https://verifone.cloud/docs/application-development-kit-version-47/pg_prx_users_guide
Updated: 17-Sep-2025

ADK-Proxy Programmers Guide

This document contains information on how to use and integrate Cloud Proxy to access the Verifone Cloud.

Audience

This guide helps application developers to install and utilize the functionality of Cloud Proxy.

Organization

This guide is organized as follows:

Chapter 1, Overview: Provides the introduction and overview for Cloud Proxy.

Chapter 2, Supported Platforms & System Requirements: Shows the OS platforms Cloud Proxy is provided for and specifies prerequisites to the environment.

Chapter 3, Environments and Download Packages: Description of Cloud Proxy installation packages and packages to activate or add additional cloud environments.

Chapter 4, Environment Configuration File: Description of Cloud Proxy environment configuration file and parameters.

Chapter 5, Local HTTP Proxy Support: Description how to configure a local HTTP proxy

Chapter 6, Getting Started: How to getting Cloud Proxy started on several platforms.

Chapter 7, Error Handling: Internal Error messages of Cloud Proxy and its meaning.

Chapter 8, Troubleshooting: Measures for identification and fixing problems.

Chapter 9, CP Log Forwarder: How to setup and configure the CP Log Forwarder.

Chapter 10, Cloud Proxy Configuration Interface: How to use Cloud Proxy Configuration Interface.

Appendix Version History and Greenbox Migration provides additional information about Cloud Proxy Version History. Depending on version, Cloud Proxy will migrate to Greenbox (GSC endpoints).

Related Documentation

To learn more about the ADK framework, please refer to the following documents:

- ADK GUI Programmers Guide
- ADK SYS Programmers Guide
- ADK Communication Service Programmers Guide
- ADK Logging Programmers Guide

Acronyms Definition

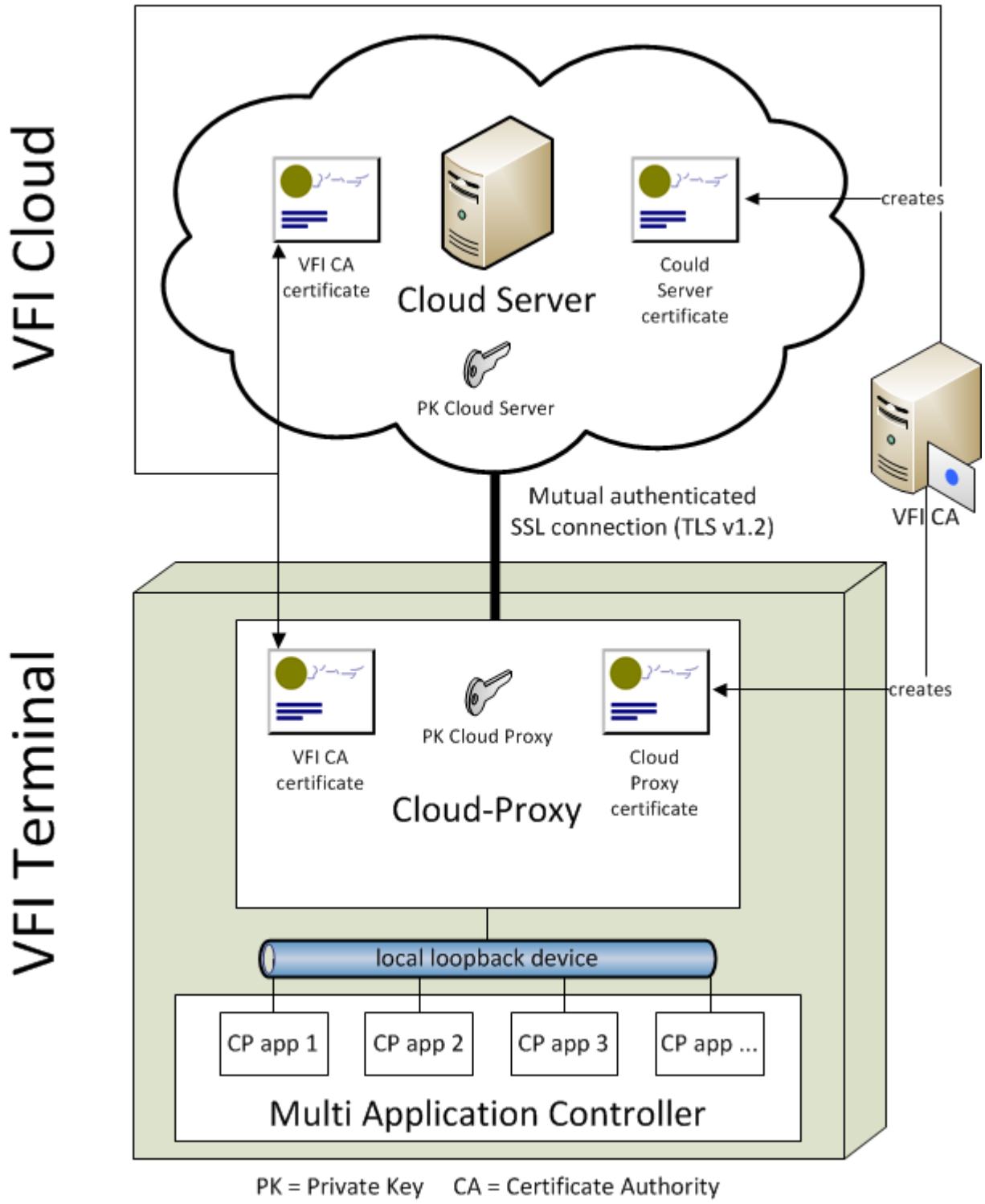
Acronym	Definitions
ADK	Application Development Kit
API	Application Programming Interface
DNS	Domain Name System
EOF	End of File
IPC	Inter Process Communication
JSON	JavaScript Object Notation
OS	Operating System
SDK	Software Development Kit
TCP/IP	Transmission Control Protocol/Internet Protocol
VOS	Verifone Operating System
VOS2	Verifone Operating System (Version 2)
VFI Cloud	Computer network providing online services and applications to Verifone terminals
CP	Commerce Platform
CPR	Commerce Platform Runtime
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
MAC	Multi Application Controller
TLS	Transport Layer Security
SSL	Secure Socket Layer
PK	Private Key
CA	Certificate Authority
URL	Uniform Resource Locator
PEM	Privacy Enhanced Mail
base 64	Binary-to-text encoding schemes that represent binary data in an ASCII string format
sysmode	System mode, administration GUI front-end on Fusion/Engage
RTT	Round Trip Time
CpDev	CP Development Unit
OsDev	OS Development Unit

ASL	ADK system launcher
LAN	Local area network

Overview

With Verifone Commerce Platform Runtime (CPR) a new application type was introduced: **Commerce Platform Applications (CP applications)**.

CP applications are considered as alternative applications formed by HTML/CSS/JS and can run beside native C/C++ applications that are typically in use for payment processing. CP applications usually operate online and they use HTTP requests to communicate with remote services of the VFI Cloud. Due to security reasons, CP applications must not be allowed to open arbitrary connections to external servers. Also direct incoming connections must not be allowed. For this reason, CP applications run in a controlled environment, some kind of sandbox, which is provided by the Multi Application Controller (MAC).



