile, *oldfile -- pointers to structures indicating
current and new name of a server-side speech file.

Description:

This function renames a server-side speech file, allowing it to be moved between different user directories as
well as changing the filename.

Return value:

0 for success, non-zero to indicate that the function could not be
completed (i.e. if the old file or new directory does not exist).

FUNCTION PROTOTYPE:

```
int sv_listen_to_scbus(int cd, long *scts);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call
long *scts -- pointer to long int containing number of SCBus time slot

Description:

This function instructs the server to connect one of its own Dialogic telephony resources to an SCBus time
slot corresponding to a client-side voice board resource. This allows the server to listen to speech in
real-time, thereby speeding up recognition compared to downloading speech over the network using TCP/IP.
This function can only be called after a call to sv_recognise_speech, sv_record_speech or sv_verify_speech,
and must eventually be followed by a call to sv_unlisten_to_scbus.

Return value:

0 for success or -1 for a TCP error.

**FUNCTION PROTOTYPE:**

```
int sv_unlisten_to_scbus(int cd);
```

Parameters:

int cd -- TCP socket returned by previous sv_open_channel call

Description:

This function instructs the server to disconnect a Dialogic telephony resource from an SCBus time slot corresponding to a client-side voice board resource.

This function can only be called after a call to sv_listen_to_scbus, and should only be used when it is known that the corresponding verification, recognition or recording process has concluded.

Return value:

0 for success or -1 for a TCP error.

4.2 DLL

4.2.1 Introduction

It is one thing to have a telephony application environment and a "C" language speaker verification API; it is another thing to have the one able to make calls to and receive results from the other. Knowing the API function prototypes and the application environment's ability to call C functions, it was necessary to design and program software specifically to bridge between the two elements. This software was created in the form of a standard Windows Dynamic-Linked Library (DLL) and has been used, in various versions, in all of the tests run at both CAVE test sites. A complete description of the CAVE DLL is depending from the specific application development environment used and is outside the scope of this public deliverable.

5 Application Generator

5.1 Environment

The decision was taken early on in the CAVE project to use an appropriate telephony application development environment (otherwise known as application generators or appgens) to develop the project's field test systems.

The purpose of appgens is to allow developers to leverage the functionality of their telephony hardware and APIs to develop computer telephony (CT) systems without having to program at the API level themselves in C, C++ or some other language.

Once such appgen is called Ring Open Systems Architecture from the European computer telephony company Ring!. It provides a cross-platform multi-threaded telephony development environment based on using icons to represent CT building blocks. With the help of those building blocks (or more precisely the associated CT functions) the flow in a CT application can be controlled very efficiently.

It stands out by virtue of Ring!Talk, an entire structured programming language based on Modula-2 with CT-specific extensions, in which additional procedures can be written, then compiled into the application and freely mixed with functions from the standard "palette" of icons.

In addition, ROSA features a database engine licensed from Borland Inc., access to whose API is implemented in Ring!Talk; this allows for easy integration between a CT application and SQL, Borland (now Corel) Paradox or dBase database files. The telephone board selected in CAVE is also supported.



5.2 Banking application

5.2.1 System overview

The banking test system is a full-featured telephone banking application which uses CAVE speaker verification technology to secure access to customer information. All input and output is handled using German speech. Users can find out balances or transaction histories on any one of four accounts and move funds between their own accounts. The banking test system was run using a stand-alone database of simulated account information; this, in conjunction with a competitive element was found to be the best way to attract experimental subjects to the field test.

5.2.2 System functionality

The first contact the user has with the system is by dialling the telephone number provided for enrolment. The user enters his or her customer number and PIN code for the system by TouchTone keypresses. When this information is checked in the database, the user is welcomed to the enrolment session. Subsequently in this session, recordings are made of the user speaking his account number and PIN code, and then six so-called "prompted digit sequences" (PDS) of four digits each. These sequences are specially designed so that each digit is spoken in a number of different contexts. The system speaks each digit sequence to the user and asks him or her to repeat the sequence at the sound of a tone. All enrolment speech is recognised and dumped in the appropriate server-side directory .

Assuming a successful enrolment, subsequent attempts to access the system proceed as follows.

The user is greeted and speaks his or her nine-digit account number. This is recognised and compared with a known list of valid account numbers.

