



## **A Comparison of the APIs in the Dialogic® Gammalink Development Kit and the Dialogic® Diva® Software Development Kit**

## Executive Summary

This application note provides a high-level comparison of the GRT API in the Dialogic® Gammalink Development Kit (GDK) and the Dialogic® Diva® Software Development Kit (SDK) API. This application note uses a fax application as a basis of comparison and can be used as a guide for porting an existing application from the GRT API to the Diva® API.

## Table of Contents

Introduction.....	4
Architecture .....	4
GDK API Overview.....	5
Diva® SDK Overview .....	5
GRT and Diva API Comparison.....	5
Programming Considerations .....	6
Sample Application Flow .....	6
Header Files and Libraries.....	10
Device Discovery .....	10
Library Level Initialization .....	11
Device Level Initialization and Registration.....	11
Setting Initial Call Parameters and Connecting.....	11
Sending a Fax.....	12
Receiving a Fax.....	13
Completing a Fax Call.....	14
Event Handling.....	15
Exiting Applications .....	15
A Comparison of Commonly Used Functions .....	16
Event Processing .....	17
GRT API.....	17
Diva API.....	17
Setting Fax Parameters.....	17
References.....	18
Acronyms .....	18

## Introduction

This application note provides a comparison of the GRT API in the Dialogic® Gammalink Development Kit (GDK) to the Diva® API in the Dialogic® Diva® Software Development Kit (SDK). A fax application is used as the basis for comparison, and the application note assumes that the fax application using the GDK is based on Interactive Programming mode and uses the GRT (Gammalink Real Time) API.

Because the comparison highlights the key differences between the GRT API and Diva API for implementing fax applications, it can be used as porting guide to modify an existing GRT-based fax application to be able to use the fax functionality of the Diva API.

This application note provides the following:

- A high-level comparison of the GRT API and Diva API architecture
- A review of the typical call flow of an application based on the GRT API
- A review of the Diva API function calls that provide functionality similar to that of the GRT API
- A parallel listing of GRT API functions and Diva API functions

## Architecture

Figure 1 illustrates the basic architecture of the Gammalink and Diva systems.

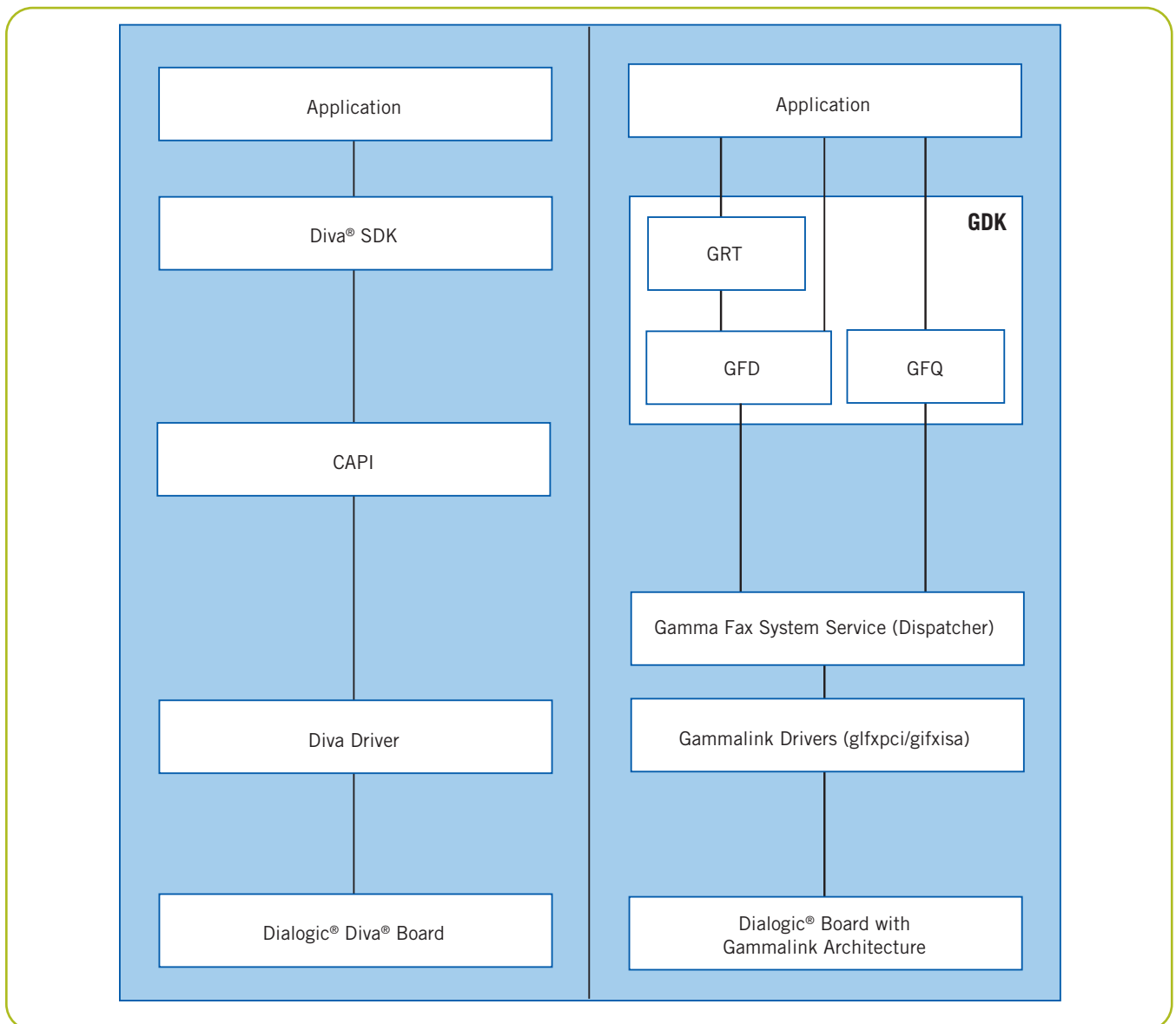


Figure 1. Gammalink and Diva Systems in Parallel

Procedure	Diva API	GRT API
Callback method	Yes	No
Polled method	Yes	Yes — interactive mode
Explicit call control	Yes	Generally no
Derive network interface from another board	Not necessary — all Diva boards have network interfaces	Yes
Must act on received events	Generally optional — only an event reporting a disconnect requires an action to free resources	Unnecessary
Uses Transparent Named Pipes	No	Yes – to connect application and Dispatcher