Cloud Proxy Overview

All connections initiated by CP applications are established and processed in context of MAC, which uses the local loopback device to connect to the Cloud Proxy running on the same terminal. MAC uses HTML based ADK GUI component, which allows CP applications to use JavaScript (JS) interface `xmlHTTPRequest()` for sending and receiving HTTP requests and responses. Cloud Proxy acts as HTTP gateway for CP applications and provides a secure SSL channel to VFI Cloud (TLSv1.2). For establishing the mutual authenticated SSL channel both instances, Cloud Proxy and Server have certificates and private keys required for secure encryption and authentication:

Cloud Proxy:

Data file	Description
Cloud Proxy certificate	Certificate containing the public key of the Cloud Proxy signed by Verifone Certificate Authority (VFI CA). This is used to authenticate the Cloud Proxy on the Cloud Server side.
VFI CA certificate	Certificate of the VFI CA. Cloud Proxy uses this certificate to verify the signature of the Cloud Server certificate to authenticate the Cloud Server.
PK Cloud Proxy	Private key of the Cloud Proxy, usually stored in secure area of the terminal. This key corresponds to the public key stored in Cloud Proxy certificate and it is used authenticate the Cloud Proxy on the Cloud Server side.

Cloud Server:

Data file	Description
Cloud Server certificate	Certificate containing the public key of the Cloud Server signed by Verifone Certificate Authority (VFI CA). This is used to authenticate the Cloud Server on the Cloud Proxy side.
VFI CA certificate	Certificate of the VFI CA. Cloud Server uses this certificate to verify the signature of the Cloud Proxy certificate to authenticate the Cloud Proxy.
PK Cloud Server	Private key of the Cloud Server. This key corresponds to the public key stored in Cloud Server certificate and it is used authenticate the Cloud Server on the Cloud Proxy side.

> Overview illustrates the case that both certificates, Cloud Proxy and Cloud Server certificate, were signed by one common VFI CA.

Long-lived TLS connection to Cloud Server

If a local client (e.g. CP app) sends a HTTP request, Cloud Proxy will establish a new TLS connection, if it has not connected before or the existing connection was closed or is disturbed. After some validation of the HTTP document, Cloud Proxy redirects the message to the Cloud Server. When response from Cloud Server is received, Cloud Proxy keeps the existing TLS channel as long as it is not closed by the server. This will avoid high RTTs for multiple HTTP requests, which would be caused by overhead of the TLS handshake during connection setup. After SSL negotiation, more efficient symmetric ciphers are used, therefore subsequent HTTP requests can be transmitted more faster than single, sporadic requests, which require a reconnect to the server.

> Since version 2.0.0 Cloud Proxy has implemented SSL sessionID caching according RFC 5246. SSL session resumption eliminates the private key operation when reconnecting to a service and, therefore, dramatically speeds up TLS connection setup.

Supported Platforms & System Requirements

Download packages for Cloud Proxy are available for the following platforms:

- Fusion (VOS)

  System requirements: Cloud Proxy for VOS requires at least OS version 30250102 or later.
- Engage (VOS2)

  System requirements: Cloud Proxy for VOS2 requires at least OS version 30350102 or later.

Environments and Download Packages

There are several variants and endpoints of Verifone Cloud. Each variant/endpoint represents an environment, which is addressed by its own URL. Implementation and configuration of these environments are different, therefore, users must be aware of using the correct endpoint, which provides required web services for the used CP apps. In addition, these environment endpoints require different authentication settings like certificates to establish secure connection with SSL. For this reason, several download packages are provided to install and activate a specific environment.

Cloud Proxy installs several default environments with the base package (see chapter Cloud Proxy base packages). Users can install an additional activation package on top to activate a specific default environment (see chapter Default environments and activation). In addition, users have to the option to install an User Configuration Package to add own environments if required (see chapter Install additional environments).

> Since version 3.1.0 of Cloud Proxy, terminals will migrate to new GSC environments of **Verifone Greenbox**. Old default environment endpoints of Verifone Cloud Gateway (CG) can still be used for compatibility reasons, but please note that Cloud Proxy will automatically switch to new GSC endpoints when it is updated. For more details please refer to chapter Version History and Greenbox Migration. Since version 3.2.0 (and version 3.0.40), Cloud Proxy has added support to add environments with the installation of an User Configuration Package. For more details please refere to chapter Install additional environments.

Cloud Proxy base packages

The following table lists the base download packages to install or uninstall Cloud Proxy:

Description	Download package

Base package

This package installs the Cloud Proxy binary and the base configuration (URLs and certificates) for the several default environments described in the chapter Default environments and activation. As long as no activation package for a specific environment is installed, Cloud Proxy uses one environment as default, which depends on device type:

- Productive unit: `prod` environment
- Test unit (OsDev/CpDev): `staging1` environment

VOS/VOS2 platform:

- `dl.cloudproxy-X.X.X-X-prod.tar`

  (sys13 package for productive and CpDev unit)
- `dl.cloudproxy-X.X.X-X.tar`

  (sys13 package for OsDev unit)

Removal package

This package uninstalls the base package. In addition, an installed activation package (see chapter chapter Default environments and activation) is removed.

VOS/VOS2 platform:

- `dl.cloudproxy-remove-X.X.X-X-prod.tar`

  (removes base and activation package from productive and CpDev unit)
- `dl.cloudproxy-remove-X.X.X-X.tar`

  (removes base and activation package from OsDev unit)

Default environments and activation

Each environment is specified by a configuration file having name `<environment>.env`. The content and configuration settings for an environment file are described in chapter Environment Configuration File.

Cloud Proxy base package (see Cloud Proxy base packages) installs several default environments, which are listed in the following table:

Environment name	Configuration file	Description
`gsc_dev`	`gsc-dev.env`	Environment settings for GSC Development Cloud
`gsc_qat`	`gsc-qat.env`	Environment settings for GSC Test Cloud
`gsc_cst`	`gsc-cst.env`	Environment settings for GSC Staging Cloud
`gsc_prod`	`gsc-prod.env`	Environment settings for GSC Productive Cloud

`gsc_cst_us`	`gsc-cst-us.env`	Environment settings for GSC Staging Cloud **(US region)**
`gsc_prod_us`	`gsc-prod-us.env`	Environment settings for GSC Productive Cloud **(US region)**
`dev`	`old-dev.env`	Environment settings for CG Development Cloud **Deprecated, please move GSC Development soon!!!**
`test`	`old-test.env`	Environment settings for CG Test Cloud **Deprecated, please move GSC Test soon!!!**
`staging1`	`old-staging1.env`	Environment settings for CG Staging Cloud **(Default for CpDev and OsDev units)** **Deprecated, please move GSC Staging soon!!!**
`prod`	`old-prod.env`	Environment settings for CG Productive Cloud **(Default for productive units)** **Deprecated, please move GSC Productive soon!!!**

On Fusion (VOS) and Engage (VOS2) the default environment files are located in subfolder `proxy` in working directory of binary `cloudproxy` under `/home/sys13`.

In addition, Cloud Proxy base package installs the following symbolic links pointing to one environment file above:

Environment link	Environment file	Description
`dev.env`	`old-dev.env`	Environment settings for CG Development Cloud
`test.env`	`old-test.env`	Environment settings for CG Test Cloud
`statging1.env`	`old-staging1.env`	Environment settings for CG Staging Cloud
`prod.env`	`old-prod.env`	Environment settings for CG Productive Cloud

Symbolic links were introduced for the following since version 3.0.40 and 3.4.0 for the following reasons:

1. Versions 3.0.40 and 3.4.0 were synchronized use the same set of default environment files. Symbolic links are used to resolve file name conflicts in both versions.
2. For version 3.0.40 a symlink `<name>.env` will point to file `old-<name>.env` to take over existing environment, which was used in previous versions < 3.0.40.
3. For version 3.4.0 the symlinks will point to new GSC endpoints to ensure automatic migration to GSC endpoints (see chapter Version History and Greenbox Migration).

With installation of an additional activation package, the user can activate another environment to select another cloud gateway endpoint.

Depending on the device type, Cloud Proxy uses a specified environment as default, if it was started for the first time without installed activation package.

