BigWorld's How To Convert to BigWorld 1.9

BigWorld Technology 1.9.1. Released 2008.

Software designed and built in Australia by BigWorld.

Level 3, 431 Glebe Point Road
Glebe NSW 2037, Australia
www.bigworldtech.com

Table of Contents

Chapter 1. Introduction

This document details the changes made to the XML configurations and behaviour of assets and other BigWorld objects between the previous releases and BigWorld 1.9. It also explains the procedures that may be required to update existing assets to the new configurations.

Note
For details on BigWorld XML configurations, see the document File Grammar Guide.

Chapter 2. Upgrading using Asset Processor

2.1. Upgrading using the Asset Processor

Since there have been a few changes made to the asset file formats, your whole resource tree must be processed to conform to the 1.9 file formats. The Asset Processor performs this conversion automatically. Please open the file bigworld/tools/misc/asset_processor/asset_processor.txt and follow the instructions therein.

Chapter 3. Converting Patrol Paths to User Data Objects

User Data Objects are a way of embedding user defined data into chunk files. In BigWorld 1.9, a special User Data Object called **PatrolNode** has been created to replace the more restrictive **Patrol Paths** system. The old system still works, but it is encouraged that all BigWorld users change to the new PatrolNode User Data Objects. This conversion requires updating all chunk files in a space which contain entity and patrol path information.

3.1. Upgrading using the Asset Processor

Since every chunk file for a particular space needs to have its references to each entity and patrol node updated to correctly reference the new user data object file configurations it is necessary to use the **Asset Processor** tool for the conversion.

3.1.1. Asset Processor

After following chapter 2, The asset processor will have already updated all of your chunks to refer to the new User Data Object format. You will now need to make changes to your scripts as well.

3.2. Updating Definition Files and Scripts

Since the Asset Processor script only affects chunk files, the definition files and scripts related to the entities and patrol nodes need to be updated manually to work with the PatrolNode User Data Object.

The `PATROL_PATH` references in the definition files need to be changed to `PATROL_NODE` in order for the links to work. `PATROL_NODE` is just an alias for `UDO_REF` (defined in `alias.xml`).

Additionally, the script API is different from the old Patrol Path system, so entity scripts need to be changed for the entities to use the new PatrolNode User Data Object. For example, this is how an entity would use the old Patrol Paths system:

```
def doWalkToNextNode(self):
    if self.patrolList != None:
         traversableNodes =
 self.patrolList.nodesTraversableFrom(self.patrolNode)
        if len(traversableNodes) > 0:
            (nextNode, nextNodePosition) = traversableNodes[0]
            self.navigateStep( nextNodePosition, velocity, maxMove )
```

The equivalent using the new PatrolNode User Data Object system would be something similar to this:

```
def doWalkToNextNode(self):
    if self.myPatrolNodeProperty != None:
        newNode =
 self.myPatrolNodeProperty.nextPatrolNodeByImportance(self.myPreviousNode)
        if newNode != None:
            self.navigateStep( newNode.position, velocity, maxMove )
```

It is worth noting that the new User Data Object based PatrolNode has additional parameters, such as **importance** and **backtraceChance** (used with the *myPreviousNode*) which affect the result of the call to the **nextPatrolNodeByImportance** method. The Guard entity makes use of the new PatrolNode node system.

Chapter 4. Converting to Advanced Terrain

BigWorld 1.9 introduces a new and advanced terrain system. The new terrain supports advanced visual features like dynamic lighting, LOD and unlimited layers, and is recommended for best visual appearance.

The old (classic) terrain system is still supported, and it is possible to convert a space from classic terrain to the new terrain. Note this conversion can only occur one way: you cannot convert a space with new, advanced terrain to classic terrain.

It is worth noting that there may be a negative performance impact when using the new terrain due to its higher memory requirements and more advanced shaders. Therefore the decision to convert a space must be made with careful consideration (and developer profiling) of the target hardware of the end user.

4.1. Using WorldEditor to convert to Advanced Terrain

Once the space which you wish to convert to the advanced terrain is loaded in WorldEditor, select the menu option **File → Convert to Advanced Terrain...** and choose the desired advanced terrain settings in the **Convert Space** dialog box (for details, see the document Content Tools Reference Guide's section *WorldEditor* → "Menu items").

Note
The converted space may need to have its navigation data recalculated with NavGen.

Chapter 5. Advanced Terrain Server Usage

Due to the increased density of the new terrain, the same sized space in new terrain may consume a substantial amount more memory depending on how much terrain is actually loaded. This can become an issue when running a BigWorld server, as each CellApp process can potentially load the entire region of a space into memory.

As 32-bit linux environments enforce a 4Gb memory limit, both production and development environment server machines may potentially encounter situations where the amount of available memory that can be installed may not be able to meet the requirements of the servers running on the hardware. To help alleviate this issue, a `space.settings` option can be used to lower the detail level loaded by the server processes. The option `<heightMapLod>` is a per space setting representing the LoD level. For more details regarding this option, please refer to the Client Programming Guide section "`terrain` section in `space.settings`". It is important to note that as this option lowers the terrain resolution on the server, you may encounter entity placement issues where the resolution on the client differs to that on the server. If you intend on using this option in production, it is strongly advised that you perform testing with this terrain resolution difference in mind.

