The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

User Guide

Contents

Entries marked with ✕ are only available in the registered version.

INTRODUCTION

With the release of Microsoft Flight Simulator X, the new Missions feature looked like a great way of adding customisable scenarios. However, as Microsoft pointed out in their SDK docs, the supplied 'Object Placement Tool' – the mission editor – is far from easy to use.

This program is an attempt to cover what I felt to be the shortcomings in the default tool. It is designed to help you create and test your mission logic without worrying about the technical details, while still using the original tool for what it does best - placing scenery.

REQUIREMENTS

- Microsoft Flight Simulator X Deluxe Edition

- FSX SP1 or higher to use debugging features

- Microsoft Windows XP or Vista

- 25Mb Diskspace

FEATURES

- Visual interface instead of list-based

- Drag-n-drop connections for linked nodes

- Full error-checking

- Configurable layout rules, to follow your expected mission flow

- Create synthetic speech for dialogs, for quick testing

- Add comments to the display

- Based largely on Microsoft's own configuration files

- Mission Wizard to create simple point-to-point missions

... plus, for registered users ...

- 'Recipes' – collections of related nodes that do a single job

- Mission Wizard will create a mission based on a FlightPlan

- Runtime monitoring and debugging of missions

- Creates installable missions using MSI

- Create mission patches also using MSI

- Create error reports

- Create scripts for voice-actors

- Extendable using XML files to allow SimConnect extension authors to use visual editing with their custom actions

- Unlimited nodes; the free version is limited to 50

- Paste function works with unlimited nodes; the free version will only paste one node at a time

COPYRIGHTS AND LICENSING

Most software has some kind of licence restrictions, even if nobody ever reads them! This is no exception.

Let's get the 'small print' over with first. We know that anything in this big bad world can be reproduced within minutes, no matter the copy protection, no matter what we write or how much we threaten, so the following text is just meant for decent people to read. The ones we know that will take it to heart. The ones that have just PURCHASED this product! In other words: YOU! (And, by the way, thank you for that!).

The intention is that the editor will be usable for free, but limited.

If you choose to buy a licence (thank you!) you get all the features, and also some more entitlements. To enter your licence, use the "Help®Enter Licence..." menu item, paste the licence key into the box and restart the software.

If you installed using the Flight1 wrapper, it will be licensed automatically.

The mission system command extension (simvar.exe) may always be used in any mission for free. It should not be distributed on its own, only with a mission you have written that requires it.

It may also be made available by the publisher (FSAddon) as a separate download, which you can refer to in your installation instructions. However, it may only be *distributed* with your missions provided you are entitled to distribute the mission itself:

- If you are using the free version of the editor, you may use it to create missions which are themselves distributed for free. You may distribute the command extension with your missions.

- If you as an individual have paid for a licence for the editor, you may use it to create missions which will be sold as shareware for a small fee (less than $10 / €10). You may distribute the command extension with your missions.

- If you are producing missions commercially, either for your own company or contracted to another company, you need to approach FSAddon for commercial licensing permission. This explicitly includes the use of any feature, including debugging, error reporting and packaging, and is not limited to actual mission script creation.

Only when commercial licensing is agreed can you distribute the command extension with your missions.

All FSAddon.com products are commercial products and copyrighted as such. This means that no product, or any part of it, may be copied, reproduced or disassembled in any way, nor published in any way and by any means, without written agreement from FSAddon.com. The same holds true for any registration or license key or any other means of product protection.

The product is provided 'as is' and the publisher, author(s) and distributor(s) do not accept any liability for any damages of any kind resulting from the use of the product in any way.

This product must not be used for real world training or other real world usage of any kind.

After purchasing this product, FS Mission Editor by Jim Keir, you may install and use it on ONE computer only for your private use, except when agreed additional terms via a Commercial License. And you know what? If you really want to make a backup copy for safety reasons only and promise NOT to use it for anything else (like giving it to your best friend), we don't even blame you!

P3D Mission Editor software and manual is copyright of Jim Keir.

Accompanying FSAddon websites/webpages are copyright of François Dumas.

All publishing rights for the licensed/commercial versions reside with FSAddon Publishing.