In addition, since versions 3.0.40 and 3.4.0 different CG and GSC endpoint are used as defaults. Please refer to documentation of corresponding Cloud Proxy version to get more details.
Since version 3.5.0 (VOS3 support added), CG endpoints are no longer installed as default environments with the base package. Only versions >=3.0.50 can use it, which is available for VOS/VOS2 only.

The following activation packages are provided with this Cloud Proxy release:

Environment	Description	Activation package
`gsc_dev`	**GSC Development Cloud** URL: **`dev2.test-gsc.vfims.com:443`** This cloud environment shall be used by terminal or CP application developers to test software during terminal development. The development cloud may contain stable and unstable web services. Users can enable this environment by downloading the `gsc_dev` activation package.	VOS/VOS2 platform: • `dl.cloudproxy-cfg-gsc_dev-X.X.X-X-prod.tar` (system package for productive and CpDev unit) • `dl.cloudproxy-cfg-gsc_dev-X.X.X-X.tar` (system package for OsDev unit) These download packages **cannot be used for productive units**, since the environment gateway has not installed productive AuthEX CA! If installed on a productive unit, Cloud Proxy will ignore this configuration and fallback to default environment `prod`.
`gsc_qat`	**GSC Test Cloud** URL: **`qat2.test-gsc.vfims.com:443`** This cloud environment is used by QA teams and for validation. Stable and well tested web services are installed here to test and qualify existing CP applications. Users can enable this environment by downloading the `gsc_qat` activation package.	VOS/VOS2 platform: • `dl.cloudproxy-cfg-gsc_qat-X.X.X-X-prod.tar` (sys13 package for productive and CpDev unit) • `dl.cloudproxy-cfg-gsc_qat-X.X.X-X.tar` (sys13 package for OsDev unit) These download packages are allowed to be used for both, productive and test units!

`gsc_cst`	**GSC Staging Cloud** URL: **`cst2.test-gsc.vfims.com:443`** This cloud environment is used for demos or final development tests before a web service is deployed on a productive system. It can also be used to provide a stable web service as kind of a sandbox for external users (e.g. third-party developers). Users can enable this environment by downloading the `gsc_cst` activation package.	VOS/VOS2 platform: • `dl.cloudproxy-cfg-gsc_cst-X.X.X-X-prod.tar` (sys13 package for productive and CpDev unit) • `dl.cloudproxy-cfg-gsc_cst-X.X.X-X.tar` (sys13 package for OsDev unit) These download packages are allowed to be used for both, productive and test units!
`gsc_prod`	**GSC Productive Cloud** URL: **`gsc.verifone.cloud:443`** This cloud environment is used for productive purposes and terminals in field. Only stable and well tested web services are installed here. Users can enable this environment by downloading the `gsc_prod` activation package.	VOS/VOS2 platform: • `dl.cloudproxy-cfg-gsc_prod-X.X.X-X-prod.tar` (sys13 package for productive unit) These download packages are allowed to be used on **productive units only**! If installed on a test unit (CpDev or OsDev), Cloud Proxy will ignore this configuration and fallback to default environment `staging1`.

`gsc_cst_us`	**GSC Staging Cloud (US region)** URL: **`uscst-gb.gsc.vficloud.net:443`** Same as `gsc_cst`, but this endpoint is intended for use in US region. Users can enable this environment by downloading the `gsc_cst_us` activation package.	VOS/VOS2 platform: • `dl.cloudproxy-cfg-gsc_cst_us-X.X.X-X-prod.tar` (sys13 package for productive and CpDev unit) • `dl.cloudproxy-cfg-gsc_cst_us-X.X.X-X.tar` (sys13 package for OsDev unit) These download packages are allowed to be used for both, productive and test units!
`gsc_prod_us`	**GSC Productive Cloud (US region)** URL: **`us.gsc.verifone.cloud:443`** Same as `gsc_prod`, but this cloud environment is used for productive purposes and terminals in US region. Users can enable this environment by downloading the `gsc_prod_us` activation package.	VOS/VOS2 platform: • `dl.cloudproxy-cfg-gsc_prod_us-X.X.X-X-prod.tar` (sys13 package for productive unit) These download packages are allowed to be used on **productive units only**! If installed on a test unit (CpDev or OsDev), Cloud Proxy will ignore this configuration and fallback to default environment `staging1`.

`dev`	**CG Development Cloud (deprecated)** URL: **`dev.cgateway.verifone.com:443`** Old cloud environment for development. This environment is **deprecated** and just kept for compatibility reasons or to use old web services, which won't move to new environemnt `gsc_dev`. Users should migrate to the new environment soon, since environment `dev` will go offline after some time. Users can enable this environment by downloading the `dev` activation package.	VOS/VOS2 platform: • `dl.cloudproxy-cfg-dev-X.X.X-X-prod.tar` (sys13 package for productive and CpDev unit) • `dl.cloudproxy-cfg-dev-X.X.X-X.tar` (sys13 package for OsDev unit) These download packages **cannot be any longer used for productive units** , since the environment gateway has not installed productive AuthEX CA! If installed on a productive unit, Cloud Proxy will ignore this configuration and fallback to default environment `prod`.
`test`	**CG Test Cloud (deprecated)** URL: **`test.cgateway.verifone.com:443`** Old cloud environment for QA and testing. This environment is **deprecated** and just kept for compatibility reasons or to use old web services, which won't move to new environemnt `gsc_qat`. Users should migrate to the new environment soon, since environment `test` will go offline after some time. Users can enable this environment by downloading the `test` activation package.	VOS/VOS2 platform: • `dl.cloudproxy-cfg-test-X.X.X-X-prod.tar` (sys13 package for productive and CpDev unit) • `dl.cloudproxy-cfg-test-X.X.X-X.tar` (sys13 package for OsDev unit) These download packages are allowed to be used for both, productive and test units!

`staging1`	**CG Staging Cloud (deprecated)** URL: **`staging1.cgateway.verifone.com:443`** Old staging cloud environment. This environment is **deprecated** and just kept for compatibility reasons or to use old web services, which won't move to new environemnt `gsc_cst`. Users should migrate to the new environment soon, since environment `staging1` will go offline after some time. Users can enable this environment by downloading the `staging1` activation package. This cloud environment is used as default for test units (CpDev and OsDev). Therefore, these packages are not required for initial setup of these terminals.	VOS/VOS2 platform: • `dl.cloudproxy-cfg-staging1-X.X.X-X-prod.tar` (sys13 package for productive and CpDev unit) • `dl.cloudproxy-cfg-staging1-X.X.X-X.tar` (sys13 package for OsDev unit) These download packages are allowed to be used for both, productive and test units!

`prod`	**CG Productive Cloud (deprecated)** URL: **`cgateway.verifone.com:443`** Old staging cloud environment. This environment is **deprecated** and just kept for compatibility reasons or to use old web services, which won't move to new environemnt `gsc_prod`. Users should migrate to the new environment soon, since environment `prod` will go offline after some time. Users can enable this environment by downloading the `prod` activation package. This cloud environment is used as default for productive units. Therefore, this package is not required for initial setup of these terminals.	VOS/VOS2 platform: - `dl.cloudproxy-cfg-prod-X.X.X-X-prod.tar` (sys13 package for productive unit) These download packages are allowed to be used on **productive units only**! If installed on a test unit (CpDev or OsDev), Cloud Proxy will ignore this configuration and fallback to default environment `staging1`.

Install additional environments

Since version 3.2.0 (and version 3.0.40), Cloud Proxy has added support to install User Configuration Packages to add additional environments with required SSL certificates. User Configuration Packages are provided by the Cloud Proxy end users (application or integration team), thus, these packages can be signed having sponsor signing previleges (e.g. with regional singing cards for VOS/VOS2).

