Agent Instance

At a general level, agents are responsible for carrying out tasks for remote users. They can connect over the network to any other type of instance. At minimum, agents must be associated with one server.

Instance Configuration

```
{
  "network" : {
    "servers" : String(),
    "timeout" : Number(), # The server connection timeout in milliseconds
    "strict_certs" : Boolean(), # The agent will refuse to connect to a server
that presents an invalid certificate
    "polling_interval" : Number(), # The connection poll interval in
milliseconds
  }
}
```

Connection Modes

There are two connection modes that have an impact on performance and latency.

Continuous

In continuous mode, the agent maintains its primary connection at all times. If the connection is lost, the agent will periodically attempt to reestablish the connection using the same parameters it used to establish the initial connection.

The connection mode can be changed on-the-fly by a user or scheduled to change automatically according to the time and day.

Polling

In polling mode, the agent intentionally closes the primary connection unless there exists an active stream. On a configurable schedule, the agent reconnects to a server, flushes any cached data, and checks for any new work items. After executing all available work items, the primary connection is closed again.

The agent may attempt a spontaneous connection outside of the regular schedule if an internal agent process triggers it.

Plugins

Agents can optionally support plugins to enhance functionality beyond the standard feature set. Upon initial connection, the agent provides a list of plugin versions that it has loaded. The server responds with a list of plugin archives that the agent should install.

Standard Feature Set

The standard feature set is the minimum amount of functionality an agent implementation must provide.

AgentMetadata

Upgrades

There are two ways to upgrade the agent:

- automatically by sending the update command to the server,
- manually by generating a new installer and executing it on the agent

Manual Upgrade

A manual upgrade is triggered when an installer is executed on the agent and the relevant base directory is already populated with an installation. If the agent is not running, the installer will overwrite the base directory and install itself. Any data that the agent has cached but not sent to the server will be lost!

Advantages

- This is the only way to upgrade if the agent can no longer connect to the server

Disadvantages

- Manual intervention required
- Cached data may be lost

Automated Upgrade

If the agent is connected to a server, it can be upgraded remotely. This will cause the server to fetch the agent configuration, generate a new installer, and transfer it to the agent. The agent then executes the new installer and terminates.

Container Resident

Boot Agent

Boot agents are a special subset of agent instances that run in a custom UEFI boot environment rather than on a host's OS.

Configuration

The boot agent accepts the following specific configuration options.

Property	Default	Description
`s7s.agent.boot.network.interface`		A specific network interface to use
		Whether the boot agent will attempt to obtain

`s7s.agent.boot.network.dhcp`	true	a DHCP address
`s7s.agent.boot.network.ip`		A static IPv4 that will be used instead of DHCP
`s7s.agent.boot.network.netmask`		The network mask for the static IPv4 address

Boot Environment

The boot agent environment occupies an entire partition on the host and contains:

- A bootloader
- A Linux kernel
- `efibootmgr` executables
- Boot agent executables
- All library dependencies

Startup

The boot environment does not provide a shell and merely executes the boot agent on startup.

The boot agent immediately spawns an infinite connection loop for the configured server.

Installation

Agent instances are capable of installing a boot agent on their host machine if installed with sufficient permissions.

downloads a boot agent image to the partition and writes a new configuration according to parameters of the host OS or overrides provided by a client.

Once the partition is written, the agent adds it to the end of the existing EFI boot order.

Host Media

In order to install the boot agent to the existing disk, a new partition of at least 250 MiB must first be created manually. The agent installs an additional bootloader to the existing ESP.

Removable Media

The agent can erase an existing disk and create a new ESP and boot agent partition.

Uninstallation

To uninstall, an agent must remove the boot agent's entry in the EFI boot order and overwrite all data in the boot agent partition.

Reboot into boot agent

Agents can launch the boot agent indirectly by setting the `BootNext` variable to the index of the boot agent EFI entry and rebooting the machine.

Client/Server Messages

Message	Sources	Destinations	Description
RQ_ServerBanner	`client`	`server`	Request the server's banner
RS_ServerBanner	`server`	`client`	Response containing the server's banner
RQ_Logout	`client`	`server`	Request that the current login session be terminated
RS_Logout	`server`	`client`	
RQ_Login	`client`	`server`	Request a new login session

Message Formats

RS_ServerBanner

Field	Type	Requirements	Description
maintenance	bool		Indicates that only superusers will be allowed to login
version	string	5 - 32 characters	The server's version string
message	string	0 - 128 characters	The banner text message
image	bytes	0 - 1 MiB PNG format	The banner image

RS_Logout

Field	Description
LOGOUT_OK	Indicates the logout attempt was successful

RQ_Login

Field	Type	Requirements	Description
username	string	5 - 32 characters	The username
password	string	5 - 32 characters	An unsalted SHA512 hash of the password
token	int32		A TOTP token

RS_Login

Field	Description
LOGIN_OK	Indicates the login attempt succeeded
LOGIN_INVALID_USERNAME	Indicates the supplied username

RQ_CreateUser

Field	Type	Requirements	Description
username	string		
password	string		
email	string		
phone	string		

expiration	int64

RS_CreateUser

Field	Description
CREATE_USER_OK	
CREATE_USER_ACCESS_DENIED	
CREATE_USER_INVALID_USERNAME	
CREATE_USER_INVALID_PASSWORD	
CREATE_USER_INVALID_EMAIL	
CREATE_USER_INVALID_PHONE	

RQ_DeleteUser

Field	Type	Requirements	Description
username	string		

RS_DeleteUser

Field	Description
DELETE_USER_OK	
DELETE_USER_ACCESS_DENIED	

RQ_DeleteGroup

Field	Type	Requirements	Description
name	string		

RS_DeleteGroup

Field	Description
DELETE_GROUP_OK	
DELETE_GROUP_ACCESS_DENIED	

RQ_DeleteListener

Field	Type	Requirements	Description
name	string		

RS_DeleteListener

Field	Description
DELETE_LISTENER_OK	
DELETE_LISTENER_ACCESS_DENIED	

RQ_BuildAgent

Request to build an agent for the given group.