Table 1. Comparison of GRT and Diva APIs

## GDK API Overview

The GDK has two main programming models:

- **batch (GFQ)** — used when the application does not need to monitor the fax transaction phases. In general, this model is not recommended and is seldom used. It is also referred to as the “queue model” or “dispatcher mode.”
- **interactive (GRT)** — enables the application to monitor call states and take appropriate action when required.

**Note:** The information in this application note applies only to the interactive or GRT model. References are to the GRT API only.

GRT uses polled mode for retrieving call events, and the events do not require a response in order for the call to continue. Responses to events are usually used to change the already set course of the call. For example, a response might terminate a call immediately.

To initiate a call, queue records that contain call properties are built and submitted to the Dispatcher, which handles call control, event reporting for enabled events, and data transfer. In most cases, the Gammalink board has a network interface, and by default the application does not handle call control explicitly.

## Divas® SDK Overview

The Diva SDK provides a callback and a polled mode for call event retrieval. This application note discusses callback mode only.

The Diva SDK does not provide a model similar to the GDK’s GFQ. However, a batch model similar to GFQ can be implemented by writing an application to set up and monitor a queue file. A sample of such an application

(called “Multi-threaded Fax Sample”) can be found at <http://www.dialogic.com/support/helpweb/dssdk/samples/cppFaxMultiThreads.txt>. This sample application is also included in the Diva SDK v4.5 download package.

Another batch-model sample application is available at <http://www.dialogic.com/support/helpweb/dssdk/samples/vcsharpBatchFax.txt>.

As with GDK, events must be enabled before they can be sent to the application. However, unlike in GDK, some events must be acted on in order for the call to proceed when Diva SDK callback mode is used. A fax application based on the Diva SDK performs call control itself and then transfers fax data when event reporting indicates that a data channel has been negotiated and data can be sent or received.

## GRT and Diva API Comparison

Table 1 compares the GRT and Diva APIs and summarizes the material already discussed.

Two notes about Table 1 are important.

- Although Gammalink boards can have SCBus connectivity, this application note only discusses Gammalink boards running as fax servers. For this reason, “generally no” is the response given for “explicit call control” in Table 1.
- Table 1 notes that it is “unnecessary” for the GRT API to act on received events. A response to received events is only necessary to change call parameters or to terminate a call. Because the Dispatcher handles call control, the default response is usually “continue.” However, event processing can be used to monitor fax events.

## Programming Considerations

The following programming languages are supported by the APIs:

- **Diva SDK** — C, C++, Visual Basic, VBScript, C#
- **GDK** — C, C++

The following are also important points of comparison:

- **Architecture** — GRT API is channel-based; Diva API is call-based.
- **Call control** — GRT API handles call control; Diva API faxes as part of a call.
- **Mode** — GRT API is implemented only in polled mode; Diva API can be implemented in either callback or polled mode.
- **Configuration** — GRT API commands may be set directly in the registry or programmatically using a remote request function; parameters for the Diva API are set through API functions.

## Sample Application Flow

Figure 2 illustrates the typical high-level call flow of a simple incoming or outgoing fax call using the GRT API. Only one flow diagram is needed because once the channel is set to default receive mode, the application does not need to do anything in order to receive a fax except to optionally process events associated with incoming faxes. However, this additional processing is not required.

Sample code snippets for Figure 2 were taken from the GRTBasic demo application at [http://resource.dialogic.com/telecom/support/Gammalink/winnt\\_source/grtbasic.c](http://resource.dialogic.com/telecom/support/Gammalink/winnt_source/grtbasic.c)

