Name 1 Name 2 Unit (NZNNU)

<Company 1>

CTI/IVR System Architecture

Work Request: 123456
Version: 1.0c
Status: Draft

- Stephen McGregor wrote this document from scratch and in its entirety.
- This document is only for private evaluation.
- Do not retain this document after use.
- Do NOT distribute this document.

This document is only for use in association with hiring Stephen McGregor as a technical writer.

<COMPANY 2> (New Zealand) Limited
Address 1
Address 2
New Zealand

Amendment History

CR# (optional)	Document Version #	Approval Date	Modified By	Section, Page(s)and Text Revised
	0.1a	15/05/2006	<removed>	Initial draft
	0.1b	22/08/06	<removed>	Updated draft
	1.0c	15/11/2006	Stephen McGregor	Complete re-write.

Table of Contents

6 DOCUMENT INFORMATION

6.1 DOCUMENT DESCRIPTION

About this document
The objective of this document is to document the current <Company 1> (WXYZ) CTI/IVR system architecture.

Intended audience
This document has been written for information specialists/analysts involved in development and technical support of WXYZ CTI/IVR.

Use Company Names	
<COMPANY 2>	<Company 2> (New Zealand) Limited
WXYZ	<Company 1>

Related Reading Material	
REF 1	CTI PARM document
REF 2	CTI Infrastructure Document
REF 3	Speech Navigator Detailed IVR Design. Version: ____
REF 4	CTI / IVR RS 6000 Operations Guide
REF 5	Genesys Framework 7.2 Deployment document
REF 6	Genesys Outbound Contact 7.2 Deployment document
REF 7	Genesys Universal Routing 7.2 Deployment document
REF 8	Genesys Configuration Management 7.2 Deployment document
REF 9	Genesys CCPulse 7.2 Deployment document
REF 10	Genesys GIS 7.2 Deployment document
REF 11	CTI/IVR Enterprise Architecture document
REF 12	CTI/IVR Interface document

6.2 DOCUMENT ACCEPTANCE

Introduction

Acceptance of this document signifies that <Company 1>'s requirement for YZ 123456 has been met and that the requirements specified for this document have been satisfied to a baseline level.

<COMPANY 2>

This document has been reviewed and approved for release on behalf of <COMPANY 2>, by:

<Name>
<Title>

Date: ……

WXYZ

This document is accepted on behalf of WXYZ, by:

<Name>
<Title>

Date: ……

7 INTRODUCTION

7.1 OVERVIEW

This document is intended to be used to describe, at a high level, the system architecture for <Company 1>'s CTI/IVR implementation.

7.2 PURPOSE

This document provides an architectural overview of the CTI/IVR systems so as to reflect the current environment, as-built. This work forms part of the CTI/IVR Replacement Project and part of the CTI/IVR Growth and Stability Project.

7.3 AUDIENCE

The intended audience for this document is:

- information specialists/analysts involved in development and technical support of the documented applications
- the relevant project team,
- Operations Managers,
- Application Support, including QA-Managers and Testing Specialists.

7.4 SCOPE

7.4.2 INCLUDED

The scope of the document includes:

- Logical views of the systems, including architectural views of each component.
- processes throughout the system.

7.4.3 EXCLUDED

- Functional Requirement
- Usability Requirements
- Threading and concurrency are not addressed
- Any deployment views of the System, including nodes, processor, physical network topologies or the protocols used.
- Any implementation view of the System, including the software packages and their components

7.5 REFERENCES

As per ‘Reading material’ under section 1.1

Referenced Material	
REF 1	CTI PARM document
REF 2	CTI Infrastructure Document
REF 3	CTI IVR Architecture Version: ____
REF 4	CTI / IVR RS 6000 Operations Guide
REF 5	Genesys Framework 7.2 Deployment document
REF 6	Genesys Outbound Contact 7.2 Deployment document
REF 7	Genesys Universal Routing 7.2 Deployment document
REF 8	Genesys Configuration Management 7.2 Deployment document
REF 9	Genesys CCPulse 7.2 Deployment document
REF 10	Genesys GIS 7.2 Deployment document
REF 11	CTI/IVR Enterprise Architecture document
REF 12	CTI/IVR Interface document

8 BACKGROUND

Historically, the configuration of <Company 1>'s IVR system included as its main features:

- an IVR Server at each call centre
- each IVR machine connected behind the local PABX; a behind-the-switch configuration.
- the IBM Callpath/Directtalk model
- ongoing issues: capacity problems, thus a need to upgrade the system's hardware
- ongoing issues: CallPath and DirectTalk have ceased to be supported by IBM
- After investigation a new IVR architecture was developed. This new architecture included:
- an IVR system connected directly to the PSTN, in front of the PABXs (an infront-of-switch configuration),
- the latest version of DirectTalk, now known as WebSphere Voice Response (WVR), running in the IBM WebSphere environment
- routing/outbound now controlled by the Genesys suite of telephony applications.
- The existing call-recording system was replaced by the NICE call-monitoring suite.
- On the users' desktops the Callpath/DirectTalk CTI Toolbar was replaced by an improved application known as the "Genesys Softphone", a non-Genesys dialog written by <COMPANY 2> in Visual Basic.

Following on from this re-work of the IVR system, further improvements have been initiated and to varying degrees implemented. At the time of writing (November 2006) these include:

- Speech Navigator, a voice recognition system that is in the process of replacing some parts of the IVR structure
- The incorporation of <Company 1>'s Directory Assistance business into the Genesys component of the IVR
- <Company 1> Outbound Calling likewise utilising the new Genesys suite.
- For full description of these and other components of <Company 1>'s IVR system, see:
- CTI/IVR Enterprise Architecture Documentation
- CTI/IVR PARM

and the other documents listed in the references, page 9.

9 IVR SYSTEM STRUCTURE

9.1 BRIEF SYSTEM DESCRIPTION

9.1.2 COMPONENTS

The hardware and software entities involved in the <Company 1> IVR are:

- PSTN — The Public (Switched) Telephone Network
- ICM — A server that runs the load balancing over the six IVR machines
- WVP — WebSphere Voice Platform (WVP). This is the software environment implementing the IVR.
- StateTables — "StateTables" are a collection of programs that implement the IVR patterns, playing voice recordings (RANs) and collecting key-presses (DTMFs). These programs interact with the "Back-End" systems, including with the Genesys Environment.
- Genesys — <Company 1>'s Genesys environment decides which IVRs to play and, if the call is sent to a Customer Service Rep after the IVR is executed, Genesys manages the transfer of the call to the CSR
- Back-end Systems — A number of <Company 1> applications, information sources, and servers are required for particular IVRs
- PABX — Calls transferred to the <Company 1> Call Centre briefly queued on a local PABX.
- NICE — A Call-recording and Screen-dump capture system. Used both to monitor CSRs and to make some transaction records

< Diagram removed >

9.2 CALL PASSAGE THROUGH THE IVR

Calls entering <Company 1>'s IVR system are initially passed to ICM. ICM re-numbers the call so as to spread load over <Company 1>'s six IVR machines. After testing the new number, ICM passes the call back to the PSTN.

The PSTN now delivers the call to <Company 1>'s IVR environment. This environment comprises six IBM RS/6000 machines, each running IBM's WebSphere Voice Platform (WVP). The part of the WVP that concerns us here – the Interactive Voice Response system – is a collection of programs referred to as StateTables. Each StateTable implements a small group of similar IVRs.

Each StateTable plays dynamically generated messages to the caller ("prompts"), collects responses from the caller's touch-tone phone, interacts with other systems on the RS/6000 machine – which may in turn connect to remote systems, and manages the telephone call itself. Effectively the StateTables *are* the IVR system.

The other systems on the RS/6000 machine that the StateTables interact with are collectively named "Custom Servers". These are specialised programs written by <COMPANY 2> or <Company 1> to supply information from, or deliver information to, a large number of <Company 1> applications, databases, processes and the like. For example, one Custom Server sends and receives data to and from <Company 1>'s ICMS Customer Management system.

Call routing actions are determined by the Genesys system. These decision are which IVR to play, what variation of the IVR to play and, if the caller needs to speak to a Customer Service Representative, which CSR will receive the call and how to transfer it. StateTables communicate with Genesys via the "Genesys Driver for AIX", usually known as "d2is".

The call remains on the RS/6000 while the IVRs are playing, while data goes back and forth to back-end systems and Genesys sends instructions and receives data.

Many calls terminate after the IVR has played. If the call needs to be transferred to a CSR, Genesys is informed of this and, using the information gathered so far during the IVR, chooses which CSR should receive the call. To do this it uses its routing server (the URS), availability and queue times, and "skills". Each CSR has a collection of skills, and, based on the data gathered by Genesys, the caller will require certain level of a particular skill. Genesys takes all these into account when selecting target CSRs. During this time the call remains on the RS/6000 machine and receives some sort of hold treatment, probably Dave Dobbyn interspersed with the Captain and Tennille.

When an appropriate CSR becomes available, Genesys transfers the call to (a Genesys TServer on) their local PABX, wherever in the country that might be. The call will ring a couple of times on the PABX and the CSR will answer it. As soon as this happens Genesys instructs the PABX to re-route the call directly from the PSTN to the agent's phone, thus absolutely minimising load on each local PABX.

After the CSR finishes with the call it is disconnected normally and records are written to both the Genesys Call Concentrator database for customer analysis, and independently gathered by the Reporting StatServer for Genesys Reporting. In the <Company 1> environment it is this second stream that is retained in the Data Warehouse.

9.3 IVR TRAFFIC DISTRIBUTION

The distribution of incoming calls is managed by ICM. This application actions calling plans designed to allocate the calls evenly, thus minimising the load on each machine. It also tests each line before it transfers a call, effectively eliminating dropped connections. ICM's routing plans can be easily changed to accommodate problems, such as unavailable machines and unforecasted peaks.

