CITRIX®

Citrix Receiver for Android 3.8

OEM Reference Guide

Contents

Disclaimer

This document is furnished "AS IS". Citrix Systems, Inc. disclaims all warranties regarding the contents of this document, including, but not limited to, implied warranties of merchantability and fitness for any particular purpose. This document may contain technical or other inaccuracies or typographical errors. Citrix Systems, Inc. reserves the right to revise the information in this document at any time without notice. This document and the software described in this document constitute confidential information of Citrix Systems, Inc. and its licensors, and are furnished under a license from Citrix Systems, Inc.

About Citrix

Citrix (NASDAQ:CTXS) is leading the transition to software-defining the workplace, uniting virtualization, mobility management, networking and SaaS solutions to enable new ways for businesses and people to work better. Citrix solutions power business mobility through secure, mobile workspaces that provide people with instant access to apps, desktops, data and communications on any device, over any network and cloud. With annual revenue in 2014 of $3.14 billion, Citrix solutions are in use at more than 330,000 organizations and by over 100 million users globally. Learn more at www.citrix.com.

About this document

The purpose of this document is to support Original Equipment Manufacturers (OEMs) who integrate Citrix® Citrix Receiver™ for Android® into their own or customers' deployments. The document helps you:

1. Modify or replace the Citrix Receiver for Android installation
2. Customize the Citrix Receiver for Android settings

This document is intended for developers of products that include Citrix Receiver for Android. If you are planning to modify the user interface of Citrix Receiver for Android, Citrix recommends that you read the entire manual.

Resources to aid customization

OEMs can make use of the following:

1. Citrix Receiver for Android (.apk file), which is available for download from the Citrix website, http://www.citrix.com/ or this is available in the Google Play Store.
2. The OEMs can create a *Citrix receiverconfig.txt* file that can be used to customize some behaviors of the Citrix Receiver for Android, such file **must** reside under the /sdcard/ folder

About Citrix Receiver for Android 3.8

Citrix Receiver enables your users to access Windows applications published on XenApp or XenDesktop from their Android devices. You publish applications on XenApp or XenDesktop to make them available to your users.

Citrix Receiver for Android supports the Mobile SDK for Windows Apps and Citrix deployments, such as XenDesktop 7 and XenApp 6.5 Feature Pack 2, which include built-in mobility features. Mobility features improve the experience of Citrix Receiver users working in supported Windows applications and published server desktops on mobile devices.

Citrix Receiver for Android 3.8 consumes the latest HDXSDK for launching the session.

Related components

Citrix Receiver deployments involve other Citrix components. These typically include XenDesktop, XenApp, StoreFront (which replaces Web Interface as the mechanism for publishing applications), and Secure Gateway or NetScaler® Gateway. Configuring and customizing these related components is not covered in this document.

Configuration

Configuring Session resolution

Basic resolution settings can be modified using the Citrix Receiver display settings. These allow coarse-grained control over the size of the session and are generally sufficient for the user.

If these settings are not sufficient for the OEMs use case, then a specific resolution can be set by using the receiverconfig.txt file.

Warning: This causes forces the session to the particular resolution and any user-defined display settings have no effect.

Following are the steps, example of forcing the session resolution to 1280x720:

Example for editing in the receiverconfig

1. Create a receiverconfig.txt file
2. Add the line "SessionResolution=1280x720" (without quotations)
3. Save the file. (The file must reside in /sdcard/ location of the device).
4. Type this command in the command prompt
    a. "adb push receiverconfig.txt /sdcard/receiverconfig.txt"(if using adb)
5. Restart the Citrix Receiver.

Note: The file name and path are case sensitive.

Configuration of Hangeul/Eng Key

Enabling the Hangeul support using the receiverconfig file setting. By setting the value of the

"HangeulScanCode" in the receiverconfig file creates a synthetic RIGHT-Alt key event,whenever the hanguel Key is pressed and configured in the receiverconfig.

- Add the line " HangeulKeyScanCode=122" (without quotations) instead of step 2 in example for editing in the receiverconfig file.

Note : If this HanguelKeyScanCode is not zero, any key with that scan-code will be sent to the server as right-Alt, whichinvokes the Hangeul function of the Korean IME.

