Porting and Integration Guide

Rovi TotalGuide 1.2.3

Porting and Integration Guide : Rovi TotalGuide 1.2.3

This document is intended to address your needs. Please provide feedback if you see any way that it might be improved, or if you have any problems with the documentation. Generated from source revision 28519 .

Publication date 24-Jan-2012

About this book

This book describes how to implement Rovi TotalGuide on a new product, typically a set-top box, television, or similar appliance.

Following an overview of the TotalGuide product, the book describes how to port TotalGuide to a new hardware platform and how to integrate it with OEM host applications.

This book provides an overview and general guidance. For the definitive reference to the porting layer and host APIs, refer to the API reference documentation that is included in the product distribution.

1

Introduction

This manual describes how to port Rovi's TotalGuide software to a new platform and integrate it into a host application. This chapter provides an overview of TotalGuide and this process.

Prerequisites

This section describes the requirements that need to be met before porting Rovi TotalGuide to a new platform, or integrating it with a host application.

Platform requirements

Rovi TotalGuide can be ported to any platform that meets a specified set of prerequisites. In summary, the requirements are:

- A multithreaded operating system such as Linux, with a POSIX-compliant filesystem
- Adequate memory, CPU performance, and storage
- Suitable graphics support

Development host requirements

Your development host (typically a Linux system) and toolchain must be compatible with, or adaptable to, the Rovi development libraries. Typically any system with an Ansi C compiler will be suitable.

Prerequisite knowledge

A thorough knowledge of the C programming language is the main requirement for porting and integrating TotalGuide, together with comprehensive knowledge of the target system. An experienced developer should

be able to use this guide, along with the Host API reference and Rovi Porting Layer API reference, to implement the required interface layers.

Overview

Introducing TotalGuide

TotalGuide is a media guidance solution that enables easy discovery of television, movies, music and photos, regardless of source. It supports delivery of both TV and broadband content. It can be configured in several different "flavors" to incorporate some or all of these content streams: TV+broadband, for example, or TV only. Regardless of the flavor configured, TotalGuide uses Rovi's rich set of media metadata to provide enhanced listings and navigation across all discovered content.

As a media guide, TotalGuide provides sophisticated ways to browse metadata-enhanced media listings, recommendations, bookmarking and personal preference configurations, and other advanced features.

TotalGuide architecture: context (on host)

Rovi TotalGuide is supplied for integration as a platform-independent set of libraries that run on top of the Rovi Porting Layer. Host applications interact with these libraries via the Host API. These relationships are shown in Figure 1.1.

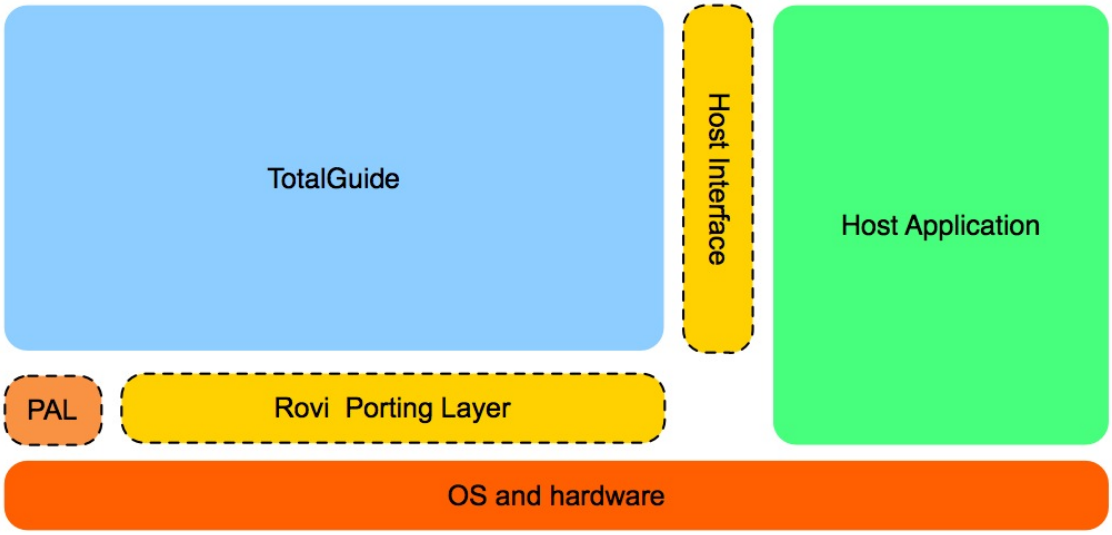

Figure 1.1. TotalGuide context diagram

The areas enclosed in dashed lines represent the interfaces that require re-implementation to adapt TotalGuide to a new hardware platform and host application. Re-implementing these is the focus of this guide.

TotalGuide architecture: ecosystem

TotalGuide draws much of its power from its interaction with the Rovi Device Gateway, which supplies rich and targeted metadata to the TotalGuide device. Figure 1.2 shows this interaction, along with the delivery of content from all potential sources.