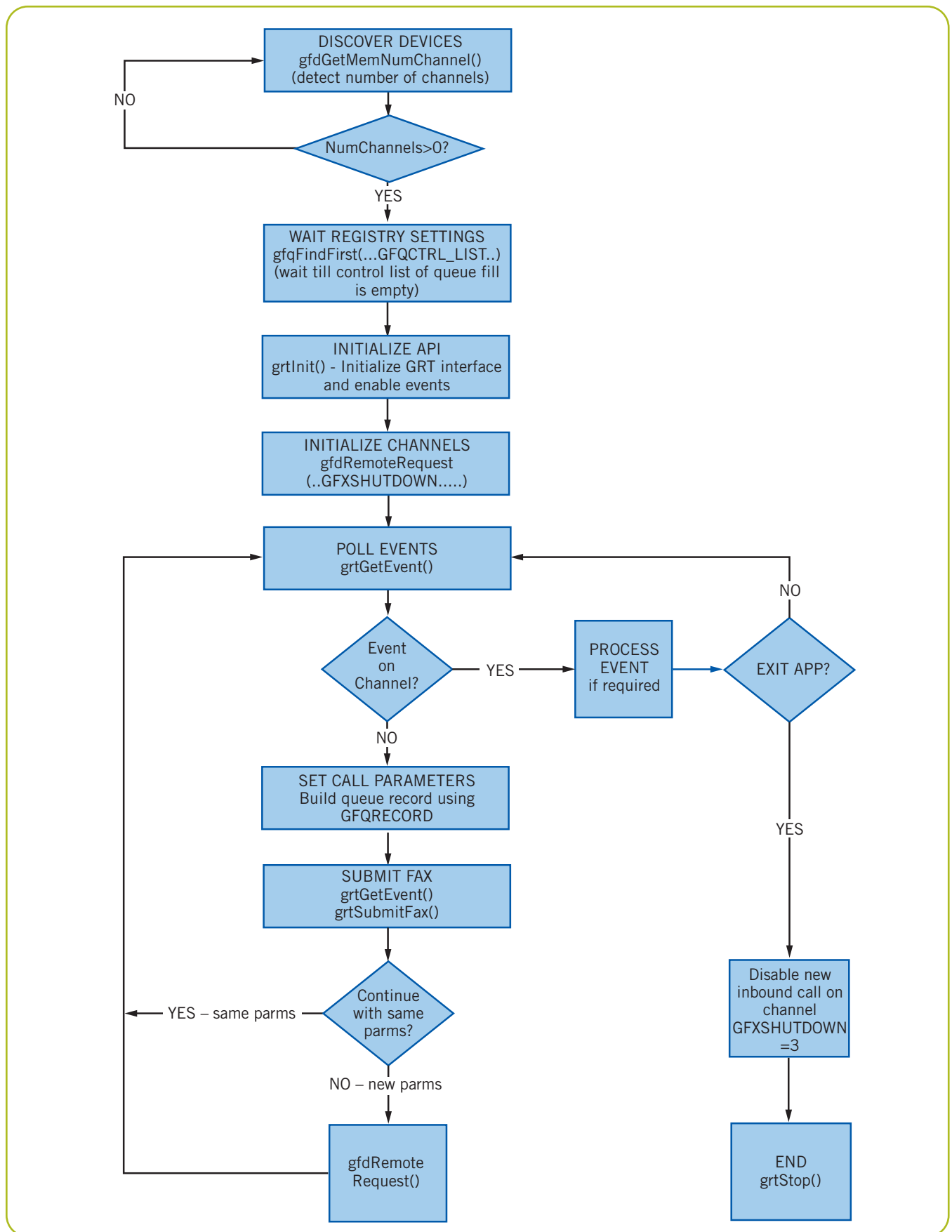


Figure 2. GRT API Call Flow

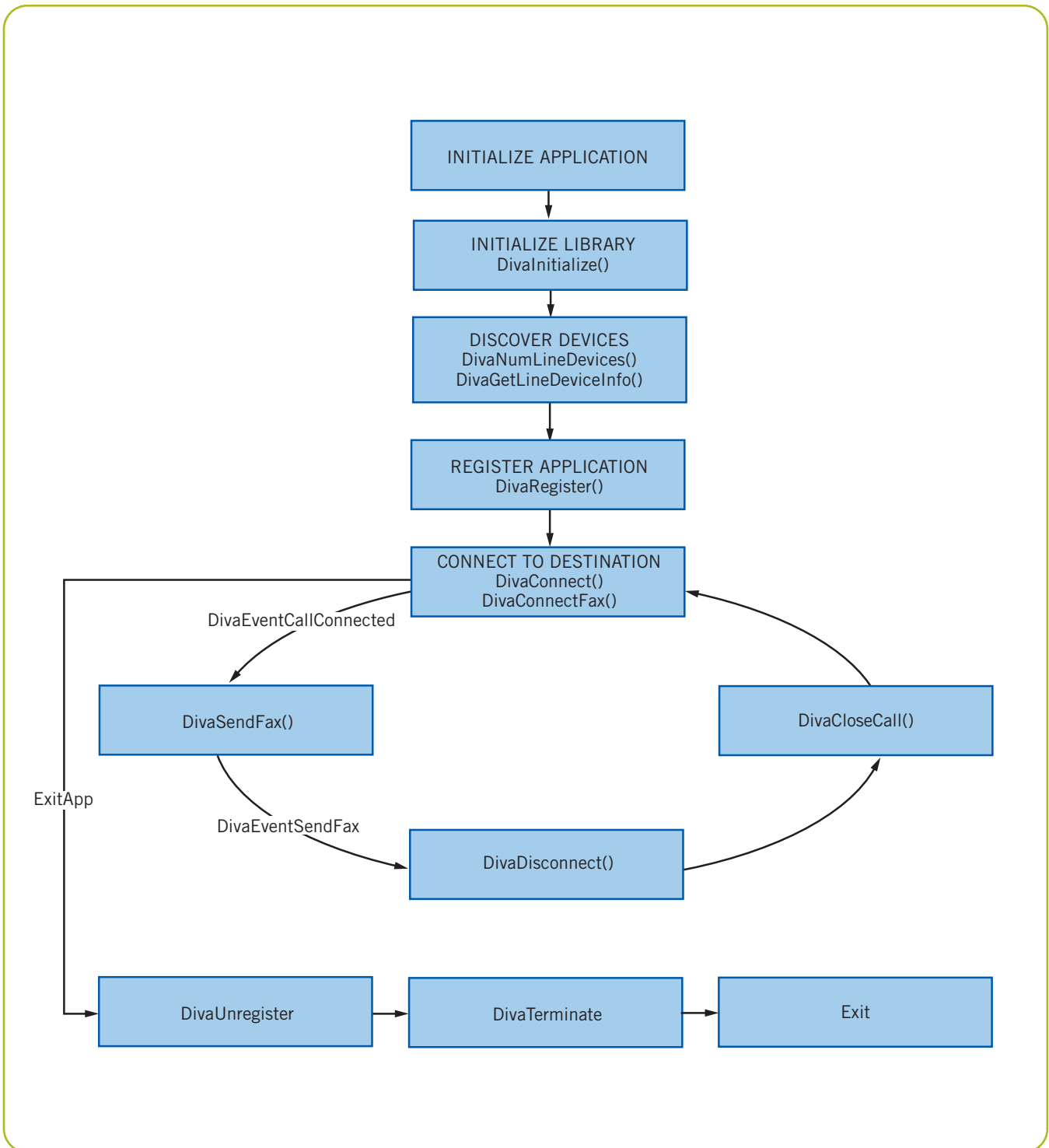


Figure 3. Diva API Outbound Call Flow

Figure 3 illustrates the typical high-level call flow of a simple outbound fax call using the Diva API.

