



Comparing Fax Implementations Using Dialogic® APIs and Dialogic® Diva® SDK APIs



Executive Summary

This application note compares fax implementations using Dialogic® Global Call and Dialogic® R4 APIs to those using the Dialogic® Diva® Software Development Kit 4.5 (Diva SDK 4.5) API. It can also be referenced when porting an existing application from the Global Call and R4 APIs to the Diva API, or for integrating applications that can use both types of APIs.



Table of Contents

Introduction.....	2
A Note on Terminology.....	2
Sample Configuration.....	2
Getting Started.....	2
Basic Information.....	2
Important Concepts.....	2
Sample Application Overview.....	4
Running the Application.....	4
Application Files.....	4
Application Architecture and Sample Porting Process.....	4
Introduction.....	4
Function and State Overview.....	7
Sample Application Porting Process.....	7
Summary.....	13
Acronyms.....	14
For More Information.....	14

Introduction

Dialogic® boards with Springware or DM3 architecture and Dialogic® Diva® boards with Dialogic® Diva® architecture offer a robust set of media resources. Both board technologies offer developers C/C++ programming interfaces to allow for customer implementation of different types of media solutions, including fax.

This application note compares the fax implementations using Dialogic Global Call and Dialogic R4 APIs to those using the Diva API in the Diva Software Development Kit 4.5 (Diva SDK 4.5). A fax application is used as the basis for comparison. Because the comparison highlights the key differences between the Global Call and R4 APIs and the Diva API for implementing fax applications, it can be used as a reference when modifying an existing fax application using the Global Call and R4 APIs so that the application can also use the fax functionality of the Diva API.

A Note on Terminology

Two types of Dialogic board technologies are discussed in this application note:

- **Dialogic boards** — Dialogic boards with Springware or DM3 architecture will be referred to as Dialogic boards.
- **Dialogic Diva boards** — Dialogic Diva boards with Diva architecture will be referred to as Diva boards.

These boards use different APIs:

- **GC/R4 APIs** — Dialogic® System Release software contains the Global Call and R4 APIs used by Dialogic boards. These APIs will be referred to as GC/R4 APIs.
- **Diva APIs** — The Diva SDK contains the Dialogic® Diva® Component APIs. These APIs will be referred to as the Diva APIs.

Sample Configuration

The following configuration was used in preparing this application note and in designing and testing the sample application DivaGcR4FaxDemo. A Zip file containing the DivaGcR4FaxDemo and other components can be downloaded from the web (see the *For More Information* section).

Dialogic System Release Software and Boards

- Dialogic® System Release 6.0 PCI for Windows®
- Dialogic® DM/IP481-2T1-PCI-100BT
- Dialogic® D/480JCT-2T1

The DM/IP481-2T1 board is used for T1 ISDN call control and has DM3 architecture. The D/480JCT-2T1 board has Springware architecture. They are both supported by the same System Release software, namely System Release 6.0 PCI for Windows.

Diva Board and Diva SDK 4.5

- Dialogic® Diva® Driver for Windows, Version 8.2
- Dialogic® Diva® PRI/T1-24 Media Board

Getting Started

This section discusses some basic concepts to keep in mind when porting from the GC/R4 APIs to the Diva APIs.

Basic Information

This application note builds on a previous application note published by Dialogic, *Using Dialogic® Boards with DM3/Springware and Diva Architecture in a Common Server* (see the *For More Information* section). It contains information about installing and configuring Dialogic and Dialogic Diva boards, and a basic comparison of the GC/R4 and Diva APIs.

The demo application (DivaGcR4FaxDemo) provided with this second application note also uses a similar architecture to the demo application included with the first application note (DivaGCR4AnsrDemo). A Zip file containing DivaGCR4AnsrDemo and other components can be downloaded from the web (see the *For More Information* section).

Important Concepts

This section contains information about concepts that are relevant when porting Diva API functions to an application originally written to use the GC/R4 APIs. The concepts are:

- Call versus Resource
- Individual Channel Addressing
- Incoming/Outgoing Events
- Release Call Event
- Fewer Events

The examples used to explain these concepts relate to handling faxes.

For information on altering the Diva API application state machine, see the Diva SDK 4.5 online documentation (see *For More Information*).

Call versus Resource

When using the Diva API, each call is given a type; for example, a fax call is one type. Setting the call type automatically causes the Diva API to allocate the correct resources for that type. In contrast, using the GC/R4 API requires the call and resource to be managed separately. For example, a fax call and fax resources must be managed separately.

If detailed call progress analysis is needed when using the Diva API, the call can first be connected as a voice call and later switched to *fax* mode. For additional information on how to modify call modes in runtime, refer to the DivaSetCallType functions in the *Diva API Developer's Reference Guide* (see *For More Information*).

