Frame App

Frame App Getting Started, Configuration, IGEL, IGEL Profiles

Introduction to Frame App

Dizzion DaaS and Cloud PC are built on a **browser-first foundation**. From day one, accessing virtual desktops and applications through a standard web browser has been core to the Frame experience—not a fallback option. While many other virtual desktop solutions treat the browser for accessing Virtual Desktops and Apps as a “Plan B,” Frame delivers a high-performance experience directly in the browser via our Frame Remoting Protocol (FRP). No plugins. No Workspace App, No Receiver, No clients required.

With just a modern browser, users can take advantage of rich capabilities such as multi-monitor support, copy/paste, local printing, audio and video playback, webcam and microphone redirection, file upload/download, and WIA scanner redirection. This makes it easy to work from any device, anywhere, without installation or configuration.

Why use the Frame App?

Most users will find everything they need in a browser. However, certain advanced use cases require capabilities that a browser cannot fully support today. This is where the **Frame App** comes in.

The Frame App provides all the benefits of browser access, plus additional features—most notably **generic USB redirection** for specific peripherals (e.g., serial devices, USB authentication keys, unique scanners). For organizations with specialized hardware needs or standardized thin-client deployments, the Frame App delivers an optimal fit.

Platform Support

The Frame App is available for:

- **Microsoft Windows**
- **Apple MacOS (Intel and ARM based)**
- **Linux** — widely used on thin clients such as IGEL, Stratodesk, 10ZIG, ZeeTim, HP, Dell, and more.

In summary: choose the browser for maximum flexibility and instant access from any device. Choose the Frame App when your environment requires enhanced USB or endpoint hardware capabilities. Either way, you get the same seamless, secure virtual desktop and app experience powered by Dizzion.

Getting Started

Deploying Frame App for your end users can be done in a few steps. The following guide will outline how to download, install, update Frame App.

Installer Downloads

Frame App installation and setup are uncomplicated and should only take a few minutes.

Frame App Installer Download by Administrator

Frame administrators can download the Frame App Installer for Linux, macOS, and Windows in two ways:

1. They can log in to the Frame Console and go to their customer entity, click on the kebab icon, and go to Update > Settings. On the Settings page, they can download the Frame App installers for their desired operating system(s) and share the installer(s) with their end users.
2. Or, they can download the Frame App installers for their desired operating system(s) from our Downloads page, and share the installer(s) with their end user

Frame Console - Frame App Installer Downloads

Frame Console - Frame App Installer Downloads

Frame App Installer Download by User

Frame Administrators can enable the ability for their end users to download the Frame App installer from the Launchpad. This can be done by enabling the "Enable Frame App download from Launchpad" toggle on their Frame Customer entity settings.Frame Console - Frame App Installer Downloads

Frame Console - Frame App Installer Downloads

1. Once this toggle is enabled, end users can navigate to their Launchpad in a browser to their profile name.

image.png

2. The end users would click on their profile name and select “Download Frame App for ---” from the menu. The browser will automatically detect the endpoint device operating system and show the appropriate Frame App installer.

image.png

3. If the user wishes to download the Frame App installer for a different operating system, they can click on the down arrow next to the “Download Frame App for _” and one of the other Frame App installers can be selected.

Installation

Windows Installer

1. Once the Frame App Installer for Windows is downloaded, run the installer.\n\n

Frame App Installer for Windows - Welcome

Frame App Installer for Windows - Welcome

Follow the prompts provided by the setup wizard, you will be asked to define the directory where you want the application installed.\n3. Click “Finish” after running through the installation wizard.

Frame App Installer for Windows - Complete

Frame App Installer for Windows - Complete

Silent Installation for Windows

Customers wishing to deploy Frame App through a silent installation process for Windows can do so by invoking `msiexec` as a Windows administrator with the Frame App MSI package, starting with Frame App 5.11. A basic silent install of Frame App 6.11 would have a command line: `msiexec /i Frame-app.msi /qn\n`

To generate an installation log during the silent install, a sample command line would be: `msiexec /i Frame-app.msi /L*V "FrameApp.log" /qn\n`

Note that the following msiexec.exe commands can be used:

Argument	Details
`/i`	Normal install
`/norestart`	Do not restart the device after the installation completes.
`/qn`	No UI during the installation process
`/quiet`	No user interaction required