Sample code snippets for Figure 3 were taken from the following demo applications:

- FaxOutSimple at <http://www.dialogic.com/support/helpweb/dssdk/samples/linux/faxoutsimple/src/faxoutsimple.txt>
- FaxinSimple at <http://www.dialogic.com/support/helpweb/dssdk/samples/linux/faxinsimple/src/faxinsimple.txt>

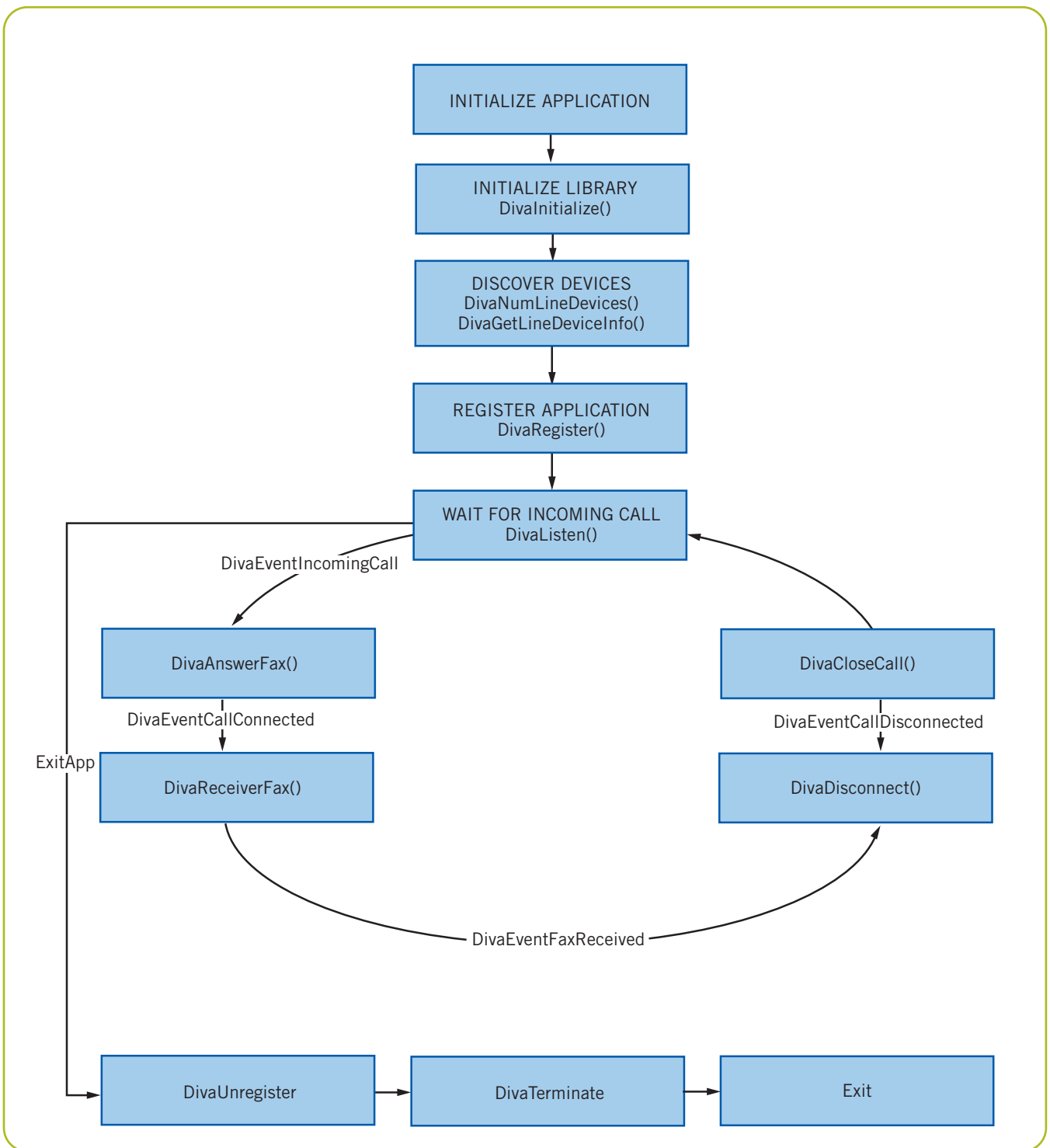


Figure 4. Diva API Incoming Call Flow

Figure 4 illustrates the typical high-level call flow of a single-page incoming fax call using the Diva API. Multi-page faxes require intermediate events, which have been excluded to keep the illustrated call flow as simple to understand as possible.

## Header Files and Libraries

### **GDK**

Header files include gamma.h, gfq.h, genra.h. Library is NTGDK.lib.

GDK provides one static library named NTGDK.lib in two versions that are compiler specific. The file path for the Visual C++ 6 version is <install dir>\fax\VC6lib. The file path for the version that is backward compatible with Visual C++ 5 is <install dir>\fax\VC5lib.