Configuring the InsessionMenuBar

Change for the LGCNS - The config file settings helps the OEM to disable to the Android Citrix Receiver InSession menu bar

- Add the line " HideInSessionMenu=true" (without quotations) instead of step 2 in example for editing in the receiverconfig.

Configuring the AndroidNotificationBar[1]

Change for the LGCNS along with the InsessionMenu Bar- The config file settings helps the OEM to disable to the Android Citrix Receiver Notification Bar. Also a full screen session experience is possible with this setting[2].

- Add the line " HideAndroidNotifyBar=true" (without quotations) instead of step 2 in example for editing in the receiverconfig.

Note: When this option is switched on there will not be any system soft keyboard, only extended keyboard will be enabled.

Launching ICA Files with the Citrix Receiver for Android

The Citrix Receiver for Android allows users to access their desktops and applications from any suitable Android device. The default launching method is to use the Citrix Receiver for Android app itself, but other methods can be used. Ultimately, all launching methods involve passing a single ICA File to the Citrix HDX Engine, which is currently embedded in the Citrix Receiver for Android. This document describes how to do so.

Android Intents

The standard Android method for starting an Activity – such as the Activity used to compose an email, view a web page, or showing a remote Citrix session – is the Intent. An app can create an Intent and then start an Activity using that Intent.

The Intent is passed to the Android system, which then routes it to the appropriate app for handling. If there are multiple apps that can handle the Intent, the Android system will ask the user to choose which app to use, and provide the user with the ability to make that app the default.

Apps indicate to the Android system that they can handle Intents by various means – the scheme (http, ftp, *etc.*) of the URL, the MIME type of the data, the path of the URL, *etc.* The MIME type for ICA files is *application/x-ica*, and the Citrix HDX Engine handles files of this type.

Sending a ICA File to the Citrix HDX Engine

The procedure for sending an ICA File to the Citrix HDX Engine is very similar to the process for launching any other content type.

---

[1] currently not useful with devices which won't display Android Notification Bar

[2] The extended keyboard won't be displayed on top of the soft keyboard when android notification bar is off.

- Obtain the ICA File
- Store the ICA File
- Create an Intent
- Start an Activity using the Intent

Each of these steps will be described below. This procedure assumes that the ICA File will be stored in the local file system before being sent to the HDX Engine.

It is possible to use an in-memory representation of the ICA File for launching purposes, either through a Citrix proprietary class or through the Android Content Provider system. This document currently details the use of the explicit file stored in the local file system.

Obtain the ICA File

The exact procedure by which the ICA File is obtained is not detailed in this document. The external app must be aware of the following constraints on the lack of reusability of ICA

Files:

• Brokered connections result in single-use ICA Files.

• Brokered connections have a short period during which the ICA File – specifically the single-use credentials therein – are valid. Using an ICA File from a brokered connection after the validity period will not result in a successful connection.

• Non-brokered connections are not possible to XenDesktop VDAs.

• Non-brokered connections are not possible to XenApp VDAs from XenApp 7.5 or later.

Only a single ICA File may be launched at a time, and only a single session may be active at any time, with the current release of the Citrix Receiver for Android.

Store the ICA File

Once obtained, the ICA File should be stored in the local file system. The location is not inherently important, however the location must be readable by an application other than the external app, otherwise the HDX Engine will not be able to read the ICA File.

The ICA File should be stored with world-readable permissions, and must be stored exactly as is. Encoding transformations or editing of the ICA File are not supported.

Create an Intent

The external app should create an Android Intent that specifies the URL of the stored ICA File and the MIME Type “application/x-ica”. The action of the Intent should be ACTION_VIEW.

The Intent should not have any categories specified or, if it does, they should only be ACTION_DEFAULT or ACTION_BROWSABLE. Any other categories will result in the Intent filter not matching.

Start an Activity using the Intent

The Intent should be passed to the startActivity() method of an android.content.Context object, such as an Activity in the external app.

The HDX Engine does not set a specific result code; therefore, using startActivityForResult() in the Activity object will always result in a RESULT_CANCELED being passed to the onActivityResult() handler. As such, Citrix does not recommend the use of startActivityForResult() for launching ICA Files.

