# How to Compile DreamGrid

Help files are Word docx's with style sheets and photos in made in View->Outline mode. The source is at https://github.com/Outworldz/DreamGridManuals.

The master word doc is "[Dreamgrid Manual.docx](https://github.com/Outworldz/DreamGridManuals/blob/main/Dreamgrid%20Manual.docx" \o "Dreamgrid Manual.docx)"  which has an automatic table of contents that makes the main manual with includes from about 70 other word docs.  These files get saved as "Web Page (filtered) in the Outworldzfiles\Help folder.

## Adding a new manual

For a NEW manual only, this procedure is necessary. After the manual is finished, saved turn on View Field codes. Go to Fuile->Options at the bottom left of Word.

Now go to the Quick Access Toolbar area of the Word Options dialog box. In the list of Categories, select Commands Not in the Ribbon. Scroll through the long list of commands and select View Field Codes from the list. Click Add.



Enable this by clicking the menu:



After clicking the {A} you just added, you see this:

Graphical user interface, text, application, website

Description automatically generated

These are all the manuals, in alphabetical order as a link.

For a new manual, highlight all the text in it. Go to the section (alphabetical order) in the DreamGrid Manual.docx. Do not paste it in. Use “ Paste Special”, “Paste Link” “Formated Text”:



A perl script "[Make\_zip\_v3.pl](https://github.com/Outworldz/DreamWorld/blob/V4/Make_zip_v3.pl)"  copies the web pages and adds them to both the main and backup Outworldz.com web sites.

## Icons

Add the icons to Robust, pCampbot and Opensim Properties:

Robust is server.ico.  pCampbot is cube\_yellow.ico, and Opensim is cube\_blue.ico.

You right click the name in the properties window and select "Properties". The type the name in.

Graphical user interface, text, application, email

Description automatically generated

## jOpensim and other modules:

I can't compile eZombies, though. That's an interesting module with attacking zombies

I have not added jOpensim Money yet, or jOpensim 4 yet. This is still V 3.9.

We will need PHP8 for it. Curl has been replaced with a new curl which changes Search/Datasnapshot too.

## Opensim compile from the beginning

Make a folder and do a git clone from

git clone <git://opensimulator.org/git/opensim>

Merge in the Diva and other modules in addon-modules.

Table

Description automatically generated

Search and replace all of these in addon-modules. I used Notepad++.

Or you can select each Addon program in Visual Studio, go to the Properties of the module, and select Dot Net 4.8

<TargetFrameworkVersion>v4.6</TargetFrameworkVersion>

with

<TargetFrameworkVersion>v4.8</TargetFrameworkVersion>

Also all v4\_6 to v4\_8 in the addon-modules folder.

Then in all .csproj:

// <autogenerated />  
using System;  
using System.Reflection;  
 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETFramework,Version=v4.8", FrameworkDisplayName = ".NET Framework 4.8")]

search and replace all the above [assembly:...  lines.

## Visual Studio 2019

Download the Visual Studio 2019 Community Edition. Start it and select the Visual Basic and Visual C# development system

You will need the Developer version of Microsoft Dot Net 4.8. You can get it from

[**https://dotnet.microsoft.com/en-us/download/visual-studio-sdks**](https://dotnet.microsoft.com/en-us/download/visual-studio-sdks)

Now load the **Opensim.sln** file in Visual Studio and it should compile in Release or Development mode.

## DreamGrid modifications

### GridService.cs

This shows all regions on the Diva home page, including paused Smart Start regions.

\addon-modules\1DivaOpenSimServices\GridService.cs

In Visual Studio this is in Diva.OpensimServices\GridService.cs

Line 64 is:   
 const RegionFlags flags = RegionFlags.RegionOnline;

Line 64 should be this to show Smart Start Regions on the home screen. These are marked as persistent on first boot by the Opensim setup in DreamGrid

const RegionFlags flags = RegionFlags.RegionOnline | RegionFlags.Persistent ; // DreamGrid to display Smart Boot on home page

BasehttpServer.cs

This allows DreamGrid to determine when Robust is ready.

OpenSim\Framework\Servers\HttpServer\BaseHttpServer.cs

Added this code to the HTTP handler for robust. It allows DreamGrid to determine more accurately when Robust is ready to accept new commands.

Line 2445:

// Smartstart

if (httpRequest.QueryFlags.Contains("version"))  
 {  
 httpResponse.StatusCode = (int)HttpStatusCode.OK;  
 httpResponse.RawBuffer = Encoding.ASCII.GetBytes(VersionInfo.Version);  
 return;  
 }

### llLoginService.cs

This allows people lioggion in to be redirected to welcome or to get a the destination region thawed.  
File Opensim.Services.llLoginService.llLoginService.cs

Insert between italicized code

98: *//IConfig m\_ClientsConfig;*

:99

100: // SmartStart

101: protected bool m\_SmartStartEnabled = false;

102: protected string m\_SmartStartUrl = string.Empty;

103: protected string m\_SmartStartMachineID = string.Empty;

104:  
105: *public LLLoginService…*

--------------------------------------------------------------------------------------------------------

*if (!string.IsNullOrWhiteSpace(agentService))*

*m\_UserAgentService = ServerUtils.LoadPlugin<IUserAgentService>(agentService, args);*

*// SmartStart*

*IConfig SmartStartConfig = config.Configs["SmartStart"];*

*if(SmartStartConfig != null)*

*{*

*m\_SmartStartEnabled = SmartStartConfig.GetBoolean("Enabled", m\_SmartStartEnabled);*