The different types of APIs require different application models:

- **GC/R4** — The fax call is connected using Global Call. Next, the resource is routed, and the fax is started.
- **Diva** — The call type is specified, and the API connects and routes the device. The Diva API handles fax tone detection and does not return a connected state if a fax is not detected.

The different models may require some redesign if the application using the GC/R4 API uses a simple fax resource pool. The fax send/receive must select the fax mode at the start of the call to identify the call type rather than immediately posting a connect function. Also, if separate data structures are used for the fax resource and for the call control, the data structures must be merged.

Individual Channel Addressing

The Diva API automatically opens all devices, and it is not necessary to manually control each time slot separately. The Diva API allocates time slots automatically as calls are made and received. In addition, no handles are available for referencing individual time slots. If an application uses handles as a reference or for a map, an alternative addressing method must be used (for example, a lookup using a lookup of Call Reference Number [CRN]).

Because line, slot, and DNIS information can be obtained using DivaCallInfo, an internal application handle can be invented based on this data. An alternative method is to track call instances rather than ports/channels. This strategy works especially well when all channels are used for fax because the channels effectively become a *pool* of like resources, which do not require individual names.

To create a pool of like resources, build a *call instance* structure when each call arrives and destroy it when the call terminates. Relevant information can be stored, such as page count, DNIS, ANI, fax mode, and the hSdkCall handle that is passed to Diva API functions. The address of this call instance structure can be passed, instead of the *index* used in the sample application provided with this application note, allowing each event to carry with it the data associated with a call.

This structure would then abstract out the existing data structure(s). This structure is not used here, but the Diva SDK 4.5 sample code has examples for using this structure.

Incoming/Outgoing Events

The Diva API uses the same events to connect and disconnect both incoming and outgoing calls. In implementations using the GC/R4 API, the events differ (for example, GCEV_ANSWER versus GCEV_CONNECTED) so it may be necessary to track call direction to differentiate the application state machine.

The current direction of a call and the reason for any disconnects, along with other event-specific information, can be obtained using DivaGetCallInfo.

Release Call Event

The Diva API does not generate an event equivalent to the GC/R4 API's ReleaseCall function. The application state machine for the Diva API can be altered to handle ReleaseCall functionality.

Fewer Events

The Diva API handles a considerable amount of internal resource allocation and call control, resulting in fewer steps and work for the application — no route functions, CPA, etc. If additional logic is triggered by any of the *missing* steps, the application state machine may have to be altered. The Diva SDK 4.5 documentation provides information about call status events that can be enabled to resolve this problem.

Sample Application Overview

Running the Application

The fax application takes several command line arguments. The usage is as follows:

```
DIVAGCR4FaxDemo [-?] [-f filename] [-i #] [-o #] [-diva]
```

Where:

- ? prints the help
- f sets the fax filelist
- i sets the number of incoming channels, default=1
- o sets the number of outgoing channels, default=1
- diva sets application into diva mode, default=off (GCR4 mode)

Application Files

- DivaGcR4FaxDemo.exe — The executable for the application.
- FaxList.txt — The file list that contains the fax file names and numbers that are used for outgoing faxing. If this file is not present, default faxname and numbers are used.
- Default.tif — The sample fax file that is used by the application. Use a different file by specifying the name inside the FaxList.txt.
- DivaGcR4FaxDemo.cpp — The source file for the application. All logic was placed inside a single file for ease of use.
- InboundXXX.tif — The incoming faxes that have been received.

Application Architecture and Sample Porting Process

Introduction

The purpose of this application note is to show similar functionality of Diva SDK 4.5 and GC/R4 API product lines in a combined application. In order to do this, each application state processing was broken down into a single function. That function contains one section for each technology. When needed, the application passes as an argument for which section to execute. The only exception to this is the ProcessEvent handlers, for which a separate argument is used for each technology.

The application that has been modified is a simple Global Call-based fax demo. The application does the following:

- Detect and allocate available fax resources
- Answer an incoming call and attempt to receive faxes
- Make an outgoing fax call and send files from a fax list

For other fax scenarios a programmer may want to implement, see the programming guides listed in the *For More Information* section. In addition, other sample applications can be reviewed, as follows:

Diva:

FaxOutSimple

FaxInSimple

FaxInProcedureInterrupt

FaxServer

FaxMultiThreads

These Diva applications are part of the Diva SDK 4.5 and are also available (see the *For More Information* section).

GC/R4:

SCFax

GC Basic Call Model

These GC/R4 apps are shipped as part of the Dialogic System Release and are located in the demo directory.

This fax application was developed using the C API. It is a single-threaded application and used the ASYNC Callback model for both technologies.

States

This fax application was based on an architecture similar to the one referenced inside the application note, *Using Dialogic® Boards with DM3/Springware and Diva Server Architecture in a Common Server*. Additional information about the similarities and differences between the APIs in the various states can be found inside that application note.