MacOS Installer

1. Once the Frame installer dmg file is downloaded, double click on the dmg file.
2. Drag the file into the applications folder.
3. Before starting Frame App, navigate to your System Preferences > Security & Privacy. Scroll down to Input Monitoring, unlock the dialog to make changes, and check the box next to Frame. Lock in your preferences. This is critical if USB devices are to be used in a Frame session.
4. Scroll down to Input Monitoring, unlock the dialog to make changes, and check the box next to Frame. Lock in your preferences. This is critical if USB devices are to be used in a Frame session
5. Launch Frame App when you're ready to begin using it.

image.png

Linux Installer

> Frame App for Linux is only supported on Ubuntu and Ubuntu-based thin client OS that are Frame Ready validated (e.g. IGEL OS, Stratodesk NoTouch OS, Zeetim ZeeOS, and 10ZiG PeakOS)

1. On your Linux endpoint, open a terminal window and navigate to the location where you have downloaded Frame App. Take note of the version number of Frame App.
2. Run sudo apt install plus the file name/location, as shown below.
3. Once installed, you should be able to access Frame App from your App Launcher.

Installing Frame App for Linux

Updates

Once installed, Frame App will automatically check for updates every time you launch it. You will be prompted to update your version of Frame App if any newer versions are available. To check your version of Frame App, simply click on the **Frame** menu and select **About**.

About Frame

Configuring Frame App

Frame App behavior can be configured on startup of the application (Command Line Arguments) and through configuration files/Windows registry keys. If you are installing Frame App on Windows endpoints and would like to configure the silent installation process, you can find additional documentation here.

Command Line Arguments

Command line arguments or “launch arguments” are parameters that are automatically passed to a program (via command line) to modify its behavior. Some organizations may choose to use these modifiers to better fit their use case. We will start by providing a list of the available command line arguments and then explain how to configure them based on your operating system.

To launch Frame App with certain command line arguments, start by opening command prompt/terminal on your local machine. Specify the file path for Frame App (include the executable). Append your command line argument(s) after file path.

You may launch Frame App with the specified parameters. Use the table below to understand command line argument options and their syntax.

Frame App 7

Launch Frame App with arguments following the pattern:

```
# Starting in "/usr/bin/"

# Single argument
./frame --arg=value

# OR

# Multiple Arguments
./frame --arg1 --arg2=value
```

Command Line Argument	Description and Syntax Example

`--startup-url`	Designates the startup URL, as you would on the "Preferences" page of Frame App. Single Argument Example: `./frame --startup-url="https://use.difr.com/"`
`--check-for-updates-on-startup`	Frame App will check for updates on startup when this argument is set to **ON**. Single Argument Example: `./frame --check-for-updates-on-startup=on`
`--clear-cache-on-startup`	Frame App will automatically clear the local cache on startup when argument is set to **ON**. Single Argument Example: `./frame --clear-cache-on-startup=on`
`--enable-camera`	Frame App will launch with camera functionality enabled when this argument is set to **ON**. Single Argument Example: `./frame --enable-camera=on`

`--enable-microphone`	Frame App will launch with microphone functionality enabled when this argument is set to **ON**. Single Argument Example: `./frame --enable-microphone=on`
`--full-screen`	Instructs Frame App to launch in full screen when this argument is set to **ON**. Single Argument Example: `./frame --full-screen=on`
`--start-session-in-full-screen`	Instructs Frame App to start a session in full screen when this argument is set to **ON**. Single Argument Example: `./frame --start-session-in-full-screen=on`
`--help`	Provides a help menu. This command must be called on the precise application executable. Single Argument Example: `/usr/bin/frame --help`
`--version`	Shows Frame App version. Single Argument Example: `./frame --version`

`--hide-menu-bar=on`	Hide Menu Bar. Useful in kiosk mode scenarios. Single Argument Example: `./frame --hide-menu-bar=on`
`--hide-toggle-fullscreen=on`	Hide toggle-fullscreen option from the View menu. Also used for kiosk mode scenarios. Single Argument Example: `./frame --hide-toggle-fullscreen=on`
`--use-experimental-gpu-flags=on`	Enable Hardware Acceleration. Mainly used on IGEL client devices. Single Argument Example: `./frame --use-experimental-gpu-flags=on`

Command Line Arguments	**Description and Syntax Example**
`displays-auto-arrange`	Frame App will launch with virtual displays configured to match your local environment. *Example:* `./Frame --displays-auto-arrange`
`kiosk`	Instructs Frame App to launch in full screen, also known as “Kiosk mode.” *Example:* `./Frame --kiosk`
`url`	Designates the startup URL, as you would on the “Preferences” page of Frame App. *Example:* `./Frame --url=console.nutanix.com`