VOS/VOS2 User Configuration Packages

Basically, Cloud Proxy User Configuration Packages for VOS/VOS2 are installed to following read-only device location:

`/etc/config`

User Configuration Packages for Cloud Proxy must use subfolder `proxy` so that the related files will be located on the devices at follows:

`/etc/config/proxy`

User Configuration Packages may contain the following data:

1. Bundle with arbitrary name (user: `usr1-usr16`):
    - Package with arbitrary name (user: `usr1-usr16`, group: `share`, type: `config`):

Destination	File	Description

`/etc/config/proxy`	`<environment>.env`	Environment configuration file as specified in chapter Environment Configuration File. Flag `activate` in section `cloud` in the configuration file can be used to activate the environment with the next startup of Cloud Proxy (after the User Configuration Package is installed). The filename `<environment>.env` must differ to default environment files as installed with the Cloud Proxy base package (see chapter Default environments and activation). If a name conflicts, the environment file `<environment>.env` of the User Configuration Package is ignored.
`/etc/config/proxy`	`*.pem, *.p12`	Additional certificates/key files, which are referred by the environment file `<environment>.env`. For more details refer to descritiption of `ca-file` and `client-id` in chapter Environment Configuration File.

For more details how to create such an User Config Package, please refer to user guide Verifone Secure Installer ERS.

A sample User Configuration Package for the corresponding platform comes along in subfolder `example/load/usr_config` of documentation package `prx-doc-X.X.X-X.zip`:

- **dl.prx-usrconfig-sample-env-X.X.X-X.tar**

  This package installs a sample environment file `sample.env` with a root CA certificate `mycloud-ca.pem`. The certificate is refered by `ca-file` in the environment file, which is used to autenticate the server for cloud endpoint `url`.

Environment Configuration File

This chapter describes the content and settings of an environment configuration file. Several default environments come along with installation of Cloud Proxy base package (see chapter Cloud Proxy base packages), which can be activated with additional packages installed on top of the base installation. In addition, users have the option to install additional environments with a User Configuration Package (see chapter Install

additional environments).

The configuration file for the active environment is read at startup of Cloud Proxy and contains the following settings:

(Example configuration file: `gsc_qat` environment in file `test.env`)

```
[cloud]
; URL of the VFI-Cloud environment, e.g. vfi-cloud.verifone.com:8843, URL
is mandatory, default port: 8843
url=qat2.test-gsc.vfims.com:443
; name of the environment (alias). This is used as 'environment name' with
the Cloud Proxy Configuration Interface
; (see parameters cloud_env_list and cloud_env_list2). If not set, the
environment filename is used without
; extension *.env. Please note that alias must be identical crossover
environment configuration files and also not
; conflict with other environment filenames, if these files don't use an
alias.
alias=gsc_qat
; productive environment, set to value 'true', if this environment uses
productive AuthEx CA, which can be used by productive units only.
; Value 'false' implies that only cp-dev and os-dev devices can use this
environment having DEV AuthEx CA (default: false).
; Please note variable 'allow_prod_unit', which allows productive units to
use non-prod environment with AuthEx!
prod=false
allow_prod_unit=true
; activate this environment after it is installed with user config package,
 default: false
; activate=false -> default configuration coming along with Cloud Proxy
base package
; name of the file containing the root CA certificate of VFI cloud,
default=cloud-ca.pem
; Note: In case of using 'new-ca-file', 'ca-file' contains 2 CA root
certificates, which
; are used during transition phase to migrate VFI cloud to a new CA (CA
update):
; 1. CA certificate: for VFI cloud with the current CA to be replaced
(before the CA update)
; 2. CA certificate: for VFI cloud with the new CA (after the CA update)
ca-file=cloud-ca.pem
; name of the file containing the new root CA certificate of VFI cloud
after a pending
; CA update, default=empty
; Note: This certificate must equal the new 2. CA certificate specified in
file 'ca-file',
;       since Cloud Proxy switches from 'ca-file' to 'new-ca-file', after
it has connected
;       to the updated VFI cloud for the first time. From this point, Cloud
 Proxy will not
;       be able to connect to VFI cloud using the old CA.
; new-ca-file=new-cloud-ca.pem

[proxy]
; filename of the PEM or PKCS12 file containing an alternative client
```

```
certificate/private key.
; This is used, if device hasn't installed AuthEx certificate/key or AuthEx
 was disabled Cloud Proxy Configuration Interface.
; default: empty (force usage of AuthEx certifcate/key if installed)
; client-id=
; label string used by cloudproxy for the client certificate (if empty
'client-id' w/o file extension is used)
; client-id-label=
; client timeout in sec when waiting for replies from the cloud,
default=30sec
timeout=30
; listen port number for incoming connections from the clients,
default=8888
port=8888

[logservice]
; URL of the log gateway (usually a relative path)
url=/cp
; host as destination for the logs (host used for routing the logs)
host=logging-service.verifone.com
; initial log level used for CP app logging, default=5
loglevel=7

[connectivity]
; URL of the availability service (usually a relative path)
url=/DeviceConnectivity/Check
; availability service host
host=gsc
```

Description of settings in configuration file `<environment>.env`:

Section	Parameter	Description
`cloud`	`url`	URL of the VFI Cloud gateway, which specifies a host name (and a port). Both parameters are separated by ':'. If no port is specified, port `8843` is used as default. Please note that the cloud host name is mandatory and Cloud Proxy will not start, if it is not configured.
`cloud`	`alias`	Name or alias used for the environment. Parameter `alias` is optional. If not set, the environment filename is used without extension `*.env` (e.g. `prod.env` will use name `prod`). The environment name must be unambiguous crossover environment configuration files. If it conflicts with another environment name, the environment file is ignored. The environment name is used Cloud Proxy Configuration Interface for parameters `cloud_env`, `cloud_env_list`, `cloud_env_list2`.

`cloud`	`prod`	Flag set to `true`, if the environment is productive and uses productive AuthEx CA. This CA can be used by productive units only. Value `false` means that the environment is not productive and can only be accessed by CpDev and OsDev devices (as long as `allow_prod_unit` is not set to `true` (see below). Default value for `prod` is `false`.
`cloud`	`allow_prod_unit`	Flag set to `true`, if the non-productive environment (`prod` set to `false`) is allowed to be used by a productive unit using AuthEx. This value is ignored, if `prod` is set to `true`. Default value for `allow_prod_unit` is `false`.
`cloud`	`activate`	Flag set to `true`, if the environment specified with this configuration files shall be activated, if it is installed (default: `false`). Default environments installed with Cloud Proxy base package don't use this flag. This was introduced for additional environments installed with an User Configuration Package (see chapter Install additional environments).
`cloud`	`ca-file`	File containing the X.509 root CA certificate of the VFI cloud. Cloud Proxy uses this certificate to verify the signature of the Cloud Server certificate to authenticate the Cloud Server. The file is in PEM format (Privacy Enhanced Mail) and stores the certificate base 64 encoded. If parameter `ca-file` is not specified, Cloud Proxy uses ca-file.pem as default. If the file does not exist, Cloud Proxy will not start and aborts with an error. If a relative file path is used, Cloud Proxy binary will lookup the certificate file relative to this environment configuration file. In case of using `new-ca-file`, `ca-file` contains 2 CA root certificates, which are used during transition phase to migrate VFI cloud to a new CA (CA update): 1. CA certificate: for VFI cloud with the current CA to be replaced (before the CA update) 2. CA certificate: for VFI cloud with the new CA (after the CA update)