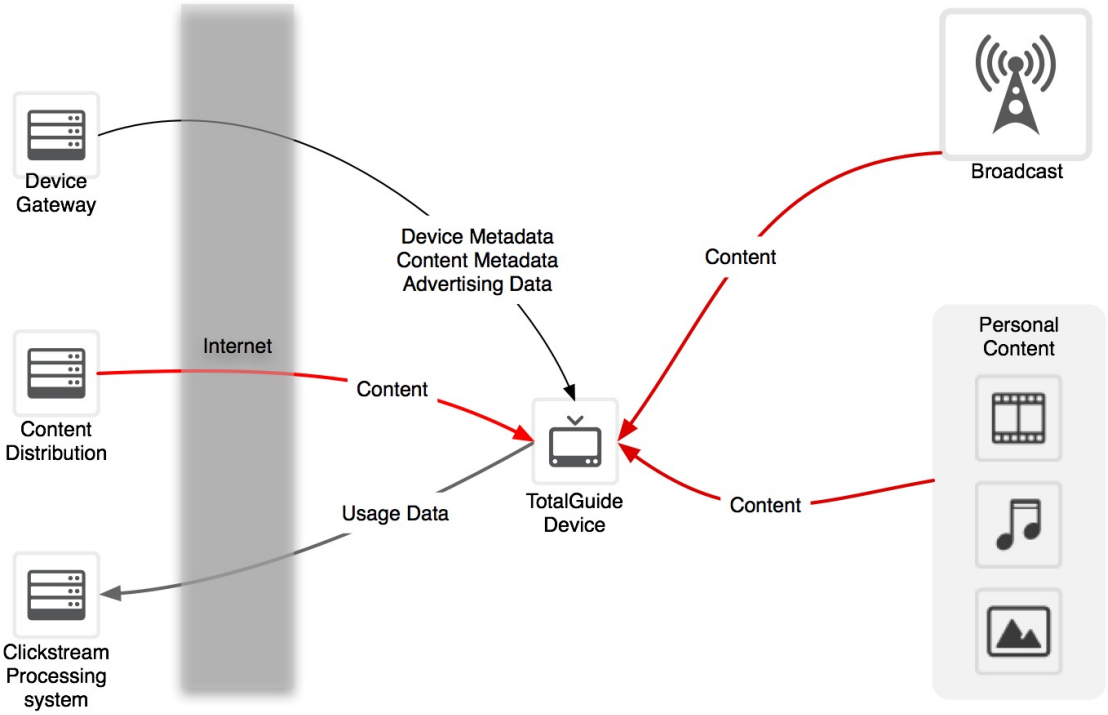

Figure 1.2. TotalGuide ecosystem diagram

The flow of usage data back to the processing system, also shown in Figure 1.2, provides the basis on which the medata is tailored to match the user's profile.

TotalGuide is compliant with privacy legislation in its target markets.

TotalGuide architecture: internal overview

The internal architecture of TotalGuide is shown in Figure 1.3. For most deployments, you do not need to be concerned with the internals of the product architecture; one notable element is that the media player component -- which requires the player abstraction layer to be ported -- can itself be reimplemented if needed.

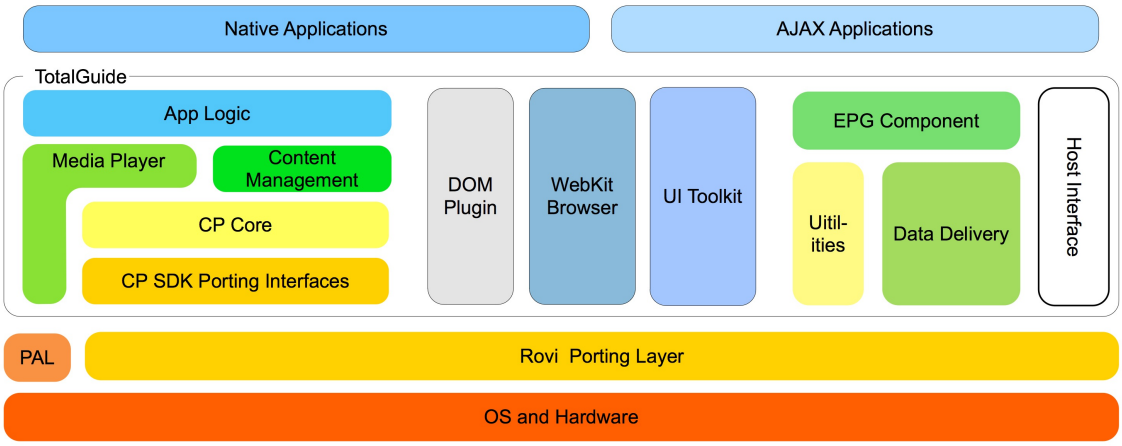

Figure 1.3. TotalGuide components diagram

Process overview

This section provides an overview of the main processes that are described in this manual.

Porting TotalGuide

Rovi TotalGuide runs on top of a system abstraction known as the Rovi porting layer. The default implementation of the porting layer matches the Rovi reference platform.

Porting TotalGuide to a new platform requires the re-implementation of the porting layer API on the new platform. Provided that the platform allows complete implementation of the porting layer API, TotalGuide can run unchanged on the updated porting layer.

The porting layer can be divided into two broad parts, as shown in Figure 1.3. The Player Abstraction Layer will need to be updated to use the specific video interfaces available on the target hardware. The remainder of the porting layer deals with access to system resources such as memory, files, networking, and so on. For POSIX-compliant Linux systems, this aspect of the port should be trivial.

Integrating TotalGuide with a host

TotalGuide is generally implemented as part of a host solution such as a set-top box, television, or similar appliance. This means that it must be integrated into that host system. The Host API provides a way for the host to invoke features in TotalGuide and for TotalGuide to call out to the host.

Integrating TotalGuide into a host system requires the re-implementation of the Host API. A sample application is included with TotalGuide that shows the default implementation in use; integration with an actual host involves changing the API implementation using features specific to the target host system.