'Flight Simulator X' is copyright of Microsoft Corporation.

FURTHER READING

What this document won't do is teach you how to build missions. There are others which will give you a headstart, not least the SDK "Mission Creation" document, as well as two 'getting started' guides available on http://www.fsinsider.com written by Microsoft:

http://www.fsinsider.com/developers/Pages/MissionBuildingTips.aspx

It is also extremely useful to look at others' missions, in particular the example missions that are supplied with the P3D Missions SDK. Any mission downloaded from e.g. http://www.flightsim.com or http://www.avsim.com can be used for learning too.

For more mission-building tips, there are some particularly useful places:

http://www.fsdeveloper.com/forum/forumdisplay.php?f=59

http://fsxmission.com/live/modules.php?name=Forums

http://www.fsdeveloper.com/wiki/index.php?title=Mission_Creation

Finally, the support forum for the editor can be found at:

http://forums.fsaddon.eu/viewforum.php?f=22

SUPPORT

If you need any support in installing or using P3D Mission Editor, you can get it in different ways:

Register on the FSAddon.eu forums (go here: http://forums.fsaddon.eu/index.php and click on "Register") and then go to the P3D Mission Editor Support Form which you can find by clicking here: http://forums.fsaddon.eu/viewforum.php?f=22

1) You MUST register before you can write messages.

OR

2) For COMMERCIAL ISSUES write us an email:

    francois@fssupport.com

OR

3) Write to simMarket support (if you bought it there) :

    https://secure.simmarket.com/ticket_create.php

We believe that support of a product, and especially products released for such a specialized audience as flight simmers, is of the utmost importance. Being flightsim freaks ourselves you can be assured that support has our fullest attention… Although we do not imagine you would need much with this product.

In any case, even if you just want to tell us what you think you are welcome on the FSAddon Forums and on the simFlight Forums. There you will also find a bunch of like-minded flight simmers to chat with you about this title, and about just anything else flightsim related. Give it a try!

INSTALLATION

Installation is like most other Windows programs – run Setup.exe and it will check that you have everything you need, then install the editor.

Before you can use it, it must have access to some of the P3D configuration files. This means that it must be installed at least once on a PC with P3D installed.

INSTALLATION WITH FSX

The first time it is run, and anytime after that it detects changes, it will offer to correct some errors in a file provided by P3D called "`<P3D Path>\propdefs\propmission.xml`". Normally you should allow these changes to be made. If you are not sure, or want to make the changes for yourself, this is what is being changed (and why):

1) The "LandingType" definition is incorrectly defined, because XML is case-sensitive. To fix this, search for "LandingType". Change 'type = "Enum"' to 'type = "ENUM"'. It should look like this:

```
<PropertyDef
    name        = "LandingType"
    id          = "{61FC2643-E3A7-4a1c-8F33-C83637A4D3CB}"
    type        = "ENUM"
    default     = "FullStop"
    descr       = "Type of Landing to detect">
```

2) The "PropertySource" definition has no default entry although both FSX and the default mission designer use one. This needs to be added. Search for "PropertySource" and add a new attribute after "type", as 'default = "Reference"'.

```
<PropertyDef
    name        = "PropertySource"
    id          = "{6D13F898-3B25-475a-93EA-0951FFB6CABB}"
    type        = "ENUM"
    default     = "Reference"
```

3) The "StopTime" property has no default entry although both FSX and the default mission designer use one. Search for the first occurrence of "StopTime" and add a new attribute after "type", as 'default = "5"'.

```
<PropertyDef
    name        = "StopTime"
    id          = "{7656871B-A49E-4b34-B4F0-AA5931E8CC98}"
    type        = "FLOAT"
    default     = "5"
```

The editor will work without these changes being made, but you will lose the ability to use drop-down lists in LandingTriggers and some types of Condition and you will get unnecessary warnings when loading missions created using the default FSX Object Placement Tool.

CHECK OBJECT PLACEMENT TOOL INSTALLATION

If you are having trouble getting the Object Placement Tool (OPT) set up correctly, choose the 'Help→Check OPT Setup' menu item in the editor. This will verify that you have the right version of the SDK installed, and that it is registered properly with FSX. If it isn't, it will register it for you.