		File containing the new X.509 root CA certificate for the VFI cloud after a pending CA update. The file is in PEM format (Privacy Enhanced Mail) and stores the certificate base 64 encoded. If parameter `new-ca-file` is not specified, Cloud Proxy is not prepared for a CA update and and uses the root CA certificate specified by `ca-file` . If `new-ca-file` is specified and the file does not exist, Cloud Proxy will not start and aborts with an error.
`cloud`	`new-ca-file`	The new root certificate must equal the new 2. CA certificate specified in file `ca-file`, since Cloud Proxy switches from `ca-file` to `new-ca-file`, after it has connected to the updated VFI cloud for the first time. From this point, Cloud Proxy will not be able to connect to VFI cloud using the old CA. If a relative file path is used, Cloud Proxy binary will lookup the certificate file relative to this environment configuration file.
		For testing purposes (e.g. development tests) or due to legacy reasons the filed `client-id` can optionally used to provide a file containing a X.509 certificate and a private key, which will be used as alternative to AuthEx. The certificate is transmitted to Cloud Server during connection setup to authenticate the terminal as a trusted client (mutual authenticated SSL connection). The file can be provided in PEM format (base 64 encoded, plain text) or in PKCS12 format, which data is usually encrypted with a password.
`proxy`	`client-id`	It is definitely **not recommented** to use an own client certificate/key, since storing the private key in file system is potentially insecure! Therefore, the field `client-id` should be removed (or kept empty) that will enable AuthEx certificate/key as preferred authentication method. AuthEx is usually available with pre-installed Warrenty Keys on all terminals. Warrenty Keys are stored in the terminal's vault and cloudproxy will use secure SSL engine to access the keys. If a relative file path is used, Cloud Proxy binary will lookup the certificate file relative to this environment configuration file.
`proxy`	`client-id-label`	Label string used by Cloud Proxy for the client certificate specified by `client-id`. The label is returned Cloud Proxy Configuration Interface in certificate list for each in environemnt (see parameter `cloud_env_list2`). If this parameter is empty, Cloud Proxy used filename `client-id` w/o file extension.

`proxy`	`timeout`	Timeout in seconds which is used by Cloud Proxy to connect or receive data from the Cloud Server. If no connection could be established or no data is received from Cloud Server within the specified timeout, the Cloud Proxy returns a HTTP error response to the requesting client. For more details please refer to chapter Error Handling.
`proxy`	`port`	This parameter specifies the listen port for incoming connections from clients. Please note that the listen port on productive units is restricted to local loopback device and external incoming connections are declined. No authentication is implemented and all local processes are allowed to connect. This means that access is not restricted to CP applications and MAC only. Default port is `8888`. For testing and development purposes the port is enabled for external access when using a test unit (CpDev and OsDev).
`logservice`	`url`	URL of the webserver providing the *Verifone CP Logging Service*. This is usually a fully qualified URL with a host name (and a port). If the webserver is behind a reverse proxy the URL might be relative path. Optional parameter `host` can be specified to address a downstream server behind the proxy. Host name and port of the URL are separated by ':'. If no port is specified, port `80` is used as default. Please note that this URL is mandatory and CP Log Forwarder is disabled, if it is not configured or empty. For more details please refer to chapter CP Log Forwarder.
`logservice`	`host`	This value specifies the destination host for the logs. Usually this parameter is used for routing, if the webserver for the *Verifone CP Logging Service* is behind a proxy. This could also be third party host receiving the logs. CP Logging data is forwarded to the VFI Cloud gateway, therefore, value `host` is not interpreted, just added as field `Host` in HTTP header of the CP logging message. The parameter `host` is optional.
`logservice`	`loglevel`	This is the initial log level used for CP app logging. The default is 5. CP log messages with a logging level above `loglevel` are filtered and not send to *Verifone CP Logging Service*. The VFI Cloud may update this value per CP app during runtime.
`connectivity`	`url`	URL of the webserver providing the *Verifone Connectivity Service*. This is usually a fully qualified URL with a host name (and a port). If the webserver is behind a reverse proxy the URL might be relative path. Optional parameter `host` can be specified to address a downstream server behind the proxy. Host name and port of the URL are separated by ':'. If no port is specified, port `80` is used as default. Please note that this URL is mandatory and connectivity check is disabled, if it is not configured or empty. For more details please refer to chapter Cloud Connectivity Check.

`connectivity host`	This value specifies the destination host for the connectivity check requests. Usually this parameter is used for routing, if the webserver for *Verifone Connectivity Service* is behind a proxy. HTTP messages for connectivity check are forwarded to the VFI Cloud gateway, therefore, value `host` is not interpreted, just added as field `Host` in HTTP header of the connectivity check request. The parameter `host` is optional.

Local HTTP Proxy Support

Since version 3.4.0 (and version 3.0.40), Cloud Proxy has added support to establish Cloud connections over a HTTP proxy. This HTTP proxy usually is available in LAN and provides HTTP clients access to outer world. Instead of connecting to Cloud endpoint directly (which is specified by Environment Configuration File), Cloud Proxy first establishes the TCP/IP connection to this local HTTP proxy. As second step it sends a HTTP CONNECT request to this proxy to establish a transparent TCP/IP channel to the Cloud endpoint. Finally, Cloud Proxy does the TLS handshake over this spliced channel to establish a secure end-to-end TLS connection to the Cloud endpoint.

The activation of a local HTTP proxy is done by configuration file `httpproxy.cfg`.

The file may contain multiple sections to configure a HTTP proxy for one or more specific Cloud environments.

```
; productive environment gsc_prod uses this proxy (specified by prod.env)
[gsc_prod]
proxy=192.168.188.250:8888
attribute_1=Proxy-Authorization: Basic YW5kcmU6ZHJvd3NzYXA=
attribute_2=X-VFI-TEST: Some proxy data
attribute_3=X-VFI-TEST2: More proxy data

; pseudo environment name 'default' is used to configure a default
 HTTP proxy
[default]
proxy=192.168.188.250:8080
attribute_1=X-VFI-TEST: Some proxy data
attribute_2=X-VFI-TEST2: More proxy data
```

The section name represents the environment name, which is either the environment alias specified in (see Environment Configuration File) or the basename of environment configuration file w/o extension, in case no alias is used.

> Default environment configurations provided with Cloud Proxy base package use aliases, which are listed in table of chapter Default environments and activation as environment name. Thus, user must specify the section by these names to match an environment, e.g. `gsc_prod` is used for environment file `gsc-prod.env`.

Pseudo environment name `default` represents the default HTTP proxy, which is used, if the name of the active environment does not match any other section. Section `default` can be used as one and only section, if only one proxy shall be used for all environments. If section `default` is omitted and no corresponding section matches to the name of the active environment, no local HTTP proxy will be used.

> If additional environments are installed by User Configuration Package (see chapter Install additional environments), users should avoid to use `default` as environment name. In this case section `default` would be applied for this environment only.

Description of section parameters in configuration file `httproxy.cfg`:

Parameter	Description
`proxy`	URL of the HTTP proxy, which specifies a host name or IP address (with a port). Both parameters are separated by ':'. If no port is specified, port `8888` is used as default. Please note that parameter `proxy` is mandatory and section skipped, if it misses mandatory data or contains an invalid format. IPv6 IP address are bracketed with `[]`, example: `proxy=[::1]:12345`.
`attribute_<number>`	HTTP header attribute(s) added to HTTP CONNECT request, which is sent to HTTP proxy to establish the TCP/IP channel. Attribute `<number>` starts with 1, numbers of following attributes must have ascending order. Attributes for HTTP CONNECT might be required, if the HTTP proxy requires authentication with `Proxy-Authorization`.

HTTP proxy configuration file `httpproxy.cfg` is installed with an User Configuration Package. User Configuration Packages are provided by the Cloud Proxy end users (application or integration team), thus, these packages can be signed having sponsor signing previleges (e.g. with regional singing cards for VOS/VOS2).

