HerdAgent

Project: HerdAgent

This file has been automatically generated. Please do not edit it

API Reference

Herd.Files.Appversion

Class Herd.Files.AppVersion

Source: AppVersion.cs

Methods

`AppVersion BestMatch(List versions)`

- **Summary**

  Returns the best match (assuming versions are ordered by preference) to the local machine's architecture

- **Parameters**

  - *versions*: Input list of app versions

Herd.Files.Experiment

Class Herd.Files.Experiment

Source: Experiment.cs

Methods

`void AddVariable(string variable)`

- **Summary**

  Adds the variable.

- **Parameters**

  - *variable*: The variable.

Herd.Files.Experimentalunit

Class Herd.Files.ExperimentalUnit

Source: ExperimentalUnit.cs

Methods

`public ExperimentalUnit(string name, List appVersions, RunTimeRequirements runTimeRequirements)`

- **Summary**

  Minimal constructor for testing

- **Parameters**

  - *name*:

`public ExperimentalUnit(XmlNode configNode, string baseDirectory, LoadOptions loadOptions)`

- **Summary**

  Constructor used to load the experimental unit from a file

- **Parameters**

  - *configNode*:
  - *baseDirectory*:
  - *loadOptions*:

`bool LogFileExists(string relExplUnitPath, string baseDirectory)`

- **Summary**

  Use this if you want to load an experimental unit only depending on whether the log file exists or not

- **Parameters**

  - *logFileName*:

- **Return Value**

  Use this if you want to load an experimental unit only depending on whether the log file exists or not

`void RequestRuntimeRequirements()`

- **Summary**

  Executes the app to retrive the runtime requirements

`bool IsHostArchitectureCompatible(AppVersion version)`

- **Summary**

  Returns whether an app version is compatible with the host architecture

- **Parameters**

  - *version*: The appversion.

`AppVersion BestHostArchitectureMatch(List appVersions)`

- **Summary**

  Returns the app version from the list that best matches the host architecture

- **Parameters**

  - *appVersions*: The application versions.

- **Return Value**

  Returns the app version from the list that best matches the host architecture

Herd.Files.Experimentbatch

Class Herd.Files.ExperimentBatch

Source: ExperimentBatch.cs

Methods

`int CountExperimentalUnits()`

- **Summary**

  This method counts the number of experimental units loaded. Load() must be called before!!

- **Return Value**

  This method counts the number of experimental units loaded. Load() must be called before!!

`int DeleteLogFiles(string batchFilename)`

- **Summary**

  This method loads the batch file and deletes any log file found

- **Parameters**

- *batchFilename*:

- **Return Value**

  This method loads the batch file and deletes any log file found

`int CountExperimentalUnits(string batchFilename, LoadOptions.ExpUnitSelection selection)`

- **Summary**

  This method loads the experiment batch file and counts the type of experimental units required: All, only finished, or only unfinished

- **Parameters**

  - *batchFilename*:

  - *selection*:

- **Return Value**

  This method loads the experiment batch file and counts the type of experimental units required: All, only finished, or only unfinished

Herd.Files.Function

Class Herd.Files.Function

Source: FunctionLog.cs

Methods

`void GenerateBitmaps()`

- **Summary**

  Generates all the bitmaps for all the samples

Herd.Files.Functionlog

Class Herd.Files.FunctionLog

Source: FunctionLog.cs

Methods

`void Load(string filename)`

- **Summary**

  Loads the specified function log file

- **Parameters**

  - *filename*: The filename

`Function GetFunctionFromId(int id)`

- **Summary**

  Gets a function from an identifier

- **Parameters**

  - *id*: The identifier

- **Return Value**

  Gets a function from an identifier

`Function ReadFunctionDeclaration(BinaryReader binaryReader)`

- **Summary**

  Reads a function declaration from the function log file

- **Parameters**

  - *binaryReader*: The binary reader

- **Return Value**

  Reads a function declaration from the function log file

Herd.Files.Functionsample

Class Herd.Files.FunctionSample

Source: FunctionLog.cs

Methods

`void ReadData(BinaryReader binaryReader, int sizeX, int sizeY)`

- **Summary**

  Reads the data of a function sample with size= sizeX*sizeY

- **Parameters**

  - *binaryReader*: The binary reader
  - *sizeX*: The size x
  - *sizeY*: The size y

`void CalculateValueRange(int sizeX, int sizeY, ref double minValue, ref double maxValue)`

- **Summary**

  Calculates the value range of the function

- **Parameters**

  - *sizeX*: The size x
  - *sizeY*: The size y
  - *minValue*: The minimum value. By reference so that we can use it to calculate the absolute min
  - *maxValue*: The maximum value. By reference so that we can use it to calculate the absolute max

`void GenerateBitmap(int sizeX, int sizeY, double minValue, double maxValue)`

- **Summary**

  This method must be called after reading the sample header and finding the declaration of the function to request the size of the function.