As an example:

> A Customer called <removed> IVR from their Mobile phone. This call goes through from the PSTN to the Intelligent Network where it has been defined as a number that ICM is interested in, therefore passed into ICM. ICM will have several call plans set up (i.e. 33% calls to IE1360, 33% calls to IE1362, 34% calls to IP736 being one call plan, 33% calls to IE1361, 33% calls to IE1362, 34% calls to IP736 being another), and the active one will pass the call back to the PSTN to be distributed to the right Server.

All IVRs are available on each one of the six RS/6000 machines.

< Diagram removed >

9.4 PSTN TRUNKS INTO THE IVR

Coming into each of the six IVR machines, the as-built configuration is:

- 1 x Primary Rate Trunk (PRA) handling <removed> pre-pay top up calls. About five million of these calls are processed each month.
- 2 x Primary Rate Trunks (PRA Lites) handling *<removed> (pre-pay account balance) calls. Twelve million of these are processed each month.
- 5 x Primary Rate Trunks (PRA Lites) handling all the remaining IVR calls.

Each Primary Rate contains 30 telephone lines, terminating on 30 ports on a particular RS/6000 IVR machine.

< Network Diagram Removed >

10 RS/6000: EACH OF THE SIX RS/6000'S

10.1 RS/6000: IVR SYSTEM

10.1.2 COMPONENTS

10.1.2.1 Operating System; WebSphere

Each RS/6000 runs an AIX operating system. AIX is IBM's Unix variant (AIX, V5.2 (program number 5765-E62) with a minimum maintenance level of 2. An IBM Communications Server for AIX, V6.1 is required for remote connections.

On top of the operating system runs an WebSphere Application Server, V5.0. The WebSphere Voice Platform runs "within" this environment.

10.1.2.2 IBM WebSphere Voice Platform

WebSphere Voice Response is a packaged, interactive, window-based system implementing telephone and data exchange functionalities. The WVR functionalities utilised in the <Company 1> environment are:

- State Tables
- some system monitoring
- management of connections to the PSTN
- "Custom Servers" ; programs that interface to State Tables in order to connect to many of <Company 1>'s back-end systems

10.1.2.3 StateTables

State Tables are BASIC/Cobol-like programs used to implement the core IVR abilities.

10.1.2.4 IVR Scripts

At a very simple level, IVR scripts are composed of:

- Voice Segments: spoken words, like "Good Morning.", "Five.", "The time is."
- Prompts: Dynamically constructed collections of Voice Segments, like "The time is." + "Five" + "Oh" + "two".
- State Tables: Programs that implement:
  - Prompts
  - collections of touch-tone phone responses
  - executions of Custom Servers, applications that connect to back-end <Company 1> systems
  - logic patterns: the caller pushed '0', divert the call to a CSR.

The IVR scripts also interface with the Genesys environment via the d2is driver.

10.1.2.5 Genesys Driver for AIX, known as d2is

The Genesys "IVR Driver for WVR for AIX" is known as d2is. This exposes an API with a number of functions for sending and obtaining information from the Genesys environment.

10.1.2.6 Custom Server

From *WebSphere Voice Response for AIX with DirectTalk Technology Custom Servers*:

> "A custom server is a program, using the C or C++ language, that provides an interface between data on host computers and WebSphere Voice Response, or performs other processes, such as speech recognition and speech synthesis, generation of fax output, or coordinated call and data transfer. The data on the host system can include business information held in a database, or digitized voice data."

The custom servers on the RS/6000 IVR machines are Custom Servers that provide:

- back functionality, via a connected fax modem,
- integration with ICMS via the PINAPPS server,
- integration with the Vantive VIC Payphones system, also via the PINAPPS server,
- integration with the EAI for prepaid functionality,
- single line test functionality via the NTS and NAS systems,
- access to the MPA EAI instance for the purpose of OTAF PRL downloads.

10.1.2.7 Holds calls

This provides a place to park calls and provides queuing functionalities.

10.1.3 OPERATION

This details the top-level operation of the Master IVR State Table, either GenStart or IStart. A summarised, and in many ways better, diagram follows.

1. A call from the PABX arrives on the RS/6000 machine via one of the trunks connected directly to it. On Infront-of-Switch machines, this triggers the IStart IVR script; GenStart is run on boxes functioning in the behind-switch pattern. At this stage a connection to the appropriate log file is made.
2. This script uses the StateTable call AnswerCall( ) to answer the call
3. The Custom Servers:
   - GenTools
   - d2is
   - WXYZ_EXP_TS
   - WXYZ_CREC_KK
   - WS_PortMonitor

   are opened. If these fail to open then the script still continues, heading towards a default IVR execution.
4. System variables are set up and copies of these are sent to the PortMonitor application via the WS-PortMonitor custom server.
5. A connection is opened to the WXYZ_HMenu Custom Server.
6. Genesys is informed about the call by the NotifyCallStart( ) function of the d2is (Genesys interface) custom server. If this fails, then the script jumps to a default handling of the call.
7. The number dialled (the DNIS) and the caller's number (CLI) are sent to Genesys
8. Genesys sends back a lot of data regarding call routing.
9. GenTools.ParseCallInfo( ) is used to make sense of this information.
10. Key Value pairs are built-up. KVPs are used by Genesys to attach "additional" information to the call
11. Special CLIs and DNISs are handled.
12. The KVPs are sent to Genesys
13. If necessary, a PIN number is prompted for
14. The GetIVRInfo( ) function of the GenTools custom server determines the routing start-point
15. Calling the RouteStart function of the d2is custom server (Genesys interface) implements this start point.
16. A copy of the routing information is held on the RS/6000 machine. At this point, if it is difficult to get data from the URS then the local routing information is used instead.
17. A loop, responding to the caller's key-presses now starts:
    a. GenTools.ParseCallInfo( ) is again used, this time to sort out the routing information available. This information is logged, including which IVR script is going to be run.
    b. d2is.SendReply( ) is used to get some more information from Genesys.

c. Which voice segment to play to the caller is decided by WXYZ_EXP_TS.ran_lookup( )

d. either: Faults or Table-Driven are activated

e. or: A chosen IVR voice segment is played

18. If the caller stays on the line, and doesn't need to be transferred, control goes back to the top of the loop.

19. If the Customer needs to be transferred, control is passed to Genesys to manage the transfer.

The remainder of GenStart is concerned with default (error situation) handling, and the PortMonitor logging and monitoring application.

< UML diagram removed >

10.2 RS/6000: STATETABLES

As noted above, StateTables are Cobol-Basic like programs that together implement the core IVR environment. Their other main attributes are:

- The running of voice "prompts", which are dynamically constructed collections of "voice segments" that together pronounce a meaningful message
- gathering of the caller's touch-tone phone key presses. These are referred to as DTMFs – Dial Tone Multi Frequency.
- usage of "Custom Servers", specialised C/C++ programs for interaction with remote <Company 1> Systems
- Interaction with the Genesys environment through the use of the API presented by the d2is driver.

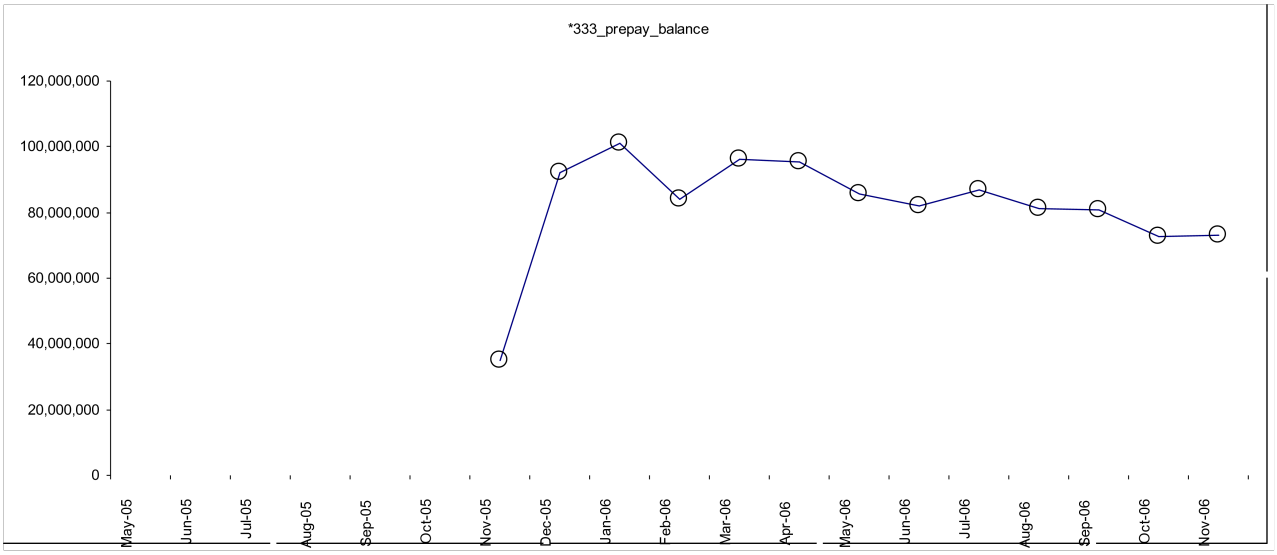
10.3 RS/6000: IVR CATEGORIES

Listed in decreasing order of volume. These have been grouped from the thousands of SV248, each potentially representing a separate IVR.

10.3.2 <REMOVED>: PREPAY BALANCE

The largest IVR, receiving around 100 million calls a month. Two primary rate trunks on each of the six IVR machine are dedicated to <removed>.

calls in the 13 months year to <date removed>	588,777,966
Total IVR Time	<removed>
Average IVR Time	<removed>
Total CSR Time	<removed>
Average CSR Time	<removed>
Back-end systems used	<removed>
StateTables used	<removed>



< Ten pages of data removed >

< Diagram removed >

10.3.2.1 Interface to IS_EAI (for Tango): WS_EAI_T

The “Tango” project implements <Company 1>’s <removed> system. WS_EAI_T is the IVR’s interface to the back-end systems implementing the Tango project.

The WS_EAI_T server uses a SOAP over HTTP connection to communicate with the IS_EAI <Company 1> server, which actions the back-office activities required for a Tango / freedom activation.