Figure 1 shows the generic application flow and Figure 2 shows the GC/R4 application flow.

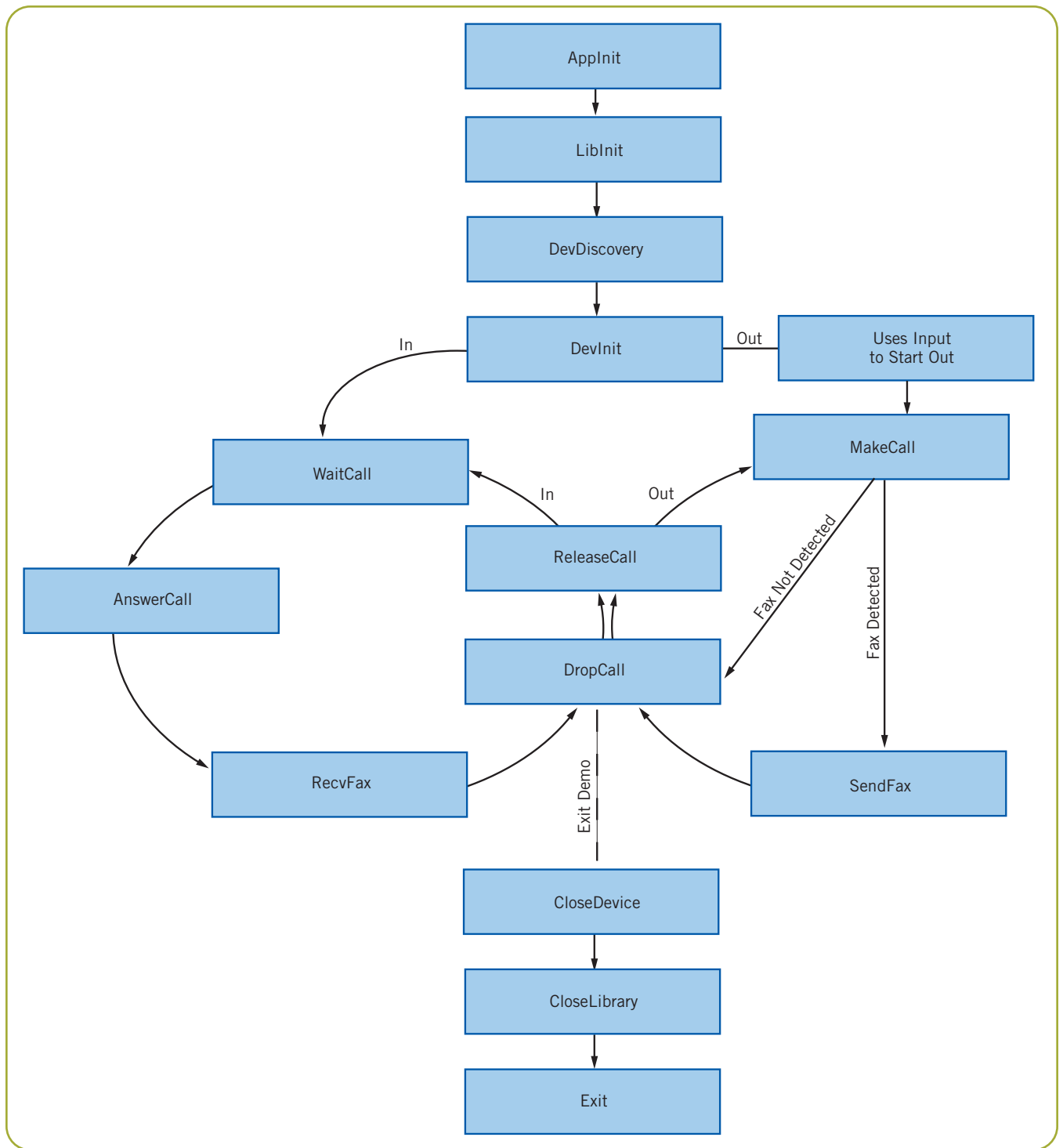


Figure 1. Generic Application Flow

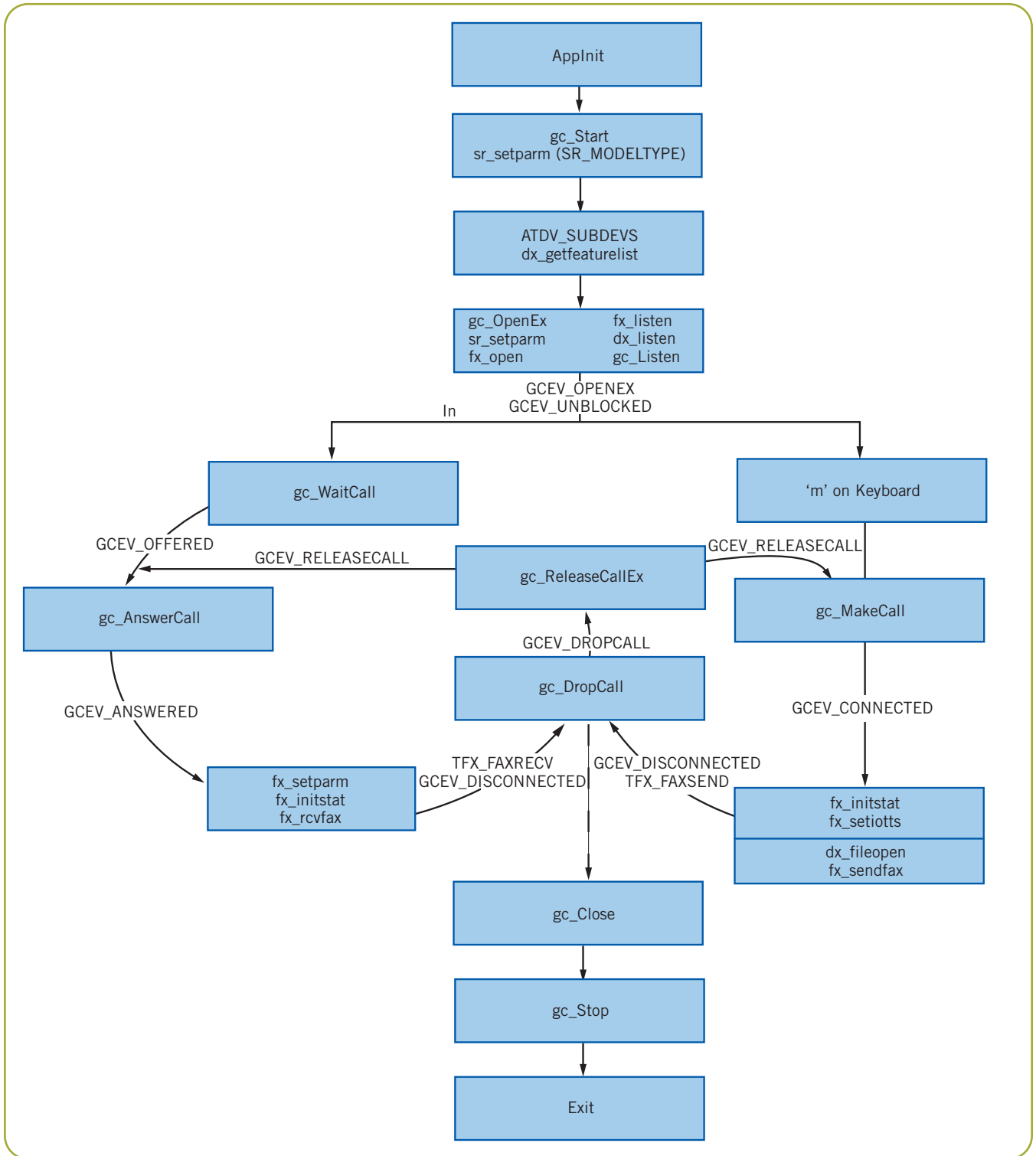


Figure 2. GC/R4 Application Flow

Function and State Overview

- **Application Init** — AppInit initializes the application. The command lines are parsed, and all data structures and global variables are initialized.
- **Library Init** — LibInit initializes the GC/R4 and Diva API libraries.
- **Device Discovery** — DevDiscovery discovers the fax devices in the system.
- **Device Init** — DevInit opens all devices and registers them with their callback handlers.
- **WaitCall** — WaitCall puts a device in wait mode for incoming calls.
- **AnswerCall** — AnswerCall answers incoming calls.
- **RecvFax** — RecvFax receives and incoming fax. These faxes are named InboundXXX.tif, where XXX is a number that increments with each new incoming fax.
- **DropCall** — DropCall disconnects an active call on the line.
- **ReleaseCall** — ReleaseCall releases the call on the line.
- **MakeCall** — MakeCall makes an outgoing call. Numbers for the outgoing call are selected from the FaxList.txt.
- **SendFax** — MakeCall transmits a fax to the outgoing call. Fax sent is selected from the FaxList.txt.
- **CloseDevice** — CloseDevice resets and closes a device.
- **CloseLibrary** — CloseLibrary closes and deinitializes support for the library.

Sample Application Porting Process

This section discusses the steps taken to add the Diva API support inside the fax application.

Note: In the final source code, sections that were modified are tagged with the DIVAADD comment to highlight what was added.

The approach was to follow through the application state machine for the original GC/R4 implementation (see Figure 2) and add in sections with their Diva API equivalents. Then, after each of the functions was updated, the Event Processor was updated to include the Events and transitions to tie the application state machines together.

The new application flow diagram for the Diva Component is shown in Figure 3.