### **Diva SDK**

Header file is dsSDK.h. Library is dsSDK.lib.

The Diva SDK provides a single library file dsSDK.lib in two versions, one static and the other dynamic. When using the dynamic version, the dssdk.dll will be loaded at runtime and is part of the driver software. The static library file path is <install dir>\basic\lib\static and the dynamic library file path is <install dir>\basic\lib\dynamic.

## Device Discovery

### **GDK**

gfdGetMemNumChannel() returns the number of available channels. This function may be called before the call to grtInit(), since it uses the GFD API.

### **Diva SDK**

When using the Diva SDK, device discovery cannot be done until the Diva API has been initialized.

DivaGetNumLineDevices() returns the number of line devices that the Diva API may access. This function cannot be called until after the DivaInitialize() function has been successfully executed, as discussed below. After the number of line devices is detected, the DivaGetLineDeviceInfo() function can be used to determine if a line is fax capable. Here is an example:

```

unsigned long linecount;
DivaLineDeviceInfo info;
int subdevs;

if( DivaGetNumLineDevices( &linecount) != DivaSuccess){
    printf("DIVA: DivaGetNumLineDevices() Failed\n");
    return -1;
}
for(c=1; c <= (int)linecount; c++){
    //Now we have to find out how many sub-devices on each board.
    // Note that the size field of the info struct must be
    // set to the size of the struct
    info.Size = sizeof(DivaLineDeviceInfo);
    if(DivaGetLineDeviceInfo(c, &info) != DivaSuccess){
        printf("    Failed DivaGetLineDeviceInfo for %d\n",c);
    }
    else{
        subdevs=info.Channels;
        printf("    %d has %d devices\n",c,subdevs);
        //Inside the info structure there is a flag to say if
        // fax is supported. If fax is supported, it is supported
        // on all channels.
        if(info.bFaxSupported){
            faxdevcnt+=subdevs;
        }
    }
}
}

```

## Library Level Initialization

### ***GDK***

Library-level initialization is not available in GDK, but is necessary when using the Diva SDK. Before `grtInit()` is called, the application sends configuration commands from the registry to the channels via the `GFQCTRL_LIST`. It is advisable, but not absolutely necessary, to use the `gfqFindFirst()` function to ensure that the list is empty, which implies that all the settings from the registry have been sent to the channels. Otherwise, values remaining in the list may overwrite values set using `gfdRemoteRequest()`.

### ***Diva SDK***

`DivaInitialize()` takes no parameters but initializes the Diva API, and must be called before any other function. The following is an example of a call to initialize the API:

```
        if (DivaInitialize() != DivaSuccess){
            return -1;
        }
```

## Device Level Initialization and Registration

### ***GRT API***

`grtInit()` initializes the run-time interface for specified channels, and enables GRT API events and GRT API event responses. It must be called before any other GRT API function. `grtInit()` specifies the channels to be initialized and enables the events that should be returned on those channels. Further channel initialization or updating are performed separately following the `grtInit()`, using `gfdRemoteRequest()` with appropriate `GFXSHUTDOWN` values.

### ***Diva API***

`DivaRegister()` registers the application with the Diva API and sets up global parameters such as the maximum number of connections, buffer sizes, and event reporting modes. Also, the callback handler (event handler function name) is specified as a parameter of this function. The major difference between the GRT API and the Diva API is that in the GRT API, specific events are listed as parameters in `grtInit()`, while only the event handler (callback handler) name is needed for event monitoring in the Diva API. Here is an example of a simple Diva API registration:

```
        if ( DivaRegister ( &hApp, DivaEventModeCallback, (void *)
CallbackHandler, 0, 0, 7, 2048 ) != DivaSuccess )
        {
            return -1;
        }
```

In this example, the `&hApp` is the application handle, and because it is used by many of the Diva API calls, it must be saved in an accessible location. The second parameter indicates the event mode (in this case, callback) and the third parameter indicates the function to be called when the event is received. The other parameters are used to configure internals, and the maximum values used in the example are recommended.

## Setting Initial Call Parameters and Connecting

### ***GRT API***

Before submitting an outbound fax, the GRT API builds a queue record, using the `GRQRECORD` data structure. `GRQRECORD` is submitted to the Dispatcher using the `gfdRemoteRequest()` function, and the Dispatcher handles call control from that point.

**Diva API**

If `DivaConnectFax()` is used to connect to the remote fax machine, some fax call parameters will already be set in the connection, and the remaining parameters will be set in `DivaSendFax()`, `DivaConnect()`, or `DivaConnectFax()`. The application must maintain the call control state machine and issue the appropriate fax call. The next section describes how these function calls work in more detail.

**Sending a Fax****GRT API**

`grtSubmitFax()` initiates an outgoing fax. This API call will take care of call control, fax device detection, and fax transmission by submitting a queue recording that contains all the information the Dispatcher needs. One component of the Dispatcher extracts the fax number from the queue record and sends the information to the fax channel, which dials the number and makes the connection. When the connection is made, another component of the Dispatcher provides other fax call properties, which are used in transmitting the fax document. Notification events can be configured to monitor fax progress.

When sending a fax with the GRT API, the following events can be enabled: `GRT_DIAL`, `GRT_RECV_DIS`, `GRT_PAGE_BREAK`, and `GRT_CALL_TERM`.

**Diva API**

Unlike with the GRT API, the application must maintain the call control state machine and issue appropriate fax function calls. Diva API uses `DivaConnect()`, `DivaConnectFax()`, and `DivaSendFax()` in establishing a fax call.