The applications connected to by the RS/6000’s custom servers are discussed in additional detail in the section “Related External Systems”, on page 62.

10.3.2.2 Interface to Single Line Testing: SLT servers

Singe Line Testing is a name given to the process of checking a telephone line for faults. <Company 1> Service Express, Faults, and Business Single Line may each activate the Single Line Testing IVR.

The Single Line Testing State Table uses the WXYZ_SLT Custom Server to activate and complete the line-test.

The applications connected to by the RS/6000's custom servers are discussed in additional detail in the section "Related External Systems", on page 62.

10.3.2.3 Interface to PINAPPS: WXYZ_TUX server

The WXYZ_TUX servers are the interface to the PINAPPS server, which is in turn a connection to both <Company 1>'s ICMS customer information system and to its application for ordering Phone Cards to be delivered to shops, "VicP". The WXYZ_TUX server is required for the execution of all IVRs.

Two Custom Servers, WXYZ_TUX and WXYZ_TUX _2 are used for this important connection to PINAPPS.

10.3.2.4 Interface to OLTP: IVR - EAI System Integration

The Online-Transaction-Processing system is the busiest part of the IVR, with about 12 million calls a month. Similar to the other Customer Servers it is constructed of three parts, as described above, and connects to the EAI_OLTP server.

The applications connected to by the RS/6000's custom servers are discussed in additional detail in the section "Related External Systems", on page 62.

10.3.2.5 Interface to OTAF: OTAF Server

Over The Air Functionality (OTAF) is a system for adding advanced services and functionality to customers' mobile phones. Currently it is only used to add international roaming (actually a PRL - Preferential Roaming List) though the system is sized to handle large amount of traffic. The OTAF set-up is slightly different from the other back-end systems in that

- it connects to a non-<Company 1> application managed by Lucient
- works with the customers physical phone directly, rather than acting on their account or a database.

The WXYZ_CrossWorlds Custom Server supplies the interface between the State Tables and the OTAF_EAI, which in turn connects to the Lucient OTAF system.

As with the other servers, a slightly longer description appears in the appendix "Related External Systems", on page 62.

10.4 CUSTOM SERVER SOURCE CODE STRUCTURE

The arrangement of custom server code is:

XXX_main.C - the main( ) function, a switchboard that processes messages
XXXmain.C - hold the functions called by the main( ) function
XXXinit - initializes memory, and the start up. Called at the start of main( )
XXX_ext.C - the detailed structures used to communicate with the State Tables
XXXterm.C - to halt the program
XXX_hdr.h - header file; declarations and the like.

For example, the source code for the WS_EAI is held in the following files: WSEAImain.C, WSEAIterm.C, WS_EAI_ext.C, WS_EAI_hdr.h, WS_EAI_main.C, WSEAIinit.C.

10.5 EXAMPLE CUSTOM SERVER: MORE DETAIL

10.5.2 WXYZ_TUX_MAIN

As the name suggests, WXYZ_TUX_Main.C contains the main( ) function, the initial routine run when the server starts. This function (main( )) is a huge switchboard: receiving messages, calling appropriate functions, and sending information back to the State Table.

An infinite loop, receiving messages *DT_msg_info_st* via CA_Recieve_DT_Msg( )

```
while (TRUE)
 {
 if (CA_Receive_DT_Msg (&DT_msg_info_st, CA_WAIT) == -1)
 {
```

For each message ***DT_msg_info_st*** handle it via a (very large) switch statement. A switch statement executes instructions selected by the value of the variable in the switch( ) statement, in this case ***DT_msg_info_st.func_id***.

```
 switch (DT_msg_info_st.func_id)
 {
```

The switch statement directs instruction flow to the appropriate '***case:*** ' statement, depending on the ***DT_msg_info_st.func_id***. value. The handling of each func_id value is the same:

1. Get the information for the execution of the upcoming function

- use ***CA_Get_DT_Parameters( )*** to get parameters
- if there is an error getting the parameters call ***CA_Send_DT_Error( )*** and move onto waiting for the next message

2. Call the selected function. This interacts with an external back-office system.

- call the appropriate function to action the value of the message. This returns a dynamically allocated character value which is assigned to a pointer-to-char.
- If nothing is returned (the pointer is NULL), ***CA_Send_DT_Error( )*** raises an error and sends the message itself as a parameter to the error process.

3. Now get information for the appropriate interaction with the local Direct Talk box

- ***CA_Put_DT_Parameters( )*** gets the parameters. These are written to the string "***DT_msg_info_st***"
- If an error is encountered, ***CA_Send_DT_Msg( )*** is used to raise an error

4. Send the information to the local State-Table system.

- ***CA_Send_DT_Msg( )*** send the DT_msg_info_st back to the Direct Talk environment

Using the example of the value of ***DT_msg_info_st.func_id*** being ***TUX_CheckGAPs_func_id*** then we have:

1. Get the information for the execution of the upcoming function

```
case TUX_CheckGAPs_func_id:
if (CA_Get_DT_Parameters (
&DT_msg_info_st,
TUX_CheckGAPs_debug,
&TUX_CheckGAPs_chan_num,
TUX_CheckGAPs_pline,
TUX_CheckGAPs_gap_codes,
TUX_CheckGAPs_dump_flag,
TUX_CheckGAPs_pin)!= 0)

{
CA_Send_DT_Error (&DT_msg_info_st);
break;
}
```

2. Call the selected function. This interacts with an external back-office system.

```
TUX_CheckGAPs_output_ptr = (char *) TUX_CheckGAPs (
TUX_CheckGAPs_debug,
TUX_CheckGAPs_chan_num,
TUX_CheckGAPs_pline,
TUX_CheckGAPs_gap_codes,
TUX_CheckGAPs_dump_flag,
TUX_CheckGAPs_pin);
if (TUX_CheckGAPs_output_ptr == NULL)
{
CA_Send_DT_Error (&DT_msg_info_st);
break;
}
```

3. Now get information for the appropriate interaction with the local Direct Talk box

```
if (CA_Put_DT_Parameters (
```

```
&DT_msg_info_st,
TUX_CheckGAPs_output_ptr)!= 0)
CA_Send_DT_Error (&DT_msg_info_st);
```

4. Send this information to the local State-Table environment

```
else
CA_Send_DT_Msg (&DT_msg_info_st);
break;
```

10.5.3 TUXMAIN.C

This contains the code of each of the functions called in WXYZ_Tux_main.c as listed above.

Each of the functions in this file may return a different data structure. The union ***csStruct*** holds all of these structures and is used to return them to the state table.

The functions in TUXmain.c

```
ARRAY_Allocate( )
ARRAY_FreeAllHandles( )
ARRAY_FreeHandle( )
ARRAY_GetNumField( )
ARRAY_GetStringField( )
ARRAY_SetNumField( )
ARRAY_SetStringField( )
TUX_ApplyCharge( )
TUX_ApplyExtCRED( )
TUX_Check4Prod( )
TUX_CheckGAPs( )
TUX_CheckLineStatusCRED( )
TUX_ChgPINEXPCID( )
TUX_ChkContrID( )
TUX_ChkElgbltyCRED( )
TUX_ChkUserID001( )
TUX_ChkUserIDEXPCID( )
TUX_ChkUserIDEXPCUST( )
TUX_ContactLog( )
TUX_CredExtAllowedCRED( )
TUX_CustProfile( )
TUX_FMGetCallPlanDetails( )
TUX_GetAccDetailsEXP( )
TUX_GetAccounts( )
TUX_GetBrochureDetail( )
TUX_GetCallPlanDetails( )
TUX_GetCustContactDetails( )
TUX_GetCustListDetails( )
TUX_GetFreeMinsCallPlan( )
TUX_GetLastPaymentsEXP( )
TUX_GetMailInfo( )
TUX_GetNextBillDate( )
TUX_GetSTInfoEXP( )
TUX_GetSTNextLastInfoEXP( )
TUX_GetSubType( )
TUX_GetSubTypeDetail( )
TUX_GetTotalBalance( )
TUX_GetVANAccess( )
TUX_PutCustListDetails( )
TUX_RefreshPB( )
```

```
TUX_RemovePIN( )
TUX_ReqMinutesUsedWill( )
TUX_ResetConEXP( )
TUX_RetAgedDebtCRED( )
TUX_RetrievePIN( )
TUX_SaveMailInfo( )
TUX_SaveMailInfoLogged( )
TUX_SendSTReprintEXP( )
TUX_UpdateSO( )
TUX_ValidateOrder( )
TUX_ValidatePIN( )
TUX_ValidateRetailer( )
TUX_cc_payment( )
TUX_cc_pmt_reconnect( )
TUX_get_outst_reconn( )
TUX_is_line_disconnected( )
TUX_just_rcpt( )
TUX_just_reconnect( )
TUX_rcpt_reconnect( )
```

10.5.4 WXYZ_TUX_EXT.C

This file holds a few large tables detailing the structures of the records passed back and forwards to the State Tables.

10.5.5 TUXSUB

This file contains all the sub-routines used in TUXmain.C and TUXinit.C

10.5.6 TUXINIT

Reads the EXPcfg.vi initialisation file and then initialises all the objects needed for the WXYZ_TUX server.

Specifically, the TUXinit( ) function:

- allocates shared memory
- initialises something called the brochure array
- uses DTArrayInitParent( ) to set-up memory required for various types of interaction with the DirectTalk State Tables
- If necessary forks a number of duplicates of itself (i.e. the WXYZ_TUX Custom Server)

10.5.7 TUXTERM.C

Contains a single function TUXterm( ), for shutting down the custom server.

10.5.8 TUXCFG.C

Contains a few functions “for accessing the Tuxedo Configuration”. This is probably obsolete, but still part of the build.