Command Line Arguments	Description and Syntax Example
`x11-window`	Switch from GTK (default) to X11 Windows. This argument should be used with HP ThinPro OS clients. *Example:* `./Frame --x11-window`

Frame App 7

Launch Frame App with arguments following the pattern:

```
# Starting in "/Applications/Frame.app/Contents/MacOS/"

# Single argument
./Frame --arg=value

# OR

# Multiple Arguments
./Frame --arg1 --arg2=value
```

Command Line Argument	Description and Syntax Example
`--startup-url`	Designates the startup URL, as you would on the "Preferences" page of Frame App. Single Argument Example: `./Frame --startup-url="https://use.difr.com/"`
`--check-for-updates-on-startup`	Frame App will check for updates on startup when this argument is set to **ON**. Single Argument Example: `./Frame --check-for-updates-on-startup=on`

Command Line Argument	Description and Syntax Example
`--clear-cache-on-startup`	Frame App will automatically clear the local cache on startup when argument is set to **ON**. Single Argument Example: `./Frame --clear-cache-on-startup=on`
`--enable-camera`	Frame App will launch with camera functionality enabled when this argument is set to **ON**. Single Argument Example: `./Frame --enable-camera=on`
`--enable-microphone`	Frame App will launch with microphone functionality enabled when this argument is set to **ON**. Single Argument Example: `./Frame --enable-microphone=on`
`--full-screen`	Instructs Frame App to launch in full screen when this argument is set to **ON**. Single Argument Example: `./Frame --full-screen=on`

Command Line Argument	Description and Syntax Example
`--start-session-in-full-screen`	Instructs Frame App to start a session in full screen when this argument is set to **ON**. Single Argument Example: `./Frame --start-session-in-full-screen=on`
`--help`	Provides a help menu. This command must be called on the precise application executable. Single Argument Example: `/Applications/Frame.app/Contents/MacOS/Frame --help`
`--version`	Shows Frame App version. Single Argument Example: `./Frame --version`
`--hide-menu-bar=on`	Hide Menu Bar. Useful in kiosk mode scenarios. Single Argument Example: `./Frame --hide-menu-bar=on`

Command Line Argument	Description and Syntax Example
`--hide-toggle-fullscreen=on`	Hide toggle-fullscreen option from the View menu. Also used for kiosk mode scenarios. Single Argument Example: `./Frame --hide-toggle-fullscreen=on`
`--use-experimental-gpu-flags=on`	Enable Hardware Acceleration. Mainly used on IGEL client devices. Single Argument Example: `./Frame --use-experimental-gpu-flags=on`

Starting Frame App in the macOS Terminal

Frame App 7

Launch Frame App with arguments following the pattern:

```
# Starting in "C:\Program Files\Frame>"

# Single argument (Requires -- before flag)
& ./Frame.exe -- --arg=value

# OR

# Multiple Arguments
& ./Frame.exe --arg1 --arg2=value
```

Command Line Argument	Description and Syntax Example

`--startup-url`	Designates the startup URL, as you would on the “Preferences” page of Frame App. Single Argument Example: `& ./Frame.exe -- --startup-url="https://console.nutanix.com/"`
`--check-for-updates-on-startup`	Frame App will check for updates on startup when this argument is set to **ON**. Single Argument Example: `& ./Frame.exe -- --check-for-updates-on-startup=on`
`--clear-cache-on-startup`	Frame App will automatically clear the local cache on startup when argument is set to **ON**. Single Argument Example: `& ./Frame.exe -- --clear-cache-on-startup=on`
`--enable-camera`	Frame App will launch with camera functionality enabled when this argument is set to **ON**. Single Argument Example: `& ./Frame.exe -- --enable-camera=on`

`--enable-microphone`	Frame App will launch with microphone functionality enabled when this argument is set to **ON**. Single Argument Example: `& ./Frame.exe -- --enable-microphone=on`
`--full-screen`	Instructs Frame App to launch in full screen when this argument is set to **ON**. Single Argument Example: `& ./Frame.exe -- --full-screen=on`
`--start-session-in-full-screen`	Instructs Frame App to start a session in full screen when this argument is set to **ON**. Single Argument Example: `& ./Frame.exe -- --start-session-in-full-screen=on`
`--help`	Provides a help menu. This command must be called on the precise application executable. Single Argument Example: `C:\Program Files\Frame\app-7.5.0\Frame.exe -- --help`