VOS/VOS2 User Configuration Packages

A VOS/VOS2 User Configuration Package is installed to `/etc/config`, whereas Cloud Proxy uses subfolder `/etc/config/proxy` (see also chapter Install additional environments).

An User Configuration Package to active/configure a HTTP proxy may contain the following data:

1. Bundle with arbitrary name (user: `usr1-usr16`):
   - Package with arbitrary name (user: `usr1-usr16`, group: `share`, type: `config`):

Destination	File	Description
`/etc/config/proxy`	`httpproxy.cfg`	HTTP proxy configuration file with content specified above. With restart of Cloud Proxy, the configuration will be read and applied for the environments.

A sample User Configuration Package for the corresponding platform comes along in subfolder `example/load/usr_config` of documentation package `prx-doc-X.X.X-X.zip`:

- **dl.prx-usrconfig-sample-proxy-X.X.X-X.tar**

  This package installs a sample configuration file `httpproxy.cfg` for using one HTTP proxy `192.168.188.34:8888` for all environments:

```
; all environments use 'default' proxy
[default]
proxy=192.168.188.34:8888
attribute_1=Proxy-Authorization: Basic YW5kcmU6ZHJvd3NzYXA=
attribute_2=X-VFI-TEST: Hello
attribute_3=X-VFI-TEST2: World
```

Getting Started

Depending on platform do the following steps to install Cloud Proxy:

- VOS/VOS2 platform:
  - Get latest load distribution package `prx-vos-load-X.X.X-X.zip` or `prx-vos2-load-X.X.X-X.zip` from Artifactory and extract it. Alternatively download full package `prx-full-X.X.X-X.zip` containing packages for all platforms.
  - On terminal enter sysmode (system mode) and put terminal into Netloader download mode
  - On PC start MX9 Downloader tool and download the base package `dl.cloudproxy-X.X.X-X.tar` (see chapter Cloud Proxy base packages)
  - After installation of the package in home directory of user `sys13`, the terminal reboots and cloudproxy service is automatically launched by launched by MAC. Please note that cloudproxy get launched only, if at least one CPApp is installed.
  - If required, user can install an additional activation package to enable a specific default environment (see chapter Default environments and activation). For this, just repeat the steps above with the corresponding activation package.

    Please note that Cloud Proxy binary depends on ADK libraries `libvfiipc` and `liblog`. Instructions for installation on the corresponding platform can be found in documentations of *ADKIPC* and *ADKLOG* projects.

Error Handling

After Cloud Proxy has accepted a new client connection to process a HTTP request, there might occur several errors for a variety of reasons. Behaviour and actions of Cloud Proxy in this error situations depend on phase of

the live circle of the HTTP request. The following table lists the session states of Cloud Proxy in which HTTP request is processed and the possible errors, which will lead to the resulting actions:

Session State	Errors	Actions
Cloud Proxy receives the request HTTP header from the client	1. HTTP header is invalid (e.g. format error) 2. HTTP header is not received or incomplete (receive timeout) 3. I/O error while receiving the HTTP header (e.g. connection is closed by client)	1. Cloud Proxy returns HTTP response **400 (Bad Request)** and closes the client socket 2. Cloud Proxy returns HTTP response **408 (Request Timeout)** and closes the client socket 3. Cloud Proxy closes the client socket
Cloud Proxy connects to VFI Cloud gateway	1. No TCP/SSL connection can be established (e.g. Gateway not reachable or SSL handshake failed)	1. Cloud Proxy returns HTTP response **503 (Service Unavailable)** and closes the client socket
Cloud Proxy sends the request HTTP header to VFI Cloud gateway	1. I/O error while sending the HTTP header (e.g. connection is closed by VFI Cloud gateway)	1. Cloud Proxy returns HTTP response **500 (Internal Server Error)** and closes the client socket
Cloud Proxy forwards the request HTTP document from client to VFI Cloud gateway	1. HTTP document is invalid (e.g. format error) 2. HTTP document is not received or incomplete (receive timeout) 3. I/O error while receiving the HTTP document (e.g. connection is closed by client) 4. I/O error while sending the HTTP document (e.g. connection is closed by VFI Cloud gateway)	1. Cloud Proxy returns HTTP response **400 (Bad Request)** and closes the client socket 2. Cloud Proxy returns HTTP response **408 (Request Timeout)** and closes the client socket 3. Cloud Proxy closes the client socket 4. Cloud Proxy returns HTTP response **500 (Internal Server Error)** and closes the client socket

Cloud Proxy receives the response HTTP header from the VFI Cloud gateway	1. HTTP header is invalid (e.g. format error) 2. HTTP header is not received or incomplete (receive timeout) 3. I/O error while receiving the HTTP header (e.g. connection is closed by VFI Cloud gateway)	1. Cloud Proxy returns HTTP response **502 (Bad Gateway)** and closes the client socket 2. Cloud Proxy returns HTTP response **504 (Gateway Timeout)** and closes the client socket 3. Cloud Proxy returns HTTP response **500 (Internal Server Error)** and closes the client socket
Cloud Proxy sends the response HTTP header to client	1. I/O error while sending the HTTP header (e.g. connection is closed by client)	1. Cloud Proxy closes the client socket
Cloud Proxy forwards the response HTTP document from VFI Cloud gateway to client	1. HTTP document is invalid (e.g. format error) 2. HTTP document is not received or incomplete (receive timeout) 3. I/O error while receiving the HTTP document (e.g. connection is closed by VFI Cloud gateway) 4. I/O error while sending the HTTP document (e.g. connection is closed by client)	1. Cloud Proxy closes the client socket 2. Cloud Proxy closes the client socket 3. Cloud Proxy closes the client socket 4. Cloud Proxy closes the client socket

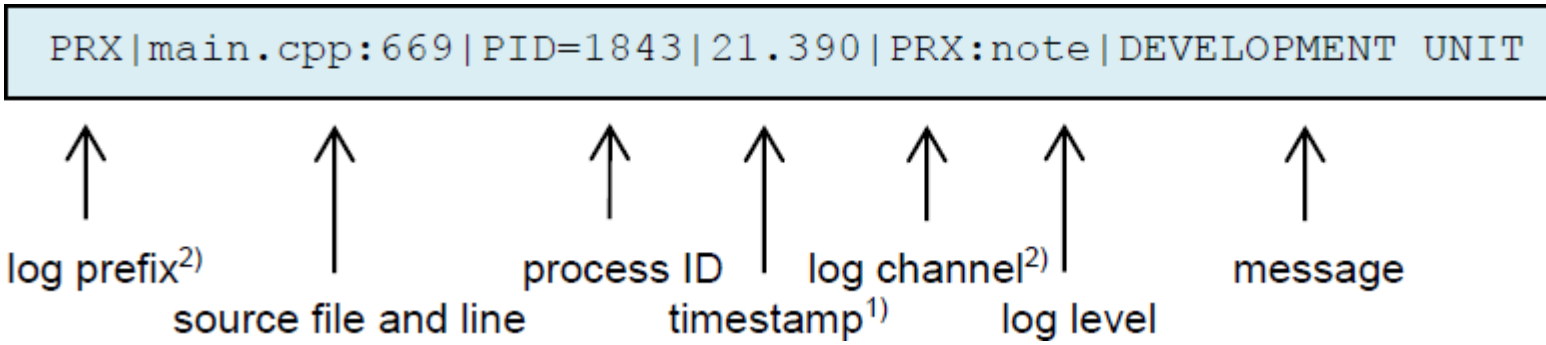
Troubleshooting

On all platforms logging messages can be enabled by setting environment variable `PRX_LOGMASK`. This variable is defined as a bitmask consisting of following decimal values:

```
1   = LOG_EMERG:   log messages for conditions, if system is unusable2   =
LOG_ALERT:   log messages, which action must be taken immediately4   =
LOG_CRIT:    log messages for critical conditions8   = LOG_ERR:     log
messages for error conditions16  = LOG_WARNING: log messages for warning
conditions32  = LOG_NOTICE:  log messages for normal but significant
conditions64  = LOG_INFO:    log messages with informational contents128 =
LOG_DEBUG:   log debug-level messages
```

Default value for `PRX_LOGMASK` is `0`, which means that no logging outputs are activated by default. `LOG_EMERG` represents the lowest logging level, which only generates messages, if Cloud Proxy cannot be started or is not working at all. The highest level `LOG_DEBUG` will produce many messages of low-level I/O routines and should only be enabled for debugging purposes. For first analyses it is recommended to set `LOG_ERR`, which will provide information about error conditions, e.g. communication problems.