10.6 RS/6000: INSTALLATION / SETUP

10.6.2 D2IS INSTALLATION

This is described in more and sufficient detail in either

IVR Interface: System Administrator's Guide for IVR Interface Driver for DirectTalk

or

IVR Driver for WVR for AIX: System Administrator's Guide

As a brief overview, the process involves:

1. Use the WebSphere Voice Response for AIX (WVR) application to import the d2is driver, probably off the CD on which it is supplied.
2. Again using the WVR application, install the d2is driver, via the Application > Custom Server dialog box
3. Connection to the Genesys IVR server, and some of the connection attributes to the local RS/6000 machine, are detailed via the properties dialog. These are specific, and described in the System Administrator's Guide.
4. The installation can be tested with Genesys's "Customer Test Package"
5. Start the local RS/6000 DirectTalk IVR environment
6. Use the WVP dialog, Applications > Custom Server Manager to initiate a single manual activation of the d2is driver or to set up automatic activation of the driver.

10.7 RS/6000: DEPENDENCIES

All IVR's depend on

- The WebSphere Voice Platform and StateTables
- the WXYZ_TUX server to connect them to the ICMS Customer Information system.
- both PINAPPS, and the ICMS system it connects to

Credit Card payments, <removed> residential mobile, <removed> account balance, and pre-pay top-up also depend on:

- WS_EAI
- the EAI OLTP system
- the ICMS customer information system

International roaming IVR depends on

- OTAF_IVR, which depends on
- OTAF_Server (locally on the RS/6000)
- OTAF_EAI
- OTAF Server; a back-end system managed by Lucient

The Tango IVR also depends on

- WS_EAI_T server

10.7.2 MACHINE FAILURE

Situation

Calls routed to the RS/6000 correctly receive a disconnected line signal. At the time of writing (November 2006) the ICM continues to route calls to the now disabled machine. Soon, by the start of 2007, the ICM will test successful initiation of its calls and re-route upon failure. From that time onwards all callers should remain unaware of any RS/6000 failure.

Resolution

- Change the routing in ICM such that no calls are routed to the failed machine.
- Reboot the box normally.

10.7.3 INTERNAL APPLICATION FAILURE

10.7.3.1 WVP / StateTable Failure

Situation

The callers already directed to that IVR machine would never have their call answered.

Resolution

- Call the CTI/ IVR system administrator to re-boot the machine.

10.7.3.2 d2is failure

Situation

Nothing immediately. The RS/6000 has a local copy of the routing rules produced by the Genesys URS, which is used after failing to get information from the URS. This local copy is updated once a day, if the d2is driver is available.

No changes to call routing could be effected on this machine while the d2is interface is down.

Resolution

- A system administrator would telnet to the RS/6000 box, run the system control GUI, and restart the d2is

10.7.3.3 WXYZ_TUX server failure

Situation

The IVR system would cease to work, as all IVRs rely on the WXYZ_TUX server. Callers will receive a “Due to technical difficulties ….” message.

Resolution

- A system administrator would telnet to the RS/6000 box, run the system control GUI, and restart the WXYZ_TUX server

10.7.3.4 WS_EAI_T failure

Situation

All callers trying to activate <Company 1>’s “<removed>” (Tango) would not be able to do so, and will receive a “Due to technical difficulties ….” message.

Resolution

- A system administrator would telnet to the RS/6000 box, run the system control GUI, and restart the WS_EAI_T server

10.7.3.5 WS_EAI failure

Situation

All callers trying to do

- Credit-card payments
- <removed> residential mobile calls
- <removed> account balance calls
- prepay top-up

would not be able to do so and would receive a “Due to technical difficulties ….” message.

Resolution

- A system administrator would telnet to the RS/6000 box, run the system control GUI, and restart the WS_EAI server

10.7.3.6 OTAF Server Failure

Situation

All callers trying to organise any advanced functionality on their cellphones, e.g. International Roaming, would not be able to. the will receive a “Due to technical difficulties ….” message.

Resolution

- A system administrator would telnet to the RS/6000 box, run the system control GUI, and restart the OTAF server

10.7.3.7 SLT servers failure

Situation

All callers trying to have a telephone line tested would not be able to, and, likewise, receive a “Due to technical difficulties ….” message.

Resolution

- A system administrator would telnet to the RS/6000 box, run the system control GUI, and restart the SLT server

10.7.4 External Application failure

With the exception of Genesys, none of the external applications are the direct responsibility of <COMPANY 2>. Genesys is under the control of <Company 1> but has <COMPANY 2> System Administration.

10.7.4.1 Genesys failure, or Network Failure

It is extremely unlikely that the entire Genesys system would go down. A loss of the network is a much more probable and common occurrence. If either of these did happen

1. The IVR system would continue to function normally, each machine using its local back-up copy of the URS database.
2. If the caller dialled ‘0’ to speak to a CSR, their call would be directed one of the default numbers (ACD queues) on the call centre’s PABX machines
3. There would be no Genesys skill-based routing of course.
4. Most customers would not notice any change.

10.7.4.2 PINAPP failure

The IVRs will not function; all callers will receive a “Due to technical difficulties ….” message. PINAPP is the interface between the IVR system and the ICMS customer information system.

10.7.4.3 ICMS Failure

Likewise, if ICMS fails none of the IVRs will function and all callers will receive a “Due to technical difficulties ….” message.

10.7.4.4 IS_EAI failure

No Customers calling to set-up <Company 1>'s "<removed>" (Tango) program would be able to do so. Each would receive a "Due to technical difficulties …." message.

10.7.4.5 EAI OLTP failure

All customers calling:

- Credit Card payments
- <removed> residential mobile,
- <removed> account balance,
- pre-pay top-up

would be able to do so. Each would receive a "Due to technical difficulties …." message.

< Network diagram removed >

11 GENESYS: GENESYS MACHINES

11.1 GENESYS: MACHINES' SERVERS

11.1.2 AUCKLAND

11.1.2.1 Genesys 1

HN_IVRInFront_TServer_MDR	TServer to connect to the Hamilton IVR server running in the Mayoral Drive / Airedale Street Call Centre
CH_TServer_Primary	TServer connecting to the Christchurch PABX
HN_TServer_Primary	TServer connecting to the Hamilton PABX
	et cetera….

< nine pages of data tables removed >

11.2 GENESYS: OVERVIEW OF SERVERS' ROLES

11.2.2 GENESYS MANAGEMENT

11.2.2.1 Configuration Server

The Configuration Server

- Stores and makes available configuration data from the configuration database
- controls access to the Configuration database

In the <Company 1> environment applications use a proxy – the Configuration Server Proxy – for their actual work. The Proxy then connects to the Configuration Server.

11.2.2.2 Configuration database

Hold all of the configuration information.

11.2.2.3 Local Control Agent

The application which controls and monitors the Genesys applications on each machine.

11.2.3 CORE ROUTING SERVERS

11.2.3.1 TServer

In the Genesys environment ,T-Servers provide the interface to external devices. Specific T-Servers are available for interaction with different types of devices

11.2.3.2 I-Server

An I-Server is a TServer specialised for use in IVR environments. It presents two interfaces, a TServer interface to Genesys and a socket-like interface to the IVR environment.

11.2.3.3 URS

The Universal Routing Server provides routing information and distributes incoming calls to CSRs.

11.2.3.4 Hot and Warm Back-up Servers

In the Hot-backup configuration a connection to the backup device has already been made and all that one has to concern one's self about are the clients. Warm back-up machines maintain current information but still need to make connections upon fail-over, which may take 10 seconds. For many applications warm-back up is adequate.

< Network diagram removed >

11.2.4 MONITORING

11.2.4.1 Message Server

Message Servers are the intermediaries for Genesys inter-process messages, which they write to the LOGDB database.

11.2.4.2 Genesys Interface Server

The GIS application supplies an http/web interface to both the Configuration Sever and to the web-based Statistics Server analytical environment. An application using the GIS runs on CSR's desktops.

11.2.4.3 Call Concentrator

Call Concentrator is a Database and Server. The Server collects call-by-call information from one or more TServers and writes this information to the Call Concentrator database. This database is used for customised reports and analyses.

11.2.4.4 StatServer

A web-based analytical environment closely integrated with Genesys data. A StatServer collects information from on or more TServers, does some processing on this data and makes it available to applications that have registered with the TServer.

11.2.4.5 CCPulse

CCPulse is the Genesys Call Centre monitoring application.

< Network diagram removed >

11.2.5 Data Collection and Reporting

The Genesys reporting structure is discussed in more detail in the ‘Genesys: Data’ section on page 39.

11.2.5.1 Reporting StatServer

The Reporting StatServer collects call, agent and call centre information from the T-Server. This is read every 15 minutes by the Data Sourcer.

11.2.5.2 DataSourcer

The DataSourcer takes 15 minute snapshots from the Reporting StatServer and writes these to the Operational Data Store.

11.2.5.3 Operational Data Store

The ODS hold a series of snap-shots of Call Centre action. ETL has the task of summarising these and transferring them to the Data warehouse. The ODS is periodically purged by the ETL, probably every few hours.

11.2.5.4 ETL

A process: the process of Extracting, Transforming, and Loading data from a range of sources into a data-warehouse, or into some other depository. As just noted, its main task is to extract data from the ODS, transform these data into the form desired by the DataMart, and then insert data into the correct repository.

11.3 GENESYS: DATA

11.3.2 SUMMARY

11.3.2.1 Data Collection

There are four, effectively independent, data collection systems in a Genesys system:

- Call Concentrator,
- Reporting StatServer to the Data Mart,
- the Genesys logging process, and
- Call Centre Monitoring

The Call Concentrator collects all call-by-call data, in a very detailed format, and makes it available for custom analysis.

The Reporting StatServer collects system statistics, which are transferred to the DataSourcer, summarised into the Operational Data Store (ODS) and finally processed by the ETL and written into the DataMart.

All Genesys inter-application messages are recorded into the LOGDB database by the Genesys MessageServer.

CCPulse is the standard Genesys Call Centre monitoring application. The Application gathers data from the ccpulse_statserver which monitors local and remote TServers. This data can be accessed by custom-written, in-house applications via the use of the Genesys Interface Server, a collection of interfaces especially designed for StatServer data.

11.3.2.2 Data Storage

The Genesys databases are summarised in the following table.