*if (m\_SmartStartEnabled)*

*{*

*m\_SmartStartUrl = SmartStartConfig.GetString("URL", m\_SmartStartUrl);*

*if(string.IsNullOrEmpty(m\_SmartStartUrl))*

*m\_SmartStartEnabled = false;*

*else*

*{*

*OSHHTPHost tmpSmartStartURL = new OSHHTPHost(m\_SmartStartUrl, true);*

*if (!tmpSmartStartURL.IsResolvedHost)*

*{*

*m\_log.Error("[LLOGIN SERVICE]: Could not parse or resolve SmartStart URI");*

*throw new Exception("LLOGIN SERVICE init error: SmartStart URI");*

*}*

*m\_SmartStartUrl = tmpSmartStartURL.URI;*

*m\_log.Info("[LLOGIN SERVICE]: SmartStart Url " + m\_SmartStartUrl);*

*m\_SmartStartMachineID = SmartStartConfig.GetString("MachineID", m\_SmartStartMachineID);*

*}*

*}*

*}*

*m\_log.Info("[LLOGIN SERVICE]: SmartStart " + (m\_SmartStartEnabled ? "Enabled" : "Disabled"));*

*// Get the Hypergrid inventory service (exists only if Hypergrid is enabled)*

public LLLoginService(IConfigSource config) : this(config, null, null)

{

}

--------------------------------------------------------------------------------------------------------

//DreamGrid SmartStart

public UUID GetSmartStartALTRegion(UUID regionID, UUID agentID)

{

// !!! DreamGrid Smart Start sends requested Region UUID to Dreamgrid.

// If region is on line, returns same UUID. If Offline, returns UUID for Welcome, brings up the region and teleports you to it.

if (m\_SmartStartEnabled)

{

string url = $"{m\_SmartStartUrl}?alt={regionID}&agent=UUID&agentid={agentID}&password={m\_SmartStartMachineID}";

m\_log.DebugFormat("[LLoginService]: GetSmartStartALTRegion Sending request {0}", url);

HttpWebRequest webRequest;

try

{

webRequest = (HttpWebRequest)WebRequest.Create(url);

}

catch

{

m\_log.Debug("[LLoginService]: GetSmartStartALTRegion failed to create url");

return UUID.Zero;

}

webRequest.Timeout = 30000; //30 Second Timeout

webRequest.AllowWriteStreamBuffering = false;

try

{

string tempStr;

using (HttpWebResponse webResponse = (HttpWebResponse)webRequest.GetResponse())

{

using(StreamReader reader = new StreamReader(webResponse.GetResponseStream()))

tempStr = reader.ReadToEnd();

}

if (string.IsNullOrEmpty(tempStr))

{

m\_log.Debug("[LLoginService]: GetSmartStartALTRegion returned null");

return UUID.Zero;

}

m\_log.Debug("[LLoginService]: GetSmartStartALTRegion returned " + tempStr);

regionID = UUID.Parse(tempStr);

}

catch (Exception ex)

{

m\_log.Warn("[LLoginService]: GetSmartStartALTRegion exception: " + ex.Message);

}

}

return regionID;

}

public Hashtable SetLevel(string firstName, string lastName, string passwd, int level, IPEndPoint clientIP)

{

--------------------------------------------------------------------------------------------------------

GridRegion dest = FindDestinationNormal(account, scopeID, pinfo, sessionID, startLocation, home,

out gatekeeper, out where, out position, out lookAt, out flags);

if(!m\_SmartStartEnabled || dest == null)

return dest;

//RegionFlags.RegionOnline should had a major role here, but it is still unreliable

//Jump out if a special region that must be always up

// for now the smartstart wait region must have one such flags

//if ((dest.RegionFlags & (RegionFlags.Hyperlink | RegionFlags.DefaultRegion | RegionFlags.FallbackRegion | RegionFlags.DefaultHGRegion)) != 0)

// return dest;

UUID rid = GetSmartStartALTRegion(dest.RegionID, account.PrincipalID);

if (rid == dest.RegionID)

return dest;

if (rid == UUID.Zero)

return null;

return m\_GridService.GetRegionByUUID(scopeID, rid);

}

protected GridRegion FindDestinationNormal(

### Scene.cs

File Opensim.Region.Framework.Scenes.Scene.cs

Line 818: insert new code between italicized code

*818: public GridInfo SceneGridInfo;*

*819:*

820: // SmartStart

821: protected bool m\_SmartStartEnabled = false;

822: protected string m\_SmartStartUrl = string.Empty;

823: protected string m\_SmartStartMachineID = string.Empty;

824:

*825: #endregion Properties*

Line 827 insert new code  
  
827: #region Constructors

// SmartStart

public UUID GetSmartStartALTRegion(UUID regionID, UUID agentID)

{

// !!! DreamGrid Smart Start sends requested Region UUID to Dreamgrid.