Building standalone RPL and HostAPI

For details of how to build RPL and the Host API without building TotalGuide, refer to the files README_Building.txt and README_RPL.txt at the root directory of the TotalGuide source code.

Customizing and extending TotalGuide

Rovi TotalGuide can be customized and extended in a wide variety of ways, including:

- Skinning
- Localization and string replacement

Refer to the *TotalGuide Skinning and Customization Guide* for further information.

API reference documentation

Complete reference documentation for the host API and the Rovi porting layer API is included in the product distribution.

2

Integrating with the host API

This chapter describes how to integrate TotalGuide with a host application using the Host API. This is different activity from porting TotalGuide to a new platform (described in Chapter 3) although it may be part of the same project.

Overview

TotalGuide is generally implemented as part of a host solution such as a set-top box, television, or similar appliance. This means that it must be integrated into that host system. The Host API provides a way for the host to invoke features in TotalGuide and for TotalGuide to call out to the host, as depicted in Figure 2.1.

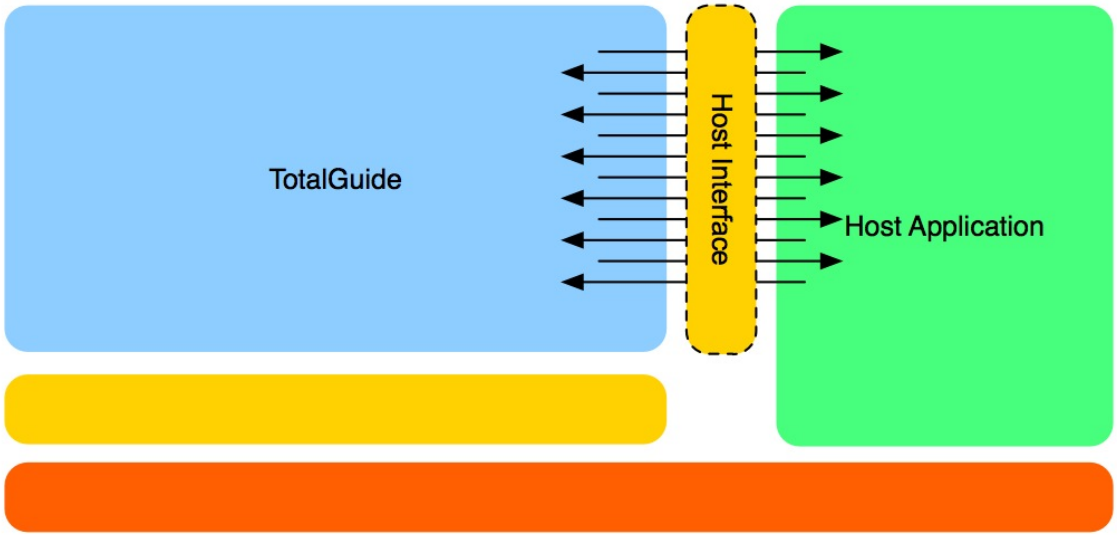

Figure 2.1. Implementation of the host API

To integrate TotalGuide into a host system you need to redevelop the Host API, providing your own implementations for functions that TotalGuide requires in the host. A sample application is included with TotalGuide to provide default implementations of these functions. Many functions must be implemented or may be invoked regardless of the flavor of TotalGuide being built; other groups of functions are optional, meaning they require implementation (or may be invoked) only for specific flavors of TotalGuide.

Additionally, you must implement logic in your host application to invoke functions in TotalGuide that set up and invoke its features correctly. the section called “Use cases” summarizes the main use cases, which are also detailed in the host API reference.

Finally, there is a set of configuration options that let you control aspects of how the host application and TotalGuide interact, described in *Building and Running TotalGuide* and the *Building and Customization Guide*.

API files

The files at .../src/PortingLayer/Include/HostAPI include the following Host API header files:

- gp_appmain.h
- gp_auxtypes.h
- gp_chanlist.h
- gp_channel.h
- gp_chantypes.h
- gp_datetime.h
- gp_device.h
- gp_devicetypes.h
- gp_joblist.h
- gp_jobs.h
- gp_jobtypes.h
- gp_setup.h
- gp_thirdpartyapp.h

- gp_userinterface.h

Some of these files define functions that are exported by TotalGuide (GP_ functions) for the host application's use, as well as "callback" functions (GPCB_ functions) that TotalGuide may invoke in the host application, which may require implementation in the host application.

Not all the function prototypes contained in these files require implementation, or are available to use, depending on the specific TotalGuide components that will be integrated with the host application. The following sections define which functions correspond to different "flavors" of TotalGuide builds.

gp_appmain.h

The functions in this file must be implemented regardless of which TotalGuide components are integrated with the host application:

- `GPCB_AppGuideInited(...)`
- `GPCB_AppGuideExited(...)`
- `GPCB_AppGetHostState(...)`
- `GPCB_AppSetGuideState(...)`
- `GPCB_AppHostVersion(...)`
- `GPCB_AppHostResetRequest(...)`
- `GPCB_AppGuideRequestHostFactoryReset(...)`
- `GPCB_AppGuideRequestHostTogglePowerState(...)`
- `GPCB_FactoryRestoreFinish(...)`

This file also defines the following functions exported by TotalGuide which must be called by the host application to coordinate behavior:

- `GP_AppGuideInit (...)`
- `GP_AppGuideExit(...)`
- `GP_AppGuideHostStartupFinished(...)`
- `GP_AppSetHostState(...)`
- `GP_AppGetLastStartupFlag(...)`

The following functions in gp_appmain.h are also exported by TotalGuide, and may be called by the host application as required:

- `GP_AppGetGuideState(...)`
- `GP_AppGuideVersion(...)`
- `GP_AppGetROFileSystemPath(...)`
- `GP_AppGetRWFileSystemPath(...)`

gp_chanlist.h

The functions in this file are exported by TotalGuide TV, and may be invoked by the host application only if that component exists in the TotalGuide implementation being integrated:

	TV	**Broadband**
GP_ChannelListSize(...)	available	no
GP_ChannelListGetList(...)	available	no

	TV	Broadband
GP_ChannelListSetList(...)	available	no
GP_ChannelListSetEnableChannel(...)	available	no
GP_ChannelInfoForChannelID(...)	available	no
GP_SetDVBFrequencyMap(...)	available	no
GP_ScanSaveChannelList(...)	available	no

gp_channel.h

The following functions in this file are required by TotalGuide TV, and must be implemented if that component exists in the TotalGuide implementation being integrated with the host application:

	TV	Broadband
GPCB_ChannelChangeRequest(...)	required	no
GPCB_ChannelCurrentChannel(...)	required	no
GPCB_ChannelListChanged(...)	required	no
GPCB_ScanChannelScanStatus(...)	required	no
GPCB_ScanRequestChannelScan(...)	required	no

This file also includes prototypes for the following functions exported by TotalGuide, which may be invoked only if TotalGuide TV is being integrated with the host application:

	TV	Broadband
GP_ChannelChanged(...)	available	no

gp_datetime.h

The callback function in this file must be implemented regardless of which TotalGuide components are integrated with the host application:

- `GPCB_TimeZoneChanged(...)`

The following functions in gp_datetime.h are exported by TotalGuide, and may be called by the host application as required, when using any "flavor" of TotalGuide:

- `GP_GetRoviDataTimeZoneInfo(...)`
- `GP_SetTimeZoneInfo(...)`

gp_device.h

The following functions in this file are required by TotalGuide TV, and must be implemented if that component exists in the TotalGuide implementation being integrated with the host application:

	TV	Broadband
GPCB_DevicesChanged(...)	required	no

This file also includes prototypes for the following functions exported by TotalGuide, which may be invoked only if TotalGuide TV is being integrated with the host application:

	TV	Broadband
GP_DevicesChanged(...)	available	no
GP_DeviceGetNumDevices(...)	available	no
GP_DeviceGetDeviceList(...)	available	no
GP_DeviceGetInput(...)	available	no
GP_DeviceSetInput(...)	available	no
GP_DeviceGetRecorder(...)	available	no
GP_DeviceSetRecorder(...)	available	no
GP_DeviceGetRecDevNumQuals(...)	available	no
GP_DeviceGetRecDevQualityList(...)	available	no
GP_DeviceGetRecDevQuality(...)	available	no
GP_DeviceSetRecDevQuality(...)	available	no
GP_DeviceApplySetDevices(...)	available	no
GP_DeviceGetConfiguredServices (...)	available	no

gp_joblist.h

The following functions in this file are required by TotalGuide TV, and must be implemented if that component exists in the TotalGuide implementation being integrated with the host application:

	TV	Broadband
GPCB_JobSetRecordStart(...)	required	no
GPCB_JobSetRecordEnd(...)	required	no

This file also includes prototypes for the following functions exported by TotalGuide, which may be invoked only if TotalGuide TV is being integrated with the host application:

	TV	Broadband
GP_JobGetJobListByTime(...)	available	no
GP_JobGetJob(...)	available	no
GP_JobSetJob(...)	available	no
GP_JobQueryResolution(...)	available	no

In addition to the functions listed in the table above, gp_joblist.h includes prototypes for the following deprecated functions:

- `GP_JobGetJobExpInfo(...)`
- `GP_JobFreeJobExpInfo(...)`

Do not use these deprecated functions.

gp_jobs.h

The following functions in this file are required by TotalGuide TV, and must be implemented if that component exists in the TotalGuide implementation being integrated with the host application:

	TV	Broadband
GPCB_JobInfoListsChanged(...)	required	no

This file also includes prototypes for the following functions exported by TotalGuide, which may be invoked only if TotalGuide TV is being integrated with the host application:

	TV	Broadband
GP_JobInfoListsChanged(...)	available	no

gp_setup.h

The callback function in this file must be implemented regardless of which TotalGuide components are integrated with the host application:

	TV	Broadband
GPCB_SetupChanged(...)	required	required

The following table shows the functions in this file that are exported by TotalGuide to the host application depending on the specific TotalGuide components being integrated.

	TV	Broadband
GP_SetupChanged(...)	available	available
GP_SetupGetSetupInfo(...)	available	available
GP_SetupSetSetupInfo(...)	available	available
GP_SetupRestoreFactoryFresh(...)	available	available
GP_SetupSetCountryList(...)	available	available
GP_SetupGetCountryList(...)	available	available
GP_SetupSetLanguageList(...)	available	available
GP_SetupGetLanguageList(...)	available	available
GP_SetupVerifyLocationCode(...)	available	available
GP_SetupGetDataRxMethod(...)	available	no
GP_SetupSetDataRxMethod(...)	available	no
GP_SetupSetNetworkProxy(...)	available	available
GP_SetupIsUserSetupComplete(...)	available	available
GP_SetupGetDataDownloadProgress(...)	available	no
GP_SetupGetDeviceRegion(...)	available	available
GP_SetupGetNoConnectionTime(...)	available	no
GP_SetupGetOvernightDownload (...)	available	no

	TV	Broadband
GP_SetupSetOvernightDownload (...)	available	no

gp_thirdpartyapp.h

The callback functions must be implemented for broadband builds that will interact with third-party provider applications co-resident on the host system.