Field	Type	Description

group	string	The group ID
config	core.instance.AgentConfig	The agent configuration
generator_options	GeneratorOptions	Options for the generator component
packager_options	PackagerOptions	Options for the packager component
deployment_options	DeploymentOptions	Options for the deployment component

RS_BuildAgent

null

Field	Description
BUILD_AGENT_OK	0
BUILD_AGENT_FAILED	1

RQ_CreateGroup

null

Field	Type	Description
name	string	The group's name

RS_CreateGroup

null

Field	Description
CREATE_GROUP_OK	0
CREATE_GROUP_FAILED_ACCESS_DENIED	1

RQ_DeleteGroup

null

Field	Type	Description

RS_DeleteGroup

null

Field	Description
DELETE_GROUP_OK	0
DELETE_GROUP_FAILED_ACCESS_DENIED	1

RQ_UpdateGroup

null

Field	Type	Description

RS_UpdateGroup

null

Field	Description
UPDATE_GROUP_OK	0
UPDATE_GROUP_FAILED_ACCESS_DENIED	1

RQ_CreateListener

null

Field	Type	Description
address	string	The listening address
port	int32	The listening port

RS_CreateListener

null

Field	Description
CREATE_LISTENER_OK	0
CREATE_LISTENER_ACCESS_DENIED	1
CREATE_LISTENER_INVALID_PORT	2

RQ_DeleteListener

null

Field	Type	Description

RS_DeleteListener

null

Field	Description
DELETE_LISTENER_OK	0
DELETE_LISTENER_ACCESS_DENIED	1

RQ_UpdateListener

null

Field	Type	Description

RS_UpdateListener

null

Field	Description
UPDATE_LISTENER_OK	0
UPDATE_LISTENER_ACCESS_DENIED	1

RQ_Login

Request a login from the server

Field	Type	Description
username	string	The login username
password	string	The password
totp	int32	Time-based One-Time Password token

RS_Login

null

Field	Description
LOGIN_OK	0
LOGIN_FAILED	1
LOGIN_FAILED_EXPIRED_USER	2
LOGIN_INVALID_USERNAME	3
LOGIN_INVALID_PASSWORD	4
LOGIN_INVALID_TOKEN	5

RQ_Logout

Request that the current user be logged out

Field	Type	Description

RS_Logout

null

Field	Description
LOGOUT_OK	0

RQ_ServerBanner

Request for the server's banner

Field	Type	Description

RS_ServerBanner

Response bearing the server's banner

Field	Type	Description
maintenance	bool	Maintenance mode indicates that only superusers will be allowed to login
version	string	The 3-field version of the server
banner	string	A string to display on the login screen
banner_image	bytes	An image to display on the login screen

RQ_CreateUser

null

Field	Type	Description

username	string	The user's immutable username

RS_CreateUser

null

Field	Description
CREATE_USER_OK	0
CREATE_USER_ACCESS_DENIED	1

RQ_DeleteUser

null

Field	Type	Description

RS_DeleteUser

null

Field	Description
DELETE_USER_OK	0
DELETE_USER_ACCESS_DENIED	1

RQ_UpdateUser

null

Field	Type	Description

RS_UpdateUser

null

Field	Description
UPDATE_USER_OK	0
UPDATE_USER_ACCESS_DENIED	1

Deployer

Deployer instances are responsible for installing, updating, or removing agent and probe instances.

If an existing agent or probe was originally installed by a package manager, it cannot be updated or removed by a deployer.

Instance Configuration

```
{
  "agent_type"   : String(), # The type of agent to install
  "callback"     : {
    "address"    : String(), # The callback address
    "identifier" : String(), # The callback identifier
  },
  "install_dir"  : String(), # The installation's base directory
  "autorecover"  : String(), # Whether the agent can disregard elements of the
config in case of failure
  "autostart"    : Boolean(), # Whether the agent should be started on boot
  "kilo"         : {
    "modules" : [
      {
        "group"       : String(), # The artifact's maven group identifier
        "artifact"    : String(), # The artifact's identifier
        "filename"    : String(), # The artifact's filename
        "version"     : String(), # The artifact's version string
        "hash"        : String(), # The artifact's SHA256 hash
      }
    ]
  }
}
```

Callbacks Connections

If the install/update operation fails, and callbacks are configured, the deployer will establish an encrypted "callback" connection with a server and transfer details on the error.

Instance

The following sections apply to all Sandpolis instances.

Instance Types

Every instance belongs to one of five mutually exclusive *instance types*. There is often more than one implementation in each category.

Instance Type	Description
`server`	A headless application that coordinates interactions among instances in the network
`agent`	A headless application that runs continuously on hosts in the Sandpolis network
`probe`	A headless application that provides strictly read-only data to servers
`client`	A UI application used for managing agents and probes
`deployer`	A headless application that installs or updates agents and probes

Instance Flavors (subtypes)

Each instance type may have multiple implementations (or flavors) to support a variety of use cases. Flavors are identified by a codename and also have a user-friendly "official" name.

Type	Flavor codename	Implementation languages	Official name
`server`	`vanilla`	Java	Server
`client`	`lifegem`	Java, Kotlin	Desktop Client
`client`	`ascetic`	Java	Terminal Client
`client`	`lockstone`	Swift	iOS Client
`client`	`brightstone`	JavaScript	Web Client
`agent`	`kilo`	Java	Agent
`agent`	`micro`	Rust	Native Agent
`agent`	`boot`	Rust	Boot Agent
`probe`	`nano`	C++	Probe
`deployer`	`rust`	Rust	Agent deployer (Rust)
`deployer`	`java`	Java	Agent deployer (Java)

Instance Configuration

```
{
  "runtime" : {
    "residency": Boolean(default=False), # Whether the instance is running in a container
  }
}
```

Build Metadata

```
{
  "build_platform"   : String(), # The build platform
  "build_timestamp"  : Number(), # The build timestamp
  "instance_version" : String(), # The instance's version
  "gradle_version"   : String(), # The Gradle version
  "java_version"     : String(), # The Java version
  "kotlin_version"   : String(), # The kotlin version is applicable
  "rust_version"     : String(), # The rust version if applicable
  "dependencies"     : [
    String(), # The artifact coordinates in G:A:V format
  ]
}
```

Data Model

There are three layers in the Sandpolis data model. Of which, client implementations are required to support at least two (ST and OID layers).

The ST Layer

The State Tree layer is the lowest layer and is concerned with storage and persistence. Every instance maintains a global tree called the "ST Tree". The tree is seldomly manipulated directly. Instead, higher layers make changes to the ST Tree on behalf of consumers.

The ST tree is composed of two components: `Attributes` and `Documents`.

Attributes

Attributes contain data of a specific type and meaning. All data in the ST tree is stored in attributes.

Retention

The history of an attribute can optionally be recorded with *tracked attributes*.

AttributeChangedEvent

Documents

Documents are a set of attributes and sub-documents.

DocumentAddedEvent

Indicates that a document has been added to the tree. No futher events will be fired for all children of the added document as a direct result of the addition.

DocumentRemovedEvent

Indicates that a document has been removed from the tree.

Entanglement

A concept that exists at the ST layer is **entanglement**: ST trees that reside on remote instances can synchronize their state. The relation can be bidirectional or unidirectional and last as long as necessary. All changes to the source of an entanglement pair will be propagated to the destination in real-time.