The HDX Engine provides no other result information; once an ICA File has been launched the HDX Engine controls the entire lifecycle of the connection. In particular, the external app will not necessarily be able to determine when the connection has ended.

Code Example

The following code snippets demonstrate the use of an Intent to launch an ICA File. They rely on two suggested additional methods that are not implemented by the Citrix Receiver for Android – getIcaFileFromStore() and downloadIcaFileToStorage(). These methods, respectively, return a URL from which to download an ICA File for a specific application, and then download the ICA File to the local storage, writing to a provided File object. These methods must be provided by the external app (either directly or through some additional libraries).

It is entirely possible to perform these steps in other ways; the HDX Engine merely requires that an ICA File has been obtained in some manner.

The launching approach shows how to provide the URI of a downloaded ICA File to an Intent, and then

```
/** Create an Intent for launching an ICA File */
public Intent createIcaIntent(URI data) {
     Intent icaIntent;
     icaIntent = new Intent();
     icaIntent.setDataAndType(data, "application/x-ica");
     icaIntent.setAction(Intent.ACTION_VIEW);
     icaIntent.addCategory(Intent.CATEGORY_DEFAULT);
     return icaIntent;
}
```

how to use that Intent to start the HDX Engine and hence launch a connection. This snippet is a simple method for creating the Intent from a URI of a stored ICA File.

This snippet is a method demonstrating the ideas necessary to obtain a download URL for an ICA File, then downloading the ICA File to a specified storage location, before finally returning a URI to the downloaded ICA File.

```
/** Download an ICA File from a store given the URL, application name and a
* Credentials object.
* The implementation of the getIcaFileFromStore() method and the details of
* the Credentials object are left unspecified. */

public URI downloadIcaFile(URL store, String appName, Credentials credentials) {
      /* Get ICA File URL.
      * Implementation is not provided by the Receiver for Android. */
      URL icaFileUrl = getIcaFileFromStore(store, appName, credentials);

      /* Get storage directory for downloaded ICA File */
      File downloadFolder = Environment.getExternalStoragePublicDirectory(
      Environment.DIRECTORY_DOWNLOADS);

      /* Create temporary file. */
      File tempIcaFile = File.createTempFile("ctxLaunch", ".ica", downloadFolder);

      /* Download ICA File, writing to file system.
      * Implementation is not provided by the Receiver for Android. */
      boolean success = downloadIcaFileToStorage(icaFileUrl, tempIcaFile);

      /* Prior to this point, the example code has obtained and stored a valid
      * ICA File. Other approaches are possible.
      * Once a valid ICA File has been stored, it is necessary to find the
      * correct URI to use in the Intent. */

      /* At this point, if success is true, an ICA File has been downloaded.
      * Now, find the URI for the downloaded ICA File to use in an Intent. */
      if (success) {
            /* Obtain URI of downloaded ICA File */
            return tempIcaFile.toURI();
      }

      return null;
}
```

This snippet shows how to use the preceding two methods to obtain and launch an ICA File.

```
/* Obtain ICA File */
Uri data = downloadIcaFile();

/* Create an Intent to launch the ICA File */
Intent launchIntent = createIcaIntent(data);

/* Launch the ICA File */
activity.startActivity(launchIntent);
```

Thinwire Modes and Rendering

Two new decoding/rendering modes were added to the Android Citrix Receiver in version 3.6. Both of these new Thinwire (TW) modes use OpenGL to render the session image to the screen. The difference between these two new modes is how the H264 frames are decoded to uncompressed pixels.

These OpenGL modes are only used when the session is in an H264 Thinwire mode. This includes H264-only TW mode and H264+Lossless Overlays TW mode. If the session is not in an H264 TW mode – then it falls back to Conventional (Legacy) TW mode or Conventional (Compatibility) TW mode – OpenGL rendering is not used. Furthermore even if the session is in an H264 TW mode, the Citrix Receiver may not use OpenGL rendering if e.g. the session resolution is larger than the OpenGL maximum texture size. When OpenGL rendering is not used, ImageView/Bitmap rendering is used.