`--version`	Shows Frame App version. Requires `| Write-Output` modifier. Single Argument Example: `& ./Frame.exe -- --version | Write-Output`
`--hide-menu-bar=on`	Hide Menu Bar. Useful in kiosk mode scenarios. Single Argument Example: `& ./Frame.exe -- --hide-menu-bar=on`
`--hide-toggle-fullscreen=on`	Hide toggle-fullscreen option from the View menu. Also used for kiosk mode scenarios. Single Argument Example: `& ./Frame.exe -- --hide-toggle-fullscreen=on`
`--use-experimental-gpu-flags=on`	Enable Hardware Acceleration. Mainly used on IGEL client devices. Single Argument Example: `& ./Frame.exe -- --use-experimental-gpu-flags=on`

Starting Frame App on the Windows command line

User Preference Policies

Frame Administrators can enforce user preference policies across all installations of Frame App for Linux, Frame App for macOS, and/or Frame App for Windows through a preferences.conf file (Linux), plist (macOS), and group policy objects (Windows), respectively.

Frame App 7

For Frame App for Linux 7 and greater, Frame administrators can control user preferences by creating and saving a configuration file at the location `/etc/frame/preferences.conf`.

The conf file contains name-value pairs in the format `KEY` = `VALUE`. Spaces will be ignored. The list of supported keys and associated values are described in the following table.

```
STARTUP_URL=https://use.difr.com/
CLEAR_CACHE_ON_STARTUP=ON
```

Key Name	Description	Value(s)
`STARTUP_URL`	Designates the startup Uniform Resource Location (URL). Default Value: `https://console.nutanix.com/`	URL
`CHECK_FOR_UPDATES_ON_STARTUP`	Frame App will check for updates on startup when this argument is set to **ON**. If there is an update available, Frame App will ask the user if they wish to download and install the update. Default Value: `ON`	`ON` or `OFF`
`CLEAR_CACHE_ON_STARTUP`	Frame App will automatically clear local cache on startup when this argument is set to **ON**. Default Value: `ON`	`ON` or `OFF`
`ENABLE_CAMERA`	Frame App will launch with camera functionality enabled when this argument is set to **ON**. Default Value: `ON`	`ON` or `OFF`

Key Name	Description	Value(s)
`ENABLE_MICROPHONE`	Frame App will launch with microphone functionality enabled when this argument is set to **ON**. Default Value: `ON`	`ON` or `OFF`
`FULL_SCREEN`	Instructs Frame App to launch in full screen when this argument is set to **ON**. Default Value: `OFF`	`ON` or `OFF`
`START_SESSION_IN_FULL_SCREEN`	Instructs Frame App to start a session in full screen when this argument is set to **ON**. Default Value: `OFF`	`ON` or `OFF`

Frame App 7

For Frame App for macOS 7 and greater, Frame administrators can control user preferences by creating and saving a plist file at the location `/Users/Shared/frame/preferences.plist`.

The plist file contains name-value pairs in the format:

```
<key>VALID_KEY</key>
<string>VALID_VALUE</string>
```

An example plist file would be:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>STARTUP_URL</key>
    <string>https://console.nutanix.com/</string>
    <key>CLEAR_CACHE_ON_STARTUP</key>
    <string>ON</string>
  </dict>
</plist>
```

Key Name	Description	Value(s)
`STARTUP_URL`	Designates the startup Uniform Resource Location (URL). Default Value: `https://console.nutanix.com/`	URL
`CHECK_FOR_UPDATES_ON_STARTUP`	Frame App will check for updates on startup when this argument is set to **ON**. If there is an update available, Frame App will ask the user if they wish to download and install the update. Default Value: `ON`	`ON` or `OFF`
`CLEAR_CACHE_ON_STARTUP`	Frame App will automatically clear local cache on startup when this argument is set to **ON**. Default Value: `ON`	`ON` or `OFF`
`ENABLE_CAMERA`	Frame App will launch with camera functionality enabled when this argument is set to **ON**. Default Value: `ON`	`ON` or `OFF`
`ENABLE_MICROPHONE`	Frame App will launch with microphone functionality enabled when this argument is set to **ON**. Default Value: `ON`	`ON` or `OFF`
`FULL_SCREEN`	Instructs Frame App to launch in full screen when this argument is set to **ON**. Default Value: `OFF`	`ON` or `OFF`
`START_SESSION_IN_FULL_SCREEN`	Instructs Frame App to start a session in full screen when this argument is set to **ON**. Default Value: `OFF`	`ON` or `OFF`

Frame App 7

Frame App for Windows (version 7 or greater) enables Windows administrators to set user preferences via group policy objects (GPOs). This allows Windows administrators to use Windows GPOs to change those preferences for all users or specific users.

Administrators can update Frame App user preferences either in `HKEY_LOCAL_MACHINE` or `HKEY_CURRENT_USER`. If user preference registry keys are set in `HKEY_LOCAL_MACHINE`, those user preferences will take precedence as they will apply to all users. The relative path to the registry keys is `\SOFTWARE\Frame\Preferences`.

User Preferences in Windows Registry

User Preferences in Windows Registry