// If region is on line, returns same UUID. If Offline, returns UUID for Welcome, brings up the region and teleports you to it.

if (m\_SmartStartEnabled)

{

string url = $"{m\_SmartStartUrl}?alt={regionID}&agent=UUID&agentid={agentID}&password={m\_SmartStartMachineID}";

m\_log.DebugFormat("[LLoginService]: GetSmartStartALTRegion Sending request {0}", url);

HttpWebRequest webRequest;

try

{

webRequest = (HttpWebRequest)WebRequest.Create(url);

}

catch

{

m\_log.Debug("[LLoginService]: GetSmartStartALTRegion failed to create url");

return UUID.Zero;

}

webRequest.Timeout = 30000; //30 Second Timeout

webRequest.AllowWriteStreamBuffering = false;

try

{

string tempStr;

using (HttpWebResponse webResponse = (HttpWebResponse)webRequest.GetResponse())

{

using (StreamReader reader = new StreamReader(webResponse.GetResponseStream()))

tempStr = reader.ReadToEnd();

}

if (string.IsNullOrEmpty(tempStr))

{

m\_log.Debug("[LLoginService]: GetSmartStartALTRegion returned null");

return UUID.Zero;

}

m\_log.Debug("[LLoginService]: GetSmartStartALTRegion returned " + tempStr);

regionID = UUID.Parse(tempStr);

}

catch (Exception ex)

{

m\_log.Warn("[LLoginService]: GetSmartStartALTRegion exception: " + ex.Message);

}

}

return regionID;

}

Line 1163 insert code before #endregion  
  
*m\_update\_temp\_cleaning = startupConfig.GetInt("UpdateTempCleaningEveryNSeconds", m\_update\_temp\_cleaning);*

1163:  
1164: // SmartStart

IConfig SmartStartConfig = config.Configs["SmartStart"];

if (SmartStartConfig != null)

{

m\_SmartStartEnabled = SmartStartConfig.GetBoolean("Enabled", m\_SmartStartEnabled);

if (m\_SmartStartEnabled)

{

m\_SmartStartUrl = SmartStartConfig.GetString("URL", m\_SmartStartUrl);

if (string.IsNullOrEmpty(m\_SmartStartUrl))

m\_SmartStartEnabled = false;

else

{

OSHHTPHost tmpSmartStartURL = new OSHHTPHost(m\_SmartStartUrl, true);

if (!tmpSmartStartURL.IsResolvedHost)

{

m\_log.Error("[SCENE]: Could not parse or resolve SmartStart URI");

throw new Exception("SCENE init error: SmartStart URI");

}

m\_SmartStartUrl = tmpSmartStartURL.URI;

m\_log.Info("[SCENE]: SmartStart Url " + m\_SmartStartUrl);

m\_SmartStartMachineID = SmartStartConfig.GetString("MachineID", m\_SmartStartMachineID);

}

}

}

m\_log.Info("[SCENE]: SmartStart " + (m\_SmartStartEnabled ? "Enabled" : "Disabled"));

#endregion Region Config

### EntitityTransferModule.cs

File Opensim.Region.CoreModules.Framework.EntityTransfer.EmntityTransferModule.cs

Line: 42, insert the RegionFlags line

42: *using GridRegion = OpenSim.Services.Interfaces.GridRegion;*

43: using RegionFlags = OpenSim.Framework.RegionFlags;

Line 755-758. After this insert the following code:

755: *m\_log.DebugFormat(*

*756: "[ENTITY TRANSFER MODULE]: Failed validation of all attachments 756: for teleport of {0} from {1} to {2}. Continuing.",*

*758: sp.Name, sp.Scene.Name, finalDestination.RegionName);*

// Smartstart

//This is still a test.

// this possible should only be called if query fails with a limited set of errors like connection refused.

if (reg.RegionLocY != 0) // not on HG

{

if ((finalDestination.RegionFlags & (RegionFlags.Hyperlink | RegionFlags.DefaultRegion | RegionFlags.FallbackRegion | RegionFlags.DefaultHGRegion)) == 0)

{

UUID regID = sp.Scene.GetSmartStartALTRegion(finalDestination.RegionID, sp.ControllingClient.AgentId); // fkb

if (regID != UUID.Zero && regID != finalDestination.RegionID)

{

if (regID == sp.Scene.RegionInfo.RegionID)

{

sp.ControllingClient.SendTeleportFailed("Destination region Loading. Teleport will happen soon");

return;

}

finalDestination = sp.Scene.GridService.GetRegionByUUID(sp.Scene.RegionInfo.ScopeID, regID);

if (finalDestination == null)

{

sp.ControllingClient.SendTeleportFailed("Destination region Loading. Teleport will happen soon");

return;

}

reg = finalDestination;

}

}

}

### GateKeeper.cs

File Opensim.Services.HypregridService.Gatekeeper.cs

Line 41: insert Using:

40: *using OpenMetaverse;*

41: using RegionFlags = OpenSim.Framework.RegionFlags;

42: *using Nini.Config;*

After line 79 insert new code:

*79: private static string m\_messageKey;*

80:

81: // SmartStart

82: protected static bool m\_SmartStartEnabled = false;

83:protected static string m\_SmartStartUrl = string.Empty;

84:protected static string m\_SmartStartMachineID = string.Empty;

85:

*86: public GatekeeperService(IConfigSource config, ISimulationService simService)*

After line 182 (italic) insert new code

*if (messagingConfig != null)*

*m\_messageKey = messagingConfig.GetString("MessageKey", String.Empty);*

// SmartStart fkb

IConfig SmartStartConfig = config.Configs["SmartStart"];

if (SmartStartConfig != null)

{

m\_SmartStartEnabled = SmartStartConfig.GetBoolean("Enabled", m\_SmartStartEnabled);

if (m\_SmartStartEnabled)

{

m\_SmartStartUrl = SmartStartConfig.GetString("URL", m\_SmartStartUrl);

if (string.IsNullOrEmpty(m\_SmartStartUrl))

m\_SmartStartEnabled = false;

else

{

OSHHTPHost tmpSmartStartURL = new OSHHTPHost(m\_SmartStartUrl, true);

if (!tmpSmartStartURL.IsResolvedHost)

