Sabre Red Apps

Developer Toolkit Overview

25 October 2012

Red Apps are optional, authorized applications that extend the capabilities of *Sabre® Red™ Workspace*. *Red Apps* are *Sabre*'s branded version of an Eclipse plug-in. They plug into and become part of the *Red Workspace*. A *Red App* can be something as simple as a script, an app that intelligently integrates mapping information to help agencies better serve their customers, or a pre-populated form that helps agencies improve customer service. *Sabre Red App* Certified Developers design and code these applications, and then wrap them as plug-ins for integration into *Sabre Red Workspace.*

Red Apps leverage the technical capabilities of the Eclipse Rich Client Platform, the open source software upon which the *Red Workspace* platform is built and operates. *Red Apps* are developed by *Sabre* teams and *Sabre Red App* Certified Developers. They are marketed and deployed through the *Sabre® Red™ App Centre* using *Sabre Red Workspace's* provisioning technology.

Red Apps are visually and physically integrated within the *Red Workspace* and share data with other elements of the *Red Workspace*, thereby providing a whole new level of integration. By choosing *Red Apps* that meet specific business needs, and even the specific needs of each agent in an office, certified developers will be able to create a truly custom *Red Workspace* environment for optimal agent productivity and customer service.

Learn more:

Sabre Red Workspace

Sabre Red App Centre

Sabre Red App Developer Toolkit. Read about the types of *Red Apps* that *Sabre Red App* Certified Developers can build, the functionality in the *Red App Developer Toolkit*, and skills for developers.

Technical Description of a Red App

Developing, Purchasing, Provisioning, and Maintaining Red Apps. Review the process, from on boarding and development, to desktop deployment and upgrades.

Sabre Red App Developer Toolkit

You can build *Red Apps* with a variety of internet and programming technologies, and add communications with *Sabre* data stores, such as the *Sabre*® GDS, and other *Red Apps*. Your *Red Apps* can be integrated into *Sabre Red Workspace* visually, or they can collect data and perform tasks behind the scenes.

The *Sabre Red App Developer Toolkit* (or simply, "*Developer Toolkit*") is a robust toolkit with all the tools, documentation, sample plug-ins — including source code — and utilities developers need to build *Red Apps* that run seamlessly in *Sabre Red Workspace*.

Why Was the Developer Toolkit Created?

The *Developer Toolkit* was created to give developers greater flexibility with building solutions that integrate within the *Workspace* in new ways when compared to existing developer tools. By using *Red App* technology, developers are able to build *Red Apps* with functionality that is not provided within *Sabre Red Workspace*, opening a whole new world of possibilities, decreasing time to market for innovative apps, and giving travel buyers the power to choose the tools they need to stay ahead.

Choose from a Variety of Development Technologies

Your choice of a programming language will depend on your development background and experience and the scope of the *Red App* that you want to build.

Because the technology behind the *Sabre Red Workspace* platform (Eclipse Rich Client Platform) allows for plug-ins (*Red Apps*) to be built using a variety of technologies, your options range from cutting-edge technologies to more simple domain-driven technologies. Examples of cutting edge technologies include Java SWT and web technologies that allow advanced, limitless integration options in the *Workspace*. Simpler domain-driven technologies include examples such as using *Sabre Scribe* and Qik for developing custom workflows or automating and customizing *Sabre* system processes.

Supported List of Technologies:

- Java, SWT, Swing
- Web-based Internet technologies: HTML, Flash, Flex, JavaScript. If running on the server side, your website can be based on Java technologies, for example, servlets or JSP. Your website can also be based on other technologies, such as Microsoft .NET Framework or PHP.

  A *Red App* that is a web app or a web page can use a custom implementation of an editor or a default implementation of an editor and a view. The custom implementation requires Java code; integration as an editor, and is based on HTML, Adobe Flash, or Adobe Flex. The default editor and view implementations do not use Java code and are intended for simpler apps that do not dynamically create a URL at runtime or manage the life cycle of the editor. A JavaScript-based *Red App* must use these default editor or view classes.

From the perspective of *Red Apps*, all of these technologies are HTML.