Chapter 6. File Format Changes

This chapter describes changes made to BigWorld's file formats between this version and the previous version.

6.1. Converted `Entity` Format

A sample BigWorld pre-1.9 Entity section looks like this (sections that remain the same have been omitted):

```
<entity>
   <properties>
      <patrolList> graph_id </patrolList>
      <patrolPathNode> node_id </patrolPathNode>
   </properties>
</entity>
```

- **`patrolList` (section `properties`)**

  The ID of the graph which the node belongs to.

- **`patrolPathNode` (section `properties`)**

  The ID of the node which the entity links to.

The new BigWorld 1.9 Entity section using **User Data Objects** for the same entity looks like this:

```
<entity>
   <properties>
      <linkName>
         <guid> node_id </guid>
         <chunkId> chunk_id </chunkId>
      </linkName>
   </properties>
</entity>
```

- **`linkName` (section `properties`)**

  The name of the link as defined by the entitiy's definition file, for more information on definition files see the document Server Programming Guide's section *Physical Entity Structure for Scripting* → "The Entity Definition File".

- **`guid` (section `linkName`)**

  The ID of the PatrolNode User Data Object which the entity links to.

- **`chunkId` (section `linkName`)**

  The ID of the chunk which the PatrolNode User Data Object that the entity links to resides in.

Note
The **`chunkId`** section is only used by the editor.

6.2. Converted `Patrol Node` To `User Data Object`

A BigWorld pre-1.9 Patrol Node section in a chunk file looks like this:

```
<station>
   <id> patrol_node_id </id>
   <position> float float float </position>
   <graph> graph_id <g/graph>
   <userString> string </userString>
   <link>
      <to> node_id </to>
      <traversable> [true|false] </traversable>
   </link>
</station>
```

- **id**

  The ID of the patrol node.

- **position**

  The position in the chunk of the patrol node.

- **graph**

  The ID of the graph which the node is part of.

- **userString**

  A user defined string

- **link**

  A link to or from another patrol node or entity.

- **to** (section **link**)

  The ID of the patrol node or entity which this patrol node links to or is linked from.

- **traversable** (section **link**)

  Specifies whether this is a forward link (true) or a back link (false).

The new BigWorld 1.9 Patrol Node User Data Object section inside a chunk file looks like this:

```
<UserDataObject>
   <type> PatrolNode </type>
   <Domain> 2 </Domain>
   <transform> matrix </transform>
   <guid> patrol_node_id </guid>
   <properties>
      <patrolLinks>
         <item>
            <guid> node_id </guid>
            <chunkId> chunk_id </chunkId>
         </item>
      </patrolLinks>
   </properties>
   <backLinks>
      <link>
         <guid> node_id </guid>
         <chunkId> chunk_id </chunkId>
      </link>
   </backLinks>
```

```
</UserDataObject>
```

- **type**

  The type of the User Data Object. When replacing old the Patrol Paths, they get replaced by User Data Objects of type PatrolNode.

- **Domain**

  This value specifies whether the PatrolNode User Data Object should be created by the server or the client. It is defined in the `PatrolNode.def` file.

- **transform**

  The position of the PatrolNode User Data Object in the chunk.

- **guid**

  The ID of the PatrolNode User Data Object.

- **patrolLinks** (section **properties**)

  This section holds information about all forward links. It is defined as a array of UDO_REF property in the `PatrolNode.def` file.

- **item** (section **patrolLinks**)

  A single link to another PatrolNode User Data Object, stored as a UDO_REF.

- **guid** (section **item**)

  The ID of the PatrolNode User Data Object that this PatrolNode User Data Object links to.

- **chunkId** (section **item**)

  The ID of the chunk which the linked PatrolNode User Data Object resides in.

- **backLinks**

  Internal editor use only. This section holds information about the back links.

- **link** (section **backLinks**)

  Internal editor use only. Links from other PatrolNode User Data Objects.

- **guid** (section **link**)

  Internal editor use only. The ID of the PatrolNode User Data Objects that links to this node.

- **chunkId** (section **link**)

  Internal editor use only. The ID of the chunk which the linked PatrolNode User Data Object resides in.

6.3. Space settings

The `space.settings` file now has an additional section called `<terrain>`. This section contains version information, and for new terrain only - per-space terrain configuration values. For more information on these values, see "Client Programming Guide", Chapter 4. Note that spaces without a `<terrain>` section are assumed to be classic (simple) terrain.

Chapter 7. Watcher Network Protocol v2

BigWorld 1.9 introduces a new version of the Watcher network protocol (referred to here-after as Watcher protocol version 2). Prior to BigWorld 1.9 the Watcher protocol was entirely string based which limited the ability of sending complex data via Watchers. Version 2 of the protocol sends watcher requests and responses in a binary format to facilitate more complex watcher interactions.

As a result of the Watcher protocol changes most components of BigWorld that interact with Watchers have been modified to interact with the new protocol.