```
STARTUP_URL=https://console.nutanix.com/
CLEAR_CACHE_ON_STARTUP=ON
```

Frame App 7

Key Name	Description	Value(s)
`STARTUP_URL`	Designates the startup Uniform Resource Location (URL). Default Value: `https://console.nutanix.com/`	URL
`CHECK_FOR_UPDATES_ON_STARTUP`	Frame App will check for updates on startup when this argument is set to **ON**. If there is an update available, Frame App will ask the user if they wish to download and install the update. Default Value: `ON`	`ON` or `OFF`
`CLEAR_CACHE_ON_STARTUP`	Frame App will automatically clear local cache on startup when this argument is set to **ON**. Default Value: `ON`	`ON` or `OFF`
`ENABLE_CAMERA`	Frame App will launch with camera functionality enabled when this argument is set to **ON**. Default Value: `ON`	`ON` or `OFF`

Key Name	Description	Value(s)
`ENABLE_MICROPHONE`	Frame App will launch with microphone functionality enabled when this argument is set to **ON**. Default Value: `ON`	`ON` or `OFF`
`FULL_SCREEN`	Instructs Frame App to launch in full screen when this argument is set to **ON**. Default Value: `OFF`	`ON` or `OFF`
`START_SESSION_IN_FULL_SCREEN`	Instructs Frame App to start a session in full screen when this argument is set to **ON**. Default Value: `OFF`	`ON` or `OFF`

Local Cache

Frame App functions similarly to web browsers. Frame App contains a local cache that saves cookies, session tokens, and specific user preferences (e.g., automatically use all local displays). The cache can be deleted manually when required.

Follow the steps below based on the operating system on which Frame App is installed. Replace `$USER` with the name of your user.

Frame App 7.X Linux

```
/home/$USER/.config/Frame/cache
```

Frame App 7.X MacOS

```
/Users/$USER/Library/Application Support/Frame/cache
```

Frame App 7.X Windows

```
C:\Users\$USER\AppData\Roaming\Frame\cache
```

> The removal of the local cache will not remove the Startup URL value as this is stored either in the Preferences database or in the registry (for Windows).

Log Location

Frame App writes its logs to the following locations. Replace `username` with the name of your user.

Frame App 7.X Linux

```
/home/$USER/.config/Frame/Logs
```

Frame App 7.X MacOS

```
/Users/$USER/Library/Application Support/Frame/Logs
```

Frame App 7.X Windows