Database	Instance	Description	Location
GENREPDB	ODS	Operational Data Store – Data Processing	HP1326 – Ak Data Collection Server
GENREPDB	DATAMART	DataMart	HP1326 – Ak Data Collection Server
CCONp	CCON	CallConcentrator – Call Records	HP1320 – Ak Outbound DB Server
CCONp	LOGDB	Log Database	HP1320 – Ak Outbound DB Server
IVRDIPp	IVRDIPp	IVR Information – Production	HP1320 – Ak Outbound DB Server
IVRDIPPP	IVRDIPPP	IVR Information – Standby	HP1336 – Hm Outbound DB Server
CONFIGp	CONFIGp	Configuration – Production	HP1320 – Ak Outbound DB Server
CONFIGpp	CONFIGpp	Configuration – Standby	HP1336 – Hm Outbound DB Server
OCSp	OCSp	Outbound Call Server DB – Production	HP1320 – Ak Outbound DB Server
OCSpp	OCSpp	Outbound Call Server DB – Standby	HP1336 – Hm Outbound DB Server

11.3.3 DATA COLLECTION: RELEVANT TO THE IVR

11.3.3.1 RS/6000 StateTable logs

Each StateTable writes data detailing each call's path through the IVR system. These are available for seven days on each machine, and available in parsed and processed form for ninety days in <Company 1>'s TRIAD data warehouse.

11.3.3.2 TRIAD IVR data

The first place to look for IVR data in TRIAD is the IG12370T table. This holds detailed day-by-day summaries of each component of the IVR system's activities. For records on each and every IVR_Call the IVR_Calls table is available. It, with other tables, is summarised into the IG123470T table.

The Call-by-Call records are held for 90 days before being purged.

11.3.3.3 Call Concentrator

Call Concentrator forms a collection of a very detailed call-by-call records. This information is gathered from PABX T-Servers by the Call_Concentrator_Primary server before it, without any processing, writes the data to the Call Concentrator Database tables in the CCONp Oracle database.

The IVR information is recorded as a string of sequence numbers, indicating the specific passage of the call through the IVR system. Call-waiting times are also available.

The Call Concentrator Database has recently been moved to the Auckland Outbound Database machine, where it now resides.

The Call Concentrator Database tables are described on pages 44 to 47. These are the only Genesys records containing IVR data. A simple utility, "Brio", is available to generate SQL type reports from Oracle databases. This may be the easiest way to obtain historical Genesys data.

11.3.3.4 Port Monitor

PortMonitor collects detailed IVR information on each of the RS/6000 machines by receiving the informational messages emitted by the top-level IVR StateTables, GenStart and IStart

As this is a new application (end of 2006), at the time of writing only real-time information is displayed. In the future a database of historical records will be built up.

11.3.4 DATA COLLECTION: NON-IVR

11.3.4.1 Reporting StatServer to DataMart

No IVR relevant data is accumulated here, even though "Reporting StatServer to DataMart" is the main Genesys record keeping.

Like the Call Concentrator process, the Reporting StatServers register with the PABX TServers so as to collect a wide range of interesting call and call-centre statistics. However, this system has more of a meta-data orientation that the call-level records of the Call Concentrator. At any one time the Reporting StatServer has a detailed up-to-the-second account of the state and through-put of the Call Centres associated with the TServers that it is monitoring.

Every fifteen minutes, the DataSourcer takes a 'snap-shot' from each of the reporting StatServers and writes this snap-shot to the Operational Data Store (ODS).

Over the next few hours, or at least every day, the ETL process summarises this information, currently built up in the ODS, before writing a summary to the Data Mart.

11.3.4.2 Logging

In Genesys, all inter-process messages are handled by the Message Servers, which writes every one of these as a log record to the LOGDB database. The Message Server may identify some messages as warranting the attention of a human. This messages are sent to the Solution Control Centre (SCS) from where an appropriate alarm, or possibly a page, may be raised.

An associated process is the Local Control Agent (LCA) server monitoring the LCA on each individual machine. If the central server cannot connect to any local LCA it realises that the local machine has crashed (if the LCA isn't available Genesys will not operate) and likewise sends a message to the SCS. These messages are not logged into LOGDB.

11.3.4.3 Call Centre Monitoring

The CCPulse StatServer monitors the local TServer, gathering information relevant to the operation of the local call centre. CCPulse, and any other applications based on GIS, use this data to provide feedback regarding the state of the local Call Centre.

This data is ephemeral: no hard-storage or database is associated with this process

11.3.4.4 Outbound Workforce Management (legacy)

In the outbound contact centres, a StatServer is dedicated to recording agent activity, part of observing agent "compliance": that agents are active at the times their schedule says they should be. This StatServer data is written to a local database via the StatServer_DBServer and then analysed by the ResourcePro workforce management application.

This is non-call related data and is not gathered by Call Concentrator.

11.3.5 DATA STORAGE: RS/6000 STATE TABLE LOGS

As described, each State Table, including the top level IStart and GenStart scripts, record a log record of every call through its IVR. This effectively holds all IVR information.

As an example, the IStart top level IVR script writes log records similar to the following example:

Server=ie1360,STB=IStart,Sys Date=20061107,Sys Time=091601,Channel=96,SV188=,SV128(trunksignaltype)=24,SV32(calltype)=-1,SV185(callednumber)=0109,SV186(callingnumber)=42989443,SV189(call origin)=0,Debug=3,XFER_CDN=1078386183,GetApp,NotifyCallStartTime=091602,NotifyCallStartEndTime=091602:NotifyCallStart:result=1,GetCallInfoStartTime=091603,GetCallInfoEndTime=091603,err_code=0000,err_desc=,CLI=42989443,DNIS=0109,connid=352101637975eefd,lastEvent=EventEstablished,EventEstablishedTime=091603,HostName=ie1360:GetIVRInfo:ErrCode=0000,ErrDesc=,result=1,vrp=10001,XFER_CDN1=1078386183,IVR_SV248=12,IVR_NAME=BSS,RouteStartTime=091603,RouteStartEndTime=091603:RouteStart:vrp=10001,result=1,reqid=8937017,GetRequestStartTime=091604,GetRequestEndTime=091604:GetRequest:result=1,rreqid=8937024,rdata=|IVR|DNIS|12|LABEL|BSS|,GetRequestTime=091604:ParseRData:err_code=0000,err_desc=,type=IVR,ivr_dnis=12,route=,dest=,label=BSS,logName=BSS:SendReply:result1,rreqid=8937024,rdata=:RouteAbort:result=1,StartIVRTime=091604,seq_no=4136:RouteRequest:vrp=10002,RouteRequestTime=091655,GenRingbackStart=091655,RouteRequest:result=1,RouteRequestTime=091655,rdata=1033740280,dest=1033740280,type=Destination,GenRingbackEnd=091655,SV177=4,ChannelGroupTransfer=HSF,HSF_PAUSE=30,Dest=1033740280,TranferCallDone:CallTransfer:result=1,NORMAL EXIT,NotifyCallEnd,Sys Time=091659,ErrCode=0502

which is:

Server=ie1360
STB=IStart
Sys Date=20061107
Sys Time=091601
Channel=96
SV188=
SV128(trunksignaltype)=24
SV32(calltype)=-1
SV185(callednumber)=0109
SV186(callingnumber)=42989443
SV189(call origin)=0
Debug=3
XFER_CDN=1078386183
GetApp
NotifyCallStartTime=091602
NotifyCallStartEndTime=091602:
GetCallInfoStartTime=091603
GetCallInfoEndTime=091603
NotifyCallStart:result=1
err_code=0000
err_desc=
CLI=42989443
DNIS=0109
connid=352101637975eefd
lastEvent=EventEstablished
EventEstablishedTime=091603
HostName=ie1360:GetIVRInfo:ErrCode=0000
ErrDesc=
result=1
vrp=10001
XFER_CDN1=1078386183
IVR_SV248=12
IVR_NAME=BSS
RouteStartTime=091603
RouteStartEndTime=091603:
result=1
RouteStart:vrp=10001
reqid=8937017
GetRequestStartTime=091604
GetRequestEndTime=091604:
GetRequest:result=1
rreqid=8937024
rdata=|IVR|DNIS|12|LABEL|BSS|
GetRequestTime=091604:
err_desc=
type=IVR
ParseRData:err_code=0000
ivr_dnis=12
route=
dest=
label=BSS
logName=BSS:SendReply:result1
rreqid=8937024
rdata=:RouteAbort:result=1
StartIVRTime=091604
seq_no=4136:RouteRequest:vrp=10002
RouteRequestTime=091655
GenRingbackStart=091655
RouteRequest:result=1
RouteRequestTime=091655
rdata=1033740280
dest=1033740280
type=Destination
GenRingbackEnd=091655
SV177=4
ChannelGroupTransfer=HSF
HSF_PAUSE=30
Dest=1033740280
TranferCallDone:CallTransfer:result=1
NORMAL EXIT
NotifyCallEnd
Sys Time=091659
ErrCode=0502

Other logs are available:

Log File	Description	Log File	Description
0800r	internal	lutinddf	TSE_SubIVR
0800td	internal	moa2	PostShops_direct_IVR
08moafb	ignore	OfferSpecial	marketing
AbcMain	127	otaf	*22800 OTAF
AnytimeFront	TSE_SubIVR	payph	VicVantage
attxfer	TSE_SubIVR	PP_Nmain - check	08323232_prepay_TopUp
bfivr	125_BusFaults	PP333_Nmain - check	<removed>_prepay_balance
bsivr	126	ppivr - check	08323232_prepay_TopUp
calin	Call_investigation	PS_Main	PostShops_direct_IVR
CallSpecial	marketing	rsivr	123
ccivr	128	rsmob	IVR_to_GVP_SpeechNav
cfivr	IVR_to_GVP_SpeechNav	SmartPack	marketing
contr	<Company 1>_contractor	SP_Smarties	marketing
CRAN_main	internal	TangoReg	Freedom_Tango
FlyBuys	TSE_SubIVR	<Company 1>_Clock	TSE_SubIVR
FreeMins	<removed>_postpaid	TRAN_main	internal
FreeMins2	<removed>_postpaid	tranfail	internal
go_4_5	marketing	TseDD	TSE_SubIVR
hit2_main	credit_from_disconnect	tsexp	TSE_MainMenu
Homeline	marketing	two_on	marketing
Int_Tolls	marketing	XtraCredit	ignore
LineRental	ignore	YourCall	marketing

These logs are all parsed by TRIAD and written to its IVR_Call table. From here it is summarised into the IG12370T table which holds very detailed day-by-day summaries of each component of the IVR system's activities.