Websites with an applet must be wrapped as Swing apps.

- *Sabre Scribe* scripts: You can wrap your compiled *Sabre Scribe* script files as plug-ins for *Sabre Red Workspace*. The wrapping of a *Sabre Scribe* script does not change the functionality of the script, but instead, it improves the deployment and version control of the scripts. *Sabre Scribe* scripts can also read to or write from files in a plug-in, and update files when you have a new version of your plug-in. Your *Sabre Scribe Red App* can use active listening to trigger your scripts. In addition, you can use the *Red App* Bundle wizard to secure your scripts from unauthorized use. The Open SabreScript dialog in *Red Workspace* lists your scripts.
- *Sabre Qik Developer* apps: Qik apps can be wrapped as plug-ins. Developers can build robust workflows, automation, and user interfaces that are compiled into a series of instructions which the Qik Runtime Engine processes from within *Sabre Red Workspace*. Qik scripts are wrapped as editors in *Red Workspace*. An end-user can run one Qik *Red App* at a time.

Note: Technology choice will dictate the visual and functional integration capabilities that can be achieved through the *Red Apps* that are built. The *Red App Developer Toolkit* describes detailed specifications for integration.

What Design Options Are Available?

Design Red Apps with Multiple User Interface Options

Using the *Red App Developer Toolkit*, you can design and build *Red Apps* with a variety of options for UI elements:

- Full application editors, which *Red Workspace* renders as tabs
- Views (assistant tools): These are tools that complement the capabilities of a primary editor/tab inside *Sabre Red Workspace*. For this reason, end-users refer to views as assistant tools. You can design your view as either a horizontal or a vertical layout.
- Pop-up dialogs: These objects are very similar to a window. Dialogs in *Sabre Red Workspace* are the same as dialogs for most other applications. They can have a variety of GUI controls, such as radio buttons, check boxes, text fields, and combo boxes. Dialogs are most commonly modal, which means that they interrupt the workflow and require the user to interact with the window before continuing.
- Notification Services dialogs: A notification is a type of message box that most typically provides information to end-users in *Sabre Red Workspace*. A *Red App* may consist solely of Notification Services or notifications may complement other *Red Apps*. Notification Services dialogs are non-modal.

  You can create the following types of *Red App* notifications:

  - A basic text notification
  - A notification with a link to reference other actions of a *Red App*
  - A custom notification with a progress bar indicating a specific task or process is running in the background

- Basic menu and complex menu contributions: You can provide menu contributions for your *Red App* that are visible and menus that are activated conditionally, based on handler policies. The *Red App* help provides guidelines for menus based on application type.

Developers can also develop *Red Apps* that do not have a UI component tied to them, such as running tasks in the background.

Red App Communications Services and Events

To make communications possible, *Red Workspace* relies on the SRWRuntime communications bus. SRWRuntime is the single gateway between *Red Apps* and services that the *Sabre Red Workspace* platform exposes, from intercepting *Sabre* emulator commands for markup, to communicating with *Sabre Web Services* or *Sabre* emulator (*Sabre* emulator is classic view) to get *Sabre* GDS content, or app-to-app communication.

You can add an asynchronous event listener to *Red Apps*. Event listeners are notified about SRWRuntime calls either before or after an actual call to SRWRuntime. *Sabre* has several published events. A *Red App* can filter the events by using an action code to pass a command prefix. An example of an event listener is a *Red App* that subscribes to an Air Availability command and triggers a certain *Red App* action when the event is generated.

Red Apps can also listen for events that graphical view publishes.