Snapshots and Merging

The VST Layer

The OID Layer

Every node in a ST Tree is uniquely identified by an OID.

Path

The OID path is a sequence of / separated strings that describe how to reach the corresponding node from the root node.

Elements of the path are called *components* which may consist of any number of alphanumeric characters and underscores. If a component equals the wildcard character (*), then the OID corresponds to all possible values of that component and is known as a *generic* OID. If an OID is not generic, then it's *concrete*.

Namespace

OIDs have a namespace string that identifies the module that provides the OID. This allows modules to define OIDs without the possibility of collisions. The namespace string must equal the name of the module that defines an OID.

Namespace notation is to prefix the namespace string and a `:`, similar to the protocol section of a URI:

```
com.sandpolis.plugin.example:/profile/*/example
```

Temporal Selector

In order to select historic values of an attribute, concrete OIDs may include a timestamp range selector or an index selector.

Timestamp Selector

To select all values within an arbitrary timestamp range, specify the inclusive start and end epoch timestamps separated by a `..` in parenthesis. If either timestamp is omitted, then the range is extended to the most extreme value possible.

```
/profile/ba4412ea-1ec6-4e76-be78-3849d2196b52/example(1628216870..1628216880)
```

Index Selector

To select an arbitrary amount of values, specify inclusive start and end indicies separated by a `..` in square brackets. If either index is omitted, then the range is extended to the most extreme value possible. Index 0 is the oldest value.

```
/profile/ba4412ea-1ec6-4e76-be78-3849d2196b52/example[2..7]
```

To select one value, omit the range specifier entirely:

```
/profile/ba4412ea-1ec6-4e76-be78-3849d2196b52/example[1]
```

Message Format

The Sandpolis network protocol is based on [protocol buffers (https://github.com/protocolbuffers/protobuf)](https://github.com/protocolbuffers/protobuf).

Request/Response messages

Request messages are named with a RQ_ prefix.

Event messages

Event messages are named with a EV_ prefix and are typically part of a stream.

RQ_Ping

Ping request.

Field	Type	Description

RS_Ping

Ping response.

Field	Type	Description

RQ_PluginOperation

Request: Attempt a plugin operation.

Field	Type	Description
operation	PluginOperation	null
plugin_config	core.instance.PluginConfig	null

RS_PluginSync

null

Field	Type	Description
plugin_config	core.instance.PluginConfig	null

RQ_ArtifactDownload

A request to download an artifact.

Field	Type	Description
coordinates	string	The coordinates of the artifact to download
location	bool	Whether it's acceptable to receive the external location of the artifact only

RS_ArtifactDownload

The download response bearing the artifact binary or the artifact's address.

Field	Type	Description
binary	bytes	The artifact binary
url	string	The artifact's HTTP location
coordinates	string	The artifact's maven central coordinates

RQ_PluginInstall

A request to install a server plugin.

Field	Type	Description

plugin_binary	bytes	The plugin binary
plugin_url	string	The plugin's HTTP location
plugin_coordinates	string	The plugin's maven central coordinates

RS_PluginInstall

null

Field	Description
PLUGIN_INSTALL_OK	0
PLUGIN_INSTALL_FAILED	1
PLUGIN_INSTALL_FAILED_DISABLED	2
PLUGIN_INSTALL_FAILED_CERTIFICATE	3
PLUGIN_INSTALL_INVALID	4

RQ_STStream

null

Field	Type	Description
stream_id	int32	null
permanent	bool	null
oid	string	null
whitelist	string	null
direction	Direction	null
update_period	int32	null

RS_STStream

null

Field	Description
ST_STREAM_OK	0
ST_STREAM_INVALID	1
ST_STREAM_FAILED	2

EV_STStreamData

null

Field	Type	Description
oid	string	The object's relative OID
removed	bool	Whether the object corresponding to the OID was removed
value_type	ValueType	The attribute value type
timestamp	int64	The timestamp associated with the attribute value
bytes	bytes	null
bytes_array	bytes	null
string	string	null
string_array	string	null

integer	int32	null
integer_array	int32	null
long	int64	null
long_array	int64	null
boolean	bool	null
boolean_array	bool	null
double	double	null
double_array	double	null

Networking Module

Sandpolis Protocol

Sandpolis uses a custom binary protocol based on [protocol buffers (https://developers.google.com/protocol-buffers)](https://developers.google.com/protocol-buffers) for all inter-instance network communications. By default, the server listens on TCP port **8768**.

Most communication happens over TCP connections among instances and the server, but the server can also coordinate direct TCP or UDP "sessions" between any two instances that need to transfer high-volume or low-latency data.

Streams

Many operations require real-time data for a short-lived or long-lived session.

All streams have a *source* and a *sink* and can exist between any two instances (a stream where the source and sink reside on the same instance is called a *local stream*). The source's purpose is to produce *stream events* at whatever frequency is appropriate for the use-case and the sink's purpose is to consume those stream events.

Multicasting

Stream sources can push events to more than one sink simultaneously. This is called multicasting and can save bandwidth in situations where multiple users request the same resource at the same time.

Messages

Message	Sources	Destinations	Description
RQ_Session	`client, agent`	`server`	
RS_Session	`server`	`client,agent`	
RQ_AddConnection	`client`	`server`	
RQ_CoordinateConnection	`server`	`client, agent`	
EV_NetworkChanged	`server`	`client,`	Indicates that some node in the network

		agent	has changed in connection status
RQ_InstallPlugin	client	server	Request that a new plugin be installed
RQ_STStream	client, agent	server	Request a new state tree sync stream
EV_STStreamData	server		
RQ_CloseStream			Request that a stream be closed

Session

Clients and agents maintain an ephemeral session which consists of a session identifier and authentication state.

Session identifiers are 4-byte unsigned integers that have the instance type and instance flavor encoded in them.

```
 0         1         2           3
 012345678901234567890123 45678 901
[        Base CVID       | FID |IID]
```

RQ_CoordinateConnection

Request that the receiving instance establish a new connection to the given host. The receiver should attempt the connection as soon as possible.

Field	Type	Requirements	Description
host	string	An IP address	The connection host
port	int32	A valid port number	The connection port
protocol	string	`tcp` or udp	The connection protocol
encryption_key	bytes	64 bytes	The initial encryption key for the new connection

RQ_STStream

Field	Type	Requirements	Description
stream_id	int32		
oid	string		
whitelist	repeated string		
direction	string	"upstream", "downstream", or "bidirectional"	

RQ_Session

Request that a new session be created. Any previous sessions associated with the instance are invalidated.