	TV	Broadband
GPCB_PartnerAppReqExec(...)	no	optional
GPCB_PartnerAppReqTerm(...)	no	optional

The third-party application registration function has relevance only for broadband builds that use third-party provider applications co-resident on the host system.

	TV	Broadband
GP_PartnerAppReg(...)	no	available

gp_userinterface.h

The following table shows the functions in this file that are required by TotalGuide and must be implemented by the host application, depending on the specific TotalGuide components being integrated.

	TV	Broadband
GPCB_UIRequestScreenOn(...)	required	required
GPCB_UIRequestMiniScreenOn(...)	required	no
GPCB_UIRequestScreenOff(...)	required	required
GPCB_UIRequestKeys(...)	required	required
GPCB_UIRequestPIPTurnOn(...)	required	no
GPCB_UIRequestPIPTurnOff(...)	required	no
GPCB_UIRequestPlayVideo(...)	required	required
GPCB_UIRequestPlayVideoFinish(...)	required	required
GPCB_UISetVideoVol(...)	required	required
GPCB_UIGetVideoVol(...)	required	required
GPCB_UISetPIPParamter(...)	required	required
GPCB_UIGetPIPParamter(...)	required	required

The following table shows the functions in this file that are exported by TotalGuide to the host application depending on the specific TotalGuide components being integrated.

	TV	Broadband
GP_UITurnScreenOn(...)	available	available
GP_UITurnMiniScreenOn(...)	available	no

	TV	Broadband
GP_UITurnScreenOff(...)	available	available
GP_UIRemoteKeyInput(...)	available	available
GP_UISetScreenResolution(...)	available	available
GP_UISetMiniScreenEnabledSetting(...)	available	no
GP_UISetMiniScreenPosAdjust(...)	available	no
GP_UIGetScreenResolution (...)	available	available
GP_UIGetMiniScreenEnabledSetting(...)	available	no
GP_UIGetMiniScreenPosAdjust(...)	available	no
GP_UISetPlayVideoReady(...)	available	available
GP_UISetPlayVideoFinishReady(...)	available	available
GP_UIBlockPIP(...)	available	no
GP_UIUnblockPIP(...)	available	no
GP_UISetInfoBoxText(...)	available	available

In addition to the functions listed in the tables above, gp_userinterface.h includes prototypes for the following deprecated functions:

- `GP_UIGetScreenPosition(...)`
- `GP_UIGetPIPDimensions (...)`
- `GP_UIGetPIPPosition(...)`
- `GP_UISetScreenPosition(...)`
- `GP_UISetPIPDimensions(...)`
- `GP_UISetPIPPosition(...)`
- `GP_UISendCharacter(...)`
- `GP_UISetPIPParamter(...)`
- `GP_UIGetPIPParamter(...)`
- `GP_UISetScreenDimensions(...)`
- `GP_UIGetScreenDimensions(...)`

Do not use these deprecated functions.

Use cases

This section describes a variety of use cases that require coordination of the host application and TotalGuide using the host API. The host application needs to invoke functions in TotalGuide to set up and invoke its features correctly; this section summarizes the main use cases. The use cases are presented in greater detail in the host API reference, including sequence diagrams to describe the interactions.

Startup

TotalGuide startup consists of two principal phases:

- Load and initialize the TotalGuide library -- parameters passed in by the host indicate the type of startup so TotalGuide can behave correctly

- Startup Sequence -- once TotalGuide is initialized: fetches regional settings and system configuration and fires up

Shutdown

Either the host or TotalGuide may initiate shutdown; the host API supports a variety of specific shutdown cases.

- Power Down: routine shutdown requested by the host to stop TotalGuide, when the host goes to low-power standby or (soft) AC power off.
- Factory Reset: a normal termination requested by the host to stop TotalGuide and restart with factory settings
- Upgrade Reset: If the TotalGuide upgrade feature is enabled, this can be used by TotalGuide to trigger the host to reset when an upgrade is available.
- Soft Reset: This is an abnormal termination triggered by TotalGuide when it discovers a fatal error. TotalGuide will free up resources, shut down, and notify the host. (The Host should react as if it is reloading the TotalGuide library.)
- Hard Reset: This is an abnormal termination triggered by TotalGuide when it encounters a second fatal error within 5 minutes of an earlier one. (This is a request to the Host to restart itself.)
- Exception Reset: This type of abnormal reset can be triggered by TotalGuide or the host. Upon restart, TotalGuide will ask the host for the exception history and then perform further diagnostics.
- Emergency Shutdown: This is a way for the host to terminate TotalGuide as quickly as possible. Recent TotalGuide data updates will not be saved to NVM, and there is a small of problems when TotalGuide restarts.

Device management

The host must notify TotalGuide of its device list so that TotalGuide can enable features and support relevant to the device (such as support for reminders and recordings). Device list exchange normally happens in the following situations:

- During the startup sequence after Guide startup
- Whenever the Host changes the device list
- Whenever TotalGuide changes the device list

Each of these use cases is detailed in the API reference documentation.

Channel management

In some modes of operation, TotalGuide depends on the Host's scanned channel list to find channels containing TotalGuide data stream. When the host has not provided this list, TotalGuide can request a channel scan.