{

m\_log.Error("[GATEKEEPER SERVICE]: Could not parse or resolve SmartStart URI");

throw new Exception("GATEKEEPER SERVICE init error: SmartStart URI");

}

m\_SmartStartUrl = tmpSmartStartURL.URI;

m\_log.Info("[GATEKEEPER SERVICE]: SmartStart Url " + m\_SmartStartUrl);

m\_SmartStartMachineID = SmartStartConfig.GetString("MachineID", m\_SmartStartMachineID);

}

}

}

*m\_log.Debug("[GATEKEEPER SERVICE]: Starting...");*

### llLoginService.cs

File Opensim.Services.llLoginService.llLoginService.cs

Line 47: insert Using

46: *using OpenSim.Services.Connectors.Hypergrid;*

47: using RegionFlags = OpenSim.Framework.RegionFlags;

Line 100 insert new code

99*: //IConfig m\_ClientsConfig;*

100:

101: // SmartStart

102: protected bool m\_SmartStartEnabled = false;

103: protected string m\_SmartStartUrl = string.Empty;

104: protected string m\_SmartStartMachineID = string.Empty;

105:

106: *public LLLoginService…*

Line 199: insert new code

196:  *if (!string.IsNullOrWhiteSpace(agentService))*

197: *m\_UserAgentService = ServerUtils.LoadPlugin<IUserAgentService>(agentService, args);*

198:

199: // SmartStart

IConfig SmartStartConfig = config.Configs["SmartStart"];

if (SmartStartConfig != null)

{

m\_SmartStartEnabled = SmartStartConfig.GetBoolean("Enabled", m\_SmartStartEnabled);

if (m\_SmartStartEnabled)

{

m\_SmartStartUrl = SmartStartConfig.GetString("URL", m\_SmartStartUrl);

if (string.IsNullOrEmpty(m\_SmartStartUrl))

m\_SmartStartEnabled = false;

else

{

OSHHTPHost tmpSmartStartURL = new OSHHTPHost(m\_SmartStartUrl, true);

if (!tmpSmartStartURL.IsResolvedHost)

{

m\_log.Error("[LLOGIN SERVICE]: Could not parse or resolve SmartStart URI");

throw new Exception("LLOGIN SERVICE init error: SmartStart URI");

}

m\_SmartStartUrl = tmpSmartStartURL.URI;

m\_log.Info("[LLOGIN SERVICE]: SmartStart Url " + m\_SmartStartUrl);

m\_SmartStartMachineID = SmartStartConfig.GetString("MachineID", m\_SmartStartMachineID);

}

}

}

m\_log.Info("[LLOGIN SERVICE]: SmartStart " + (m\_SmartStartEnabled ? "Enabled" : "Disabled"));

Line 273 insert new code:

*public LLLoginService(IConfigSource config) : this(config, null, null)*

*{*

*}*

273: //DreamGrid SmartStart

public UUID GetSmartStartALTRegion(UUID regionID, UUID agentID)

{

// !!! DreamGrid Smart Start sends requested Region UUID to Dreamgrid.

// If region is on line, returns same UUID. If Offline, returns UUID for Welcome, brings up the region and teleports you to it.

if (m\_SmartStartEnabled)

{

string url = $"{m\_SmartStartUrl}?alt={regionID}&agent=UUID&agentid={agentID}&password={m\_SmartStartMachineID}";

m\_log.DebugFormat("[LLoginService]: GetSmartStartALTRegion Sending request {0}", url);

HttpWebRequest webRequest;

try

{

webRequest = (HttpWebRequest)WebRequest.Create(url);

}

catch

{

m\_log.Debug("[LLoginService]: GetSmartStartALTRegion failed to create url");

return UUID.Zero;

}

webRequest.Timeout = 30000; //30 Second Timeout

webRequest.AllowWriteStreamBuffering = false;

try

{

string tempStr;

using (HttpWebResponse webResponse = (HttpWebResponse)webRequest.GetResponse())

{

using (System.IO.StreamReader reader = new System.IO.StreamReader( webResponse.GetResponseStream()))

tempStr = reader.ReadToEnd();

}

if (string.IsNullOrEmpty(tempStr))

{

m\_log.Debug("[LLoginService]: GetSmartStartALTRegion returned null");

return UUID.Zero;

}

m\_log.Debug("[LLoginService]: GetSmartStartALTRegion returned " + tempStr);

regionID = UUID.Parse(tempStr);

}

catch (Exception ex)

{

m\_log.Warn("[LLoginService]: GetSmartStartALTRegion exception: " + ex.Message);

}

}

return regionID;

}

*public Hashtable SetLevel(string firstName, string lastName, string passwd, int level, IPEndPoint clientIP)*

After line 690 function definition

        protected GridRegion FindDestination(  
            UserAccount account, UUID scopeID, GridUserInfo pinfo, UUID sessionID, string startLocation,  
            GridRegion home, out GridRegion gatekeeper,  
            out string where, out Vector3 position, out Vector3 lookAt, out TeleportFlags flags)  
        {

paste in: ( line 696)

....start

             GridRegion dest = FindDestinationNormal(account, scopeID, pinfo, sessionID, startLocation, home,  
                out gatekeeper, out where, out position, out lookAt, out flags);  
  
            if(!m\_SmartStartEnabled || dest == null)  
                return dest;  
  