A console logging message consist of the following fields:



1) *timestamp:*

The timestamp format is `<seconds>.<milliseconds>`. The value represents the time since beginning of capture (startup or first usage of IPC component).

2) *log prefix, log channel:*

The field `log prefix` is used to identify Cloud Proxy as logging source. For identification of the subcomponent the field `log channel` was introduced. Recently this is always `PRX`, but other names can be used for future extensions.

> On Fusion (VOS), Engage (VOS2) and VOS3 Cloud Proxy is running as system process and environment variables can be set with intallation package only. Due to the fact that system packages require additional privileges to modify and resign them, users should prefer ADKLOG logging mechanism for these platforms, which is descriped below.

Support for ADKLOG: Logging with liblog library

Since version 2.0.1 the Cloud Proxy also has added support for ADKLOG component. ADKLOG is required to implement the new ADK logging concept, which provides the Logging Control Panel (LCP) used as central instance to configure and enable logging for the several ADK components. For this, LCP uses a configuration files (recently one for Cloud Proxy), which are read by `liblog` library. If installed on the system, `liblog` library will be used by Cloud Proxy to output logging messages. In this case, the logging messages are passed to `liblog` library instead of using console logging mechanism (with stderr), which is described above.

In order to lookup the corresponding configuration files, ADKLOG uses component identifiers, which are reserved for each ADK component. Cloud Proxy uses the following:

Component Id	Configuration file	Description
`PRX`	`PRX_log.conf`	Cloud Proxy related log settings

The configuration files contain several settings for logging like output channels, verbosity and a logging mask, which is similar to environment variable `PRX_LOGMASK`. For more details about configuration settings or

logging message formats, please refer to documentation of ADKLOG project.

> If environment variable `PRX_LOGMASK` is set, console logging is preferred and ADKLOG with `liblog` is disabled. Only if the environment variable is unset, Cloud Proxy will lookup `liblog` library from the system to enable logging via ADKLOG.

CP Log Forwarder

CP applications may generate logging messages, which needs to be transmitted to *Verifone CP Logging Service*. This service is coupled to the VFI Cloud server and the basic usage is to provide an interface to CP apps for statistics and error reporting. Without the need initially of a sensitive data logging filter the messages must be transferred in a secure way. Therefore, Cloud Proxy has implemented a CP Log Forwarder, which redirects messages to the *Verifone CP Logging Service* over the secure SSL channel to VFI Cloud. The CP Log Forwarder receives logging data from CP apps with notification `_CPLog` over interface of *ADKIPC* component. No result notification is returned to CP applications by the CP Log Forwarder. Content and format of Log HTTP messages are described in *CPR* documentation. For details about notification interface, please refer to *ADKIPC* documentation and the programmers guide.

CP Log Forwarder is configured in environment configuration file `<environment>.env` (see chapter Environment Configuration File).

Cloud Proxy Configuration Interface

Since version 3.0.0 of Cloud Proxy, applications are allowed to use the remote configuration interface to read out information and settings of Cloud Proxy. Recently, the interface is fully implemented by CPDownloader application, which comes along with ADKGUI component. CPDownloader allows to show information like installed Cloud Proxy version or which environment or certificates are being used. In addition, a small amount of settings can be configured into Cloud Proxy, for instance, it is possible to switch the cloud environment during runtime. This chapter provides an overview about the Cloud Proxy Configuration Interface and instructions for applications that want to implement the interface instead of using CPDownloader.

Cloud Proxy Configuration Interface is provided via notification service of *ADKIPC* component. An application can send configuration requests as IPC notification with notification `CloudProxyConf`. After Cloud Proxy has processed the request, it responds with notification `CloudProxyConfResponse`. Both, configuration request and responses have a JSON encoded payload and use a JSON object `proxy` as container, which contains all data. An overview about the supported commands and its parameters is given by the following chapters.

> Please note that an application must initialize IPC notification service with a unique application ID. For more details please refer to *ADKIPC* documentation and the programmers guide.

Read Static Information

Static information parameters like version and build details can be requested with command `info`. Static information parameters won't change until next reboot.

`CloudProxyConf` parameters:

Container	Parameter	Content
`proxy`	`command`	`info`

Example:

```
{  "proxy" : {    "command" : "info"  }}
```

`CloudProxyConfResponse` parameters:

Container	Parameter	Content
`proxy`	`auth_ex_support`	AuthEx key/certificate for SSL client authentication is installed and supported by the system: `1`=supported `0`=unsupported
`proxy`	`buildinfo`	Build information string of format: `<name> <version> <month> <day> <year> <hour>:<min>:<sec>`
`proxy`	`device_type`	Detected device type string: `prod`=productive unit `os-dev`=Fusion/Engage OsDev unit `cp-dev`=Fusion/Engage CpDev unit
`proxy`	`version`	Version string in format: `<major number>.<minor number>.<bugfix number>-<build number>`
`proxy`	`result`	Result: `success`=command successful `error`=command failed

Example:

```
{  "proxy" : {    "auth_ex_support" : 0,    "buildinfo" :
"cloudproxy 3.0.0-1 Mar 14 2018 12:16:56",    "device_type" : "prod",
```

```
"version" : "3.0.0-1",    "result" : "success"  }}
```

Read Configuration

Configuration settings can be read with command `readconfig`.

`CloudProxyConf` parameters:

Container	Parameter	Content
`proxy`	`command`	`readconfig`

Example:

```
{  "proxy" : {    "command" : "readconfig"  }}
```

`CloudProxyConfResponse` parameters:

Container	Parameter	Content
		AuthEx certificate for SSL client authentication: `1`=enabled `0`=disabled
`proxy`	`auth_ex`	This parameter is optional. If not supplied, AuthEx certificate is not installed or not supported by the system. AuthEx key/certificate is installed with productive warrenty keys. If AuthEx certificate is disabled or not supported, Cloud Proxy will use the alternative certificate specified in environemnt configuration file (see section `proxy`, key `client-id`). For more details, please refer to chapter Environment Configuration File.
`proxy`	`cloud_env`	Name of recently enabled cloud environment, a string of environment list provided by parameter `cloud_env_list` (see below).
`proxy`	`cloud_env_file`	Configuration filename of recently enabled cloud environment. Depending on platform this can be an absolute or relative path. Base of the relative path is the working directory of Cloud Proxy binary.

			List of available cloud environments to enable, an array of strings with environment names (see table of chapter Default environments and activation).
`proxy`	`cloud_env_list`		
			This list is dynamic, e.g. productive environments will not be returned on test units (CpDev and OsDev).
			List of available cloud environments to enable as JS Object array. Each entry consists of a JS object per environment with the following data:

Parameter	Type	Description
`name`	String	Environment name (same as provided by parameter `cloud_env_list`).
`prod`	Boolean	Flas set to true, if the environment is productive and can be used with productive AuthEx key/certificate only.
`certs`	Array of Strings	Array of strings with certificate names, which are available for this environment. Devices having installed AuthEx key/certificate the list contains the string `auth-ex`.