Field	Type	Description
instance_uuid	string	The UUID of the requesting instance
instance_type	core.instance.InstanceType	The instance type of the requesting instance
instance_flavor	core.instance.InstanceFlavor	The instance flavor of the requesting instance

RS_Session

Respond to a session request with a successful result.

Field	Type	Description
instance_sid	int32	A SID for the requesting instance
server_sid	int32	The SID of the server
server_uuid	string	The UUID of the server

RQ_DirectConnection

Request the server for a new direct connection.

Field	Type	Description
cvid	int32	The requested node
type	LinkType	The type of connection
port	int32	An optional listener port. If specified, the requested node will attempt
a connection on this port. Otherwise, the server will coordinate the connection.		

RS_DirectConnection

Server response to a connection request.

Field	Type	Description
request	RQ_Connection	The connection request

RQ_Connection

Request the instance to establish a new connection.

Field	Type	Description
host	string	The host
port	int32	The port
type	LinkType	The link type

EV_EndpointClosed

Indicates that a connection has been lost.

Field	Type	Description
cvid	int32	The node that has lost connection

EV_NetworkChange

Indicates that the network has been updated.

Field	Type	Description
node_added	NodeAdded	null
node_removed	NodeRemoved	null
connection_added	LinkAdded	null
connection_removed	LinkRemoved	null

RQ_StopStream

null

Field	Type	Description
id	int32	The stream ID of the stream to stop

RS_StopStream

null

Field	Description
STOP_STREAM_OK	0
STOP_STREAM_INVALID	1

Probe Instance

Probes are similar to agents, but are only allowed to egress data to a server. They cannot receive messages, so their configuration is immutable unless the system also runs an agent capable of managing probe instances.

Probe instances are designed to be extremely lightweight and run on almost any hardware.

Probe Configuration

```
{
  "network" : {
    "server_address" : [
      String() # An IP address or DNS name with port info
    ],
    "connection": {
      "timeout" : Number(), # The connection timeout in milliseconds
      "interval" : Number() # The connection poll interval in milliseconds
    }
  },
  "collectors": {
    "/memory/total_used"
  }
}
```

Connection Mode

The only connection mode supported by probes is the *polling* mode. On a configurable schedule, the probe reconnects to a server, flushes any cached data, and closes the connection.

The probe may attempt a spontaneous connection outside of the regular schedule at any time.

If a server is overloaded, the probe's connection attempt may not be accepted. In this case, the probe may choose another server or wait and try again later.

Connection Security

Unlike agents, probes do not use TLS. Rather, outgoing messages are encrypted with AES256 using a sesion key derived from a master key embedded in the configuration.

No two connections will use the same session key, so compromising it will only yield an attacker decrypted data for the remainder of the connection.

Server Instance

Every Sandpolis network must include one server instance at minimum. Servers are responsible for coordinating interactions among instances and persisting data.

Listening port

The Sandpolis server listens on TCP port **8768** by default, but can be configured to listen on a different port or multiple ports concurrently.

Instance Configuration

```
{
  "storage" : {
    "provider" : String(default="ephemeral"), # The database storage provider
    "mongodb" : {
      "host"     : String(), # The address of the mongodb host
      "username" : String(), # The mongodb user's username
      "password" : String(), # The mongodb user's password
    }
  },
  "geolocation" : {
    "service"    : String(values=["ip-api.com", "keycdn.com"], default="ip-
api.com"), # The name of the geolocation service to use
    "key"        : String(), # The service API key
    "expiration" : Number(), # The cache timeout in hours
  }
}
```

First Start

If the server starts with ephemeral storage or an empty database, the server enters "first start" mode. This mode has the following implications:

Default admin password

The admin password will be randomized and printed in the server log. All clients are required to force users to change the admin password and setup multi-factor authentication before proceeding after the first login.

Connection Blocking

The server will refuse connections from IP addresses on a configurable blocklist or those that trigger the global rate-limiting policy.

IP address on a configurable whitelist are exempt from rate-limiting.

Permissions

All user accounts are subject to a set of permissions controlling what server operations are authorized. The inital admin user has complete and irrevocable permissions. By default, additional user accounts are created without permissions and consequently are allowed to do almost nothing.

Permissions list

Permission	Description
`server.generate`	Rights to use the generator
`server.users.list`	Right to view usernames and permissions of all other users
`server.users.create`	Right to create new users (of lesser or equal permissions)
`server.net.view`	Right to open the network control panel
`server.listener.create`	Right to create a new listener on the server
`server.listener.list`	Right to view all listeners on the server
`server.group.create`	Right to create a new authentication group on the server
`server.group.list`	Right to view all authentication groups on the server
`agent.system.power`	Right to shutdown, reboot, etc the agent

Agent Groups

Agent groups are sets of agents that share one or more authentication schemes. Every group has exactly one owner and zero or more (user) members.

Password Authentication Scheme

After establishing a connection, agents may present an unsalted SHA512 hash of a password entered by the user to the server. The server compares the password to each agent group until it finds a match. If a match is found, the agent is becomes authenticated to the matching agent group. Otherwise, the connection is closed if more than 5 attempts were made on that connection.

Since a user must type the password manually, the server will attempt to configure the certificate authentication scheme for all subsequent connections.

Token Authentication Scheme

The agent may provide an 8 character alphanumeric time-based token periodically generated by the server from an agent group's secret key. Since a user must type the token in manually, the server will attempt to configure the certificate authentication scheme for all subsequent connections.

Certificate Authentication Scheme

The agent may provide an X509 "client" certificate signed by an agent group's secret key during the initial connection attempt. If the agent certificate was found to be valid, the connection is automatically authenticated without any additional message exchanges.

Agent Certificate Expiration

The default lifetime for an agent certificate is six months. The following section implies an agent must connect to a server at least once every 1.5 months otherwise it loses its ability to authenticate.

Agent Certificate Renewal

Once 75% of the lifetime of an agent certificate elapses, the server attempts to issue a new certificate and installs it on the agent.

Agent Generators

A `Generator` is a routine which produces some installation artifact according to the parameters set out in an authentication group. The installation artifact can then be used to install an agent on a remote system.

Deployers

On execution, deployers set up the agent base directory according to its configuration and executes the agent. If the target directory already contains an installation, the old installation is entirely overwritten.

Packager

A packager is responsible for creating a deployer binary according to the parameters set out in an authentication group.

Distributor

A distributor is responsible for transferring and executing generated deployer artifacts to remote systems.

SSH Distributor

The SSH deployer first determines the remote system type and invokes an appropriate packager to generate an installer. The installer is then transferred to the remote host and executed.