            //RegionFlags.RegionOnline should had a major role here, but it is still unreliable  
            //Jump out if a special region that must be always up  
            // for now the smartstart wait region must have one such flags  
            if ((dest.RegionFlags & (RegionFlags.Hyperlink | RegionFlags.DefaultRegion | RegionFlags.FallbackRegion | RegionFlags.DefaultHGRegion)) != 0)  
                return dest;  
  
            UUID rid = GetSmartStartALTRegion(dest.RegionID, account.PrincipalID);  
            if (rid == dest.RegionID)  
                return dest;  
  
            if (rid == UUID.Zero)  
                return null; //??  
  
            return m\_GridService.GetRegionByUUID(scopeID, rid);  
        }  
  
        protected GridRegion FindDestinationNormal(  
            UserAccount account, UUID scopeID, GridUserInfo pinfo, UUID sessionID, string startLocation,  
            GridRegion home, out GridRegion gatekeeper,  
            out string where, out Vector3 position, out Vector3 lookAt, out TeleportFlags flags)  
        {

IAR patch

https://grimore.org/opensim/server\_patches/disable\_iar\_passwords/0.7.6

**--- ./OpenSim/Region/CoreModules/Avatar/Inventory/Archiver/InventoryArchiverModule.cs.orig 2013-10-04 19:45:02.000000000 +0000**

+++ ./OpenSim/Region/CoreModules/Avatar/Inventory/Archiver/InventoryArchiverModule.cs 2013-10-16 00:46:01.000000000 +0000

@@ -114,20 +114,19 @@

scene.AddCommand(

"Archiving", this, "load iar",

- "load iar [-m|--merge] <first> <last> <inventory path> <password> [<IAR path>]",

+ "load iar [-m|--merge] <first> <last> <inventory path> [<IAR path>]",

"Load user inventory archive (IAR).",

"-m|--merge is an option which merges the loaded IAR with existing inventory folders where possible, rather than always creating new ones"

+ "<first> is user's first name." + Environment.NewLine

+ "<last> is user's last name." + Environment.NewLine

+ "<inventory path> is the path inside the user's inventory where the IAR should be loaded." + Environment.NewLine

- + "<password> is the user's password." + Environment.NewLine

+ "<IAR path> is the filesystem path or URI from which to load the IAR."

+ string.Format(" If this is not given then the filename {0} in the current directory is used", DEFAULT\_INV\_BACKUP\_FILENAME),

HandleLoadInvConsoleCommand);

scene.AddCommand(

"Archiving", this, "save iar",

- "save iar [-h|--home=<url>] [--noassets] <first> <last> <inventory path> <password> [<IAR path>] [-c|--creators] [-e|--exclude=<name/uuid>] [-f|--excludefolder=<foldername/uuid>] [-v|--verbose]",

+ "save iar [-h|--home=<url>] [--noassets] <first> <last> <inventory path> [<IAR path>] [-c|--creators] [-e|--exclude=<name/uuid>] [-f|--excludefolder=<foldername/uuid>] [-v|--verbose]",

"Save user inventory archive (IAR).",

"<first> is the user's first name.\n"

+ "<last> is the user's last name.\n"

@@ -184,18 +183,18 @@

}

public bool ArchiveInventory(

- Guid id, string firstName, string lastName, string invPath, string pass, Stream saveStream)

+ Guid id, string firstName, string lastName, string invPath, Stream saveStream)

{

- return ArchiveInventory(id, firstName, lastName, invPath, pass, saveStream, new Dictionary<string, object>());

+ return ArchiveInventory(id, firstName, lastName, invPath, saveStream, new Dictionary<string, object>());

}

public bool ArchiveInventory(

- Guid id, string firstName, string lastName, string invPath, string pass, Stream saveStream,

+ Guid id, string firstName, string lastName, string invPath, Stream saveStream,

Dictionary<string, object> options)

{

if (m\_scenes.Count > 0)

{

- UserAccount userInfo = GetUserInfo(firstName, lastName, pass);

+ UserAccount userInfo = GetUserInfo(firstName, lastName);

if (userInfo != null)

{

@@ -230,7 +229,7 @@

}

public bool ArchiveInventory(

- Guid id, string firstName, string lastName, string invPath, string pass, string savePath,

+ Guid id, string firstName, string lastName, string invPath, string savePath,

Dictionary<string, object> options)

{

// if (!ConsoleUtil.CheckFileDoesNotExist(MainConsole.Instance, savePath))

@@ -238,7 +237,7 @@

if (m\_scenes.Count > 0)

{

- UserAccount userInfo = GetUserInfo(firstName, lastName, pass);

+ UserAccount userInfo = GetUserInfo(firstName, lastName);

if (userInfo != null)

{

@@ -272,18 +271,18 @@

return false;

}

- public bool DearchiveInventory(string firstName, string lastName, string invPath, string pass, Stream loadStream)

+ public bool DearchiveInventory(string firstName, string lastName, string invPath, Stream loadStream)

{

- return DearchiveInventory(firstName, lastName, invPath, pass, loadStream, new Dictionary<string, object>());

+ return DearchiveInventory(firstName, lastName, invPath, loadStream, new Dictionary<string, object>());

}

public bool DearchiveInventory(

- string firstName, string lastName, string invPath, string pass, Stream loadStream,

+ string firstName, string lastName, string invPath, Stream loadStream,

Dictionary<string, object> options)

{

if (m\_scenes.Count > 0)

{

- UserAccount userInfo = GetUserInfo(firstName, lastName, pass);