Data in the IVR_Call table is kept for 90 days before being purged.

11.3.6 DATA STORAGE: CALL CONCENTRATOR

The Call Concentrator process gathers all call data from the PABX T-Severs and writes it to the four main Call Concentrator database tables. The four tables are:

- GCDR: Global Call Detail Record table. This hold a large amount of data about the call
- SCDR: Segment Call Data Record table. Information about each segment of the call.
- EVREF: Connects the EVDATA table to the GCDR table record for the call
- EVDATA: The Attached Data added, for each call.

The AREC table, a combination of EVREF and EVDATA, is not used in the <Company 1> Call Centre environment.

< Diagram removed >

11.3.6.1 GCDR table

The Global Call Detail Record table which holds a large amount of data related to each call

Field Name	Description
agwttime	The time between the first appearance of a call in a call centre and the first answer by a logged-in agent.
Calls	The number of calls segments constituting the call.
CallType	Call type—internal, outbound, inbound, consultation, or unknown.
ConnID	A primary key that is an identification number assigned by T-Server to a call (for example, 392732680203).
Customer	A reference to a tenant.
Destination	The value taken from the ThirdPartyDN attribute of the EventRouteUsed event
DestLabel	Refers to the DN present in the ThirdPartyDN attribute of the EventRouteUsed event
DN	Refers to first DN that received or initiated the call.
Dur_Cons	Total duration of consultation call segments constituting the call.
Dur_Inb	Total duration of inbound call segments constituting the call.
Dur_Int	Total duration of internal call segments constituting the call.
Dur_Outb	Total duration of outbound call segments constituting the call.
Duration	Total call time from initial connection to termination.
First_CallID	CallID from the first call segment.
First_CDIG	Reserved.
First_DNIS	DNIS (Dialled Number Identification Service) from the first call segment.
First_PHONE	The ANI (Automatic Number Identification) service or an outbound dialled number for the first call segment.
First_QUEUE	The queue DN of the first queue that the call entered.
FirstTime_Delivery	Reserved.
FirstTime_Park	The duration of the treatment (if any) applied to the first call segment.
FirstTime_Queue	The amount of time the first call segment spent in queue.
FirstTime_Ring	The amount of time the first call segment spent ringing.
FirstTime_Rout	routing point from which it was successfully routed.
Flag_Abandoned	A flag indicating whether the call was abandoned at the physical telephony object event
HAgent	Refers to the first agent that participated in the call.
HResult	Not used.
MediaType	Captures any media type information attached by T-Server.
N_conf	The number of call segments that have been in a conference.
N_Cons	The number of consultation call segments among those constituting the call.
N_Inb	The number of inbound call segments among those constituting the call.
N_Int	The number of internal call segments among those constituting the call.
N_Outb	The number of outbound call segments among those constituting the call.
N_park	The number of call segments given a treatment.
N_queue	The number of call segments among those constituting the call that passed through a queue.
N_trans	The number of transferred call segments among those constituting the call.
Project	Reserved.
StParkTime	The timestamp of the time that treatment was applied to the first call segment (if any).
StQueueTime	The timestamp of the first call segment that entered a queue.
StRoutTime	The timestamp of the first EventRouteRequest message.
StTime	The start date and time of the call.
Switch	Refers to the switch where the call was initiated or to the first switch that received the call
TimeIn_conf	Total time spent by the call segments in a conference state.
TimeIn_park	Total time spent by the call segments in a treatment state.
TimeIn_queue	Total time spent by the call segments waiting in a queue.
TimeIn_trans	Total duration of the call segments that were transferred.
Tot_DialTime	Total time spent by the call segments in the dialling state. DialTime ends when the call is either answered or abandoned.
Tot_RingTime	Total time spent by the call segments in the ringing state.

11.3.6.2 SCDR Table

Holds Segment Call Data Records, one of which is recorded for every connection made by any call that could transmit voice (or data). For example, one segment for the IVR, one segment for the CSR, and one segment for a subsequent IVR or transfer. No segments are added while the call is being manipulated by Genesys.

Field Name	Description
ConnID	A foreign key that is an identification number assigned by T-Server to a call.
DialTime	The total time from when the call segment was first dialled until it made a connection or was abandoned.
DNIS	Dialled Number Identification Service.
EndTime	The timestamp when this call segment was terminated.
F_aban	A flag indicating whether the call segment was abandoned in queue.
F_conf	A flag indicating whether the call segment was conferenced.
F_dial	A flag indicating whether the call segment was dialled.
F_estb	A flag indicating whether the call segment was established.
F_inconf	A flag indicating whether the call segment was in conference.
F_obsrv	Not used
F_que	A flag indicating whether the call segment was in queue.
F_rels	A flag indicating whether the call segment was released.
F_retr	A flag indicating whether the call segment was retrieved.
F_ring	A flag indicating whether the call segment was rung.
F_tran	A flag indicating whether the call segment was transferred.
LocAgent	The DBID of the local agent.
LocDN	The DBID of the local agent.
LocLQ	The last local queue that the call segment passed through.
LocQueue	The first local queue the call segment passed through.
LocSwitch	The local switch.
LocTrunk	Optional number for the local trunk provided by some switches.
ParkTime	The duration of a call in a parked (or held) state.
Phone	The ANI service value for the inbound call segment or the dialled number for the outbound call segment.
RingTime	The total time from the time the call segment first rang.
RmtAgent	The remote agent.
RmtDN	The remote DN.
RmtLQ	The last remote queue that the call segment passed through.
RmtQueue	The first remote queue that the call segment passed through.
RmtSwitch	The remote switch.
RmtTrunk	The optional number for a remote trunk, provided by some switches.
RoutTime	The amount of time that the call segment spent being routed from the first routing point from which it was successfully routed.
SCallID	A primary key that is the call identification; a unique identifier for a call.
SCallType	Call type—internal, outbound, inbound, consultation, unknown.
SCSequence	The ordinal number of the SCDR table in the sequence of SCDR records having the same value ConnID.
SDuration	The duration of the call segment from the moment it was created to the moment it was terminated.
SHResult	Not used.
SSwitchCallID	The call ID assigned to the call segment by the physical switch.
SProject	Not used.
SResult	Call status.
StParkTime	The timestamp when the first treatment was applied to the call.
StQueueTime	The timestamp when the call segment entered the queue.
StRoutTime	The timestamp when the first routing of the call segment began.
StTime	The start date and time of the call segment.
WtTime	The duration of time that the call segment spent in all queues before being established or abandoned.

11.3.6.3 Attached Data Records: EVREF and EVDATA

All Genesys and user-specified data relevant to the call is appended to it in the form of Key-Value pairs. These are used throughout Genesys for call management.

These two tables, EVREF and EVDATA, are used for recording some of the attached data Key-Value-Pairs added to each call. Maybe 30% of the attached data is recorded, most is discarded.

11.3.6.3.1 EVREF

Field Name	Description
Agent	The agent object in the Configuration Database.
ConnID	If not zero, the field ties the EVREF record to the record in the GCDR table representing the call associated with this EVREF record.
DN	The DN object in the Configuration Database.
ESequence	An integer counter generated by Call Concentrator. The value ties the EVREF record to one or more records in the EVDATA table having the same value as the EVDATA.ESequence field.
EventType	Not used.
SCSequenceThe	ordinal number of the call segment in chronological order.
Time	The timestamp when the attached data was sent.

11.3.6.3.2 EVDATA

Field Name	Description
DataType	The type of user data stored in the EVDATA record.
ESequence	An integer counter generated by Call Concentrator.
KeyName	The key name retrieved from a key-value pair that was sent by an agent from a desktop phone application.
ValChar	The character value retrieved from a key-value pair that was sent by an agent from a desktop phone application.
ValInt	The integer value retrieved from a key-value pair that was sent by an agent from a desktop phone application.

11.3.7 DATA STORAGE: PORTMONITOR

At the time of writing the PortMonitor application is only operating in real time. It is likely that, over time, a database will be constructed and this information made available for historical analysis.

11.4 GENESYS: OUTBOUND CALLING

This document is a description of the IVR, which a part only of the *inbound* calling process. However a short description of the Outbound Calling set-up will be delivered here. Outbound Calling is undertaken for <Company 1>'s sales promotion campaigns, e.g. to encourage existing <Company 1> customers to add additional services to their phone line.

Before the campaign starts, analysts in marketing will load up the OCS database with tables of customers they have decided are appropriate to target. When the campaign starts, the call-centre supervisor uses SCS to add CSRs into campaign groups. This makes the OCS Server register with the PABX TServer to initiate monitoring of each added CSR's phone,.

To run an outbound campaign the numbers for Genesys to dial are extracted from the Outbound Call Service Database (OSCp) by way of the Outbound Call Server (OCS_Primary on the Auckland Genesys 1 machine). For each outbound agent in a "Ready" state the OCS server sends the next number to be dialled to the TServer, which forwards the number to the CSR's Softphone from

where it is automatically dialled. The customer's info and the number being dialled are shown on the CSR's computer screen.

The CSR talks to the customer and gathers information. The OCS system now:

- enters information gathered into the OCS database.
- Should the customer have initiated any changes to their account or telephone set-up, OCS also triggers the provisioning of these changes.
- Records the agent, call, call-centre data and statistics
- When the TServer detects that the CSR has completed the call, the OCS server, having registered with the TServer, receives the news that the CSR is free and initiates the dialling of the next potential customer.

11.4.2.1 Outbound Campaign call example

As an example, suppose a campaign's task is to increase the use of <Company 1>'s call-forwarding services.