It will check that:

– FSX is installed

– The SDK is installed

– The Object Placement Tool DLL is in the correct place

– The version of the OPT matches the version of FSX

– It is correctly registered and enabled in dll.xml, and will create a new one if it is damaged or missing

INSTALLATION WITHOUT FSX

You can also install this on a machine without FSX available, for example if you want to continue your mission design on the move but your laptop isn't up to running FSX.

Install FSXME as normal by double-clicking 'Setup.exe' and following the prompts. Next you need to copy a folder from your main PC, having already run FSXME on it at least once.

You will need to find your 'Common Application Data' folder. Likely locations are:

XP : `C:\Documents and Settings\All Users\Application Data\FSAddon\FSXME`

Vista : `C:\ProgramData\FSAddon\FSXME`

This folder may be hidden, but it will usually be in one of these locations. Once you have found it, you will need to copy this folder to the equivalent place on the PC without FSX installed. Note that this folder isn't in a fixed location, so take care if you are copying between different versions of Windows.

You can also find the location of this folder on the Paths tab of the 'Edit→Preferences' dialog, as 'Fallback Path'. If you're unsure about where to copy the folder to, run the editor, ignore the various warnings that appear and check this dialog.

This folder will contain copies of some of your FSX configuration files. These can't be included in the FSXME setup program because they are owned by Microsoft and can't be redistributed. It also contains other files that are generated from any third-party BGLs, such as scenery addons, that you may have. These two sets of files are used extensively for error-checking, providing drop-down lists and default values.

FIRST RUN

There are two things you need to do the first time you run the FSX Mission Editor.

The first time you run FSXME it will offer to create lists based on your particular FSX setup. It will work without this, but you will not be able to use drop-down lists when selecting models or airport, or use the Wizard to create new missions. These lists can be recreated later if need be. This should be done if you add new model BGLs which you want to use in the editor.

When it is scanning the scenery files, it uses the FSX scenery library to work out where to look. Any files you have visible to FSX should be indexed, with some exceptions; specifically, any BGL file which starts with "OBX" will not be read. This is because FSX contains a large number of these files, each of which contains a large number of generic – that is, boring – buildings. Reading these files would take a very long time, and generate a list of tens of thousands of objects, most of which would be almost useless for designing add-on scenery and missions.

If you are using Vista and have done a default FSX installation, it will also offer to create a safe place to write new missions to. See below for an explanation of why this should be done.

VISTA ODDITIES

Although the editor works with Vista (it was developed on Vista) there are some things which may cause confusion. This is because of the way that Vista tries to protect system files.

If you have done a default installation of FSX, it will be stored under "C:\Program Files" or "C:\Program Files (x86)". This is a protected area in Vista, and the system will not normally let you write any files there. Unfortunately, that's where the missions need to be stored if FSX is to see them.

Vista is one step ahead; if any files *do* get written there, they will be quietly stored somewhere else and the filesystem will make them appear to be where you originally saved them. This means that you can almost ignore the file protection that Vista has in place. If the intricacies of the Vista filesystem don't interest you, just move right along to the next chapter!

The problem is that not every program will see the corrected - or 'virtualised' - version of the filesystem, with those files in place. Notably, Explorer doesn't show them in-place. It does add an extra button called "Compatibility Files" to show that there are virtualised *files* present, but this button doesn't appear if there are only virtualised *folders*. This means that your mission folder won't be visible from Explorer. Also, any virtualised files won't be visible to other users on the same computer, including the administrator.

There are two ways to deal with this. First, you can ignore it; most programs will read the virtualised files without realising it, provided that you run them both as the same user.

The other way is to effectively bypass the virtualisation so that files really do exist on disk. The easiest way to do this is to run the editor as an administrator, although this is not recommended because you will forfeit the numerous protections that Vista offers to normal users.

RUN AS ADMINISTRATOR

To force the editor to be run as an administrator every time, using Windows Explorer go to "C:\Program Files\FSAddon\FSX Mission Editor". Right-click on "MissionDisplay.exe" and choose "Properties", then the "Compatibility" tab. Tick "Run this program as administrator" and click "OK". Every time you run it, you will first be given the UAC permission dialog.