- **Parameters**

  - *binaryReader*:
  - *sizeX*:
  - *sizeY*:

Herd.Files.Log

Class Herd.Files.Log

Source: LogFile.cs

Methods

`bool ReadStep(BinaryReader logReader, int numLoggedVariables)`

- **Summary**

  Reads the data from a single step from the log file

- **Parameters**

  - *logReader*: The log reader
  - *numLoggedVariables*: The number of variables in the log file

- **Return Value**

  Reads the data from a single step from the log file

`void ReadEpisodeHeader(BinaryReader logReader)`

- **Summary**

  Reads the episode header.

- **Parameters**

  - *logReader*: The log reader.

`bool LoadBinaryLog(string LogFileName)`

- **Summary**

  Read the binary log file. To know whether the log information has been succesfully loaded or not, BinFileLoadSuccess can be checked after calling this method.

- **Return Value**

  Read the binary log file. To know whether the log information has been succesfully loaded or not, BinFileLoadSuccess can be checked after calling this method.

Herd.Files.Requirements

Class Herd.Files.Requirements

Source: Requirements.cs

Methods

`void AddRenameRule(string original, string rename)`

- **Summary**

  Rename rules: files that must be stored in the remote machine in a different relative location Example: 64 bit runtime C++ libraries have the same name that 32-bit versions have. In the local machine, 64 bit libraries are in /bin/64, 32 libraries are in /bin, but both must be in the same directory as the .exe using them, so the 64 dll-s must be saved in /bin in the remote machine.

`void CommonInit(XmlNode node)`

- **Summary**

  Common initialization used by sub-classes of Requirements

- **Parameters**

  - *node*:

Herd.Files.Runtimerequirements

Class Herd.Files.RunTimeRequirements

Source: Requirements.cs

Methods

`public RunTimeRequirements(int numCPUCores)`

- **Summary**

  Used only for testing

- **Parameters**

  - *numCPUCores*:

Herd.Network.Herdagent

Class Herd.Network.HerdAgent

Source: HerdAgent.cs

Methods

`public HerdAgent(CancellationTokenSource cancelTokenSource)`

- **Summary**

  HerdAgent class constructor

- **Parameters**

  - *cancelTokenSource*:

`void SendJobResult(CancellationToken cancelToken)`

- **Summary**

  Sends the outputs of the job to the client (Badger)

- **Parameters**

  - *cancelToken*: The cancel token.

`Task WaitForExitAsync(Process process, CancellationToken cancellationToken)`

- **Summary**

  Waits for the process to exit asynchronously

- **Parameters**

  - *process*: The process

  - *cancellationToken*: The cancellation token

- **Return Value**

  Waits for the process to exit asynchronously

Herd.Network.Herdagentinfo

Class Herd.Network.HerdAgentInfo

Source: HerdAgentInfo.cs

Methods

`public HerdAgentInfo(string processorId, int numCPUCores, string architecture, string CUDAVersion, string herdAgentVe`

- **Summary**

  Constructor used for testing purposes

`void AddProperty(string name, string value)`

- **Summary**

  Adds a property to the agent description

- **Parameters**

  - *name*: The name of the property
  - *value*: The value of the property

`string Property(string name)`

- **Summary**

  Gets the value of the specified property

- **Parameters**

  - *name*: The name of the property

- **Return Value**

  Gets the value of the specified property

`void Parse(XElement xmlDescription)`

- **Summary**

  Parses the specified XML herd agent description filling the properties of the herd agent

- **Parameters**

  - *xmlDescription*: The XML description.