`DivaConnect()` allows a generic call to be placed, specifying the type as fax. Here is an example:

```
if ( DivaConnect ( hApp, hMyCall, &hSdkCall, OutboundNumber,
                 DivaCallTypeFax, LINEDEV_ALL ) != DivaSuccess )
return -1;
```

The above call segment will cause the `DivaEventCallConnected` to be generated.

`DivaConnectFax()` is an alternative function that enables the application to specify some fax parameters such as local fax ID, local sub address, fax header, maximum speed, and fax options. The fax options specify such properties as encoding (MR, MMR), resolution, polling, interrupt, color, and ECM. These parameters are similar to GRT API parameters that would normally be set using GRT API configuration commands such as `GFXFAXCONTROL` and `GFXMODEMCONTROL`. Note that the Diva API will attempt to match the specified parameters for the fax, but, in accordance with T.30 negotiation, will perform a conversion if necessary, such as from fine to standard resolution.

`DivaSendFax()` should be called only upon the application's receipt of the `DivaEventCallConnected` event. Its parameters are the call handle, file name containing data to be transmitted, and the fax format (`DivaFaxFormat`). The `DivaFaxFormat` data structure specifies the tiff file type, indicates whether or not the document should be auto-detected, and specifies whether or not the document should be transmitted in color. The parameters in `DivaConnectFax()` and `DivaSendFax()` are similar to the parameters built in the queue record and configuration commands when using the GRT API.

Here is an example of how a fax can be sent using the Diva API:

```
void CallbackHandler ( DivaAppHandle hApp, DivaEvent Event, PVOID Param1, PVOID
Param2 )
{
    DivaCallInfo    Info;

    memset ( &Info, 0x00, sizeof ( Info ) );
    Info.Size = sizeof ( Info );
```

```

switch (Event)
{
case DivaEventCallConnected:
    if ( DivaSendFax ( hSdkCall, FaxFile, DivaFaxFormatTIFF_ClassF ) ==
        DivaSuccess )
    {
        DivaGetCallInfo ( hSdkCall, &Info );
        printf ( "Connected, Speed: %ld\n", Info.RxSpeed );
    }
    else
        DivaDisconnect ( hSdkCall );
    break;

case DivaEventFaxSent:
    DivaGetCallInfo ( hSdkCall, &Info );
    printf ( "Fax sent. Pages: %ld Speed: %ld, ECM: %s \n", Info.dwFaxPages,
Info.RxSpeed, Info.bECMAActive ? "YES" : "NO" );
    DivaDisconnect ( hSdkCall );
    break;
}

```

## Receiving a Fax

### ***GRT API***

When a Gammalink card is used, the application receives no notification by default. However, notification events can be configured to monitor fax progress by setting LISTEN or receive mode in the correct GFXSHUTDOWN bit.

Under the GRT API, the following events can be enabled to receive a fax: GRT\_CALL\_PENDING, GRT\_RECV\_DCS, GRT\_INFO\_EXCHANGE, GRT\_PAGE\_BREAK, and GRT\_CALL\_TERM.

### ***Diva API***

As in sending a fax, multiple Diva API calls need to be made to establish the call, detect fax machine presence, and begin fax reception.

First, DivaListen() is called, and the application enters a state in which it waits for events. When an incoming call is presented to the application, the event DivaEventIncomingCall is received. Here is a DivaListen() example:

```

if ( DivaListen ( hApp, DivaListenAll, LINEDEV_ALL, "" ) != DivaSuccess )
    return -1;

```

DivaAnswer() is used to answer the incoming call and takes a call type that must be specified as DivaCallTypeFax. When a fax call is detected, a DivaEventCallConnected event is generated. If a fax call is not detected, a disconnect is generated.

**Note:** A call can be answered as a call type other than fax and then changed to a fax call later. This is done by altering the call properties after the call is connected, or by using DivaSetCallType. This technique can be useful for Call Progress Analysis (CPA). A call can be connected in “speech” mode, and the application can be set to listen for tones or speech. If tones are detected, the call becomes a fax call.

Here is an example of how to use DivaAnswer():

```

void CallbackHandler ( DivaAppHandle hApp, DivaEvent Event, PVOID Param1, PVOID
Param2 )
{
    DivaCallInfo    Info;

    memset ( &Info, 0x00, sizeof ( Info ) );
    Info.Size = sizeof ( Info );
}

```

```

switch (Event)
{
    case DivaEventIncomingCall:
        if ( hSdkCall == NULL )
        {
            hSdkCall = Param1;
            DivaAnswer ( hSdkCall, hMyCall, DivaCallTypeFax );
        }
        break;
}

```

DivaReceiveFax() takes the call handle as its first parameter because the fax devices are tied to the call and are not a separate entity. Also, this function takes just the filename for the incoming fax. The GRT API automatically uses file naming conventions.

Fax receipt will spawn several events. At the end of each page, the DivaEventFaxPageReceived event is generated. This event indicates that the fax page has been received. The application can optionally retrieve the content for each page using DivaGetCallInfo() and the page's quality from the DivaFaxPageQuality data structure. When the last page is received, the event DivaEventFaxReceived is generated. This event indicates that the fax has been successfully received, regardless of the disconnect reason reported later. Here is an example:

```

void CallbackHandler ( DivaAppHandle hApp, DivaEvent Event, PVOID Param1, PVOID
Param2 )
{
    DivaCallInfo    Info;

    memset ( &Info, 0x00, sizeof ( Info ) );
    Info.Size = sizeof ( Info );

    switch (Event)
    {
    case DivaEventCallConnected:
        if ( DivaReceiveFax ( hSdkCall, "RxFax.tif", DivaFaxFormatTIFF_ClassF ) ==
DivaSuccess )
        {
            DivaGetCallInfo ( hSdkCall, &Info );
            printf ( "Connected, Speed: %ld\n", Info.RxSpeed );
        }
        else
            DivaDisconnect ( hSdkCall );
        break;

    case DivaEventFaxPageReceived:
        DivaGetCallInfo ( hSdkCall, &Info );
        printf ( "Page %ld received. Speed: %ld, ECM: %s \n", Info.dwFaxPages,
Info.RxSpeed, Info.bECMAActive ? "YES" : "NO" );
        break;
}
}

```

## Completing a Fax Call

### ***GRT API***

Functions such as grtRespondEndCall() may be called, but such function calls are generally not required, unless a response to some event that causes call termination is needed.