The other way to make the files be actually stored where you put them is to fool FSX into reading a different part of the disk when reading missions. They will then be visible to all users on the computer. This is both cleaner and safer, because Vista's file protection is still being used. It does take a little effort to set up though.

SETTING UP A SAFE MISSIONS FOLDER

Since Vista doesn't approve of people writing files into protected parts of the disk, we would ideally store our edited missions somewhere else. The problem is that FSX only looks in it's own Missions folder. We can fool it by inserting a symlink into the Mission folder which points to a part of the disk which can be written by everyone on the computer. A symlink can be seen as a fake directory which is actually stored somewhere else.
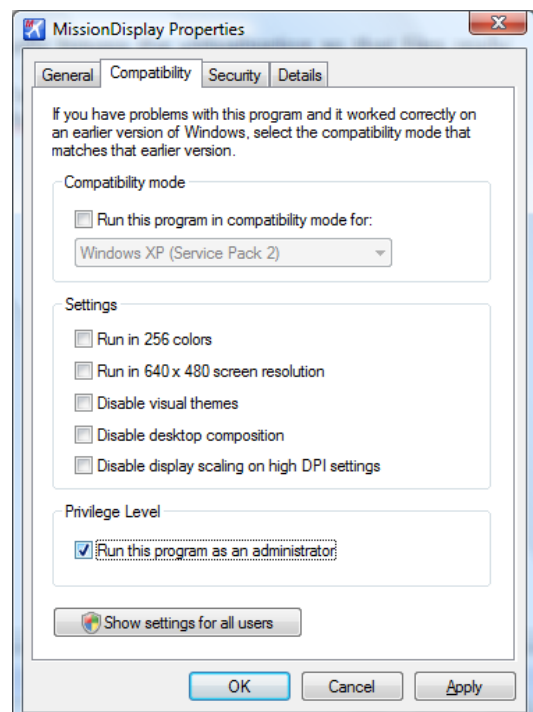
The good news is that the editor can set this up for you. When you run it the first time, it will check to see if this problem will affect you and if it will, you will be asked if you want to fix it.

Alternatively, you can do it yourself using a DOS window which has been run as an Administrator:

```
> cd "C:\Program Files\Microsoft Games\Microsoft Flight Simulator X\Missions"
> mkdir "%PUBLIC%\FSX Missions"
> mklink /D "My Missions" "%PUBLIC%\FSX Missions"
symbolic link created for My Missions <<===>> C:\Users\Public\FSX Missions
```

Either way, this will create a new folder in your FSX Missions folder called "My Missions". FSX, Explorer and all other programs will see this folder exactly as you would expect. However, the folder really exists on disk at "C:\Users\Public\FSX Missions" - and will appear in Explorer's "Public" folder.
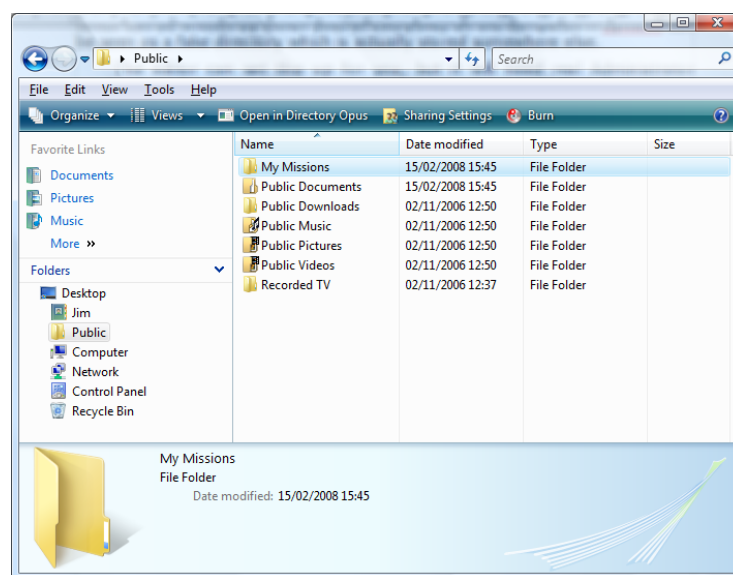
As long as you store your new missions in "[FSX]\Missions\My Missions", they will appear both in the public folder (which is safe to write to) and under "[FSX]\Missions\My Missions". If you store them under any other subfolder of "[FSX]\Missions", Vista will virtualise them, meaning they won't appear in the correct place in Windows Explorer. They will instead be stored in:

```
C:\Users\<Username>\AppData\Local\VirtualStore\Program Files\ ↵
              Microsoft Games\Microsoft Flight Simulator X\Missions\
```

WHAT'S IN A MISSION?