`AppVersion BestMatch(ExperimentalUnit experimentalUnit)`

- **Summary**

  Returns the app version of an experimental unit that best matches the herd agent

- **Parameters**

  - *experimentalUnit*: The experimental unit

- **Return Value**

  Returns the app version of an experimental unit that best matches the herd agent

Herd.Network.Job

Class Herd.Network.Job

Source: Job.cs

Methods

`public Job(List experimentalUnits, HerdAgentInfo herdAgent)`

- **Summary**

  Constructor used from Network.Dispatcher

`public Job()`

- **Summary**

  Parameter-less constructor using for reading a job from the network: Network.JobTransmitter

`void PrepareForExecution()`

- **Summary**

  Prepares the job for execution, creating the appropriate tasks from the experimental units in the job

`void AddInputFiles(List source)`

- **Summary**

  Adds a list of input files

`bool AddInputFile(string file)`

- **Summary**

Adds one input file.

`bool AddOutputFile(string file)`

- **Summary**

  Adds one output file

`string RenamedFilename(string filename)`

- **Summary**

  Returns the filename renamed according to the renaming rules

- **Parameters**

  - *filename*: The filename.

- **Return Value**

  Returns the filename renamed according to the renaming rules

`string OriginalFilename(string filename)`

- **Summary**

  Returns the original name of a renamed file

- **Parameters**

  - *filename*: The renamed filename

- **Return Value**

  Returns the original name of a renamed file

Herd.Network.Jobdispatcher

Class Herd.Network.JobDispatcher

Source: JobDispatcher.cs

Methods

`void AssignExperiments(ref List pendingExperiments

, ref List freeHerdAgents, ref List assignedJobs, JobDispatcherOptions options= null)`

- **Summary**

  Assigns experiments to available herd agents.

- **Parameters**

  - *freeHerdAgents*:

`ExperimentalUnit FirstFittingExperiment(List pendingExperiments, int numFreeCores, bool bAgentUsed, HerdAgentInfo age`

- **Summary**

  Returns the first experimental unit that fits the agent

- **Parameters**

  - *pendingExperiments*: The pending experimental units
  - *numFreeCores*: The number agent's free cores
  - *bAgentUsed*: Is the agent already being used for another experimental unit?
  - *agent*: The herd agent

- **Return Value**

  Returns the first experimental unit that fits the agent

Herd.Network.Jobtransmitter

Class Herd.Network.JobTransmitter

Source: NetTransfer.cs

Methods

`string getFileTypeXMLTag(FileType type)`

- **Summary**

  Gets the file type's XML tag

- **Parameters**

- *type*: The file type

- **Return Value**

  Gets the file type's XML tag

`void SetLogMessageHandler(Action logMessageHandler)`

- **Summary**

  Sets the log message handler

- **Parameters**

  - *logMessageHandler*: The log message handler that will be used from now on

`bool WriteAsync(byte[] buffer, int offset, int length, CancellationToken cancelToken)`

- **Summary**

  Writes asynchronous on the network stream

- **Parameters**

  - *buffer*: The buffer
  - *offset*: The start offset within the buffer
  - *length*: The length
  - *cancelToken*: The cancel token

- **Return Value**

  Writes asynchronous on the network stream

`void waitAsyncWriteOpsToFinish()`

- **Summary**

  Waits for the pending asynchronous write operations to finish.

`void SendInputFiles(bool sendContent, CancellationToken cancelToken)`

- **Summary**

  Sends the input files to the client

- **Parameters**

  - *sendContent*: true if the contents of the files must also be sent
  - *cancelToken*: The cancel token

`void SendOutputFiles(bool sendContent, CancellationToken cancelToken)`

- **Summary**

  Sends the output files to the client

- **Parameters**

  - *sendContent*: true if the content of the file must be sent too
  - *cancelToken*: The cancel token

`void SendTask(HerdTask task, CancellationToken cancelToken)`

- **Summary**

  Sends the task to the client

- **Parameters**

  - *task*: The task
  - *cancelToken*: The cancel token

`void SendJobHeader(CancellationToken cancelToken)`

- **Summary**

  Sends the job header

- **Parameters**

  - *cancelToken*: The cancel token

`void SendJobFooter(CancellationToken cancelToken)`

- **Summary**

  Sends the job footer.

- **Parameters**

  - *cancelToken*: The cancel token.