+ UserAccount userInfo = GetUserInfo(firstName, lastName);

if (userInfo != null)

{

@@ -326,12 +325,12 @@

}

public bool DearchiveInventory(

- string firstName, string lastName, string invPath, string pass, string loadPath,

+ string firstName, string lastName, string invPath, string loadPath,

Dictionary<string, object> options)

{

if (m\_scenes.Count > 0)

{

- UserAccount userInfo = GetUserInfo(firstName, lastName, pass);

+ UserAccount userInfo = GetUserInfo(firstName, lastName);

if (userInfo != null)

{

@@ -383,24 +382,23 @@

List<string> mainParams = optionSet.Parse(cmdparams);

- if (mainParams.Count < 6)

+ if (mainParams.Count < 5)

{

m\_log.Error(

- "[INVENTORY ARCHIVER]: usage is load iar [-m|--merge] <first name> <last name> <inventory path> <user password> [<load file path>]");

+ "[INVENTORY ARCHIVER]: usage is load iar [-m|--merge] <first name> <last name> <inventory path> [<load file path>]");

return;

}

string firstName = mainParams[2];

string lastName = mainParams[3];

string invPath = mainParams[4];

- string pass = mainParams[5];

- string loadPath = (mainParams.Count > 6 ? mainParams[6] : DEFAULT\_INV\_BACKUP\_FILENAME);

+ string loadPath = (mainParams.Count > 5 ? mainParams[5] : DEFAULT\_INV\_BACKUP\_FILENAME);

m\_log.InfoFormat(

"[INVENTORY ARCHIVER]: Loading archive {0} to inventory path {1} for {2} {3}",

loadPath, invPath, firstName, lastName);

- if (DearchiveInventory(firstName, lastName, invPath, pass, loadPath, options))

+ if (DearchiveInventory(firstName, lastName, invPath, loadPath, options))

m\_log.InfoFormat(

"[INVENTORY ARCHIVER]: Loaded archive {0} for {1} {2}",

loadPath, firstName, lastName);

@@ -444,10 +442,10 @@

try

{

- if (mainParams.Count < 6)

+ if (mainParams.Count < 5)

{

m\_log.Error(

- "[INVENTORY ARCHIVER]: save iar [-h|--home=<url>] [--noassets] <first> <last> <inventory path> <password> [<IAR path>] [-c|--creators] [-e|--exclude=<name/uuid>] [-f|--excludefolder=<foldername/uuid>] [-v|--verbose]");

+ "[INVENTORY ARCHIVER]: save iar [-h|--home=<url>] [--noassets] <first> <last> <inventory path> [<IAR path>] [-c|--creators] [-e|--exclude=<name/uuid>] [-f|--excludefolder=<foldername/uuid>] [-v|--verbose]");

return;

}

@@ -457,8 +455,7 @@

string firstName = mainParams[2];

string lastName = mainParams[3];

string invPath = mainParams[4];

- string pass = mainParams[5];

- string savePath = (mainParams.Count > 6 ? mainParams[6] : DEFAULT\_INV\_BACKUP\_FILENAME);

+ string savePath = (mainParams.Count > 5 ? mainParams[5] : DEFAULT\_INV\_BACKUP\_FILENAME);

m\_log.InfoFormat(

"[INVENTORY ARCHIVER]: Saving archive {0} using inventory path {1} for {2} {3}",

@@ -467,7 +464,7 @@

lock (m\_pendingConsoleSaves)

m\_pendingConsoleSaves.Add(id);

- ArchiveInventory(id, firstName, lastName, invPath, pass, savePath, options);

+ ArchiveInventory(id, firstName, lastName, invPath, savePath, options);

}

catch (InventoryArchiverException e)

{

@@ -504,9 +501,8 @@

/// </summary>

/// <param name="firstName"></param>

/// <param name="lastName"></param>

- /// <param name="pass">User password</param>

/// <returns></returns>

- protected UserAccount GetUserInfo(string firstName, string lastName, string pass)

+ protected UserAccount GetUserInfo(string firstName, string lastName)

{

UserAccount account

= m\_aScene.UserAccountService.GetUserAccount(m\_aScene.RegionInfo.ScopeID, firstName, lastName);

@@ -518,27 +514,7 @@

firstName, lastName);

return null;

}

-

- try

- {

- string encpass = Util.Md5Hash(pass);

- if (m\_aScene.AuthenticationService.Authenticate(account.PrincipalID, encpass, 1) != string.Empty)

- {

- return account;

- }

- else

- {

- m\_log.ErrorFormat(

- "[INVENTORY ARCHIVER]: Password for user {0} {1} incorrect. Please try again.",

- firstName, lastName);

- return null;

- }

- }

- catch (Exception e)

- {

- m\_log.ErrorFormat("[INVENTORY ARCHIVER]: Could not authenticate password, {0}", e);

- return null;

- }

+ return account;

}

/// <summary>

**--- ./OpenSim/Region/CoreModules/Avatar/Inventory/Archiver/Tests/InventoryArchiveTestCase.cs.orig 2013-10-16 00:52:58.000000000 +0000**

+++ ./OpenSim/Region/CoreModules/Avatar/Inventory/Archiver/Tests/InventoryArchiveTestCase.cs 2013-10-16 00:53:29.000000000 +0000

@@ -163,7 +163,7 @@

scene.AddInventoryItem(coaItem);

archiverModule.ArchiveInventory(

- Guid.NewGuid(), m\_uaLL1.FirstName, m\_uaLL1.LastName, "/\*", "hampshire", archiveWriteStream);