Data and download handling

TotalGuide downloads program listing data so that it can display current listings to the user. This download can be done in one of two ways:

- via data embedded in the broadcast signal received by the television tuner

- via the internet

In either case, before data is actually downloaded, the user setup process provides TotalGuide with the user location (post code or area code) information. This information is used to select which TV listing data to download from the set of all TotalGuide data.

Tuner download

Downloading via broadcast data using the tuner consists of two stages: data scan and scheduled listing data download.

- *Data scan* identifies which channels in the tuner carry the TotalGuide data streams
- *Scheduled download* extracts the actual program listings from the tuned data streams

The initial steps of this process vary somewhat between North American and European builds. For North America, the data scan and listing data download are scheduled at some point after user setup. For the European region, both data scan and listing data download are triggered immediately after the user setup. (However, if this immediate download process terminates for some reason, it is simply rescheduled at another time.)

The initial data download always takes place following the initial data scan. Subsequent downloads of listings data take place at scheduled times, based on data contained in the TotalGuide data stream. If a download does not complete for some reason (such as a user's channel change seizing the tuner), then it resumes at the next scheduled time.

Internet download

Downloading program listings from the internet works more simply; no tuner data scan is needed. TotalGuide begins to download program data from the internet immediately after user setup. Future data download times are assigned by the internet server and TotalGuide schedules and performs the downloads as specified.

UI access

The host interacts with TotalGuide to display TotalGuide's user interface according to Guide requirements. Usually TotalGuide requires that the following is managed by the Host:

- Guide-supported key mapping
- Screen (OSD) resources
- Picture-in-picture video display

Major UI interactions include the following:

- Turn Guide On -- a key press triggers TotalGuide to request a full Guide screen display with a PIP request to the host.

- User Turn Guide Off-- a key press triggers a full Guide screen off and the Host resumes a full screen video display.

- Host Turn Guide Off -- the host triggers a full Guide screen off based on some other event such as host power down.

3

Porting to a new platform

This chapter describes how to port TotalGuide to a new hardware/OS platform. This is different activity from integrating TotalGuide with a host application (described in Chapter 2) although it may be part of the same project.

Overview

Rovi TotalGuide runs on top of a system abstraction known as the Rovi Porting Layer, or RPL. The default implementation of the porting layer matches the Rovi reference platform.

To port TotalGuide to a new platform, you must redevelop RPL to provide implementations for its functions that are native to the target platform. This allows TotalGuide to run unchanged on the updated porting layer.

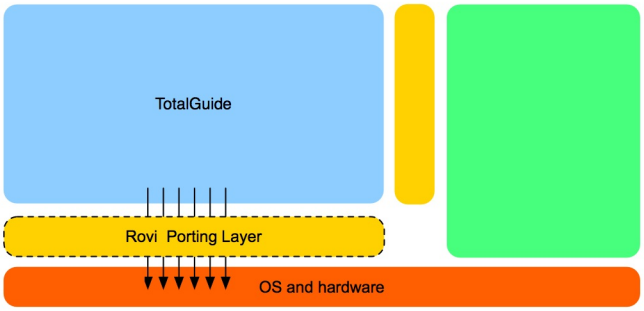

Figure 3.1. Implementing the Rovi Porting Layer

The reference implementation included with the product distribution provides a starting point for your redevelopment of the Rovi Porting Layer.

API functions

Header files

The files at .../src/PortingLayer/Include/ include the following RPL header files:

- `rpl_aud.h`
- `rpl_common.h`
- `rpl_crypto.h (default implementation is provided using OpenSSL)`
- `rpl_err.h`
- `rpl_file.h`
- `rpl_gfx.h`
- `rpl_grtypes.h`
- `rpl_init.h`
- `rpl_mpeg.h`
- `rpl_net.h`
- `rpl_os.h`
- `rpl_util.h`
- `pal/rovi_pal.h`
- `pal/rovi_pal_types.h`
- `pal/rovi_pal_utils.h`

Not all the function prototypes contained in these files require full implementation, depending on the specific TotalGuide components that will be used with RPL. The following sections define which functions must be implemented for use with different "flavors" of TotalGuide builds.

rpl_aud.h

This section describes the implementation of functions contained in the rpl_aud.h header file.

All functions in this file must be fully implemented regardless of build type.

	TV	**Broadband**
rpl_AudioPlayFile(…)	required	required
rpl_AudioPlayStop(…)	required	required
rpl_SetAudioVolume(…)	required	required
rpl_GetAudioVolume(…)	required	required
rpl_EnableAudioMute(…)	required	required
rpl_DisableAudioMute(…)	required	required
rpl_GetAudioMuteStatus(…)	required	required
rpl_EnableAudio(…)	required	required
rpl_DisableAudio(…)	required	required

rpl_crypto.h

This section describes the implementation of functions contained in the rpl_crypto.h header file.

The functions in this file must all be implemented regardless of build type.

Functions marked (stub only) are not required by TotalGuide, but some implementation is required to successfully build RPL. For these functions, the implementation may consist of a simple return statement.

	TV	Broadband
rpl_Crypto_Create(…)	*(stub only)*	*(stub only)*
rpl_Crypto_SetKey(…)	*(stub only)*	*(stub only)*
rpl_Crypto_Process(…)	*(stub only)*	*(stub only)*
rpl_Crypto_Destroy(…)	*(stub only)*	*(stub only)*

rpl_file.h

This section describes the implementation of functions contained in the rpl_file.h header file.

The functions in this file must all be implemented regardless of build type.

Functions marked (stub only) are not required by TotalGuide, but some implementation is required to successfully build RPL. For these functions, the implementation may consist of a simple return statement.