When a valid account number and PIN code combination has been obtained, speaker verification is carried out on the original account number utterance and the resulting score compared with the speaker-dependent threshold for the user, which is retrieved from the database.

If the score is higher than the threshold, then the user's identity claim is accepted, otherwise it is rejected. Database fields for the number of accesses or rejections are updated accordingly. If the user is accepted, the appropriate account information is loaded and the system asks the user for the desired transaction.

Database integration is handled through the Borland Database Engine (BDE) API, to which the Ring!Talk programming language offers access through a set of database access and modification functions.

5.2.3 Extension for CAVE field test

As stated before, the banking test faced a major problem in how to attract and maintain callers to a system which offered no obvious The solution was to introduce a competitive element to the system, as follows:

- when users first enrolled into the system, a daily routine was activated in which an amount of "cash" was placed into one of the three "savings accounts" belonging to that user. This amount would be erased from the account twenty-four hours later unless it was moved to the user's "current account". Therefore, the task for users was to dial in on a daily basis, search for the day's free gift and transfer it to the current account, the only account on which cash could be accumulated. At the end of the trial, a prize-draw took place, in which each user's chance of winning was proportional to the amount of money accumulated in the current account, up to a maximum of 5000 Swiss Francs. In addition, callers were given phonecards to compensate them for the cost of calls made to the system from personal telephones. The code to implement the daily dispersal of cash amounts was added to the nightly "housekeeping" code.



5.3 Calling card application

5.3.1 Motivation / Overview

The calling card field test application is a voice based version of an operational service.

This service allows customers to call from all over the world using toll free numbers and paying a bimonthly bill. Every client is identified by a 14 digit card number and a 4 digit PIN code.

Thanks its billing structure an automated call handling is attractive for the clients and the service provider.

The present DTMF service causes substantial security problems. For instance, it is known that many hotel PBXs store the DTMF sequences. For criminals who can gain access to an hotel PBX it is simple to retrieve card number and PIN code combinations, and sell these, e.g. to foreigners to make expensive home calls for a flat rate. Other means for obtaining card numbers and PIN codes are 'shoulder surfing' in airports and train stations, while almost all memory phones contain enough storage locations to hold the toll free country direct number, the card number and the PIN code; thus, pressing the redial button is enough to retrieve the information. The operator based version of the service has equally large security problems. The card number and PIN can be overheard, but the most serious security threat comes from temporary staff working as operators. The customer must give the card number and PIN code to the operator, who can make mental or written copies of the information.

These security holes can be avoided by using a biometric based verification method like speaker verification. Contrary to a knowledge-based authentication method like the PIN, the user does not have to remember any secret digit sequences. Thus a voice driven service that uses ASR for entering card number and phone number to dial, in addition to speaker verification to authenticate the identity of the caller, has advantages for both clients and the service operator.

The main goals of the field test of a voice driven version of the basic functionality of the calling card service are:

1. to test the technology with users who are frequent users of the service, but who are naive with respect to speech technology, and
2. to obtain subjective evaluations of the voice service by this group.

To motivate callers to participate in the field test, they got telephone calls for free, where every call is limited to 10 minutes. This is done not only to limit the costs, but also to keep the service available for all participants, because only two operational lines are available in the present implementation of the current Speech Server.

5.3.2 System functionality

The system set-up of the field test demonstrator is a close copy of the basic functionality of the existing calling card service.

Customers get a 14 digit ISO standard card number and a 4 digit PIN code. Before (s)he can use the service, each client has to enrol her/himself. This is done by speaking the card number 8 times (in the form of a digit string) in response to a system prompt. The first time the card number is prompted the system explicitly asks to respond in the form of a digit string; subsequent prompts omit this lengthy formulation.

In between each pair of card numbers the client must repeat a prompted 4 digit random sequence. The prompted digits have several goals, the most important of which is to elicit some variation in the production of the card numbers. For the enrolment a toll free number is used. The enrolment procedure is secured by the 4 digit PIN code: each client must enter the PIN -via DTMF- to obtain access to the enrolment procedure. Although research in CAVE has shown that superior client models are obtained if enrolment is done in at least two different sessions in order to capture a minimum of speaker, line and microphone variability, the current implementation uses only one enrolment session. This is so because of human factors research, which



showed that customers and service providers alike are reluctant to accept more than a single, short enrolment session. The model of the clients voice is created right after the call, and the service is ready for use in less than two minutes after the enrolment call. The whole enrolment call takes about four minutes.