+ Guid.NewGuid(), m\_uaLL1.FirstName, m\_uaLL1.LastName, "/\*", archiveWriteStream);

m\_iarStreamBytes = archiveWriteStream.ToArray();

}

**--- ./OpenSim/Region/CoreModules/Avatar/Inventory/Archiver/Tests/InventoryArchiveLoadPathTests.cs.orig 2013-10-16 00:46:32.000000000 +0000**

+++ ./OpenSim/Region/CoreModules/Avatar/Inventory/Archiver/Tests/InventoryArchiveLoadPathTests.cs 2013-10-16 01:01:58.000000000 +0000

@@ -70,7 +70,7 @@

UserAccountHelpers.CreateUserWithInventory(scene, m\_uaMT, "meowfood");

UserAccountHelpers.CreateUserWithInventory(scene, m\_uaLL1, "hampshire");

- archiverModule.DearchiveInventory(m\_uaMT.FirstName, m\_uaMT.LastName, "/", "meowfood", m\_iarStream);

+ archiverModule.DearchiveInventory(m\_uaMT.FirstName, m\_uaMT.LastName, "/", m\_iarStream);

InventoryItemBase foundItem1

= InventoryArchiveUtils.FindItemByPath(scene.InventoryService, m\_uaMT.PrincipalID, m\_item1Name);

@@ -79,7 +79,7 @@

// Now try loading to a root child folder

UserInventoryHelpers.CreateInventoryFolder(scene.InventoryService, m\_uaMT.PrincipalID, "xA", false);

MemoryStream archiveReadStream = new MemoryStream(m\_iarStream.ToArray());

- archiverModule.DearchiveInventory(m\_uaMT.FirstName, m\_uaMT.LastName, "xA", "meowfood", archiveReadStream);

+ archiverModule.DearchiveInventory(m\_uaMT.FirstName, m\_uaMT.LastName, "xA", archiveReadStream);

InventoryItemBase foundItem2

= InventoryArchiveUtils.FindItemByPath(scene.InventoryService, m\_uaMT.PrincipalID, "xA/" + m\_item1Name);

@@ -88,7 +88,7 @@

// Now try loading to a more deeply nested folder

UserInventoryHelpers.CreateInventoryFolder(scene.InventoryService, m\_uaMT.PrincipalID, "xB/xC", false);

archiveReadStream = new MemoryStream(archiveReadStream.ToArray());

- archiverModule.DearchiveInventory(m\_uaMT.FirstName, m\_uaMT.LastName, "xB/xC", "meowfood", archiveReadStream);

+ archiverModule.DearchiveInventory(m\_uaMT.FirstName, m\_uaMT.LastName, "xB/xC", archiveReadStream);

InventoryItemBase foundItem3

= InventoryArchiveUtils.FindItemByPath(scene.InventoryService, m\_uaMT.PrincipalID, "xB/xC/" + m\_item1Name);

@@ -110,7 +110,7 @@

SceneHelpers.SetupSceneModules(scene, serialiserModule, archiverModule);

UserAccountHelpers.CreateUserWithInventory(scene, m\_uaMT, "password");

- archiverModule.DearchiveInventory(m\_uaMT.FirstName, m\_uaMT.LastName, "/Objects", "password", m\_iarStream);

+ archiverModule.DearchiveInventory(m\_uaMT.FirstName, m\_uaMT.LastName, "/Objects", m\_iarStream);

InventoryItemBase foundItem1

= InventoryArchiveUtils.FindItemByPath(

@@ -180,13 +180,13 @@

mre.Reset();

archiverModule.ArchiveInventory(

- Guid.NewGuid(), userFirstName, userLastName, "Objects", userPassword, archiveWriteStream);

+ Guid.NewGuid(), userFirstName, userLastName, "Objects", archiveWriteStream);

mre.WaitOne(60000, false);

// LOAD ITEM

MemoryStream archiveReadStream = new MemoryStream(archiveWriteStream.ToArray());

- archiverModule.DearchiveInventory(userFirstName, userLastName, "Scripts", userPassword, archiveReadStream);

+ archiverModule.DearchiveInventory(userFirstName, userLastName, "Scripts", archiveReadStream);

InventoryItemBase foundItem1

= InventoryArchiveUtils.FindItemByPath(

**--- ./OpenSim/Region/CoreModules/Avatar/Inventory/Archiver/Tests/InventoryArchiveSaveTests.cs.orig 2013-10-16 00:48:52.000000000 +0000**

+++ ./OpenSim/Region/CoreModules/Avatar/Inventory/Archiver/Tests/InventoryArchiveSaveTests.cs 2013-10-16 00:50:20.000000000 +0000

@@ -110,7 +110,7 @@

mre.Reset();

m\_archiverModule.ArchiveInventory(

- Guid.NewGuid(), userFirstName, userLastName, "/", userPassword, archiveWriteStream);

+ Guid.NewGuid(), userFirstName, userLastName, "/", archiveWriteStream);

mre.WaitOne(60000, false);

// Test created iar

@@ -179,7 +179,7 @@

mre.Reset();

m\_archiverModule.ArchiveInventory(

- Guid.NewGuid(), userFirstName, userLastName, "f1", userPassword, archiveWriteStream);

+ Guid.NewGuid(), userFirstName, userLastName, "f1", archiveWriteStream);

mre.WaitOne(60000, false);