	TV	Broadband
rpl_FileFOpen(…)	required	required
rpl_FileFeof(…)	required	required
rpl_FileFClose(…)	required	required
rpl_FileFFlush(…)	required	required
rpl_FileFSeek(…)	required	required
rpl_FileFTell(…)	required	required
rpl_FileFGetC(…)	required	required
rpl_FileFGetS(…)	required	required
rpl_FileFPutC(…)	required	required
rpl_FileFPutS(…)	required	required
rpl_FileFRead(…)	required	required
rpl_FileFWrite(…)	required	required
rpl_FileFUnlink(…)	required	required
rpl_FileFmkdir(…)	required	required
rpl_FileOpendir(…)	required	required
rpl_FileReaddir(…)	required	required
rpl_FileClosedir(…)	required	required
rpl_FileFsync(…)	required	required
rpl_FileFRename(…)	required	required
rpl_FileFStat(…)	required	required
rpl_FileGetDiskStats(…)	*(stub only)*	*(stub only)*
rpl_FileEmptyDir(…)	required	required

	TV	Broadband
rpl_FileReaddirex(…)	required	required
rpl_FileFtruncate(…)	required	required
rpl_FileHasDisk(…)	*(stub only)*	*(stub only)*
rpl_FileMkTempDir(…)	*(stub only)*	*(stub only)*
rpl_FileGetMountType(…)	*(stub only)*	*(stub only)*
rpl_FileMount(…)	*(stub only)*	*(stub only)*
rpl_FileUnMount(…)	*(stub only)*	*(stub only)*
rpl_FileIsMounted(…)	*(stub only)*	*(stub only)*
rpl_FileOSCanonicalize(…)	*(stub only)*	*(stub only)*
rpl_FileScanDir(…)	*(stub only)*	*(stub only)*
rpl_FileMmap(…)	*(stub only)*	*(stub only)*
rpl_FileMunmap(…)	*(stub only)*	*(stub only)*
rpl_FileFRewind(…)	*(stub only)*	*(stub only)*
rpl_FileTelldir(…)	*(stub only)*	*(stub only)*
rpl_FileSeekdir(…)	*(stub only)*	*(stub only)*
rpl_FileRewinddir(…)	*(stub only)*	*(stub only)*

rpl_gfx.h

This section describes the implementation of functions contained in the rpl_gfx.h header file.

All functions in this file must be fully implemented regardless of build type.

	TV	Broadband
rpl_GFxInitialize(…)	required	required
rpl_GFxExit(…)	required	required
rpl_GFxShowScreen(…)	required	required
rpl_GFxHideScreen(…)	required	required
rpl_GFxCreateSurface(…)	required	required
rpl_GFxDestroySurface(…)	required	required
rpl_GFxSetSurfaceAlpha(…)	required	required
rpl_GFxSetSurfaceAbsoluteAlpha(…)	required	required
rpl_GFxLockSurface(…)	required	required
rpl_GFxUnlockSurface(…)	required	required
rpl_GFxTransformColor(…)	required	required
rpl_GFxStretchBlit(…)	required	required
rpl_GFxBitblt(…)	required	required
rpl_GFxBitbltCopy(…)	required	required
rpl_GFxCopySurfaceFromOSD(…)	required	required

	TV	Broadband
rpl_GFxDrawRect(…)	required	required
rpl_GFxFillRect(…)	required	required
rpl_GFxDrawLine(…)	required	required
rpl_GFxDrawEllipse(…)	required	required
rpl_GFxFillEllipse(…)	required	required
rpl_GFxFillTriangle(…)	required	required
rpl_GFxSetClipRect(…)	required	required
rpl_GFxGetClipRect(…)	required	required
rpl_GFxUpdateOSD(…)	required	required

rpl_mpeg.h

This section describes the implementation of functions contained in the rpl_mpeg.h header file.

Implementation requirements for the functions in this file vary by build type. Refer to the table for details.

Functions marked (stub only) are not required by TotalGuide, but some implementation is required to successfully build RPL. For these functions, the implementation may consist of a simple return statement.

	TV	Broadband
rpl_MPEGInitialize(…)	required	*(stub only)*
rpl_MPEGExit(…)	required	*(stub only)*
rpl_MPEGReset(…)	required	*(stub only)*
rpl_MPEGEnableSource(…)	required	*(stub only)*
rpl_MPEGDisableSource(…)	required	*(stub only)*

rpl_net.h

This section describes the implementation of functions contained in the rpl_net.h header file.

Some functions in this file are present in RPL, but their implementation is not required for TotalGuide. Refer to the table for details.

Functions marked (stub only) are not required by TotalGuide, but some implementation is required to successfully build RPL. For these functions, the implementation may consist of a simple return statement.

	TV	Broadband
rpl_NetInitialize(…)	required	required
rpl_NetExit(…)	required	required
rpl_NetSocket(…)	required	required
rpl_NetConnect(…)	required	required
rpl_NetBind(…)	required	required