SRWRuntime services and events provide the following functionality that you can add to a *Red App*:

- Intercepting *Sabre* emulator commands and markup
- Sending information and commands to *Sabre* emulator
- *Sabre* GDS (*Sabre* host) communications: This service can be used to send a single request or multiple, consecutive requests to the host, bypassing *Sabre* emulator and going directly to the *Sabre* host.
- Communicating with *Sabre Web Services*: This service lets *Red Apps* make a single service call or multiple, consecutive requests sharing the same *Sabre Red Workspace Sabre* session.
- An SRWRuntime locking service: This service is implemented when sending multiple, consecutive requests through the *Sabre* GDS or *Web Services*.
- Listening to *Sabre* emulator requests or responses, either before or after they are sent to the emulator

Red App Application Services and APIs

The *Red App Developer Toolkit* provides a comprehensive set of traditional OSGi services and other non-traditional plug-in services (APIs) that you can use for optimizing your development. Key services include the following:

- Agent Profile Service: This service retrieves information about an agent who is logged in to *Sabre Red Workspace* and the agent's *Sabre* session. The information includes EPR, PCC, language, country, e-mail, etc.
- ConfigService: This is a central plug-in service that stores properties from different *Red App* plug-ins. This service provides a convenient API to read the properties efficiently.
- Logger Services: A central plug-in service that provides logging services for *Red Apps*.
- Agent Work Area: This service retrieves agent work area data.
- Encode/Decode service: This service encodes or decodes text for a specified category.
- Internationalization/i18n
- Access to Resources in a plug-in: This service consists of Java methods for obtaining local resources, such as files or images, that are encapsulated inside a plug-in.
- Unpack and Copy Resources: This service automatically unpacks and copies files from your plug-in to *Sabre Red Workspace* for *Red App* websites that are run locally, and not from a JAR.
- First run actions and long-running operations in a plug-in.

Resources in the Developer Toolkit

The *Red App Developer Toolkit* provides all the resources that you need to make it all possible when it comes to building *Sabre Red Apps*.

The main features include:

- Development using the Eclipse Integrated Development Environment with the *Red App* target platform. The target platform includes the complete code base to develop and test *Red Apps* locally from your development environment.

  You set up a workspace for your *Red App* project, your *Red App* target platform, and a run configuration that lets you launch *Sabre Red Workspace* in development mode. This enables you to run the sample plug-ins and test your code. With these tools, you develop and manage your integrated *Red App* project.

  Later you will access CERT or PROD *Red Workspace* during the certification and testing phase of your *Red App*. This access is based on *Sabre's* approval.
- Integrated *Red App* wizards. These wizards enable you to easily evolve and wrap existing applications, such as websites, *Sabre Scribe* scripts, and other types of apps as “Red Apps,” and then plug the apps into the *Workspace*, or to create basic *Red App* projects that automate complex tasks by asking you a series of easy-to-answer questions.
- Integrated *Red App* help system, with detailed documentation about all *Red App Developer Toolkit* capabilities.

Note: After you set up your Red App target platform, you must install the Red App development tools to access the Red App help and Red App wizards!

- JavaDocs for specific services and APIs.
- Sample *Red App* plug-in projects, including source code, to demonstrate all the capabilities of the *Developer Toolkit* that developers can reuse and reference to get started.
- *Getting Started with Red Apps*. This document includes all the procedures you need to set up your *Red App* target platform with a run configuration, and to install the *Red App* development tools that access the *Red App* wizards and help. Procedures for upgrading a new version of the *Toolkit* and other maintenance procedures using Eclipse are also included.
- *Sabre Red Style Guide*. This has guidelines for GUI design to ensure that the look and feel of all *Red Apps* within *Sabre Red Workspace* is consistent. Obtain this on the *Dev Centre*.
- *Red App Developer Guidelines*. These guidelines help you ensure that your apps comply with *Sabre's* standards. Obtain this on the *Dev Centre*.
- *Red App Developer FAQs*. These FAQs have troubleshooting information, resources, and more.

What Skills Do Red App Developers Need?

Sabre Red App Certified Developers are required to have the following skills and expertise to develop plug-ins and clients for *Sabre Red Workspace*:

- Knowledge of the *Sabre Red Workspace* client, the technologies that the client is built upon, and the APIs that the client provides
- Java programming
- Development using the Eclipse IDE
- Knowledge of extension points, OSGi services, and APIs, if you are developing plug-ins that use Eclipse RCP functionality created by *Sabre*

Note: The technology that you choose will dictate the skill set that you need. The preceding list describes the recommended skills for *Sabre Red App* Certified Developers who want to build advanced *Red Apps* using Java and the most sophisticated capabilities in the *Red App Developer Toolkit*.