Chances are you've already got your storyline ideas. Putting them together into something that tells that story is the next step. So what do you need?

The main part of the mission is stored in an XML file. You don't need to be able to read that yourself, although you can if you want to. That's not as important as the logical structure of the mission.

There are three main types of thing you will use to create a mission. First, there are Triggers. These wait for something to happen – a timer, a counter, the player's plane reaching a certain altitude or entering or leaving a certain location etc. – and when it does, they make other things happen. These are what you use to tie the events in your mission to what the player is doing. Triggers can also be deactivated so that they don't do anything until you tell them to.

Second are Actions. These are what triggers will use to make things happen. For example you can start and stop timers, change counters, and – most importantly – activate and deactivate triggers.

Already you have the basics of how a mission would be put together. You could, for example, set up a Trigger which waits for a few seconds and then calls an Action which displays a message. Another Trigger might wait for the user to reach 1000ft AGL then display another message and enable a third trigger which will check for a safe landing at a given position or airport. Your story gets told by creating Triggers which test for everything you want the player to do, linked with Actions which provide feedback and activate the next Trigger in the story sequence.

The third type of thing you can use is scenery, both static and mobile or even AI (artificial intelligence) aircraft, cars, boats or animals. These can also be used to tell your story by placing visual clues for the player to follow.

It will probably help if you plan your mission out on paper first. You can get a clear idea of what happens when and what the conditions are. Scribbling away with an old-fashioned pencil and paper is a more intuitive way of experimenting without having to worry about working out how a new computer program works. Once you've got your plan, it should be fairly simple to translate that into sequences of Triggers and Actions to tell your story, and Scenery and AI objects to give your story somewhere to happen.

You will need some other files before your mission will work. Along with the XML mission description, you will need a starting location (.FLT), weather (.WX), some pictures and some HTML mission briefings. The FLT and WX files come from a saved flight, and the HTML briefings can be copied and modified from another mission or designed from scratch if you prefer. These are all described in the SDK's Missions document.

USING THE MISSION WIZARD

This is the quickest way of getting off the ground with a new mission. Instead of worrying about the different bits that are needed before FSX will let you start flying, you can get the computer to sort it all for you. The Mission Wizard will let you create a simple mission, either flying a circuit of an airport or flying from one airport to another, and fill in all the bits that FSX will look for. You can then edit them, and add bits to the mission, without having to worry about the mechanics of how it works.

To use the wizard, return to the Welcome page if need be by selecting 'File→New' or using the 'New Document' button on the toolbar. Now select the 'Create a basic tutorial mission' button.
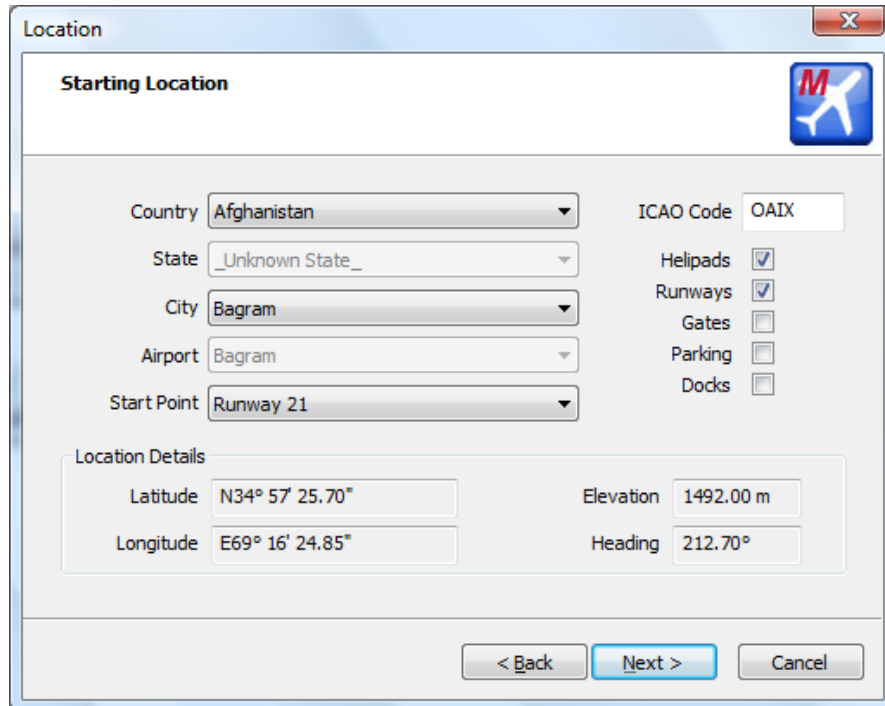
A standard introduction page appears; click Next to get into the main part of the Wizard.