Note
Due to the protocol changes, the server tools (ie: WebConsole, cluster_control.py, bop_op.py) are not backwards compatible with prior BigWorld releases (1.8.x and prior). BigWorld 1.9 servers however still support Watcher protocol version 1. This means that 1.8.x and prior tools can communicate with 1.9 servers.

7.1. Watcher Network Protocol v1

Please be aware that while BigWorld servers currently still support protocol version 1 requests, this feature may be deprecated in future releases. We advise that any custom programs or reliance on the Watcher protocol is modified to interact with Watcher protocol version 2.

Chapter 8. Migrating WebConsole and other server tools

When migrating between BigWorld 1.8.x and BigWorld 1.9 there are two important factors to consider in regards to the server tools.

- Watcher protocol incompatibility with 1.8 servers.
- Upgrading WebConsole vs re-installing.

8.1. Watcher protocol incompatibility with 1.8 servers

Due the incompatibility for BigWorld 1.9 server tools such as WebConsole to interact with BigWorld 1.8 servers, it is recommended that during the migration to BigWorld 1.9 the old server tools are maintained to allow developers to continue operating seemlessly.

Please refer to the section “Upgrading WebConsole vs re-installing” on page 12 for futher details.

Below is a complete list of BigWorld utilities that may be affected due to the Watcher protocol changes, and thus won't work against earlier server versions.

- WebConsole
- StatLogger / StatGrapher
- `control_cluster.py`
- `bot_op.py`

Please be aware that any custom applications that use the Watcher protocol should still operate correctly, however we recommend upgrading them if possible to avoid any future incompatibilities.

8.2. Upgrading WebConsole vs re-installing

During the migration phase from BigWorld 1.8 to 1.9, it is recommended to install the BigWorld 1.9 server tools on a fresh machine with a more recent linux distribution (eg: Fedora 8). BigWorld 1.9 server tools are now much easier to maintain due to their ability to run from default distribution packages (ie: those provided by `yum` and `apt`).

Note

BigWorld does not support server tools from BigWorld 1.9 running on Fedora Core 6 or earlier.

While the overhead of installing a new machine to run the server tools on may be frustrating, we believe it is the best long term solution to enable the trouble free operation of the server tools in your development environment.

8.2.1. Migrating WebConsole user database

When installing WebConsole from BigWorld 1.9, you may wish to migrate your existing WebConsole user database to the new machine. In order to do this, please follow the instructions below, step by step. NB: The following instructions assume a linux distribution of Fedora, please modify any commands as appropriate if using Debian.

- Install a new machine with linux (Note: do not install the server tools yet).

- Install a MySQL server on the new machine.
- Ensure the MySQL server has been started. `/etc/init.d/mysqld start`
- Create a database user account on the new MySQL server for the server tools. For more details, see the Server Installation Guide section Configure MySQL Server.
- Ensure WebConsole has been stopped on the original machine `/etc/init.d/bw_web_console stop`
- Create the new `bw_web_console` database to migrate the data into.

  On the new server tools machine, start a mysql connection and create the database as follows:

```
$ mysql -u DB_USERNAME -p
Enter password:
...
mysql> CREATE DATABASE bw_web_console;
Query OK, 1 row affected (0.01sec)
```

- Migrate the data from the old tools machine database to the new tools machine database.

  Login to the old tools machine, and as the root user perform the following:

```
$ mysqldump --opt bw_web_console | mysql --host=NEW_MACHINE -u DB_USERNAME
 -p -C bw_web_console
```

- Install the server tools on the new machine.

Note

If you experience issues with SELinux disabling access when trying to run **mysqldump**, set the SELinux mode to 'Disabled' and try again. After performing the mysqldump operation you can safely re-enable SELinux if desired.

Chapter 9. Migrating the Game Database (MySQL)

DBMgr in BigWorld 1.9 requires a slightly different database schema even when using the same entity definitions. When reusing a database that was working with BigWorld 1.8 you will need to run the following command to upgrade the database to the new schema:

```
$ dbmgr -upgrade
```

As this command performs schema alterations, it is recommended to backup your data before running it. The upgrade will perform the following operations:

- Adds the "NOT NULL" specification to all entity data columns. Existing NULL values will be set to the default value of the corresponding property.
- Removes the `bigworldTableMetadata` table.
- Changes `bigworldSpaces`, `bigworldSpaceData` and `bigworldGameTime` tables to use InnoDB and removes the `version` column from `bigworldSpaces` and `bigworldSpaceData`.
- Updates the version number in the `bigworldInfo` table.

Chapter 10. Client Script Changes

A few changes have been made to folder names, script paths and script functions in BigWorld 1.9 that will need to be changed.

1. The folder for scripts files has been renamed from `entities` to `scripts`.
2. The folder for entity definitions has been renamed from `defs` to `entity_defs`.
3. The `keys.py` python script has been renamed to `Keys.py` and moved to `bigworld/res/scripts/client`. This demonstrates where all global python scripts should be located.
4. The callback function definition for **BigWorld.connect()** has been changed so the status argument is a string and not an int. Refer to client python documentation for more information.