It is not necessary for developers to know about the server-side components in *Sabre Red Workspace* to integrate or implement an existing application into it.

Additionally, *Red App* Certified Developers who are creating *Red Apps* that use cardholder or personal data are required to have the following knowledge and skills, and to conform to these requirements when designing and developing *Red Apps*:

- Developers must be knowledgeable in secure coding techniques such as OWASP. (https://www.owasp.org/index.php/)
- The development of *Red Apps* must be based on secure coding guidelines to prevent common coding vulnerabilities in the software development process.

Developers must comply with all laws, including applicable data privacy laws and regulations governing the collection, storage, and use of user data.

Red App Developer Toolkit Support

The *Red App Developer Toolkit* was designed to easily allow you to build *Red Apps* that run seamlessly inside the *Sabre Red Workspace* by providing a large set of tools and documentation. If needed, additional support is also provided through the following means:

- *Sabre* Technical Support help desk specialists
- *The DevStream Sabre* Developer Community where *Red App* Certified Providers will be able to network with many other *Red App* Certified Providers and *Sabre* experts.

How Do I Get the Sabre Red App Developer Toolkit?

If you are already a *Sabre Red App* Certified Provider, log in and get started in the *Dev Centre*. There you'll find the *Developer Toolkit* and other resources to help you. (Log in on http://www.sabreredappcentre.com and then navigate to the *Dev Centre*.)

If not, simply complete the form to become Red App Certified, and we'll follow up with you soon.

Technical Description of a Red App

The components that make up a *Red App* are categorized as follows:

Components of a Red App

Red App Bundle File

For detailed information about these components, see the *Red App* help.

Components in a Red App

Plug-in projects: A *Red App* consists of a main plug-in project and one or more dependent plug-in projects. Generally, the main plug-in project has the most important or significant functionality, although dependent plug-ins can also include many types of functionality. All communications in a *Red App* must be included in the main plug-in.

plugin.xml: Every plug-in project in a *Red App* has a plugin.xml file located at the root of the project. As you add classes, extension points with properties and values, and dependencies to a plug-in project, Eclipse generates the plugin.xml file and formats these elements as XML.

Contact details: As a *Red App* provider, your *Red App* must include support contact information. This information gets published in *Sabre Red Workspace* along with your *Red App* to enable your customers to contact you if needed.

Code: Each plug-in project in a *Red App* that executes functionality has source code. Currently, you can code your *Red App* with Java, JavaScript, Adobe Flash or Flex, HTML and other browser technologies, Qik scripts, *Sabre Scribe* scripts, and Java Swing.

Background processes: A *Red App* can execute functionality in the form of background processes, for example, logging or operations on the first run of a plug-in.

UI or workspace elements: A *Red App* can have user interface elements. The choices are editors, views, menu contributions, status lines, pop-up dialogs, and notifications.

Red App ID: After *Sabre* accepts your application and *Red App* proposal, and after all agreements are signed between you and *Sabre*, *Sabre* assigns a unique *Red App* ID to your *Red App*. This ID is associated with your proposal. You will use this *Red App* ID in the redapp.xml configuration file, and when you submit your *Red App* for certification. You will also use this same ID to submit any subsequent corrections to previously failed *Red App* bundles and to patch releases.

redapp.xml configuration file: This configuration file registers your unique *Red App* ID, your publisher or company details, and all access and use of communications in a *Red App*. Communications include authorization to consume services that *Sabre* and other *Red Apps* register with the SRWRuntime communications bus, registration of event listeners, and published Event IDs. Every project has a single redapp.xml file that is located in the root of the main plug-in project.

build.properties file: This file includes all files and properties that create a binary build.

Custom folders and files: Special folders and files may include images, resources, scripts or executables, and properties files for translation.

Communications services, event listeners, and events: The redapp.xml configuration file in the main plug-in project implements communications in a *Red App*.

A *Red App* obtains an authentication token in the Java code to use synchronous services that are registered with SRWRuntime and to publish events. It does not authenticate to use its own services. Registration of an event listener also accomplishes authentication.

A *Red App* includes authorization to use services that are registered with the SRWRuntime bus and to publish its own events. A *Red App* also registers its own services, event listeners, and events in this file.