Server/Agent Messages

Message	Sources	Destinations	Description
RQ_AuthSession	`agent`	`server`	Request to authenticate an agent session
RQ_RefreshAuthCertificate	`server`	`agent`	Request to refresh an agent's authentication certificate
RQ_AgentMetadata	`server`	`agent`	Request agent metadata
RS_AgentMetadata	`agent`	`server`	Response containing agent metadata
RQ_FindBootAgents	`client`, `server`	`agent`	Request to locate all installed boot agents
RS_FindBootAgents	`agent`	`client`, `server`	Response listing boot agent installations
RQ_FindBootAgentCandidates	`client`, `server`	`agent`	Request candidate partitions and devices that may be suitable for a boot agent installation
RS_FindBootAgentCandidates	`agent`	`client`, `server`	Response listing boot agent installation candidates
RQ_InstallBootAgent	`client`, `server`	`agent`	Request a boot agent be installed on the system
RQ_UninstallBootAgent	`client`, `server`	`agent`	Request a boot agent be uninstalled from the system
RQ_LaunchBootAgent	`client`, `server`	`agent`	Request that the boot agent be started
RQ_ChangePowerState	`client`, `server`	`agent`	Request that the agent alter its power state

Agent Authentication Messages

RQ_AuthSession

Field	Type	Requirements	Description
password	string	8 - 64 characters	The password text
token	string	8 characters	The authentication token

RQ_RefreshAuthCertificate

Field	Type	Requirements	Description

General Messages

RQ_AgentMetadata

RS_AgentMetadata

Field	Type	Requirements	Description
hostname	string	0 - 64 characters	The agent's network hostname

os	OsType		The agent's OS family
arch	string		The agent's CPU architecture
boot_agent_detected	bool		Whether a boot agent was detected on the system

Boot Agent Messages

RQ_FindBootAgents

Field	Type	Requirements	Description

RS_FindBootAgents

Field	Type	Requirements	Description

RQ_InstallBootAgent

Field	Type	Requirements	Description
partition_uuid	string		The UUID of the target partition
device_uuid	string		The UUID of the target device
interface_mac	string		The MAC address of the network interface to use for connections
use_dhcp	bool		Whether DHCP will be used
static_ip	string		A static IP address as an alternative to DHCP
netmask	string		The netmask corresponding to the static IP
gateway_ip	string		The gateway IP

RS_InstallBootAgent

Field	Description
INSTALL_BOOT_AGENT_OK	
INSTALL_BOOT_AGENT_ACCESS_DENIED	

RQ_UninstallBootAgent

Field	Type	Requirements	Description
target_uuid	string		The UUID of the partition containing the boot agent

RS_UninstallBootAgent

Field	Description
UNINSTALL_BOOT_AGENT_OK	
UNINSTALL_BOOT_AGENT_ACCESS_DENIED	

RQ_LaunchBootAgent

Field	Type	Requirements	Description
target_uuid	string		The UUID of the partition containing the boot agent

RS_LaunchBootAgent

Field	Description
LAUNCH_BOOT_AGENT_OK	
LAUNCH_BOOT_AGENT_ACCESS_DENIED	
LAUNCH_BOOT_AGENT_FAILED	

RQ_ChangePowerState

Field	Type	Requirements	Description
new_state	string		The desired power state

RQ_AgentMetadata

Request metadata from an agent.

Sources : server
Destinations : agent
Response : RS_AgentMetadata

Field	Type	Description

RS_AgentMetadata

Response containing agent metadata.

Sources : agent
Destinations : server
Request : RQ_AgentMetadata

Field	Type	Description
hostname	string	The network hostname
install_directory	string	The install directory
os	core.foundation.OsType	The OS family
boot_agent_detected	bool	Whether a boot agent was detected on the system

RQ_LaunchBootAgent

Request that the boot agent be started.

Sources : client, server
Destinations : agent
Response : Outcome

Field	Type	Description
target_uuid	string	The UUID of the partition containing the boot agent

RQ_UninstallBootAgent

Request a boot agent be uninstalled from the system.

Sources : client, server
Destinations : agent
Response : Outcome

Field	Type	Description
target_uuid	string	The UUID of the partition containing the boot agent

RQ_FindBootAgents

Request to locate all installed boot agents.

Sources : client, server
Destinations : agent
Response : RS_FindBootAgents

Field	Type	Description

RS_FindBootAgents

Response listing boot agent installations.

Sources : client, server
Destinations : agent
Request : RQ_FindBootAgents

Field	Type	Description
installation	BootAgentInstallation	null

RQ_InstallBootAgent

Request a boot agent be installed on the system.

Sources : client, server
Destinations : agent
Response : Outcome

Field	Type	Description
partition_uuid	string	The UUID of the target partition
device_uuid	string	The UUID of the target device
interface_mac	string	The MAC address of the network interface to use for connections
use_dhcp	bool	Whether DHCP will be used
static_ip	string	A static IP address as an alternative to DHCP
netmask	string	The netmask corresponding to the static IP
gateway_ip	string	The gateway IP

RQ_FindBootAgentCandidates

Request candidate partitions and devices that may be suitable for a boot agent installation.

Sources : client, server
Destinations : agent
Response : RS_FindBootAgentCandidates

Field	Type	Description

RS_FindBootAgentCandidates

Response listing boot agent installation candidates.

Sources : agent
Destinations : client, server
Request : RQ_FindBootAgents

Field	Type	Description
device_candidate	DeviceCandidate	null
partition_candidate	PartitionCandidate	null

RQ_AuthSession

Request to authenticate the current session.

Sources : agent
Destinations : server
Response : Outcome

Field	Type	Description
password	string	The group password
token	string	The authentication token

RS_AuthSession

null

Field	Description
AUTH_SESSION_OK	0
AUTH_SESSION_FAILED	1

RQ_RefreshAuthCertificate

Request to refresh an agent's authentication certificate.

Sources : server
Destinations : agent
Response : Outcome

Field	Type	Description
certificate	bytes	The new "client" certificate

Boot Agent

Standard Agent

This JVM-based agent implementation has the most features and is consequently the most resource intensive. It's suitable when running on modern hardware where memory/storage is abundant.

Installation Types

There are three ways to install the agent, each with advantages and disadvantages.

Package Manager Installation

This option makes installation itself easy, but requires some effort to configure the agent to connect to a server. This installation type also means that upgrades must only occur through the package manager.

Filesystem Layout

Pacman (Arch Linux)

Path	Description
`/usr/lib/systemd/system/sandpolis-agent.service`	Systemd service definition
`/usr/bin/sandpolis-agent-config`	Configuration executable
`/usr/bin/sandpolis-agent`	Main executable
`/usr/share/java/sandpolis-agent/lib`	`LIB` location
`/usr/share/java/sandpolis-agent/plugin`	`PLUGIN` location
`/var/lib/sandpolis-agent/data`	`DATA` location

Apk (Alpine)

Deployer Installation

This option requires no explicit configuration, but the *deployer* executable must be transferred to the host and invoked explicitly.