To access the system, the customer dials another toll free number and (s)he is asked to speak the card number. The system looks up the recognised number in a data base, and depending on which of the two groups the user belongs to, (s)he is prompted for the PIN code or not. This allows to compare two levels of security: speaker verification only and speaker verification together with a PIN code. Both applications then ask the user to hold on. During that time the speaker verification is done, comparing the currently recorded card number utterance with the model of the claimed speaker.

If the verification returns a score below the threshold estimated during enrolment, the speaker is accepted and is given access to the service. The system then prompts for the desired telephone number, that must be spoken as a (connected) digit string. The application then generates an outgoing call and connects the two lines. Once accepted, the user has the possibility to place multiple calls, but the total duration of the calls may not exceed ten minutes. If a call is not answered, or if the called line is busy, the system advises the caller, and then prompts for another number to call.

In case the verification score is higher than the threshold, the system asks the user to repeat the card number. If the second attempt to verify the claimant's identity fails, the caller is refused access. The system then asks the caller to identify her/himself (in this test system) as a true client or an impostor, by means of a yes/no answer to the question 'Did you call as impostor?'. This information is necessary to establish the number of false rejects.



6 Appendix A: SS1 product description

CAVE Speech Server I

Technical Description

7.1



Introduction

The Vocalis CAVE Speech Server (SS1) is an interactive voice response platform which delivers a range of speech technologies primarily to support speaker verification pilot and demonstration systems. It has been specifically designed to support real-time application development, experimentation and evaluation of speaker verification technology within the CAVE project. The SS1 incorporates Vocalis proprietary speaker independent speech recognition and CAVE speaker verification algorithms, as well as providing some general functions such as speech storage. (In principle it could also provide a full application run-time and development environment, supporting DTMF input, audio playback and telephony functions as a standalone IVR system, but this has not been required).

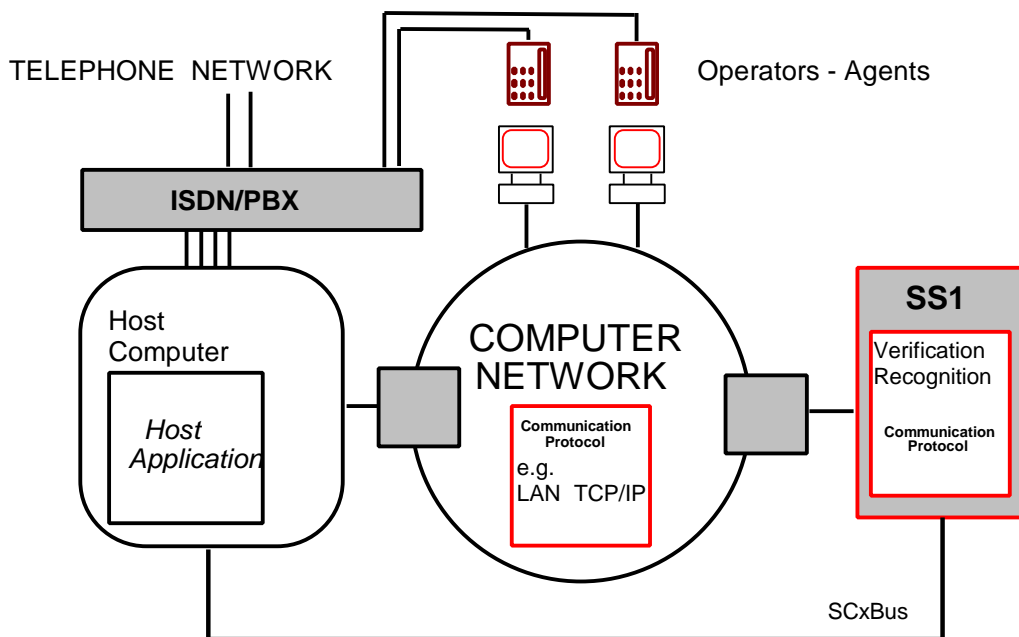
The speech server platform is designed to be used either singly or in multiple server configurations to provide scalability to large systems, and will interface with a number of different application host environments. The host machines will primarily implement all the dialogue and telephony functions of the application, including all supporting administration functions, leaving the server to provide only the speech verification and recognition functions. This architecture permits one basic server to be integrated with a wide range of applications and host environments.

