

Hyades Versioning Support

Revision history

Rev.	Date	Author	Summary of Changes
0.1	August 31, 2004	H.M. Nguyen	First draft

Important note

This is only one chapter of the overall “Hyades Protocol Specification” document. Reviewers are assumed to have read that document prior to this section.

Hyades versioning support

This section describes how Hyades supports versioning. All following components: Hyades clients, agents, the Hyades Collection Engine (HCE) and its plug-in components are all required to be versioned.

The main objectives of versioning support are to:

- Dynamically discover compatibility and supported releases (at run-time)
- Introduce and make changes to each component independently.

Versioning format

Format: **Triplet of integers:**

major-number . minor-number . patch-number

Example:

3.0.1

Versioning Component	Description
Major	Large-scale updates. Any type of change is possible. New features and APIs may be introduced. Incompatibility may occur.
Minor	Is required to maintain source compatibility (compiling without error) and binary compatibility (linking and running successfully) with older minor versions. But this allows the ability to add new functions and deprecate old functions. Deprecated features will not actually be removed until the next major release.
Patch	Changes to the APIs are not allowed. This is intended for function/method implementation changes only and mainly used for bug fixes.

Versioning support scenarios

The following scenarios are supported through versioning:

- There could be multiple HCE instances running on the same machine. Each has its own version.
- Multiple clients with different supported versions can connect and make request to the HCE.
- The HCE can load and manage multiple versions of the same agents at the same time.
- The version of the HCE is independent from the version of its plug-in components.

Versioning policy

Each client, agent and the HCE must have a version number as part of its identity.

The HCE supports the plug-in architecture (e.g. the Transport Layer, Payload Normalizer, and Command Extractor). Each of these must have its own version as well.

The versioning numbers are part of the binary and not available in the configuration XML file. As a result, all versioning checks are done at runtime.

Version Checking

HCE will use run-time checking in all cases. The objective is to loosely couple between specific versions or releases for connected components.

Here is the table of events that triggers versioning check for the HCE.

Versioning Check	Description
At HCE startup time	This is when the HCE establishes its own version and performs the following check: <ul style="list-style-type: none">• All plug-in components (transport layer, payload normalizer, command extractor)• All configured agents to be loaded at startup
At HCE connection time	Whenever the client or the agent makes a connection request to the HCE, part of the exchange data will be the requester's version info and the returned data will be the HCE's current versioning numbers. This info will be included for all connection requests (e.g. Socket or Named Pipe calls)
At agent load time	Whenever the HCE loads a new agent, it will query and validate the agent's versioning numbers.
Using an executable option	This is where the user can find out which version of the HCE they are currently using by simply executing the program with the "-version" option on the command line. For example, raserver -version
At query time	At any given time, the client can always query the HCE or the agent for its version number and perform compatibility check on its own (see "Versionable" interface below).

“Versionable” interface and its command

The following command is supported through the Hyades “Versionable” interface. Clients and agents use this command to find out the current versions of the HCE or the agents.

The client or the agent should also implement this same interface so that its users can query through the same consistent command interface.

It is noted that even though the versioning numbers are exchanged at connection time, this API is still required because the client application itself may not have access or control over the connection.

Command:

`CID_GET_VERSION`

Data:

None

Response:

`CID_OK`

Response Data:

Bytes 0-3	Major number
Bytes 4-7	Minor number
Bytes 8-11	Patch number

Description:

Clients (or agents) may use this command to find out the current version of its target (e.g. the HCE or the agent).