The decision between the two OpenGL rendering modes (OpenGL CoreAVC[3] mode and OpenGL

MediaCodec[4] mode) is made when the TW session is initialized. MediaCodec decoding is preferred, but on many devices there is no useable MediaCodec H264 decoder and so CoreAVC decoding must be used. The reasons for not using MediaCodec decoding include no MediaCodec support (older Android OS versions), no decoder that can handle the session resolution at the required H264 profile, and most commonly MediaCodec implementations that buffer frames and so can't be used for an interactive session.

---

[3] CoreAVC is a proprietary codec for decoding the H.264/MPEG-4 AVC (Advanced Video Coding) video format.

[4] MediaCodec - MediaCodec is an Android API that can be used to decode H.264 video streams using hardware decoders (if available in the device).

Using Citrix receiverconfig.txt file you can tune the modes, but the best performance depends upon the available decoder in the device. Following are the types and the default values if no Citrix receiverconfig.txt file exists on the device.

ThinwireMode

Type: list of choices (none, conventional, h264, h264plusslosslessoverlays).

Default: h264plusslosslessoverlays, h264, conventional

Determines what TW mode is preferred by the Citrix Receiver, and what TW modes are allowed by the Citrix Receiver. The first entry in the list is the preferred TW mode which the Citrix Receiver sends to the server. The entire list is the set of allowed TW modes. So if the client prefers H264 but the server does not support it, "conventional" must also be in the list because the server will insist on a conventional TW session. Note that just because a TW mode is in this list does not mean the Citrix Receiver supports it. E.g. if the width of the session is greater than 2032 pixels, H264 and H264PlusLosslessOverlays are never supported on some devices.

H264ThinwireDecoders

Type: list of choices (CoreAvc, MediaCodecToGlSurface).

Default: MediaCodecToGlSurface,CoreAVC

Determines what H264 decoder is preferred by the Citrix Receiver, and what H264 decoders are allowed by the Citrix Receiver. This setting only applies to H264/H264PlusLosslessOverlays TW modes. The first entry in the list is the preferred H264 decoder. The entire list is the set of allowed H264 decoders. So if the client prefers MediaCodecToGlSurface but the device does not support it (e.g. the MediaCodec H264 decoder implementation on the device buffers frames), CoreAVC must also be in the list or else H264 Thinwire cannot be used and connection may fail.

RenderWithOpenGL

Type: choice (Always, Never, Try),

Default: Try

Controls whether OpenGL is used to render the session image rather than an ImageView widget. If set to "Try", OpenGL will be used depending on other settings and device capabilities. This setting only applies to H264/H264PlusLosslessOverlays TW modes.

Configuration file snippet

An example of Citrix receiverconfig.txt file content for changing the OpenGL rendering mode and selecting the decoder is shown below. Save this text into a file as “Citrix receiverconfig.txt” and save it under “/sdcard/” location in the device.

```
RenderWithOpenGL=Always

H264ThinwireDecoders=MediaCodecToGlSurface
```

Transport Layer Security (TLS)

Since version 3.7 Citrix Receiver for Android supports the TLS 1.1 and TLS 1.2 protocols, those are configurable either using the user interface or through the receiverconfig, the default protocol used by Citrix Receiver for Android is TLS 1.0.

Configuration file use:

The receiverconfigfield to use for TLS configuration is SslSdkProtocolNumber and only the following values are accepted:

Value	Protocol
2	TLS 1.0
4	TLS 1.1
8	TLS 1.2
129	Force SSLv3

For instance the following line should be in place for TLS1.2:

SslSdkProtocolNumber=8

An incorrect value will result with the default protocol TLS 1.0 being used, the application should be restarted each time the value is changed in the file.

Configure SSLv3

The SSL version 3 protocol is considered unsafe and its use should be avoided for security reasons, however it is still possible to force its usage through the Citrix receiverconfig.

Login into the first store

User Interface does not allow users to modify the used protocol unless a store is added, this could result in an issue if the first store added does not use the default protocol – TLS 1.0 - since it is impossible to gain access to the needed UI to change the protocol settings from the default value.

In this case the receiverconfig comes in handy and it represents the only way to add the first store, an alternative workaround is add a TLS 1.0 compliant store as first store and then, gained access to the UI, select the desired TLS protocol version which will be used to add the following store.