Filesystem Layout

Deployers install everything under a single *base directory* and create symbolic links in certain locations outside of the base directory.

Platform	Default base directory path
Linux	`/opt/sandpolis-agent`
Windows	
macOS	

Docker Installation

This option is easy to deploy and configure, but can limit the functionality of the agent in some cases.

Certain elements of the host filesystem may be optionally mounted into the container.

Native Agent

This agent implementation is designed for performance critical deployments. It has fewer features than the standard agent, but requires significantly less resources.

Release artifacts

Identifier	Type
agent-linux-amd64	ELF
agent-linux-i686	ELF
agent-linux-aarch64	ELF
agent-linux-armv7	ELF
agent-windows-amd64	PE
agent-windows-i686	PE
agent-macos-amd64	Mach-O
agent-macos-aarch64	Mach-O
bootagent-linux-amd64	ELF
bootagent-linux-i686	ELF
bootagent-linux-aarch64	ELF

Web Client

Desktop Client

The desktop client is based on the JVM.

Installation Types

There are two ways to install the client, each with advantages and disadvantages.

Package Manager Installation

Filesystem Layout

Chocolately (Windows)

Path	Description
`C:/ProgramData/Microsoft/Windows/Start Menu/Programs`	Desktop shortcut

Docker Installation

This option is easy to deploy, but only works on Linux with X11.

Server

Installation Types

There are two ways to install the server with the recommended approach being Docker.

Package Manager Installation

Filesystem Layout

Pacman (Arch Linux)

Path	Description
`/usr/lib/systemd/system/sandpolis-server.service`	Systemd service definition
`/usr/share/java/sandpolis-server/lib`	`LIB` location
`/usr/share/java/sandpolis-server/plugin`	`PLUGIN` location
`/var/lib/sandpolis-server/data`	`DATA` location

Docker Installation

Desktop Plugin

The desktop plugin provides remote desktop and clipboard integration.

Messages

Message	Sources	Destinations
RQ_DesktopList	`client`	`agent`
RS_DesktopList	`agent`	`client`
RQ_DesktopStream	`client`	`agent`
EV_DesktopInput	`client`	`agent`
EV_DesktopOutput	`agent`	`client`
RQ_Screenshot		
RS_Screenshot		

RQ_DesktopList

Request for a listing of available desktops.

Field	Type	Requirements	Description

RS_DesktopList

Response containing all available desktops.

Field	Type	Requirements	Description
desktop::name	string		The desktop name
desktop::width	int32		The desktop width in pixels
desktop::height	int32		The desktop height in pixels

RQ_DesktopStream

Request a new desktop stream be established.

Field	Type	Requirements	Description
stream_id	int32		
desktop_uuid	string		
capture_mode	string	`poll` or `hook`	
color_mode	string	`rgb888`, `rgb565`, or `rgb332`	
compression_mode	string	`zlib`	
scale_factor	double		

EV_DesktopInput

`EV_DesktopInput` contains key, mouse, and clipboard data.

Field	Type	Requirements	Description
key_pressed	string		
key_released	string		
key_typed	string		
pointer_pressed	int32		
pointer_released	int32		
pointer_x	int32		
pointer_y	int32		
scale_factor	double		
clipboard	string		

EV_DesktopOutput

`EV_DesktopOutput` contains pixel data and clipboard data.

Field	Type	Requirements	Description
width	int32		The width of the destination block in pixels
height	int32		The height of the destination block in pixels
dest_x	int32		The X coordinate of the destination block's top left corner
dest_y	int32		The Y coordinate of the destination block's top left corner
source_x	int32		The X coordinate of the source block's top left corner
source_y	int32		The Y coordinate of the source block's top left corner
pixel_data	bytes		The pixel data encoded according to the session's parameters
clipboard	string		Clipboard data

RQ_DesktopList

Request for a listing of available desktops

Field	Type	Description

RS_DesktopList

Response containing all available desktops

Field	Type	Description
desktop	Desktop	null

RQ_DesktopStream

null

Field	Type	Description
stream_id	int32	The requested stream ID
desktop_uuid	string	The desktop to capture

RS_DesktopStream

null

Field	Description
DESKTOP_STREAM_OK	0

EV_DesktopInput

null

Field	Type	Description
key_pressed	string	
key_released	string	
key_typed	string	
pointer_pressed	PointerButton	
pointer_released	PointerButton	
pointer_x	int32	The X coordinate of the pointer
pointer_y	int32	The Y coordinate of the pointer
scale_factor	double	Scale factor
clipboard	string	Clipboard data

EV_DesktopOutput

null

Field	Type	Description
width	int32	The width of the destination block in pixels
height	int32	The height of the destination block in pixels
dest_x	int32	The X coordinate of the destination block's top left corner
dest_y	int32	The Y coordinate of the destination block's top left corner
source_x	int32	The X coordinate of the source block's top left corner
source_y	int32	The Y coordinate of the source block's top left corner
pixel_data	bytes	The pixel data encoded according to the session's parameters
clipboard	string	Clipboard data

RQ_CaptureScreenshot

null

Field	Type	Description
desktop_uuid	string	The desktop to capture

RS_CaptureScreenshot

null

Field	Type	Description
data	bytes	null

Device Plugin

The device plugin extends management functionality out to agent-less devices.

Subagents

Subagents are devices that do not have Sandpolis agent software installed, but are instead managed via a third-party protocol such as SSH, IPMI, or SNMP from an instance called the *gateway*. The gateway instance for a subagent may be an independent agent or a server.

Communicators

A subagent communicates to its gateway instance over one of the following well-known protocols. Since subagents must accept incoming connections, the gateway instance usually must reside on the same network segment.

WOL

The WOL communicator is able to send Wake-on-LAN magic packets to listening devices.

SSH

The SSH communicator establishes SSH sessions with remote devices.

Property	Description
`ssh.username`	The SSH username
`ssh.password`	The SSH password
`ssh.private_key`	The SSH private key

IPMI

The IPMI communicator runs IPMI commands on remote devices.

Property	Description
`ipmi.username`	The IPMI username
`ipmi.password`	The IPMI password

SNMP

The SNMP communicator reads and writes standard MIBs on remote devices.