The following sections describe what was added to or modified in the global variables, defines, event handler, and functions for the Diva Application Flow.

Global Variables and Defines

The global variable and defines were added or modified in the following order:

The includes and Lib files for the Diva SDK 4.5 were added. The include for dssdk.h was added in the include section, and the dsSDK.lib was added in the Lib section.

Note: VC directory settings may need to be changed to point to the location of the Diva SDK 4.5 files.

The useDivaFlag was added. This flag was used in each of the functions to indicate if the Diva or GC/R4 section should be executed. This flag is set via the `-diva` command line option.

The channel info structure was modified to include some of the Diva-specific information. The hSdkCall is the handle to the call similar to the CRN; the line and channel are there to reflect what line and channel the Diva index is pointing to.

The DivaAppHandle was added. This is the application handle that is used in the Diva API functions.

DivaEvtHdr

This event handler was added to process the Diva events. When the porting started, this function was empty except for a print. Then, as the other state functions were modified, the termination events for each of the functions were added into the function to navigate through the state machine.

Application Init

The support for the `-diva` was added to the command line option to toggle Diva application logic.

Library Ini

The ability to check the Diva Version was added, as well as the call to DivaInitialize to initialize the Diva Library for use inside the application.

Device Discovery

Inside this function, a DivaGetNumLineDevices was used to obtain the number of line devices. Then a loop was used to get the information on each of these line devices.

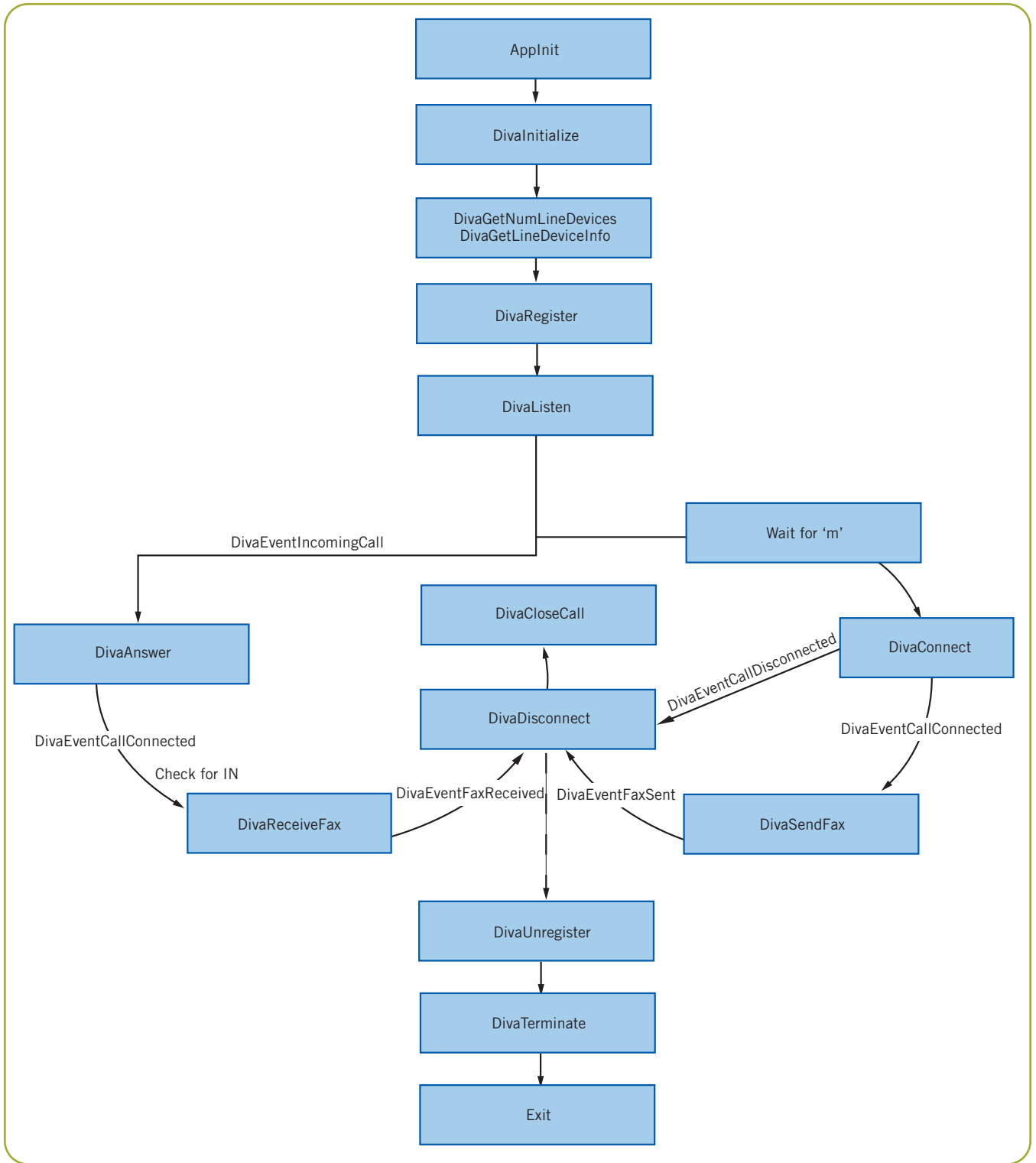


Figure 3. Diva Application Flow

Inside that `DivaLineDeviceInfo` struct, a bit indicates if the line device is fax capable. If that bit is set, that indicates that devices on that line device are capable, and the number of devices was added to the fax pool.