Before the campaign starts, appropriate customers are added to database tables in the OCSp database (Outbound Call Server DB – Production). These may be customers in particular areas, with particular demographics who have not already purchased the new service.

The Call Centre supervisor adds CSRs to the campaign. This tells OCS to register with the local TServer, indicating that it wants all information about activity on those CSR's telephones.

As described above, each number in the OCSp database table(s) is dialled and the customer information displayed; such things such as what other <Company 1> utilities they have purchased and some account information.

The customer says "OK, I'll buy, but don't ever call me again". The CSR:

- records the sale
- records that the customer is not to be included in any future campaigns.
- initiates OCS Server provisioning the Call Forwarding utility on the customer's account.

OCS observes through its connection to the local TServer that the call is terminated, so it prepares the next customer to dial.

11.5 GENESYS: INTERFACES

11.5.2 INTERFACE TO RS/6000: ISERVER

This is a standard IServer, not configured in any customised manner.

11.5.3 IVR ROUTING: URS

The Universal Routing Server sends the IVR script information to the IBM RS/6000 and its IVR system. While the content of the interaction is from the URS, the actual interface is the IServer.

11.5.4 PABX T-SERVER

TServers are used by Genesys to interface to external devices. The PABX TServer is a Symposium TServer, designed to connect to a Symposium Link server, which is itself an interface to a Meridian PABX.

11.6 GENESYS: INSTALLATION / SETUP

The general approach to install a Genesys application is:

1. Install the Genesys Framework on the machine.
2. All the Genesys applications and their machine configurations set up in the Configuration Server.
3. The machine has a working network connection, and can connect to a machine running a Configuration Server.
4. LCA , the Genesys Local Control Agent installed on each machine. This application monitors and controls the inter-Genesys connections.
5. Run the Genesys application's installer. The installer reads from the Configuration Server to correctly install and configure the application.

There are no installation dependencies, beyond the LCA and connection to a Configuration Server. The sole exception is that a Java environment must be available before a Genesys Interface Server (GIS) is set up.

11.7 GENESYS: DEPENDENCIES

11.7.2 MACHINE FAILURE: AUCKLAND

11.7.2.1 Genesys 1 (unavailable)

Situation

Machine crashed

Resolution

- Call the <COMPANY 2> Hosting Group to physically reboot the machine.
- Use Solution Control Centre to restart the servers. Do not start them from the command-line or establish automatic booting of the servers
- First start the T-Servers. They will run as back-up servers to the (now primary) servers in Hamilton
- Then start the URS
- Decide whether to switch from Hamilton to Auckland as the primary T-Servers. Usually wait till the end of the day to do this.

Situation

Very quickly, after about 10 seconds, the T-Server functionality would be picked up by the Hot-Standby T-Server running on the machine in Hamilton.

Resolution

- Nothing.

Situation

CSRs on calls at the time of the crash, that end the call before the backup T-Server takes over, will appear to Genesys to still be on the call. Genesys has missed the end-of-call.

Resolution

- A group of people - “Service Events” - will, after an event such as this, look through all open calls in Genesys and test that they are really on Calls
- Call Centre supervisors will check that all the CSR’s phones are operating properly, and fix up those that aren’t
- The CSR can make an outbound call, and then hang up. Genesys will now route calls to their phone again.

Situation

The Genesys GUIs have problems when the databases and back-end servers they are connected to switch to back-up systems. CCpulse reports an error, but is OK.

Resolution

- Nothing

11.7.2.2 Genesys 2 (unavailable)

Situation

The Reporting StatServer would switch over to Hamilton. No logs would be written for the 10 seconds or so of the switch

Resolution

- Nothing

Situation

The Configuration Proxy runs on this machine. All the Genesys applications connect to this, and not the Configuration Server itself, so each would have a brief period of disturbance while they switched over to the back-up in Hamilton. However the Genesys Information Server does not switch to the backup correctly. Thus the CSR's 'readerboad' desktop application, while continuing to run, will not accommodate any changes.

Resolution

- Nothing

11.7.2.3 Outbound DB Server (unavailable)

Situation

The Outbound Database server would be unavailable. No outbound calls could be made.

Resolution

- Nothing

Situation

The Configuration Server Database would be unavailable

Resolution

- Changes to the Genesys system could be made by writing to the back-up database in Hamilton, but then one encounters the problem of the two databases being out of synchronisation. However, in a situation with unavailable machines or other problems it might be necessary to make configuration changes and deal with synchronising the configuration databases later.

11.7.2.4 Data Collection Server (unavailable)

Situation

No Genesys system statistics would be collected ; historical data recording would cease. Specifically:

- the 15 minute summaries that are the core of Genesys reporting would not be available.
- the every-20-seconds' workforce management data transfers to the Resource Pro system would stop

Resolution

- Nothing. The data is lost.

Situation

The absence of 15-minute data summaries will reduce the hourly, daily, weekly, monthly, etc totals and statistics.

Resolution

- Nothing; unless one wants to replace the data-gaps with averages or estimates. Regardless, the unique aspects of the data are lost.

Genesys will automatically page a Genesys system administrator.

11.7.2.5 Reporting DB Server (unavailable)

The Reporting DB Server is the final repository of system performance figures collected by Genesys.

The Reporting DB Server should be restarted via use of the Genesys Solution Control interface. Genesys servers should not be run directly.

Situation

While no data would be added to the machine's databases, the originating machines cache their own data collections. When the Reporting DB server comes back up, the cached data is transferred to it.

The Reporting DB server was down for three weeks at one time. Data was cached on the originating machines, in 2 GB files, and was delivered when the machine returned.

Resolution

- Nothing.

Situation

Outbound Call data is written directly to the DataMart after every call. While the Reporting DB Server is down this data is irretrievably lost.

Resolution

- Nothing.

Situation

Up to date reports and statistics would be unavailable.

Resolution

- Nothing.

11.7.3 MACHINE FAILURE: HAMILTON

11.7.3.1 Genesys 1 (unavailable)

This box functions as a hot-backup to the Auckland machines. It becomes the primary server when the Auckland Genesys 1 machine is out of service for whatever reason, and may remain the primary Genesys server for a while afterwards, possibly until the end of the day.

The Hamilton Genesys 1 machine should be restarted via use of the Genesys Solution Control interface. Genesys servers should not be run directly

Situation

If, as usual, the Hamilton Genesys 1 machine is not operating as the Primary server nothing would be observed.

Resolution

- Restart the machine normally. It will start up, correctly, as a back-up server.

Situation

All the IVRs would be run on the Auckland IVR servers.

Resolution

- Nothing.

11.7.3.2 Genesys 2 (unavailable)

This box functions as a hot-backup to the Auckland machine. It becomes the primary server when the Auckland Genesys 2 machine is out of service for whatever reason, and may remain the primary Genesys server for a while afterwards, possibly until the end of the day.

The Hamilton Genesys 2 machine should be restarted via use of the Genesys Solution Control interface. Genesys servers should not be run directly

Situation

- If, as usual, the Hamilton Genesys 2 machine is not operating as the Primary server nothing would be observed.

Resolution

- Restart the machine normally. It will start up, correctly, as a back-up server.

11.7.3.3 Outbound Database Server (unavailable)

This box functions as a hot-backup to the Auckland machine. It becomes the primary server when the Auckland Outbound Database Server machine is out of service for whatever reason, and may remain the primary Genesys server for a while afterwards, possibly until the end of the day.

The Hamilton Outbound Database Server machine should be restarted via use of the Genesys Solution Control interface. Genesys servers should not be run directly

Situation

If, as usual, the Hamilton Outbound Database Server machine is not operating as the Primary server nothing would be observed.

Resolution

- Restart the machine normally. It will start up, correctly, as a back-up server.

11.7.4 INDIVIDUAL APPLICATION FAILURE

It is very rare for an individual server application to fail and all important servers have hot-standby running on effectively identical machines in Hamilton. If an application crashed there may be a delay of 5 – 10 seconds for full functionality to be available; maybe 5 seconds longer – to allow for a connection – for warm standby machines.

12 LOCAL GENESYS

For faster response, and lower network load, boxes are set up in each Call Centre running a local GIS and a local StatServer for CCPulse. Most of the Call Centre monitoring is about the local call centre.

12.1 LOCAL GENESYS: MACHINES' SERVERS

12.1.2 GENESYS INTERFACE SERVER (GIS)

The Genesys Interface Server provides each CSR with a display of the call centre total, how many calls waiting, how many CSRs available and similar information. Each CSR's Softphone establishes a direct link to the local GIS server, and each GIS server in turn receives call information from every call centre's PABX New Zealand-wide. However, the usual configuration is for a local server to ignore non-local events, only totalling and reporting those of interest to the local site. This data is not recorded: it is only for immediate display on the registered Softphones.

The GIS gets its set-up information from either the Hamilton or Auckland Configuration Server Proxy.

12.1.3 STATSERVER FOR CCPULSE

CCPulse is the application for monitoring Genesys call centres, the supervisors' meta-equivalent to the CSR's Softphone.

The StatServer monitors and summarises the activity over the call centre's PABX, and CCPulse gather's and displays this data, as required. There is no database involved; all data are kept in memory.

12.2 LOCAL GENESYS: DEPENDENCIES

12.2.2 GIS SERVER (UNAVAILABLE)

Situation

None of the CSR's would have Call Centre summary information available. Local GIS servers are not, currently, able to switch to the back-up Genesys Config Server proxy, as they should do when the Primary Configuration Server Proxy is unavailable. While the CSRs still receive Call Centre statistics, changes, such as a CSR starting or changing phone lines, will be incorrectly accounted for.

Resolution

- Restart the GIS normally, using the Genesys Solution Control Interface.

12.2.3 LOCAL STATSERVER SERVER (UNAVAILABLE)

Situation

None of the Supervisors' CCPulse applications would display appropriate Call Centre summary information.

Resolution

- Restart the StatServer normally, using the Genesys Solution Control Interface.