Property	Description
`snmp.version`	The SNMP version
`snmp.community`	The SNMP community string if version < 3
`snmp.privacy.type`	
`snmp.privacy.secret`	
`snmp.authentication.type`	
`snmp.authentication.secret`	

Messages

Message	Sources	Destinations
RQ_FindSubagents	`client`, `server`	`agent`
RS_FindSubagents	`agent`	`client`, `server`
RQ_RegisterSubagent	`client`	`server`
RQ_ConfigureSubagent	`server`	`agent`
RQ_IpmiCommand		
RQ_SnmpWalk		
RQ_SshCommand		
RQ_SendWolPacket		
RS_SendWolPacket		

RQ_FindSubagents

Scan the local network (if it's smaller than a /16) for devices that may be candidate subagents.

- For the `ssh` communicator, a TCP connection is attempted on port 22
- For the `snmp` communicator, probes are sent via UDP port 161
- For the `ipmi` communicator, probes are sent via UDP port 623

Field	Type	Requirements	Description
gateway_uuid	`string`		The UUID of a gateway instance
communicators	`repeated string`	`ssh`, `snmp`, `ipmi`	The communicator types to search

RS_FindSubagents

Field	Type	Requirements	Description
ssh_device::ip_address	`string`	IPv4 or IPv6 address	Device IP address
ssh_device::mac_address	`string`	MAC address	Device MAC address
ssh_device::fingerprint	`string`		Device SSH fingerprint
snmp_device::ip_address	`string`	IPv4 or IPv6 address	Device IP address
ipmi_device::ip_address	`string`	IPv4 or IPv6 address	Device IP address

RQ_SnmpWalk

Field	Type	Requirements	Description
oid	`string`	An OID string	Request an SNMP walk operation be executed

RS_SnmpWalk

Field	Type	Requirements	Description
data::oid	string		
data::type	string		
data::value	string		

RQ_FindSubagents

null

Field	Type	Description
network	string	If specified, the search will be restricted to the given networks
communicator	CommunicatorType	If specified, the search will be restricted to the given communicator types

RS_FindSubagents

null

Field	Type	Description
ssh_device	SshDevice	null
snmp_device	SnmpDevice	null
ipmi_device	IpmiDevice	null
http_device	HttpDevice	null
onvif_device	OnvifDevice	null
rtsp_device	RtspDevice	null
wol_device	WolDevice	null

RQ_RegisterSubagent

null

Field	Type	Description
ip_address	string	null
mac_address	string	null
gateway_uuid	string	The uuid of the gateway instance

RS_RegisterSubagent

null

Field	Description
REGISTER_SUBAGENT_OK	0

RQ_IpmiCommand

Request an IPMI command be executed

Field	Type	Description

command	string	The IPMI command

RQ_SnmpWalk

Request an SNMP walk operation be executed

Field	Type	Description
oid	string	The OID to retrieve

RS_SnmpWalk

Response containing the result of a walk operation

Field	Type	Description
data	Data	null

RQ_SendWolPacket

null

Field	Type	Description

RS_SendWolPacket

null

Field	Description

Filesystem Plugin

The filesystem plugin exposes agent and client filesystems to Sandpolis.

Mounting

Remote filesystems may be mounted on clients or agents with FUSE. Once established, the mount is permanent until explicitly closed by the user.

By default, the entire filesystem is mounted, but can be configured to only expose a particular subtree.

Agent mount

An agent's filesystem may be mounted to a mountpoint on another agent or on a client's machine.

Client mount

A client's filesystem may be mounted to a mountpoint on an agent's machine.

Permissions list

Permission	Description

`agent.fs.mount`	Rights to mount an agent's filesystem
`agent.fs.read`	Rights to read an agent's filesystem
`agent.fs.write`	Rights to write an agent's filesystem
`client.fs.mount`	Rights to mount the current client's filesystem
`client.fs.read`	Rights to read the current client's filesystem
`client.fs.write`	Rights to write the current client's filesystem

Messages

Message	Sources	Destinations	Description
RQ_DirectoryListing	`client`	`agent`	
RS_DirectoryListing	`agent`	`client`	
RQ_MountDirectory	`client`	`agent`	
RQ_UnmountDirectory	`client`	`agent`	
RQ_FileStats	`client`	`agent`	
RS_FileStats			
RQ_DeleteFile	`client`	`agent`	
RQ_DirectoryStream	`client`	`agent`	
EV_DirectoryChange	`agent`	`client`	

EV_FileListing

Updates to a directory listing.

Field	Type	Description
fmid	int32	null
path	string	The directory's absolute path
listing	FileListlet	Listing updates

RQ_FileListing

Request a directory listing.

Field	Type	Description
path	string	null
options	FsHandleOptions	null

RS_FileListing

Response to a directory listing request.

Field	Type	Description
path	string	The directory's absolute path
listing	FileListlet	The directory listing

RQ_FileInfo

Request for detailed file information.

Field	Type	Description
file	string	An absolute path

RS_FileInfo

Response to a file info request.

Field	Type	Description
local_icon	bytes	The file's icon on the local platform
path	string	The file's location
name	string	The file's name
ctime	int64	The file's creation time
mtime	int64	The file's modification time
atime	int64	The file's access time
size	int64	The file's size in bytes or number of elements if directory
mime	string	The file's MIME type

RQ_FileDelete

Request for one or more files to be deleted.

Field	Type	Description
target	string	A list of absolute paths to delete

RQ_FsHandle

Request to open a new filesystem handle.

Field	Type	Description
options	FsHandleOptions	null

RS_FsHandle

Response to a handle request.

Field	Type	Description
fmid	int32	The filesystem handle's ID

RQ_CloseFsHandle

Request to close the given handle.

Field	Type	Description
fmid	int32	The filesystem handle to close

Shell Plugin

The shell plugin integrates with various kinds of system shells.

Shell Capability

Label	Shell Name
bash (zsh, ash, dash)	
fish	
powershell	
cmd.exe	
ksh	
csh (tcsh)	

Messages

Message	Sources	Destinations	Description
RQ_ShellStream	`client`	`agent`	Request to start a new shell session
EV_ShellInput	`client`	`agent`	Event containing standard-input to a shell
EV_ShellOutput	`agent`	`client`	Event containing standard-output and standard-error
RQ_FindShells	`client`	`agent`	Request to locate supported shells on the system
RS_FindShells	`agent`	`client`	Response containing supported shell information
RQ_Execute	`client`	`agent`	Request to execute a command snippet in a shell
RS_Execute	`agent`	`client`	Response containing execution results

RQ_FindShells

Field	Type	Requirements	Description

RS_FindShells

Field	Type	Requirements	Description
shell::path	string	Filesystem path	The location of the shell executable
shell::capability	repeated string		A list of shell capabilities

RQ_Execute

Field	Type	Requirements	Description
shell_type			
command	repeated string		The commands to execute
timeout	int32		
ignore_stdout	bool		
ignore_stderr	bool		