The delivery platform is principally designed as a relatively low-cost unit suitable for small installations or research prototyping, and is not supplied with the full set of reliability features or documentation normally required for an operational network service. However the software and hardware configuration can be delivered in a fully specified Vocalis network platform should this be required.

Based on Vocalis's proprietary speech technology software, the SS1 can service a range of interactive spoken dialogue applications. It is intended that software upgrades will be produced during the lifetime of the project to enable continual developments in speech technology algorithms to be readily implemented, although the basic hardware should remain unchanged, with the exception of items such as the SCxBus card, planned to provide integrated digital telephony communications as this is developed later in the project. Access to Vocalis proprietary algorithms and library functions will be provided only through provided software interfaces. Source code will not be supplied other than as developed in the CAVE project.

The SS1's hardware and architecture as well as its UNIX operating system software are all industry standard, which maximises open-interconnectivity. The SS1 software is supplied with a library of modules, functions and tools implementing the CAVE Speech Server API functions, thereby allowing rapid development of custom applications.

The SS1 will typically integrate into a customer environment in the way outlined in the diagram located in next page.



Hardware Overview

The SS1 hardware is modular, enabling a wide variety of systems to be configured. The hardware is based on a standard tower or similar chassis with both Industry Standard Architecture (ISA) bus and PCI slots, with some room for expansion if required.

Basic computing resources are provided by a Pentium processor with 64MB random access memory, DAT tape cartridge drive, IDE floppy disk controller, SCSI hard disk controller, network card, SVGA display card and an SVGA monitor and keyboard. The SCSI controller supports both the hard disk and tape cartridge, and permits use of additional disk drives and CD-ROM reader for installation. A 2 Gb hard disk will be supplied to support the system services, storage of speaker recognition and verification models, and associated call speech data. This is expected to be more than sufficient storage to support all phases of the CAVE field trials, including recording of all speech data. A modem facilitates installation, support, updates and reconfiguration remotely by Vocalis.

Additional disk drives, ISA peripherals and interface cards can be connected as required by individual applications – depending on availability of back-plane slots and software driver support. Telephony interfacing and speech input/output on the host is to be managed by industry standard digital telephony cards. Provision is made for later addition of an SCxBus digital interface card, but initial system specification does not include this card, and TCP/IP connectivity will be used for both data and speech communications.



Hardware components

Item description	Qty	Reference**
Chassis	1	Standard tower or similar chassis, ISA and PCI slots, 3.5" 1.44 Mb floppy drive, with country-specific connectors
Single board computer	1	Pentium 120 Mhz ¹ , 512 Kb L2 cache, floppy controller, serial, parallel and PS/2 mouse ports
RAM	4	16 ² Mb (72 pin SIMM 9 chip 70 ns parity)
Hard disk controller	1	DPT Smartcache IV PM2144W PCI SCSI
Hard disk	1	2 Gb SCSI
Tape backup	1	DAT 4 Gb, internal
Network adapter	1	SMC 8216 ISA Ethernet 10 Mbit/s
Video controller card	1	SVGA
Modem *	1	External US Robotics Sportster 28.8 v34 (UK approved only)
Monitor	1	14" color SVGA
Keyboard	1	with standard connector
Mouse	1	Microsoft with PS/2 connector
SCxBus card *	1	Dialogic SCxBus adapter - can be added later as required

* Optional features not included in the standard specification and price quotation.

** Manufacturers and item types may be replaced by suitable alternatives

The following table describes the SS1's physical characteristics and the range of environmental conditions within which it can operate :

Temperature	0 - 55 °C. Reliable disk reading/writing 5 - 45 °C.
Humidity	0- 95% at 40°C (non condensing).
Altitude	15,000 feet / 4,500 metres.
Power Supply	95-132/180-246 V AC, 47-63Hz, 250 Watts
Physical **	254mm H x 2254mm W x 406.4mm D. Additionally, a keyboard and monitor.
MTBF	Manufacturers' figures for continuous 24 hour operation: Central processing Unit 50,000 hours Power Supply Unit 100,000 hours
Access **	0.5 metre envelope to sides, top, front, with 50cm at rear. SS1 is relatively portable, so access can be achieved by small movement of the unit, though no cables may be disconnected.

** Specifications subject to change if chassis is replaced by a suitable alternative

¹ This CPU is specified for a hardware-based recognition system. CAVE systems were actually supplied with a host software-based recogniser which required a 200Mhz Pentium Pro.