	TV	Broadband
rpl_NetListen(…)	required	required
rpl_NetAccept(…)	required	required
rpl_NetRecv(…)	required	required
rpl_NetRecvFrom(…)	required	required
rpl_NetSend(…)	required	required
rpl_NetSendTo(…)	required	required
rpl_NetClose(…)	required	required
rpl_NetSetSockOpt(…)	required	required
rpl_NetGetFdSetSize(…)	required	required
rpl_NetFdZero(…)	required	required
rpl_NetFdSet(…)	required	required
rpl_NetFdIsSet(…)	required	required
rpl_NetSelect(…)	required	required
rpl_NetGetSockName(…)	required	required
rpl_NetGetPeerName(…)	required	required
rpl_NetGetHostByNameR(…)	required	required
rpl_NetGetHostByAddrR(…)	required	required
rpl_NetInetAddr(…)	required	required
rpl_NetGetMacAddress(…)	required	required
rpl_NetGetStatus(…)	required	required
rpl_NetGetLastError(…)	required	required
rpl_NetPing(…)	required	required
rpl_NetRestart(…)	*(stub only)*	*(stub only)*
rpl_SendLinkPacket(…)	*(stub only)*	*(stub only)*
rpl_NetCbSystemOpen(…)	*(stub only)*	*(stub only)*
rpl_NetCbSystemClose(…)	*(stub only)*	*(stub only)*
rpl_NetCbSystemWork(…)	*(stub only)*	*(stub only)*
rpl_NetGetIPAddress(…)	no	no
rpl_NetFdClr(…)	no	no
rpl_NetShutdown(…)	*(stub only)*	*(stub only)*
rpl_NetGetSockOpt(…)	*(stub only)*	*(stub only)*
rpl_NetSocketPair(…)	no	no
rpl_NetIoctl(…)	required	required
rpl_NetWritev(…)	required	required

rpl_util.h

This section describes the implementation of functions contained in the rpl_util.h header file.

Some functions in this file are present in RPL, but their implementation is not required for TotalGuide. Refer to the table for details.

	TV	Broadband
rpl_UtilLogMessage(…)	required	required
rpl_UtilGetCallerAddress(…)	required	required
rpl_UtilGetCallerStackAddress(…)	required	required
rpl_UtilGetCallChain(…)	required	required
rpl_UtilGetExceptionDiagnostics(…)	required	required
rpl_UtilStoreResetDiagnostics(…)	no	no
rpl_UtilGetResetDiagnostics(…)	no	no

rpl_os.h

This section describes the implementation of functions contained in the rpl_os.h header file.

Some functions in this file are present in RPL, but their implementation is not required for TotalGuide. Refer to the table for details.

Functions marked (stub only) are not required by TotalGuide, but some implementation is required to successfully build RPL. For these functions, the implementation may consist of a simple return statement.

	TV	Broadband
rpl_OSTaskCreate(…)	required	required
rpl_OSTaskDelete(…)	required	required
rpl_OSGetTaskId(…)	required	required
rpl_OSGetNativeTaskId(…)	required	required
rpl_OSSemaphoreCreate(…)	required	required
rpl_OSSemaphoreDelete(…)	required	required
rpl_OSSemaphoreAcquire(…)	required	required
rpl_OSSemaphoreRelease(…)	required	required
rpl_OSGetTimeOfDay(…)	required	required
rpl_OSGetSystemTicks(…)	required	required
rpl_OSSleepTicks(…)	required	required
rpl_OSReserveMemory(…)	required	required
rpl_OSReclaimMemory(…)	required	required
rpl_OSReallocMemory(…)	required	required
rpl_OSCallocMemory(…)	required	required
rpl_TaskLock(…)	required	required
rpl_TaskUnLock(…)	required	required
rpl_OSMutexInit(…)	required	required

	TV	Broadband
rpl_OSMutexDestroy(…)	required	required
rpl_OSMutexLock(…)	required	required
rpl_OSCondInit(…)	required	required
rpl_OSCondDestroy(…)	required	required
rpl_OSCondWait(…)	required	required
rpl_OSTaskJoin(…)	required	required
rpl_OSTaskDetach(…)	required	required
rpl_OSTaskSetSpecific(…)	required	required
rpl_OSTaskGetSpecific(…)	required	required
rpl_OSTaskOnce(…)	no	no
rpl_OSTaskExit(…)	*(stub only)*	*(stub only)*
rpl_OSTaskSetName(…)	no	no
rpl_OSSystemRestart(…)	no	no
rpl_OSSystemShutdown(…)	no	no
rpl_OSGetOsString(…)	*(stub only)*	*(stub only)*

rpl_pal.h

This section describes the implementation of functions contained in the rpl_pal.h header file.

Some functions in this file are present in RPL, but their implementation is not required for TotalGuide. Refer to the table for details.

	TV	Broadband
rpbl_Init(…)	required	required
rpbl_Uninit(…)	required	required
rpblCapabilities_Common(…)	no	no
rpblCapabilities_Stream(…)	no	no
rpblCapabilities_Codec(…)	no	no
rpblCapabilities_Playback(…)	no	no
rpblRenderer_Create(…)	required	required
rpblRenderer_Destroy(…)	required	required
rpblRenderer_Configure(…)	required	required
rpblRenderer_GetBuffer(…)	required	required
rpblRenderer_DeliverBuffer(…)	required	required
rpblRenderer_SetState(…)	required	required
rpblRenderer_GetState(…)	required	required
rpblRenderer_SubscribeToEvent(…)	required	required
rpblRenderer_Flush(…)	required	required

rpl_init.h

The two functions in this file are provided for host initialization and de-initialization of RPL.

- `rplInit(...)`
- `rplUninit(...)`

The reference implementation of RPL allocates resources when the host calls rplInit() and frees them on rplUninit(), but there is no requirement that other implementations work this way. If your RPL implementation will be initialized and de-initialized by the host in this way, then use these functions as the control points for this. If your implementation does not require host initialization and de-initialization, then you may omit these functions.

Threading model

It is normal for multiple threads in TotalGuide to invoke RPL functions. Therefore RPL functions should always be implemented in a threadsafe fashion to ensure compatibility.