RS_Execute

Field	Type	Requirements	Description

EV_ShellInput

Field	Type	Requirements	Description
stdin	bytes	0 - 65535 bytes	The process standard-input
rows_changed	int32		
cols_changed	int32		

EV_ShellOutput

Field	Type	Requirements	Description
stdout	bytes	0 - 65535 bytes	The process standard-output
stderr	bytes	0 - 65535 bytes	The process standard-error

RQ_ShellStream

Field	Type	Requirements	Description
stream_id	int32		
capability	string		
environment	map		
rows	int32		
cols	int32		

RQ_Execute

null

Field	Type	Description
type	Shell	null
command	string	null

RS_Execute

null

Field	Type	Description
exitCode	int32	null
result	string	null

RQ_ListShells

null

Field	Type	Description

RS_ListShells

null

Field	Type	Description
listing	ShellListing	null

RQ_ShellStream

null

Field	Type	Description
id	int32	The desired stream ID
type	Shell	The stream type

environment	string	null
rows	int32	The number of rows to request
cols	int32	The number of columns to request

EV_ShellStream

null

Field	Type	Description
data	bytes	The I/O data
rows	int32	Update the number of rows
cols	int32	Update the number of columns

Snapshot Plugin

The snapshot plugin gives agents the ability to take and apply snapshots, even on filesystems that don't natively support snapshots.

Block-based snapshots

The target device is divided into blocks of variable size (power of 2 only). Each block has a corresponding block hash which is the murmur3 128-bit hash of its content.

File-based snapshots

Partition size considerations

On-premise Server

(1 TiB) = (100 MiB / s) * t

t = 1000 s ~ 16 minutes

Off-premise Server

(1 TiB) = (1 MiB / s) * t

t = 100000 s ~ 27 hours

Server

The server is responsible for storing snapshot data and uploading/downloading it to/from agents.

Snapshot format

Snapshot contents are stored in QEMU qcow2 files on the server. This format is mature and supports useful features like compression and encryption.

Boot Agent

All snapshot read/write operations are run from a boot agent rather than the regular agent. This ensures snapshots are perfectly consistent, but implies some amount of downtime during the operation.

Create Snapshot

If there exist no previous snapshots, the boot agent first determines the appropriate block size for the disk. The agent may take into account the size of the disk or the erase-block size of an SSD, but the block size must be a power of two.

If allowed, the boot agent will wipe the disk's free space before continuing. This can significantly decrease the size of the resulting snapshot because empty blocks are omitted.

If there exists a previous snapshot for the disk, the boot agent receives a stream of block hashes. A single worker thread reads blocks from the disk and compares their hashes against the block hashes retrieved from the server. If the hashes do not match, the block is passed into a send queue to be egressed to the server.

Apply Snapshot

If there exists a previous snapshot for the disk, the boot agent initiates a stream of block hashes. A single worker thread reads blocks from the disk and passes their hashes into a send queue to be egressed to the server.

Simultaneously, the boot agent receives a stream of block data which are placed into a write queue to be written to the device.

Snapshot Messages

Message	Sources	Destinations	Description
RQ_CreateSnapshot	`client`	`server`	Create a new snapshot on a target agent
RQ_ApplySnapshot	`client`	`server`	Apply an existing snapshot on a target agent
RQ_SnapshotStream	`server`	`agent`	Create a new snapshot stream
EV_SnapshotDataBlock	`server`, `agent`	`server`, `agent`	An event containing compressed snapshot data
EV_SnapshotHashBlock	`server`, `agent`	`server`, `agent`	An event containing one or more contiguous block hashes

RQ_CreateSnapshot

Field	Type	Requirements	Description
agent_uuid	string		The target agent's UUID
partition_uuid	string		The target partition's UUID

RQ_ApplySnapshot

Field	Type	Requirements	Description
agent_uuid	string		The target agent's UUID
partition_uuid	string		The target partition's UUID
snapshot_uuid	string		The snapshot's UUID

RQ_SnapshotStream

Field	Type	Requirements	Description
stream_id	int32		The stream's ID
operation	string	"create" or "apply"	The snapshot operation type
partition_uuid	string		The target partition uuid
block_size	int32		The block size in bytes

EV_SnapshotDataBlock

Field	Type	Requirements	Description
offset	int64		The block's offset
data	bytes		The block's contents compressed with zlib

EV_SnapshotHashBlock

Field	Type	Requirements	Description
offset	int64		The offset of the block that the first hash corresponds
hash	repeated bytes		A list of consecutive block hashes

Permissions list

Permission	Description
`agent.snapshot.create`	Rights create new snapshots of agent disks
`agent.snapshot.apply`	Rights apply existing snapshots to agent disks
`server.snapshot.list`	Rights to list existing snapshots stored by the server

Configuration

Property	Default	Description
`s7s.snapshot.storage.provider`	*filesystem*	The storage provider to use
`s7s.snapshot.storage.filesystem.path`		The filesystem path

RQ_CreateSnapshot

Create a new snapshot on a target agent.

Sources : client
Destinations : server
Response : Outcome

Field	Type	Description
agent_uuid	string	The target agent's UUID
partition_uuid	string	The target partition's UUID

RQ_ApplySnapshot

Apply an existing snapshot on a target agent.

Sources : client
Destinations : server
Response : Outcome

Field	Type	Description
agent_uuid	string	The target agent's UUID
partition_uuid	string	The target partition's UUID
snapshot_uuid	string	The snapshot's UUID

RQ_SnapshotStream

Create a new snapshot stream.

Sources : server, agent
Destinations : server, agent
Response : Outcome

Field	Type	Description
stream_id	int32	The stream's ID
operation	string	The snapshot operation type
partition_uuid	string	The target partition uuid
block_size	int32	The block size in bytes

EV_SnapshotDataBlock

An event containing compressed snapshot data.

Sources : server, agent
Destinations : server, agent

Field	Type	Description
offset	int64	null
data	bytes	null

EV_SnapshotHashBlock

An event containing one or more contiguous block hashes.

Sources : server, agent
Destinations : server, agent

Field	Type	Description
offset	int64	The offset of the block that the first hash corresponds
hash	bytes	A list of consecutive block hashes

Update Plugin

The update plugin integrates with system package managers.

Messages

Message	Sources	Destinations	Description
RQ_RemovePackages	`client`	`agent`	
RQ_InstallOrUpdatePackages	`client`	`agent`	

RQ_InstallOrUpgradePackages

null

Field	Type	Description
package	string	null

RQ_RemovePackages

null

Field	Type	Description
package	string	null

RQ_RefreshPackages

null

Field	Type	Description