In addition to adding communications to Java-based or browser-based *Red Apps*, you can also publish communications events, register an event listener, and use communications services that are registered with the SRWRuntime bus in a Qik script. It is also possible to trigger a *Sabre Scribe* script with active listening for commands.

If your *Red App* is connected to graphical view (GV) or acts as an add-on for GV, your app can communicate with GV with the event listening mechanism in *Red Apps*.

A *Red App* can publish events. When the event occurs, the SRWRuntime bus notifies other event listeners about the event.

Security in Red Apps and security best practices: *Red Apps* that use or access cardholder or personal data must conform to all standards that *Sabre* requires. *Red App* providers manage this data according to *Sabre's* standards. For this information, see the *Red App Centre* or the *Dev Centre*. (https://www.sabreredappcentre.sabre.com/Brand-Developer.aspx)

Red App Bundle File

When you are done with all development and local testing, you create a *Red App* bundle ZIP file that consists of all plug-in JARs, a custom bundle properties file, and the digital certificate used to sign the bundle and all JARs. You submit the bundle ZIP file to *Sabre* on the *Dev Centre* for validation and certification. You are strongly encouraged to use the *Red App* Bundle wizard in the *Red App Developer Toolkit* to create a *Red App* bundle and ZIP file automatically.

Process for Developing, Purchasing, Provisioning, and Maintaining Red Apps

1. Review this *Sabre Red Apps Developer Toolkit Overview* for information about *Red Apps*, the *Red App Developer Toolkit* functionality that you can use to build *Red Apps*, and the process for development and deployment.

2. Before you apply to become a *Sabre Red App* Certified Developer, review the security requirements for *Red Apps*. This information is available on a link on the *Sabre Red App Centre*. (https://www.sabreredappcentre.sabre.com/Brand-Developer.aspx)

   You must create an idea and proposal for your *Red App* and submit it to *Sabre* together with your application. If applicable to your *Red App*, prepare security documentation to submit with your *Red App* bundle.

3. Complete and submit an application to become a *Sabre Red App* Certified Developer. For the application, go to the Login page on http://www.sabreredappcentre.com, and click **Become a Red App Developer**.

4. As part of the application process, submit your proposal for a *Red App* to *Sabre*. On the proposal form, select the technologies, communications, services, UI workbench elements, and any security-related content that you intend to use in your *Red App*.

After you submit your proposal, you cannot change the marketing or customer-facing name of your *Red App*. Later, when you submit your *Red App* to *Sabre* for certification, *Sabre* verifies that your *Red App* includes the name and selections on your proposal.

5. After *Sabre* approves your application and *Red App* proposal, and after you and *Sabre* agree on commercial and legal terms, *Sabre* does the following:
    - Sends you a notification with your *Sabre* ID and PCC.
    - Sends you a unique *Red App* ID

Note! Retain your *Red App* ID. You use this ID in your *Red App* plug-in project and when you submit your *Red App* bundle for certification, approval, and testing on the *Dev Centre*. You also use your *Red App* ID whenever you submit a failed bundle during the certification phase or a patch release.

The following topics explain the use of your *Sabre* ID and PCC.

A Certification (CERT) Sabre ID and PCC grants the following access:

- *Sabre Red Workspace* in development mode, which you launch from Eclipse IDE. From *Red Workspace*, use your CERT login credentials in *Sabre* emulator to log in to the CERT environment of the *Sabre* GDS. You can also change environments and use your Production (PROD) credentials to log in to PROD.
- The CERT environments of *Sabre Red Workspace* and the *Sabre* GDS from *Sabre* emulator. You will install and access CERT *Red Workspace* when *Sabre* asks you to test your *Red App* in CERT.

A PRODUCTION (PROD) Sabre ID and PCC grants the following access:

- The PROD environments of *Sabre Red Workspace* and the *Sabre* GDS from *Sabre* emulator. You will install and access PROD when *Sabre* asks you to test your *Red App* in PROD.
- Agency eServices (http://eservices.sabre.com)
- *Sabre Red App Centre* (www.sabreredappcentre.com). The *Dev Centre* is one of several sections in the *Sabre Red App Centre*. You always log in to the *Red App Centre* in order to access the *Dev Centre*.