```
C:\Users\$USER\AppData\Roaming\Frame\Logs
```

IGEL

Frame provides a convenient Custom Partition for IGEL OS bundled with UMS Profiles for easy and secure integration with IGEL OS and management with IGEL's UMS. The included UMS Profiles allow admins to quickly and easily deploy Frame App tailored for your users and use-case(s).

Version Requirements

- Frame App 7.4.x+ is compatible with IGEL OS 12
- IGEL UMS 11 or higher

Frame App Custom Partition

You can bundle Frame App into an IGEL Custom Partition for use with IGEL OS 12 following the instructions below. *Building the custom partition bundle currently requires Ubuntu 18.04 with IGEL OS 11*.

Frame App IGEL Bundling instructions For Ubuntu 18.04

1. Download the latest Frame App for Linux (Debian) to your `~/Downloads` directory.
2. Download and unzip `Frame.zip` from https://github.com/IGEL-Community/IGEL-Custom-Partitions/raw/master/CP_Packages/Apps/Frame.zip.
3. Using a terminal, navigate to the unzipped directory to `/target/build/` and execute `build-frame-cp.sh`
4. Copy `frame.ini` and `frame.tar.bz2` from `/target/` to a new Frame folder in the UMS "ums_filetransfer" path depending on your OS:
   - UMS upload path on Linux: `/opt/IGEL/RemoteManager/rmguiserver/webapps/ums_filetransfer/Frame/`
   - UMS upload path on Windows: `C:/Program Files/IGEL/RemoteManager/rmguiserver/webapps/ums_filetransfer/Frame/`
5. Import Frame's Custom Profile(s) from `/igel/*`
6. Edit the profile and set up *Firmware Customization -> Custom Partition -> Download* with your UMS server info and credentials.
7. Setup env variables as instructed in the guides below.

Building the bundle will provide you with a zip file relative to the version of Frame App that was bundled. This bundle also includes Frame-provided UMS Profiles that you can quickly import and begin using with your Frame Custom Partition.

> The UMS Profiles provided by Frame serve as starting points for your implementation. However, they don't limit your options for customization. Frame is highly extensible, allowing for extensive customization of various aspects including authentication, Role-Based Access Control (RBAC), and user interface (UI). Advanced customizations can be achieved through: - Orchestrating Secure Anonymous Tokens for fine-grained control over authentication flows and RBAC - Leveraging the Session API to manage how, when, and where customers interact with your Frame resources

These tools enable you to tailor the Frame experience to your specific needs, going beyond the default configurations provided in the UMS Profiles.

Frame-Provided Profiles

Frame-provided UMS Profiles offer various configuration options to customize and optimize your IGEL environment. These profiles range from basic Frame App desktop integration to specialized kiosk modes supporting SAML2 and Secure Anonymous Tokens (SAT) authentication.

Testing New Versions of Frame App

When a new version of Frame App comes out, admins should test the new version of Frame App on a small subset of devices before rolling it out to the rest of their users. In order to configure multiple versions of Frame App in your UMS, you need to follow a few steps below to add a custom installation path of a test Frame App Custom Partition.

> When building the custom partition for Frame App 7.x+, make sure to use the `build-frame-cp.sh` script.

1. Create a new folder in your UMS file transfer server, something like `Frame-Test`. This would result in a folder at the following path: `/IGEL/RemoteManager/rmguiserver/webapps/ums_filetransfer/Frame-Test/`
2. Once that's complete, import or create a copy of an existing profile and edit it. Navigate to **Firmware Customization > Custom Partition > Download** and edit the

download URL to reference the same path.
For our example: `https://[YOUR_UMS_SERVER]:8443/ums_filetransfer/Frame-Test/frame.inf`

3. That's it! Assign the profile to your devices and they should download the new partition accordingly.

> Multiple versions of Frame App are not currently available on the same IGEL device. Admins must assign only one Frame App Custom Partition to a device at a time.

Troubleshooting

As of now, there are no specific common issues reported for Frame App 7 on IGEL OS. However, we are continuously monitoring user feedback and will update this section with any relevant troubleshooting information as it becomes available.

If you encounter any issues, please contact our support team for assistance.

Frame-Provided IGEL Profiles

Profile Options

Frame-provided UMS Profiles offer various configuration options to customize and optimize your IGEL environment. These profiles range from basic Frame App desktop integration to specialized kiosk modes supporting SAML2 and Secure Anonymous Tokens (SAT) authentication. Click here to skip ahead to instructions on how to import Frame App into UMS 12.

Basic Frame App Profile

Bundle location: `igel/frame-app-basic-profile.xml`

This "basic" UMS Profile simply enables a Frame App icon on the IGEL Desktop.

Admins can customize the default settings and launch parameters by adding command line arguments in your UMS by editing the Frame App Basic Profile Settings: **Firmware Customization > Custom Application > Frame > Settings**.

> When configuring the profile, you need to specify the Frame App version in the command field, use `/custom/frame/frame- sat-kiosk-launcher.sh v7`

Please refer to our Linux command-line arguments for Frame App for more information.

Frame SAML2 Kiosk Mode Profile

Bundle location: `igel/frame-saml2-kiosk-profile.xml`

This profile is designed to support a specific end user workflow and assumes a particular Frame configuration.

Frame App Kiosk with SAML

SAML2 Kiosk Mode User Experience

The SAML2 Kiosk Mode provides a seamless and secure user experience, integrating Frame App with third-party identity providers. Here's what users can expect:

```
graph TD
    Start([Start]) --> A[Frame App Launch]
    A -->|Clear cache| B[Present Kiosk Mode Interface]
    B --> C[Show Identity Provider Login]
    C --> D{Authentication Successful?}
    D -->|Yes| E{Launch Link Configuration}
    E -->|Desktop| F[Full-screen Desktop Session]
    E -->|Application| G[Full-screen Application Session]
    F --> H{User Action}
    G --> H
    H -->|Disconnect or Inactivity timeout| I[Show Resume Option]
    I -->|Within timeout| F
    I -->|Timeout expires| C
    H -->|End Session| J[Close Session]
    J -->|Configure Frame to redirect to IdP| C
    C -->|New user| C

    classDef default fill:#f9f,stroke:#333,stroke-width:2px;
    classDef decision fill:#ccf,stroke:#333,stroke-width:2px;
    class D,E,H decision;
```

1. Upon launch, Frame App clears its cache to ensure a fresh session and secure authentication.
2. Users are presented with a full-screen Kiosk Mode interface, supporting multiple monitors. The login screen of the configured third-party identity provider appears, prompting for credentials.
3. After successful authentication, Frame App directs users to either a desktop or specific application, based on the Launch Link configuration.
4. The Frame session begins in full-screen mode, providing an immersive remote desktop experience.
5. If users disconnect, either manually or due to inactivity, they have the option to resume their session within the account's or Launchpad's configured idle timeout.
6. When users end their session (by quitting or shutting down the Windows instance), they are logged out and returned to the identity provider's login screen, ready for the next user.

This workflow ensures a secure, stateless experience for shared devices while maintaining ease of use for end-users.

SAML2 + Kiosk Mode Requirements

To set up the SAML2 Kiosk Mode, ensure you have the following:

- A *Published* Launchpad
- A configured identity provider with associated roles/permissions allowing access to the desired Frame Account
- A Frame Launch Link with the additional "Quit and log out" URL parameter: `&qlo=1`
- (Optional) Frame account production workload VMs joined to a Windows domain, if desired

Additionally, you need to configure the IGEL UMS Custom Profile:

1. Navigate to:
   > System > Firmware Customization > Environment Variables > Predefined
2. Add the following environment variable:
   - `FRAME_LAUNCH_URL`: This is your Launch Link, obtained from the Account's *Dashboard > Launchpad > Advanced Integrations*. You'll find a configurable dialog with Launch Links there.

> While we recommend using Launch Links for Kiosk scenarios, you can also use a standard Launchpad URL for the FRAME_LAUNCH_URL value if needed.

This configuration ensures that your SAML2 Kiosk Mode is properly set up and integrated with your Frame Account and IGEL environment.

SAML2 + Kiosk Mode Configuration

1. Import the SAML2 kiosk launcher profile template (with .ipm extension) into your UMS12.
2. After importing, update the template values with your specific configuration.
3. Follow the existing steps for setting up the environment variables.

Frame SAT Kiosk Mode Profile

Bundle location: `igel/frame-sat-kiosk-profile.xml`

The Frame SAT Kiosk Custom Profile is designed to support a specific end user workflow relying on Frame's Secure Anonymous Tokens (SAT) for authentication. This flow also assumes a particular Frame configuration to support the kiosk experience as defined below.

SAT Kiosk Mode User Experience

With the SAT Kiosk Mode user experience, end users will not authenticate to a SAML2-based identity provider (this script uses the Frame Secure Anonymous Token (SAT) functionality for session authentication).

```
graph TD
    1[1. Launch Frame App in kiosk mode]
    2[2. Clear user cache]
    3[3. Authenticate using SAT]
    4[4. Direct to desktop or application]
    5[5. Start remote desktop in full-screen]
    6[6. Session ends]
    7[7. Restart Frame App with new SAT token]