13 SYMPOSIMUM LINK SERVERS

The Symposium Link machines are dedicated to running the Genesys “Symposium Link” servers, a custom interface between a Meridian PABX and the Genesys PABX TServer. There is one of these in each call centre.

13.1.2 INTERFACES

On the Genesys side a standard PABX TServer is used. The Symposium Link server is itself an interface to the Call Centre’s PABX.

13.1.3 SYMPOSIUM LINK (UNAVAILABLE)

Situation

If the server goes down, Genesys looses all connection to the call centre. It can no longer see that there are any phones there and ceases sending any calls. The call centre goes quiet and both Softphones and CCPulse cease to work.

Resolution

- There may be a range of solutions; the last time this happened skills were adjusted in other call centres so that they started to receive the calls that would have gone to the un-available call centre. Later it may have been possible for call-plans to changed in either ICM or on the IVR machines so as to send the calls directly to default queues on the PABXs. The problem with this is that one doesn’t know that the problem is not with the PABX itself.
- After the problem was investigated, the Symposium Link machine was re-booted and routing was returned to normal.

Situation

Uncertainty that the PABX itself does not have faults.

Resolution

- Nothing

14 <COMPANY 1> CALL-CENTRE ENVIRONMENT

< Diagram removed >

15 NICE CALL MONITORING AND RECORDING

The NICE call centre monitoring suite is one of a range of telephony products from the Israeli company NICE Systems. It runs in three layers:

- at the call centre level. The system is controlled at this level.
- one single central repository
- a few programs running on each agent's desktop machine

< Diagram removed >

15.1 CONTROLLING THE NICE SYSTEM

15.1.2 NICE STORAGE CENTRE

A centralised server running on a dedicated machine, HP1325, in Auckland, that holds Call Logger and Screen Logger records. This machine receives its data from each Call centre, from the NICE Screen Logger running on the call centre's "Local Genesys" machine, and from the dedicated call logging machine.

15.1.3 NICE CLS (CALL LOGGING SYSTEM)

Each Call Centre runs the NICE CLS application to manage all the other NICE components; all of the CLS applications themselves, however, run on a single dedicated box in Auckland, HP1324.

The NICE CLS:

- records and implements the decisions as to which CSRs are monitored and when.
- interfaces to a SQL Server 2000 database
- is the 'master controller' of the NICE system.

15.1.4 THE NICE TOOLBAR

A toolbar is available on the supervisors' desktops providing access to

- NICE Query, to study the records of stored data.
- NICE Administrator, to set up agents, their T-numbers (for Genesys) and extensions.
- NICE Monitor, to observe agents in real time.
- NICE Scheduler, to program many possible patterns of observing and recording agents.

Together these applications, appearing on the NICE toolbar, are also labelled the "NICE Suite".

15.2 NICE CALL CENTRE SERVERS

15.2.2 NICE CALL LOGGER

In each Call Centre a dedicated machine runs the NICE Call Logger. This obviously connects to each of the CSR's desktop boxes as well as to the central NICE Storage Centre.

15.2.3 NICE SCREEN LOGGER

Running on each call centre's 'Local Genesys' machine the screen logger collects screen shots for uploading to the central NICE Storage Centre

15.3 NICE SYSTEM ON THE CSR'S MACHINES

15.3.2 "GENESYS" SOFTPHONE

Amongst the other features of the "Genesys" Softphone is a button to trigger one of, or both, voice recording and / or screen recording. The Softphone connects to both the NICE Interactive Controller to record voice, and to the NICE Screen Agent to take screen shots, together implementing what has been named "Record on Demand (ROD)".

RODs are part of the solution to a Commerce Commission requirement to have evidence of one Telephony company having Customer permission to change to another Telephony Company. The Recordings, as part of the stipulated compliance program, are referred to as "winbacks".

15.3.3 NICE INTERACTIVE CONTROLLER

Also known as the NICE Link, the Interactive Controller records voice from the agent's softphone, sending it to the NICE Call Logger application

15.3.4 NICE SCREEN AGENT

This program, also running on the CSR's machine, sends Screen Records from the softphone to the call centre's NICE Screen Logger server. Both screen recording and voice recording can be triggered by the CSR's softphone, remotely by the supervisor, or remotely from a predetermined schedule of monitoring.

16 APPENDICES

16.1 RELATED EXTERNAL SYSTEMS

16.1.2 SOFTPHONE

The Genesys Softphone Toolbar is an application that has been built around the Genesys ActiveX Desktop Toolkit 7.0 client. The Softphone takes on the functionality of the CSR's phone, physically controlling the basic call retrieval, transfers and hangups allowing them to take calls hands-free. The Softphone interacts with other applications, namely INFO and GeneCISS (soon to be decommissioned). The Softphone is a Visual Basic application that has an access database to enable call list functionality.

Configuration of the Softphone is done at twp levels, either through amendments done within their applications' toolbar.ini file. The TCA Configuration Utility 3.0 is an application that is being rolled out with the Softphone to allow a gui interface to do these amendments.

GISstat7 is another client that runs in the background. This communicates with the Statserver and passes current statistics to the Users Softphone.

16.1.3 PINAPPS

There are three front end applications which integrate with the PIN Application. There are:

- An Administration Application known as pinapp2002 which is run on CSRs' desktops within the <removed> call centres. This is a Visual Basic application developed by the CTI/IVR Team.
- A Test Harness application implemented as a web application under Apache Tomcat, used as a test tool,
- The IVR Applications themselves access the PIN Application via the custom servers WXYZ_TUX, and WXYZ_TUX_2.

The PIN Application server itself is implemented as a free-standing java application which exposes a CORBA interface for client applications to call it.

The PIN Applications has a generic connector plug architecture so it can connect to back end systems. The connector for interfacing with ICMS is the JTB (AS400 Java Toolbox) connector. The connector for integrating with the Vantive VIC system is a generic CORBA connector which, in turn, connects to another free standing servant application running on the same platform as the Vantive VIC system: the VIC connector. The VIC Connector uses VANAPI to access routines within the Vantive run time.

The PIN connector is used to implement PIN Validation/Administration services which use JDBC to access the PIN database.

< Diagram removed >

Security has been built into the IVRs such that when a Customer asks for account information they can be prompted for a pin number to verify that they are allowed access to this information. The IVR communicates with the pinapp Servers SN881 and SN882 to validate the pin number given. If three failed attempts are made, the pin is locked and the Customer has to have this reset.

16.1.4 ICMS

ICMS is <Company 1>'s Customer management application. In the IVR environment it is accessed directly by the PINAPPS interface, IS_EAI, when adding a customer to the Tango program, or when using EAI_OLTP to action financial payments and checks, such as during cell-phone top ups.

< Diagram removed >

16.1.5 OTAF

< Diagram removed >

Integration with the OTAF environment is somewhat complex because it requires asynchronous interactions, using callbacks to propagate events back to the calling interface. State Tables do not support events driven by callbacks. Thus, every request has to be stored in a shared memory table, and a proxy, the otaf_intf_process, updates a shared memory table with response when they arrive. The custom server (WXYZ_Crossworlds) periodically checks shared memory to see when a response has been received. At the same time, the state table is playing music on hold (or talking) to the customer to keep them on the line while their mobile telephone is being programmed.

The custom server WXYZ_Crossworlds communicates with the proxy process otaf_intf_process via a SystemV message queue and a block of shared memory. The otaf_intf_process uses a CORBA client side interface for the forwards connection to the IVR Connector running on a MPA WBIS instance. It then uses a CORBA server-side interface for the asynchronous response connection back from the IVR connector. The MPA WBIS Instance uses a standard AS400 connector (custom built) which communicates with the ICMS via the AS400 Java Toolbox, RMI. A purpose built OTAF connector is used to communicate with the OTAF server (and MSC etc) for the purpose of programming the customer's mobile telephone.

16.1.6 SINGLE LINE TESTING

The SLT Server connects to the NAS system. The NAS in turn uses the NEAX machines to actually test the line. TSE, Faults and Business SLT IVRs all communicate with the HN247 NAS Server and report back to the customer whether or not NAS has detected any issue related the phone number they have provided.

In more detail, Single Line Testing is a function provided by the NTS (National Test System). It involves instructing a NEAX exchange to move a test head over a suspect faulty line and listen on the line for signs of a fault. NTS will then respond with a VER code appropriate for the test result. NTS is accessed by the NAS (National Activations Service) via an asynchronous CORBA interface.

16.1.7 OLTP

Online Transaction Processing implements <Company 1>’s payments and account checks. In the new architecture, as shown in the following diagram:

- WS_EAI is the custom server that accesses the new EAI services exposed by SOAP.
- The EAI layer is implemented using a Websphere Business Integration Server.
- Voucher Topup Functionality is provided by the replacement VSP components, USMS and VMS. USMS is a package system that does not support SOAP integration. Instead, it uses CORBA. So a CORBA gateway (CW SAI GW) is used to allow the USMS to access the EAI.
- Access from the EAI to USMS is via a purpose-built CORBA connector.
- The new billing system is a third-party product, Single eView, provided by ADC.
- Access from the EAI to Single eView is via a purpose-built TUXEDO connector.

< Diagram removed >

16.1.8 TRIAD, PROBE

Each call through the IVR system writes a detailed record to one or more RS/6000 IVR logs. These are discussed in Data Storage: RS/6000 State Table Logs on page 42.

TRIAD collects daily log files from RS/6000 machines and stores them in its IVR_Call table. From here, the data is summarised into the IG12370T table, which is a very detailed day-by-day summary of each component of the IVR system's activities. Data in the IVR_Call table is kept for ninety days before being purged.

PROBE is a Data Warehousing repository used to store, manage and support <Company 1>'s enterprise business intelligence data. It is widely used across <Company 1>'s Marketing and Finance sector community to provide reporting services, to promote customer markets and to build customer relationships. As <Company 1>'s Data Warehouse, PROBE is a vast system, with TeraBytes of data and thousands of users.

A modified version of the IVR log data that has been transferred into TRIAD is then copied into PROBE, from where it may be accessed by any authorized information-seeking user.