Sabre assigns an ID to access the *DevStream* application and community upon approval.

Tip! *Sabre* has resources about working in the *Sabre* GDS. Some especially important resources follow:

Working in the CERT System explains how to change partitions and CERT refreshes in the *Sabre* GDS. When you get your *Sabre* ID, you can obtain this document on eservices.sabre.com.

Format Finder is a reference of *Sabre* host formats. When you get your *Sabre* ID, you can log in to *Format Finder* directly on `http://format-finder.sabre.com`.

For more information and troubleshooting, see the *Red App Developer FAQs* in the documentation folder, or download it from the *Dev Centre*.

Both *Getting Started with Red Apps* and the *Red App* help also have a "Red App Resources" topic.

6. Download and install the *Red App Developer Toolkit* on the *Dev Centre*.

    - *Getting Started with Red Apps* has all instructions to install and set up your development environment. When you extract the *Developer Toolkit* ZIP file, you will find *Getting Started with Red Apps* in the documentation folder.
    - Obtain and install all required software and tools. See the topic "Required Developer Software, Tools, and IDs" in *Getting Started with Red Apps*.
    - Within Eclipse, set up your *Red App* target platform and a run configuration.
    - Within Eclipse, install the *Red App* development tools. This integrates the *Red App* wizards and *Red App* developer help into your workspace in Eclipse IDE.

7. In the *Red App* help, review all *Red App* requirements and standards. Scan the books and topics that relate to the functionality you are including in your app. The *Red App* help is organized by functionality. Other topics describe important requirements, the sample plug-ins and wizards, and more.

8. Review the security requirements for *Red Apps*. Check these requirements before you code and submit your *Red App* for periodic updates. You can review these requirements on the *Dev Centre* or without logging in on https://www.sabreredappcentre.sabre.com/Brand-Developer.aspx.

9. Design your custom application using the technologies, services, communications, UI workbench elements, and PCI or personal data that you selected in your proposal.

10. Code your app. Add the functionality, communications, authentication, authorization, and registration in your app, as needed.

11. Obtain a digital certificate from a certificate authority that *Sabre* supports. These are listed in *Getting Started with Red Apps* and in the *Red App* help.

12. Test your *Red App* by launching *Sabre Red Workspace* in development mode from Eclipse IDE.

13. Prepare use cases for all functionality in your *Red App* and marketing artifacts before submitting your *Red App* on the *Dev Centre*.

14. If applicable to your *Red App*, prepare security documentation to submit with your *Red App* bundle.

15. Create your *Red App* bundle ZIP file by using the *Red App* Bundle wizard.

16. On the *Dev Centre*, submit a first *Red App* release bundle as version 1.0.0.date-timestamp qualifier to *Sabre*.

17. After you upload your *Red App* bundle, *Sabre* begins the process for validating and certifying your *Red App*. If *Sabre* certifies your app, *Sabre* enables it in your CERT environment of *Sabre Red Workspace*, and asks you to test your *Red App* in CERT.

18. If CERT *Sabre Red Workspace* is not installed on your local machine, install it. The URL for installation is on the *Dev Centre*.

19. Test your *Red App* in the CERT environments of *Sabre Red Workspace*, the *Sabre* GDS, or *Sabre* emulator.

20. If your *Red App* meets your acceptance criteria in CERT, click to post your CERT test results on the *Dev Centre*. Posting your test results sends verification to *Sabre*.

    Identify beta testers who will test your *Red App* in PROD.

    When you post CERT test results, provide from one to five PCC_IDs of beta customers who will test your *Red App* in PROD *Red Workspace* when the time arrives for actual testing in PROD. Beta customers do not test in CERT.

    Sabre loads your *Red App* to PROD *Red Workspace* and provisions it to you and your beta customers in PROD.

21. Download and install PROD *Sabre Red Workspace*. The URL for PROD *Sabre Red Workspace* installation is on the *Dev Centre*.