// Test created iar

@@ -267,7 +267,7 @@

mre.Reset();

m\_archiverModule.ArchiveInventory(

- Guid.NewGuid(), userFirstName, userLastName, "Objects/" + item1Name, userPassword, archiveWriteStream);

+ Guid.NewGuid(), userFirstName, userLastName, "Objects/" + item1Name, archiveWriteStream);

mre.WaitOne(60000, false);

byte[] archive = archiveWriteStream.ToArray();

@@ -364,7 +364,7 @@

// When we're not saving assets, archiving is being done synchronously.

m\_archiverModule.ArchiveInventory(

- Guid.NewGuid(), userFirstName, userLastName, "Objects/" + item1Name, userPassword, archiveWriteStream, options);

+ Guid.NewGuid(), userFirstName, userLastName, "Objects/" + item1Name, archiveWriteStream, options);

byte[] archive = archiveWriteStream.ToArray();

MemoryStream archiveReadStream = new MemoryStream(archive);

**--- ./OpenSim/Region/CoreModules/Avatar/Inventory/Archiver/Tests/InventoryArchiveLoadTests.cs.orig 2013-10-16 00:46:40.000000000 +0000**

+++ ./OpenSim/Region/CoreModules/Avatar/Inventory/Archiver/Tests/InventoryArchiveLoadTests.cs 2013-10-16 01:03:30.000000000 +0000

@@ -72,7 +72,7 @@

// TestHelpers.EnableLogging();

UserAccountHelpers.CreateUserWithInventory(m\_scene, m\_uaLL1, "password");

- m\_archiverModule.DearchiveInventory(m\_uaLL1.FirstName, m\_uaLL1.LastName, "/", "password", m\_iarStream);

+ m\_archiverModule.DearchiveInventory(m\_uaLL1.FirstName, m\_uaLL1.LastName, "/", m\_iarStream);

InventoryItemBase coaItem

= InventoryArchiveUtils.FindItemByPath(m\_scene.InventoryService, m\_uaLL1.PrincipalID, m\_coaItemName);

@@ -107,7 +107,7 @@

UserAccountHelpers.CreateUserWithInventory(m\_scene, m\_uaLL1, "meowfood");

- m\_archiverModule.DearchiveInventory(m\_uaLL1.FirstName, m\_uaLL1.LastName, "/", "meowfood", m\_iarStream);

+ m\_archiverModule.DearchiveInventory(m\_uaLL1.FirstName, m\_uaLL1.LastName, "/", m\_iarStream);

InventoryItemBase foundItem1

= InventoryArchiveUtils.FindItemByPath(m\_scene.InventoryService, m\_uaLL1.PrincipalID, m\_item1Name);

@@ -171,7 +171,7 @@

// log4net.Config.XmlConfigurator.Configure();

UserAccountHelpers.CreateUserWithInventory(m\_scene, m\_uaMT, "password");

- m\_archiverModule.DearchiveInventory(m\_uaMT.FirstName, m\_uaMT.LastName, "/", "password", m\_iarStream);

+ m\_archiverModule.DearchiveInventory(m\_uaMT.FirstName, m\_uaMT.LastName, "/", m\_iarStream);

InventoryItemBase foundItem1

= InventoryArchiveUtils.FindItemByPath(m\_scene.InventoryService, m\_uaMT.PrincipalID, m\_item1Name);

**--- ./OpenSim/Region/Framework/Interfaces/IInventoryArchiverModule.cs.orig 2013-10-16 00:55:45.000000000 +0000**

+++ ./OpenSim/Region/Framework/Interfaces/IInventoryArchiverModule.cs 2013-10-16 00:57:59.000000000 +0000

@@ -59,7 +59,7 @@

/// <param name="invPath">The inventory path in which to place the loaded folders and items</param>

/// <param name="loadStream">The stream from which the inventory archive will be loaded</param>

/// <returns>true if the first stage of the operation succeeded, false otherwise</returns>

- bool DearchiveInventory(string firstName, string lastName, string invPath, string pass, Stream loadStream);

+ bool DearchiveInventory(string firstName, string lastName, string invPath, Stream loadStream);

/// <summary>

/// Dearchive a user's inventory folder from the given stream

@@ -72,7 +72,7 @@

/// the loaded IAR with existing folders where possible.</param>

/// <returns>true if the first stage of the operation succeeded, false otherwise</returns>

bool DearchiveInventory(

- string firstName, string lastName, string invPath, string pass, Stream loadStream,

+ string firstName, string lastName, string invPath, Stream loadStream,

Dictionary<string, object> options);

/// <summary>

@@ -84,7 +84,7 @@

/// <param name="invPath">The inventory path from which the inventory should be saved.</param>

/// <param name="saveStream">The stream to which the inventory archive will be saved</param>

/// <returns>true if the first stage of the operation succeeded, false otherwise</returns>

- bool ArchiveInventory(Guid id, string firstName, string lastName, string invPath, string pass, Stream saveStream);

+ bool ArchiveInventory(Guid id, string firstName, string lastName, string invPath, Stream saveStream);

/// <summary>

/// Archive a user's inventory folder to the given stream

@@ -97,7 +97,7 @@

/// <param name="options">Archiving options. Currently, there are none.</param>

/// <returns>true if the first stage of the operation succeeded, false otherwise</returns>

bool ArchiveInventory(

- Guid id, string firstName, string lastName, string invPath, string pass, Stream saveStream,

+ Guid id, string firstName, string lastName, string invPath, Stream saveStream,

Dictionary<string, object> options);

}

}

\....end