Note: Individual devices are not addressable and are automatically allocated to the call, so there is no device list to allocate.

Device Init

In this function, the `DivaRegister` was added. With this function, the number of channels was specified and associated in the `Callback` handler.

This single call to `DivaRegister` opens the support for the number of calls specified. There is no need to open each device individually.

At this point, the port info structure was initialized.

Note: The lack of a unique device handle is a different paradigm to that used in `Global Call`, so here are two frequently used approaches for device addressing:

1. Use the `Line/Channel` numbers as an identifier.
2. Use the `hSdkCall` as a search point for active calls (that is, convert the storage structure to a pool rather than a 1:1 mapping).

If individual ports have a meaning or special function (for example, on the analog boards where each channel is connected to a different physical phone line), then read the `DNIS` (called number) from the `DivaCallInfo` structure, and use this to decide how to process the call. To control which channels are for incoming calls and which are not, use the `Diva Configuration Manager` to set lines for in, out, or both.

WaitCall

The `Diva` API call `DivaListen` was added. This enabled detection of incoming calls.

When an incoming call was presented, the `DivaEventIncomingCall` was generated. So at this time, the `DivaEventIncomingCall` event processing was added to the `Event Handler` (`DivaEvtHdlr`).

Inside the processing of this event, it was necessary to find a port structure to use. In this section, the second approach mentioned in `Device Init` was used and a free port structure was searched for by the `CRN` (`hSdkCall`).

This `hSdkCall` was stored in the `Param1`. A port structure was searched for that does not have an `hSdkCall`, and ownership was seized by updating the `hSdkCall` with that `CRN`.

Note: If approach 1 is used, the `CallInfo` is queried to get the line and channel numbers. The line and channel information is populated in the `Dev Discovery`, the line count and subdev is determined for each line, and the line and channel information is then used find the unique port data structure. See the commented code in the `DivaGcR4FaxDemo`.

The direction was then set to incoming. This was necessary because in the `Diva` API, some events overlap for both incoming and outgoing (`id CallConnected`, `CallDisconnected`, etc.) and it is necessary to know the direction.

At this point, the `AnswerCall` was called to answer the incoming call.

AnswerCall

In this section the `DivaAnswer` function was used to answer the call.

The `Diva` API allows the user to pass a user context data into the function. This data is delivered to the event handler as one of the parameters. In this case, the index was stored into the info structure.

At this point, the type of call to answer as was passed. In this case, a fax call was wanted. This caused the `Diva` API to not only answer the call, but confirm that it is a fax call before returning the `DivaEventCallConnected`. The fax functions such as device allocation and fax call detection are handled automatically by the `Diva` API. If a fax call cannot be detected, then a disconnect is generated.

Note: It is possible to answer the call as another type and then change later. This is done by changing the call properties after the call is connected, or by using `DivaSetCallType`. This can be useful for `CPA`; for example, connect a call in *speech* mode, and then listen for tones or speech before deciding to connect the call as a fax call.

This `Answer` causes the `DivaEventCallConnected` to be generated, so this was added to the `EventHandler` (`DivaEvtHdlr`).

The index was then delivered as Param1, so there was no need for additional searching.

At this point, the call was connected and the fax reception was to take place.

RecvFax

In this function, the DivaReceiveFax function was used.

This function takes the call handle as the first parameter. This is because the fax devices are tied to the call rather than as separate entities.

Also, this function takes just the filename for the incoming fax. Like the fx_ equivalent, it takes care of opening and closing the file automatically.

Then the application set the mode to FAX via the DivaSetCallTypeFax function. See the *Diva API Developer's Reference Guide* for additional modes that can be set.

The Fax reception generated several events. At the end of each page the DivaEventFaxPageReceived event was generated. When the fax was complete, the DivaEventFaxReceived was received.

These events were added in the DivaEvtHdlr. For each of them, the call info was obtained to print out the specifics of the transmission and the fax receptions (num pages, encoding, speed, ECM, etc.).

After the fax was received, the DropCall was issued.

DropCall

DivaDisconnect was added to this section. This function drops the active call.