### ***Diva API***

DivaDisconnect() and DivaCloseCall() are required to disconnect a call, usually after receiving a fax termination event (DivaEventFaxSent or DivaEventFaxReceive) or notification of a call disconnect (DivaEventCallDisconnected).

DivaDisconnect() is used to terminate call control. Fax transmission will automatically be stopped when this function is called.

DivaCloseCall() is used to release the call and free the resources needed to make or receive another call.

Here is an example of how to use these two functions:

```
void CallbackHandler ( DivaAppHandle hApp, DivaEvent Event, PVOID Param1, PVOID
Param2 )
{
memset ( &Info, 0x00, sizeof ( Info ) );
    Info.Size = sizeof ( Info );
switch (Event)
    {
.
.
.
    case DivaEventFaxReceived:
        DivaGetCallInfo ( hSdkCall, &Info );
        printf ( "Fax received. Pages: %ld Speed: %ld, ECM: %s \n", Info.dwFaxPages,
Info.RxSpeed, Info.bECMAActive ? "YES" : "NO" );
        DivaDisconnect ( hSdkCall );
        break;

    case DivaEventFaxSent:
        DivaGetCallInfo ( hSdkCall, &Info );
        printf ( "Fax sent. Pages: %ld Speed: %ld, ECM: %s \n", Info.dwFaxPages,
Info.RxSpeed, Info.bECMAActive ? "YES" : "NO" );
        DivaDisconnect ( hSdkCall );
        break;

    case DivaEventCallDisconnected:
        hSdkCall = 0;
        DivaCloseCall ( Param2 );
        printf ( "Disonnected\n" );
        exit_program = 1;
        break;
    }
```

## Event Handling

### **GRT API**

grtGetEvent() polls for events on specified channels. The application may respond with a specific action, but most responses under normal circumstances would be grtRespondContinue(). The event type can then be extracted from the GRT\_EVENT structure.

### **Diva API**

The callback handler function specified in the call to DivaRegister() will catch all events. Polling is not required unless poll mode has been specified. Some events, such as DivaEventCallConnected and DivaEventIncomingCall, must be acted on in order for a call to continue.

## Exiting Applications

### **GDK API**

grtStop() stops the GRT API from managing the fax channels, that is, it stops control of a given range of channels by the current thread of execution.

**Diva API**

DivaUnregister() and DivaTerminate() undo registration and initialization, and have the same effect as grtStop().

Here is an example of how these functions should be called:

```
DivaUnregister ( hApp );
```

```
DivaTerminate ( );
```

**A Comparison of Commonly Used Functions**

Table 2 compares the Diva and GRT API functions commonly used during a fax call.

Diva API Function	GRT API	Description
DivaInitialize()	No parallel function	Initialize the API libraries
DivaRegister()	grtInit()	Register the application with the board and set event processing mode
DivaConnect()	Not applicable — performed by the Dispatcher	Make call
DivaSendFax()	grtSubmitFax()	Begin T.30 protocol to send fax. The GRT function also dials, makes a connection, and sends a transmit fax command.
DivaGetNumLineDevices()	gfdGetFileNumChannel() gfdGetMemNumChannel()	Obtain the number of line devices
DivaGetLineDeviceInfo()	glHWDetect()	Obtain information about device
DivaListen()	Not applicable — handled by the Dispatcher	With GDK Dispatcher, application goes into listen mode, provided the appropriate GFXSHUTDOWN value is set. Diva API only needs to listen once per line device (span) to activate listen on all channels.
DivaAnswer()	Not Applicable — call control handled by Dispatcher	Answers an incoming call
DivaReceiveFax()	Not Applicable — handled by Dispatcher	Begins T.30 protocol to receive fax
DivaDisconnect()	Not applicable — handled by Dispatcher	Drops an active call
DivaUnregister() DivaTerminate()	grtStop()	GRT API: Closes the application reference to a single device. Diva API: Un-register application from board.
DivaGetEvent()	grtGetEvent()	Polls for event. Required for Diva API; optional for GRT API.

Table 2. Function Comparison

## Event Processing

### GRT API

To retrieve events, a process must poll for events using `grtGetEvent()`. If this function returns a `GRT_SUCCESS` to indicate that an event is available, the API checks the `GRT_EVENT` data structure and the program calls the appropriate `grtProcessXEvent()`, where “X” is the event type.

### Diva API

The callback model, which creates a single callback handler for all channels, is recommended when using the Diva API. All events can be caught in the callback handler function specified in a call to `DivaRegister()`.

**Note:** When using the Diva API, the callback handler is called for any event on any device. With the GRT API, multiple `grtGetEvent()` calls that handle different subsets of devices on the system are possible. A programmer porting to the Diva API for fax calls may need to take this into consideration when handling events.

Table 3 compares Diva and GRT API events.