22. Test your *Red App* in the PROD environments of *Sabre Red Workspace* and the *Sabre* GDS or *Sabre* emulator. Collaborate with beta customers to test your *Red App*.

23. If your *Red App* meets your acceptance criteria in PROD, click to post PROD test results on the *Dev Centre*. Confirm that the beta customers accepted your beta app. When you provide testing results on the *Dev Centre*, this sends verification to *Sabre*.

    After you mark PROD testing as passed, *Sabre* removes the *Red App* from the beta customers, while retaining the *Red App* in your *Red Workspace*. If your beta customers want to use your *Red App*, they must purchase it on the *Sabre Red App Centre*.

    Sabre publishes your *Red App* on the *Market Centre* as is, and notifies you that your *Red App* is listed for sale.

24. Agency customers shop for and contact your company’s purchase contact on record to purchase your *Red App*.

25. You or your purchase contact work with the buyer to agree to the total purchase price. You also obtain the PCCs and *Sabre* IDs to receive your *Red App*.

26. Confirm the sale on the *Dev Centre*, and specify the end-users who are to receive the *Red App* and the total purchase price.

27. The buyer approves the purchase. You or your purchase contact invoice the buyer for the full purchase price and collect payment from the buyer.

28. *Sabre* invoices your company according to your revenue share agreement, and provisions the *Red App* to the entitled users.

29. The agency administrator in the purchasing agency uses the *Sabre Red Workspace* Agency Administrator Tools to enable your *Red App* for entitled users.

30. If your *Red App* does not meet your expectations in CERT or PROD, mark that the test failed on the *Dev Centre*.

31. *Sabre* will ask you to resolve the failure in your *Red App*. Correct the problem. At a minimum, change the date-timestamp qualifier (part 4) of the original version number and submit your bundle to *Sabre* again. *Sabre* executes the same approval process and notifies you when you can test again in CERT or PROD.

    Each version of a *Red App* bundle that you submit must have a different version number.

32. After your *Red App* is published, it is likely that you will update it. You can develop and submit a new version as either a patch or an upgrade release. A patch is a bug fix and an upgrade has new or improved functionality. An upgrade is considered to be a new product, in other words, a new *Red App*.

    Review *Red App* version standards and release types again in the *Red App* help to ensure that your updated version number complies with *Sabre's* requirements.

    For an upgrade to a *Red App*, you must submit a new proposal and provide a new name for your *Red App*. If *Sabre* approves your upgrade proposal, *Sabre* assigns a new *Red App* ID.

    For a patch, correct your *Red App* and submit it again with a new version. When you submit a patch, you can change the description of your *Red App*, the launch type, and you can also provide new screen captures of your app.

33. *Sabre* repeats the validation and certification process. If *Sabre* certifies your upgrade or patch release, the following occurs:

    - For a patch release, you must test your *Red App* in CERT and PROD again. Based on certification by *Sabre*, *Sabre* publishes the patch to the *Market Centre* and all existing users of the patch release receive the patch release automatically. The Agency Administrator in the purchasing agency uses the *Sabre Red Workspace* Agency Administrator Tools to enable your *Red App* for entitled users.
    - You do not confirm a sale for a patch release. All existing users receive the patch release automatically.
    - For an upgrade release, you repeat the process of testing in both CERT and PROD, following the same process as for a new *Red App*. *Sabre* publishes your *Red App* in the *Market Centre*. After you confirm a sale in the *Dev Centre*, you specify which end-users

are to receive the new upgrade on the *Dev Centre*, and then *Sabre* entitles your upgraded *Red App* it to the designated customers. End-users of an upgraded *Red App* may be new or existing users. The Agency Administrator in the purchasing agency uses the *Sabre Red Workspace* Agency Administrator Tools to enable your *Red App* for entitled users.

- The purchase process must be completed before *Sabre* distributes an upgrade to end-users.

Note: If support contact information for your *Red App* changes, you are required to update the contact details in your plug-in. Even if this is the only change you make to a published *Red App*, you are required to submit a new release with this updated information. It is most likely that this will be a patch release.

Sabre, Inc.
3150 Sabre Drive
Southlake, TX 76092-2199
(682) 605-1000