`proxy`	`cloud_env_list2`	This environment list is dynamic, e.g. productive environments will not be returned on test units (CpDev and OsDev). Order of environments is the same as returned in parameter `cloud_env_list`. Parameter `cloud_env_list2` was introduced with Cloud Proxy 3.0.11. Previous versions do not provide this parameter.
`proxy`	`cloud_hostname`	Hostname of VFI Cloud gateway of recently enabled cloud environment.
`proxy`	`result`	Result: `success`=command successful `error`=command failed

Example:

```
{  "proxy" : {    "auth_ex" : 1,    "cloud_env" : "dev",
"cloud_env_file" : "proxy/dev.env",    "cloud_env_list" : ["prod", "dev",
"test", "staging1"],    "cloud_env_list2" : [      {"name":"prod",
"prod":true,       "certs":["auth-ex"]      },      {"name":"dev",
"prod":false,       "certs":["auth-ex"]      },      {"name":"test",
"prod":false,       "certs":["auth-ex"]      },      {"name":"staging1"
,       "prod":false,       "certs":["auth-ex"]      }    ],
"cloud_hostname" : "dev.cgateway.verifone.com",    "result" : "success"  }}
```

Write Configuration

Configuration settings can be written with command `writeconfig`.

`CloudProxyConf` parameters:

Container	Parameter	Content
`proxy`	`command`	`writeconfig` Name of cloud environment to be enabled, a string of environment list provided with parameter `cloud_env_list` in command `readconfig` (see above).
`proxy`	`cloud_env`	Test units (CpDev and OsDev) does not allow to enable environment `prod` and in this case command `writeconfig` will fail on these devices.
`proxy`	`auth_ex`	If installed and supported, AuthEx certificate for SSL client authentication can be enabled or disabled: `1`=enable `0`=disable It is not allowed to disable AuthEx certificate for an environment, which does not provide an alternative certificate in certificate list different than string `auth-ex` (see parameter `certs` in environment list `cloud_env_list2`). In addition, AuthEx certificate cannot be enabled, if it is not installed or supported by the system (see chapters Read Static Information and Read Configuration). In both cases command `writeconfig` will fail.

Example:

```
{  "proxy" : {    "command" : "writeconfig",    "cloud_env" : "dev",
"auth_ex" : 0  }}
```

`CloudProxyConfResponse` parameters:

Container	Parameter	Content
`proxy`	`result`	Result: `success`=command successful `error`=command failed

Example:

```
{  "proxy" : {    "result" : "success"  }}
```

Cloud Connectivity Check

Applications can use command `cloudcheck` to check the connectivity to VFI Cloud gateway. With this command Cloud Proxy will establish a secure SSL connection to VFI Cloud gateway, which will forward the request to *Verifone Connectivity Service* webserver, which is behind the VFI Cloud gateway. URL and host of the *Verifone Connectivity Service* webserver is configured in environment configuration file `<environment>.env` (see chapter Environment Configuration File).

`CloudProxyConf` parameters:

Container	Parameter	Content
`proxy`	`command`	`cloudcheck`

Example:

```
{  "proxy" : {    "command" : "cloudcheck"  }}
```

`CloudProxyConfResponse` parameters:

Container	Parameter	Content
`proxy`	`error_code`	HTTP status code of Connectivity Check request, examples: • `200` (OK) • `503` (Service Unavailable) Status codes `2xx` are returned for success, others mean failure.
`proxy`	`error_reason`	Result as string (readable error reason), examples: • `OK` • `Service Unavailable`
`proxy`	`result`	Result: `success`=command successful `error`=command failed `success` is returned for HTTP status code `2xx`, else `error` is returned

Example:

```
{  "proxy" : {    "error_code" : 200,    "error_reason" : "OK",    "result"
 : "success"  }}
```

Version History and Greenbox Migration

Versions < 3.0.7 are out-dated and cannot be used any longer, since Commerce Gateway (CG) has changed server certificates to new root CA "AddTrust External CA Root" (Sectigo Limited). Therefore, older Cloud Proxy version won't be able to connect if installed.

Basically, ADK Cloud Proxy is provided as two different version variants:

- Versions < 3.1.0 are used for older ADKs (ADK <4.9), which use Verifone Commerce Gateway (CG) endpoints with base URL `cgateway.verifone.com`
- Versions >= 3.1.0 are planned for ADKs >=4.9 intened for use of new GSC environments of **Verifone Greenbox**.

With update of Cloud Proxy to version >= 3.1.0, new GSC entpoints will automatically be applied with installation of the base package (see chapter Cloud Proxy base packages).

The configured environment endpoint (used by old Cloud Proxy < 3.1.0) is mapped to the corresponding new GSC endpoint, which will be used by the updated Cloud Proxy >= 3.1.0.

The mapping of the environments during the update shows the follwing table:

CG environment with old versions (< 3.1.0)	**GSC environment with new version (>= 3.1.0)**
`dev` (dev.cgateway.verifone.com)	`gsc_dev` (dev2.test-gsc.vfims.com)
`test` (test.cgateway.verifone.com)	`gsc_qat` (qat2.test-gsc.vfims.com)
`staging1` (staging1.cgateway.verifone.com)	`gsc_cst` (cst2.test-gsc.vfims.com)
`prod` (cgateway.verifone.com)	`gsc_prod` (gsc.verifone.cloud)

Even version >= 3.1.0 might use old CG environment endpoints, if the corresponding activation package is installed.

Finally, version 3.1.0 ADK Cloud Proxy has dropped support for Activation and Parameter Service, because the services were not used in past and no longer required. For this reason, customers must disable requests to the Activation and Parameter Service when using a newer Cloud Proxy in an older ADKs with Multi Application Controller (MAC) < 3.78.3. Otherwise it won't be possible to activate installed CPApps. For more details please refer to MAC documentation or documentation of old Cloud Proxy versions.

Since version 3.2.0 Cloud Proxy has added support to add environments with the installation of an User Configuration Package (see chapter Install additional environments).

Because customers weren't able to migrate their CPApps to new GSC endpoints within the envisaged timeframe, existing ADKs still have integrated an old version 3.0.x.

Since 3.0.40 and 3.4.0 both version branches were synchronized and have a common code base, since features implemented for ADK 4.9 (e.g. GSC support) were also required for old ADKs. The only difference is version 3.4.0 will automatically switch to GSC endpoints with its installation, whereas version 3.0.40 will keep CG endpoints and must use activation packages to configure a GSC endpoint (see chapter Default environments and activation). Thus, future ADKPRX releases will be provide in 2 variants supporting the same features. These variants use the following version schemes:

- `x.y.z` used for ADKs >= 4.9
- `x.0.yz` used for older ADKs < 4.9

With version 3.5.0 Cloud Proxy has added support for VOS3 platform. In addition, old CG endpoint were removed from the base package (see Cloud Proxy base packages). Thus, old CG endpoints (`dev`, `test`, `staging1` and `prod`) are no longer available with default environments (see Default environments and activation of version 3.5.0). Old ADKs for VOS/VOS2 from now must use version 3.0.50, if they still want to keep support for old endpoints, since version 3.5.0 as dropped it. Finally, documentation of version 3.0.50 declares CG endpoints as deprecated with recommendations to migrate to GSC. The last option to use old CG endpoints with version 3.5.0 is the installation of an User Configuration package on top of the base package. (see chapter Install additional environments). Customers are advised not to do this for new productive installations, since old CG environment endpoints will go offline sooner or later.

For more version details please read release notes `prx-releasenotes-X.X.X-X.pdf` included in documentation package `prx-doc-X.X.X-X.zip`, which is provided with the ADK Cloud Proxy release.