Diva API Event	GRT API Event	Description
<code>DivaEventIncomingCall</code>	<code>GRT_CALL_PENDING</code>	Incoming call is presented
<code>DivaEventCallConnected</code>	<code>GRT_DIAL</code>	Call has been answered and is now in a connected state
<code>DivaEventCallDisconnected</code>	<code>GRT_CALL_TERM</code>	Application has dropped the call or the far end has terminated the call
<code>DivaEventFaxPageSent</code>	<code>GRT_PAGE_BREAK</code>	Single page has been transmitted
<code>DivaEventFaxSent</code>	<code>GRT_CALL_TERM</code>	Outbound faxing has been completed
<code>DivaEventFaxPageReceived</code>	<code>GRT_PAGE_BREAK</code>	Single page has been received
<code>DivaEventFaxReceived</code>	<code>GRT_CALL_TERM</code>	Fax has been received completely
No equivalent event, but is implied in <code>DivaEventCallConnected</code>	<code>GRT_RECV_DIS</code>	<code>DivaGetCallInfo()</code> can obtain equivalent information. Complete DIS information can be retrieved via the <code>DIVACPT_FaxRemoteFeatures</code> of the <code>DivaCallPropertyType</code> data structure.
No equivalent event, but information is available when <code>DivaEventCallConnected</code> is signaled for an incoming call	<code>GRT_RECV_DCS</code>	<code>DivaGetCallInfo()</code> can obtain equivalent information
No equivalent event	<code>GRT_INFO_EXCHANGRT</code>	<code>DivaGetCallInfo()</code> can obtain equivalent information

Table 3. Event Comparison

## Setting Fax Parameters

Both the Diva and GRT APIs have numerous fax connection and transmission settings, and it is outside the scope of this application note to present an exhaustive list of parameters. However, some of the commonly set parameters are compared in Table 4. In GDK, some parameters are configurable before runtime in the registry (parameter is shown in normal type), or during runtime using the `gfdRemoteRequest()` function (parameter is shown in italics).

A complete list of configurable GDK parameters is available in *GDK Version 5.0 Programming Reference Manual for Windows* at [http://resource.dialogic.com/telecom/support/releases/winnt/sr60pci/onldoc/pdffiles/gdk\\_api\\_win\\_v2.pdf](http://resource.dialogic.com/telecom/support/releases/winnt/sr60pci/onldoc/pdffiles/gdk_api_win_v2.pdf).

All Diva SDK parameters are listed in the “Diva SDK API Data Structures and Defines” section of the Diva SDK API Developer’s Reference Guide at <http://www.eicon.com/pubs/20640808.pdf>.

Div a SDK	GDK	Description
LocalNumber in DivaLineDeviceParamsFax	CSID	Local fax number or ID
FaxHeadline in DivaLineDeviceParamsFax	GFXHEADER	Fax document header
Div aFaxOptionDisableECM in DivaFaxOptions	GFXECM GFXECMMODE	Div a SDK: ECM is enabled by default; GDK: ECM is disabled by default.
Div aFaxResolution	GFXFINE GFXFINE	Div a API: Four different resolutions from Standard through Ultra-fine may be specified; GDK: Only fine resolution documents will be accepted
Div aFaxFormat	GFXFORM GFXFORMAT	GDK: Sets the tiff type with which to write received fax to disk (no color); Div a SDK: Structure specifies the tiff type for both incoming and outgoing faxes, including the color jpeg format
No equivalent parameter. The Div a configurator can be used to control which channels are for incoming calls and which are not. Channel can be set for in, out, or both.	GFXSHUTDOWN GFXSHUTDOWN	Sets or changes the fax channel state and enables incoming, outgoing, both, or none on a channel
No equivalent configuration parameter. Controlled via dial string.	GFXDIGIT	Sets the number of DTMF tones that a sender can enter
Handled internally within firmware and library; will automatically attempt to retrain	GFXRTNRETRAIN	Specifies action to be taken when an illegible page has been sent
	GFXRTPRETRAIN	Specifies action to be taken when a higher speed or change in resolution is required
Set via Div a Configuration GUI, which sets up the boards when installing Div a hardware	MODEMCTRL 1024 GFXMDMCONTROL	Changes the default dialing type when placing a call
	MODEMCTRL 2054 GFXMDMCONTROL	Sets the number of rings before answering
Uses “;” for pause in dial strings	MODEMCTRL 2066 Uses “,” for pause in dial strings	Sets the time value for a pause

Table 4. Comparison of Commonly Set Parameters

## References

For information on installing and configuring the GDK and Div a SDK, refer to the following manuals:

GDK Installation and Configuration Guide

[http://resource.dialogic.com/telecom/support/releases/winnt/sr60pci/onldoc/pdffiles/gdk\\_install\\_and\\_config\\_win\\_v1.pdf](http://resource.dialogic.com/telecom/support/releases/winnt/sr60pci/onldoc/pdffiles/gdk_install_and_config_win_v1.pdf)

GDK Programming Reference

[http://resource.dialogic.com/telecom/support/releases/winnt/sr60pci/onldoc/pdffiles/gdk\\_api\\_win\\_v2.pdf](http://resource.dialogic.com/telecom/support/releases/winnt/sr60pci/onldoc/pdffiles/gdk_api_win_v2.pdf)

Using Dialogic® Boards with DM3/Springware and Div a Server Architecture in a Common Server

[http://www.dialogic.com/network/csp/appnots/10116\\_Comparing\\_Dialogic\\_APIs.pdf](http://www.dialogic.com/network/csp/appnots/10116_Comparing_Dialogic_APIs.pdf)

Div a SDK v4.5

<http://www.dialogic.com/support/helpweb/dssdk/manuals/>

## Acronyms

API	Application Programming Interface
CPA	Call Progress Analysis
ECM	Error Correction Mode
GDK	Gammalink Development Kit
GRT	Gammalink Real Time
SDK	Software Development Kit

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 DIALOGIC® PRODUCTS. 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"). 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.

The 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 a 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.

04/07 10362-001