² 64 MB RAM wa supplied for the host-based recognition system. The number of active recognisers is dependent upon RAM size.



Speech Recognition

The SS1 is configured with a software speech recognition and verification system which may be shared across all four telephony channels, subject to reasonable resource loading. Hardware speech recognisers are not supplied as standard in the SS1, though they may be added subsequently, as needs dictate. These can be quoted for separately.

In most applications a recognition process is required for a relatively small proportion of the dialogue with the remaining time being spent in prompt playback. In this way the single software recogniser can generally service all 4 telephone channels. The number of simultaneous recognition processes possible without degrading response times is determined by the vocabulary size and recognition mode (i.e. isolated or continuous word recognition and word- or phoneme-based vocabulary).

The SS1 is delivered configured for recognition vocabularies up to 100 words. (This includes separate male and female models, and should be sufficient for all planned CAVE demonstration systems.) This figure may be increased by a Vocalis engineer in consultation with the client. Extending the vocabulary size in a software recogniser tends to introduce constraints in other parts of the system, such as response time or number of telephone channels which can be serviced.

The following speech recognition capabilities are supported:

Speaker Independent Recognition

- Flexible vocabulary creation based on phonemes (either British English, Dutch, Swedish or high German. Other languages can be supplied in consultation with Vocalis)
- Isolated word recognition - digits and keywords/keyphrases
- Keywords - yes, no, help, stop, cancel, repeat. A number of keywords can be supported using phoneme models for recognition if whole word models are unavailable.
- Continuous digit recognition.
- Top scoring candidate hypotheses with scores
- Dynamically loaded syntax trees
- Multiple mixture models

Single (later multiple) front-end parameterisation for recognition and verification

The following capabilities will also be provided in later releases of the software:

- N-best recognition, with resulting scores
- Improved word and phoneme models in formal German for Swiss subject to database provision by CAVE partners
- Improved CAVE algorithms and additional front end parameterisations

It may be possible to provide the following capabilities if required, in a separate quotation:

- Word spotting of words or phrases surrounded by any other words.
- Out of vocabulary rejection
- Talkover

Speaker verification

- Enrolment and authentication modules for speaker verification using CAVE algorithms
- Single Gaussian mixture models
- Single (later multiple) front-end parameterisation for recognition and verification



Software Description

Item description	Qty	Reference
Operating system	1	SCO OpenServer 5 Enterprise - 2 users licence (UNIX)
Telephony drivers	1	Dialogic System Release 4.2 for SCO UNIX
Recognition (runtime)	1	Vocalis Speech Recognition Runtime System (includes Speaker Verification)
Recognition (development) *	1	Vocalis Speech Recognition Development System.
Template database*	1	Informix C-ISAM 3.0 or C-ISAM 6.0

* Optional features not included in the standard specification and price quotation.

The operating system is SCO Openserver 5 (UNIX). This affords connectivity to a multitude of third party communication products which interface to LANs, WANs and other communication interfaces.

Below the operating system layer a number of device drivers handle the telephony interfaces, recognition processors and host communications hardware.

Above the operating system, the API layer provides a library of C subroutines for speech recognition, verification and basic host communications. This API will conform to the CAVE specification, initially developed to a design specified solely by project partners. Later releases will be provided implementing the final CAVE API design.

Also included is Management Information Support offering features such as call statistics collection and line and system management functions. These features are set out below:

System Administration Interface

The SS1 provides a system administration interface which allows a system administrator to:

- Perform the system shutdown (this will allow the system to be powered down.)
- Change the system administrator's password.
- Archive log files and allow an operator to extract log files from the system.
- Carry out any application-specific system administration functions.

Call Logging

Each call creates a call record entry which is kept in an ASCII file on the SS1. It is possible to process call records to provide customised reports for detailed analysis. The call record will be defined by the CAVE project and contains test and debug logging information such as:

- The date of the call.
- The time of the call.
- The total duration of the call
- The server functions called and their activity status
- The incoming line number or telephony channel on which the call was taken.



Vocalis Speech Recognition Runtime System

The Vocalis Runtime system provides the following:

- utilities to configure and load the speech recognisers installed in Speech Server with vocabularies to be recognised
- a utility to monitor the usage of the speech recognisers in Speech Server
- Speech recognition firmware
- a watchdog process to monitor recognisers in Speech Server and reboot them if they are found to be down

That Runtime system also includes full support for SV.