    1 --> 2
    2 --> 3
    3 --> 4
    4 --> 5
    5 --> 6
    6 --> 7
    7 -.-> 1
```

1. Frame App will launch in "kiosk mode" (full screen).
2. User cache is cleared prior to start and exit of Frame App to ensure no user preference settings have persisted since the prior use of Frame App.
3. End users are authenticated using Frame Secure Anonymous Token (SAT) functionality.
4. Frame App directs the end user directly to the desktop or application (depending on the Launch Link configuration).
5. When a Frame session starts, the remote desktop will be in full-screen mode.
6. Upon session disconnect or closure, Frame App will restarts with a new SAT token.

> **NOTE:**
>
> Disconnect behavior is configurable from Session Settings.

SAT + Kiosk Configuration Requirements

- A *Published* Launchpad.
- API Provider configured at the Organization entity.
- Secure Anonymous Token Provider at the Account entity level granting a role of Launchpad User for a specific Launchpad in a Frame account (under the Organization

entity).

- Frame Launch Link is used, rather than a Launchpad URL to support automatic start of the user's session and to simplify the UX.
- *Optional:* The Frame account production workload VMs can be joined to a Windows domain, if desired.
- The Environment Variables listed below:

Environment Variables

The following environment variables must be configured in the IGEL Custom Profile for this profile to work.

1. **Edit your IGEL UMS Custom Profile and go to**:

   > System > Firmware Customization > Environment Variables > Predefined

2. **Set the following environment variables:**

Environment Variable	Description
`FRAME_CLIENT_ID`	Obtained from the API provider when a set of API credentials are created.
`FRAME_CLIENT_SECRET`	Obtained from the API provider when a set of API credentials are created.
`FRAME_SAT_URL`	URL obtainable from the Playground. For example: https://api.console.nutanix.com/v1/accounts/XXXXXXXX-XXXX-XXXX-XXXX-31d09e2881cd/secure-anonymous/secure-anon-XXXXXXXX-XXXX-XXXX-XXXX-c5e2dc93df1e/tokens.
`FRAME_ACCOUNT_ID`	Sign in to Frane Console as an Admin. Locate your account, click the three-dot menu, and select "update" to view the Account's entity settings. Next, copy the Account UUID from the browser's URL bar. For example: `https://console.nutanix.com/frame/account/YOUR-FRAME-ACCOUNT-UUID-HERE/basic-info` or use the Admin API to List Accounts.
`FRAME_EMAIL_DOMAIN`	Email domain name used to create the anonymous user email addresses that will be visible in the Session Trail. Example: frame.igel.mycompany.com

Environment Variable	Description
`FRAME_LAUNCH_URL`	Obtained from an Account's Dashboard > Launchpad > Advanced Integrations to get a configurable dialog with Launch Links. While we recommend Launch Links for Kiosk scenarios, the value of FRAME_LAUNCH_URL could instead be a standard Launchpad URL.
`FRAME_TERMINAL_CONFIG_ID`	Obtainable from the Launch Link URL.
`FRAME_LOGOUT_URL`	Optional. Allows configuration of the "logout" behavior by specifying a URL. Useful when using a Frame Launch Link with additional "Quit and log out" url parameter: &qlo=1.

Frame Admin API and SAT Quick Setup Guide

1. Enable API access

   > “ Account > Users > Authentication

   Enable API
2. Add an API

   > “ Account > Users > API

   Create an API integration with with the ability to generate anonymous tokens and manage your account as an Account Administrator. These roles are **mandatory** for this custom partition's scripts; they use account-based Admin API calls to validate the current status of sessions (statuses such as "initializing", "open", "closing", etc.).
   API - Generate
3. Create a set of credentials for use with the Custom Profile.

   Manage Credentials
   Manage Credentials

   Create new API key
   Create new API key

   Copy the credentials for use in the IGEL Environment Variables. Keep it secret; keep it safe.

Secure Anonymous Access Setup

1. Enable "Secure Anonymous" access

> Account > Users > Authentication

2. Create Anonymous Access Provider

> Account > Users > Secure Anonymous

3. Add the Launchpad User role to the Provider

Note: If Launchpad User Role is not visible on the list, be sure you've created a launchpad first. If you have, refresh the page and try again.

4. Copy Provide URL from Playground Examples

Easily find and copy your SAT Provider URI:

Import a Frame App Profile

Follow these simple steps to import the Frame App profile into your IGEL environment and configure it according to your needs.

Step 1: Import the Frame App Profile

1. **Navigate to the IGEL App Portal** in the UMS server.
2. Go to the **Apps** tab (yellow tab at the top).
3. In the **left panel**, select **Cloud**.
4. Locate and select **Frame App**.

Step 2: Create a New Frame App Profile

1. Click on the **"Create new profile"** button.
2. In the **dialog box** that appears:
   - Enter a **Name** (e.g., "FrameApp-Test").
   - Enter a **Description** (optional, e.g., "Frame App application").
   - Select the **Location** (e.g., under "Profiles").
3. Click **Save** to confirm.

Step 3: Configure the Frame App Profile

1. Once the Frame App profile is created, click the **Edit Configuration** button.
2. Update the profile settings based on your requirements. Some configuration options include:
   - **Check for updates on startup**
   - **Clear cache before starting**
   - **Enable kiosk mode**
   - **Set kiosk timeout behavior**

Refer to your specific Frame use case to determine which options need to be enabled or customized.

Example Configuration

The final configuration may look similar to the following:

- A created profile named **FrameApp-Test**.
- Key settings such as startup behavior, cache clearing, and kiosk options enabled.

Frame-provided UMS profiles options are detailed at the top of the page. Pick a UMS Profile that sounds best for your IGEL use-case and import it to try it out.