When this was complete, the DivaEventCallDisconnected event was received

In the Diva SDK 4.5, the same DivaEventCallDisconnected was generated from a call drop due to the issuing of a DivaDisconnect and for the far side network drop. So the DivaEventCallDisconnected is equivalent to GCEV_DROP_CALL and GCEV_DISCONNECTED. The application proceeded directly on to ReleaseCall.

Note: There is no need to call DropCall in the case of far end disconnect.

ReleaseCall

Two main things were done inside this section:

First, the call was released by calling DivaCloseCall. Note that the function executes immediately and does not generate a termination event similar to gc_ReleaseCallEx. If the application used the ReleaseCall event to transition to the state machine, it is necessary to combine that state with the DivaEventCallDisconnected event.

Second, the call handle (hSdkCall) was reset to 0 so that that data structure entry could be used for another call.

MakeCall

The DivaConnect was used to make the outgoing call. Note that the type DivaCallTypeFax was passed into the function. This caused the Diva SDK 4.5 to automatically attempt to negotiate a fax call on the outgoing call.

Unlike the GC/R4 implementation, the Diva SDK 4.5 does not return a termination event unless the call is connected AND a fax machine is detected on the remote end. If the call is connected, but a fax machine is not detected, the call results in a DivaEventCallDisconnected.

Also in this function, the index to the info structure was passed in as an argument. This index was then able to be retrieved by the event handler. This is identical to what was done on the incoming side with the AnswerCall.

Two events are possible from the DivaConnect: DivaEventCallDisconnected or DivaEventCallConnected

The Disconnect logic is in place from the incoming side. The only thing that was added was a check to see if the channel in question was an outgoing channel so a new outgoing call could be made.

The DivaEventCallConnected logic also checks direction, and if the direction is outgoing it calls SendFax.

SendFax

DivaSendFax was used to send the fax specified in the FaxList.txt.

DivaSendFax takes in as a parameter the pointer to the call that is to receive the fax. Also, it takes a string, which is the filename to send. The Diva SDK 4.5 automatically opens this file.

The last parameter is the file format of the fax file. This parameter, DivaGetCallInfo, can be used to obtain the

information about the how the fax was negotiated (for example, Speed, Encoding, RemoteFaxID, number of pages sent, etc.)

The DivaSendFax generated a DivaEventFaxPageSent for each page sent and generated a DivaEventFaxSent when the fax completed. Upon receiving DivaEventFaxSent, the application proceeded on to disconnect the outgoing call.

CloseDevice

DivaUnregister was used to unregister the device and the callbacks.

Close Library

DivaTerminate closed the application's use of the Diva SDK 4.5.

Table 1 compares Diva SDK 4.5 and GC/R4 functions used during a fax call.

Diva SDK 4.5 Function	GC/R4 Function	Description
DivaInitialize	gc_Start	Initialize the API libraries
DivaGetNumLineDevices	sr_getboardcnt	Obtain the number of boards
DivaGetLineDeviceInfo	ATDV_SUBDEVS	Obtain information about device
DivaRegister	gc_Open dx_open dt_open sr_setparm sr_enbhdr gc_GetResourceH	For Global Call, open and initialize the device and register it with the event processing modes. For Diva, register the application with the board and set event processing mode.
DivaListen	gc_WaitCall	For Global Call, set the channel into a state where it is waiting for a call. For Diva it is only necessary to listen once per line device (that is, span) to activate listen on all channels.
DivaAnswer	gc_AnswerCall	Answer an incoming call
DivaSendVoiceFile	dx_playiottdata dx_listen gc_Listen dx_fileopen dx_fileclose	Play out voice file to a caller
DivaRecordVoiceFile	dx_reciottdata dx_playiottdata dx_listen gc_Listen dx_fileopen dx_fileclose	Record to a voice file
DivaSendFax	fx_sendfax fx_initstat dx_fileopen dx_fileclose fx_listen fx_unlisten gc_listen	
DivaReceiveFax	fx_rcvfas dx_fileopen dx_fileclose fx_listen fx_unlisten gc_Listen	
DivaSetCallProperties	dx_reciottdata DV_TPT	Used to set up termination conditions for IO functions
DivaDisconnect	gc_DropCall	Drop an active call
DivaConnect	gc_Makecall	
DivaCloseCall	gc_ReleaseCallEx	Clean up library references to the call and make the device ready for a new call
DivaUnregister	gc_Close dx_close sr_dishdr	For Global Call, close the application reference to a single device. For Diva board, unregister application from board
DivaServerStopXXX	dx_stopch	Used to stop IO functions that are in progress.
DivaTerminate	gc_Stop	Clean up and uninitialized the library

Table 1. Function Comparison

Table 2 compares Diva SDK 4.5 and GC/R4 events used during a fax call.