`void SendFile(string fileName, FileType type, bool sendContent

, bool fromCachedDir, CancellationToken cancelToken, string rename = null)`

- **Summary**

  Sends the file

- **Parameters**

  - *fileName*: Name of the file
  - *type*: The type: input or output
  - *sendContent*: If true, the content is also sent
  - *fromCachedDir*: If true, it will be read from the temp dir
  - *cancelToken*: The cancel token
  - *rename*: The name given to the file remotely

Herd.Network.Shepherd

Class Herd.Network.Shepherd

Source: Shepherd.cs

Methods

`bool IsLocalIpAddress(string host)`

- **Summary**

  Determines whether if the local IP address is local

`void CallHerd()`

- **Summary**

  Calls the herd using a UDP broadcast. Any agent in the same subnet should reply with its properties

`bool ConnectToHerdAgent(IPEndPoint endPoint)`

- **Summary**

  Connects to a herd agent via TCP

- **Parameters**

  - *endPoint*: The end point of the agent

- **Return Value**

  Connects to a herd agent via TCP

`void Disconnect()`

- **Summary**

  Disconnects from the herd agent

`void GetHerdAgentList(ref List outHerdAgentList, int timeoutSeconds = 10)`

- **Summary**

  Gets a list of the herd agent discovered last time the herd was called

- **Parameters**

  - *outHerdAgentList*: The out herd agent list
  - *timeoutSeconds*: The timeout seconds

`void SendJobQuery(Job job, CancellationToken cancelToken)`

- **Summary**

  Sends a job query to the herd agent we connected to

- **Parameters**

  - *job*: The job
  - *cancelToken*: The cancel token

Herd.Network.XmlStream

Class Herd.Network.XMLStream

Source: XMLStream.cs

Methods

`void discardProcessedData()`

- **Summary**

  Discards all already processed data in the buffer

`void WriteMessage(NetworkStream stream, string message, bool addDefaultMessageType = false)`

- **Summary**

  Writes on the network stream

- **Parameters**

  - *stream*: The stream.
  - *message*: The message.
  - *addDefaultMessageType*: If true, a default header is added to the message

`Task WriteMessageAsync(NetworkStream stream, string message, CancellationToken cancelToken, bool addDefaultMessageTyp`

- **Summary**

  Writes on the network stream asynchronously

- **Parameters**

  - *stream*: The stream
  - *message*: The message
  - *cancelToken*: The cancel token
  - *addDefaultMessageType*: If true, a default header is added to the message

- **Return Value**

  Writes on the network stream asynchronously

`void writeMessage(NamedPipeServerStream stream, string message, bool addDefaultMessageType = false)`

- **Summary**

  Writes the message on the named pipe stream

- **Parameters**

  - *stream*: The stream
  - *message*: The message
  - *addDefaultMessageType*: If true, a default header is added to the message

`Task writeMessageAsync(NamedPipeServerStream stream, string message, CancellationToken cancelToken, bool addDefaultMe`

- **Summary**

  Writes a message on the named pipe stream asynchronously

- **Parameters**

  - *stream*: The stream
  - *message*: The message
  - *cancelToken*: The cancel token
  - *addDefaultMessageType*: If true, a default header is added to the message

- **Return Value**

  Writes a message on the named pipe stream asynchronously

`string peekNextXMLItem()`

- **Summary**

  Peeks the next XML item without advancing on the buffer

- **Return Value**

  Peeks the next XML item without advancing on the buffer

`string processNextXMLItem(bool bMarkAsProcessed = true)`

- **Summary**

  returns the next complete xml element (NO ATTRIBUTES!!) in the stream empty string if there was none

- **Parameters**

  - *bMarkAsProcessed*: if true, the element is marked as processed

- **Return Value**

returns the next complete xml element (NO ATTRIBUTES!!) in the stream empty string if there was none

`string peekNextXMLTag()`

- **Summary**

  If message "harry potter" is received this method should return "pipe1", not marking those bytes as processed

- **Return Value**

  If message "harry potter" is received this method should return "pipe1", not marking those bytes as processed

`string processNextXMLTag()`

- **Summary**

  Returns the next complete xml element (NO ATTRIBUTES!!) in the stream, or an empty string if there was none

- **Return Value**

  Returns the next complete xml element (NO ATTRIBUTES!!) in the stream, or an empty string if there was none

`string getLastXMLItemContent()`

- **Summary**

  Instead of parsing pending info in the buffer, it parses m_lastXMLItem , which is set after a call to processNextXMLTag()

- **Return Value**

  Instead of parsing pending info in the buffer, it parses m_lastXMLItem , which is set after a call to processNextXMLTag()