Diva SDK 4.5 Event	GC/R4 Event	Description
DivaEventIncomingCall	GCEV_OFFERED	Incoming call is presented
DivaEventCallConnected	GCEV_CONNECTED	Call has been answered by the far end of the correct call type
DivaEventCallConnected	GCEV_ANSWERED	Call has been answered and is now in the connected state
DivaEventCallDisconnected	GCEV_DROPCALL	Application has dropped the call
DivaEventCallDisconnected	GCEV_DISCONNECTED	Far end has terminated the call
	GCEV_RELEASECALL	The call reference is cleaned up and the device is now ready to take new calls
DivaEventSendVoiceEnded	TDX_PLAY	Play file ended
DivaEventRecordVoiceEnded	TDX_RECORD	Record file ended
DivaEventFaxReceived	TFX_FAXRECV	Fax has been received
DivaEventFaxSent	TFX_FAXSEND	Fax has been sent

Table 2. Event Comparison

Table 3 compares Diva SDK and GC/R4 data structures used during a fax call.

Diva SDK 4.5 Variable	GC/R4 Variable	Description
hApp	ldevh	Handle to the device (Global Call) or to the application (Diva)
hSdkCall	crn	The reference number for the call
hMyCall	SR_USERCONTEXT	An open storage area where application developers can place information for easy of retrieval
DivaVoiceDescriptor	DX_IOTT DX_XPB	Used to show the source and format of the data files
DivaSetCallProperties	DV_TPT	Used to setup termination conditions for IO functions
Param2 (for some APIs)	ATDX_TERMMSK	Contains the reason for termination

Table 3. Data Structure Comparison

Summary

A fax application can be used as a basis of comparison between the Dialogic® Global Call and R4 APIs and the Dialogic® Diva® Software Development Kit (SDK) 4.5 API. A sample can be referenced when porting an existing application from the Global Call and R4 APIs to the Diva API, or when integrating applications that can use both types of APIs.

Acronyms

ANI	Automatic Number Identification
CPA	Call Progress Analysis
CRN	Call Reference Number
DNIS	Dialed Number Identification Service
ECM	Error Correction Mode
Diva SDK 4.5	Diva Software Development Kit 4.5
GC/R4	Global Call/R4
SDK	Software Development Kit

For More Information

A Zip file, *Comparing Fax using Dialogic APIs Diva SDK*, containing the DivaGcR4FaxDemo and other components used in this application note can be downloaded at <http://www.dialogic.com/goto/?10566>

Using Dialogic® Boards with DM3/Springware and Diva Architecture in a Common Server —
<http://www.dialogic.com/goto/?10116>

A Zip file, *Using Dialogic® Boards with DM3/Springware and Diva Architecture in a Common Server*, containing the code for the application note can be downloaded at <http://www.dialogic.com/goto/?10569>

Dialogic® Diva® API Developer's Reference Guide —
<http://www.dialogic.com/pubs/20640809.pdf>

Other Dialogic® Diva® API guides —
<http://www.dialogic.com/support/helpweb/dssdk/manuals/dsapilin/default.asp>.

Dialogic® Diva® Software Development Kit - Version 4.5 and documentation are available at http://www.dialogic.com/products/tdm_boards/development_tools/Diva_Server_Software_Development_Kit.htm

To learn more, visit our site on the World Wide Web at <http://www.dialogic.com>.

Dialogic Corporation

9800 Cavendish Blvd., 5th floor
Montreal, Quebec
CANADA H4M 2V9

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH PRODUCTS OF DIALOGIC CORPORATION OR ITS SUBSIDIARIES ("DIALOGIC"). NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN A SIGNED AGREEMENT BETWEEN YOU AND DIALOGIC, DIALOGIC ASSUMES NO LIABILITY WHATSOEVER, AND DIALOGIC DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF DIALOGIC PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Dialogic products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Dialogic may make changes to specifications, product descriptions, and plans at any time, without notice.

Dialogic and Diva are registered trademarks of Dialogic Corporation or its subsidiaries. Dialogic's trademarks may be used publicly only with permission from Dialogic. Such permission may only be granted by Dialogic's legal department at 9800 Cavendish Blvd., 5th Floor, Montreal, Quebec, Canada H4M 2V9. Any authorized use of Dialogic's trademarks will be subject to full respect of the trademark guidelines published by Dialogic from time to time and any use of Dialogic's trademarks requires proper acknowledgement.

Windows is a registered trademark of Microsoft Corporation in the United States and/or other countries. Other names of actual companies and products mentioned herein are the trademarks of their respective owners. Dialogic encourages all users of its products to procure all necessary intellectual property licenses required to implement their concepts or applications, which licenses may vary from country to country.

In the United States, the use of the Dialogic® Diva® digital board to route facsimile transmissions using DID is likely to infringe certain claims of U.S. Patent Nos. 5,488,651 and 5,291,546 owned by Cantata Technology Inc.

Copyright © 2007 Dialogic Corporation All rights reserved.

09/07 10443-01