For registered users, the first page gives you the option to load a flightplan and create a mission based around that. The flightplan must have been generated using FSX; older ones won't be read correctly.
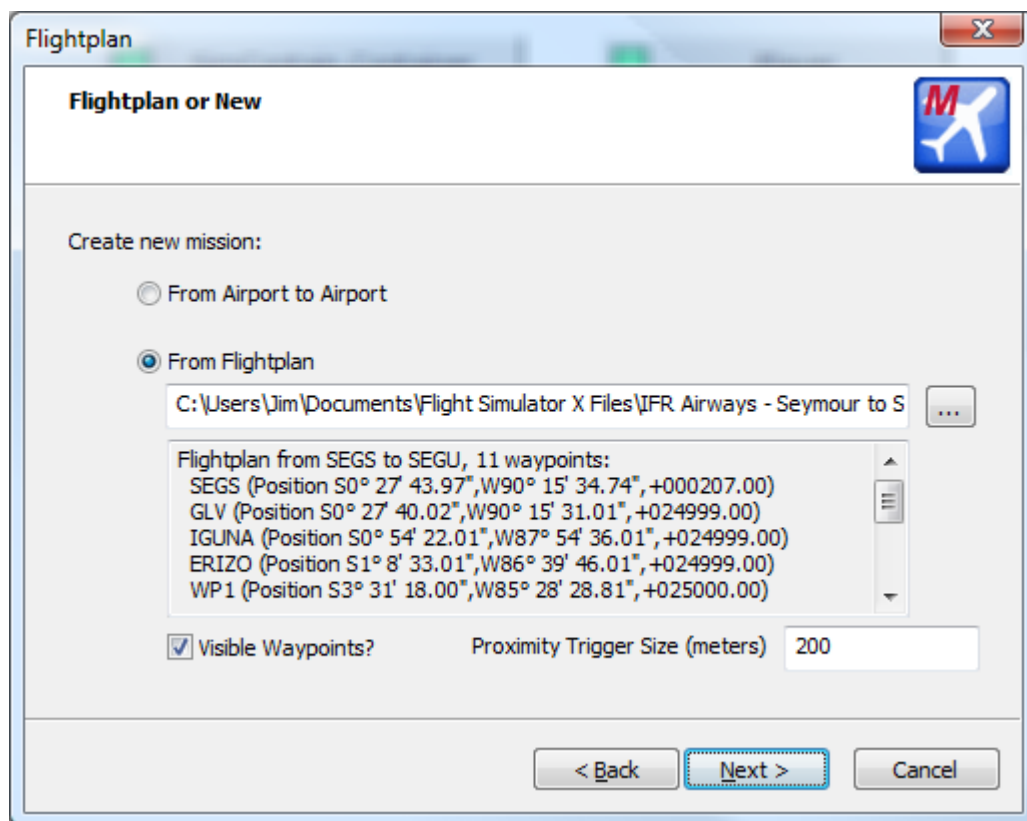
Each of the waypoints will be used to create a Point of Interest, a dialog and a trigger that activates the next waypoint. If you don't want to load a flightplan, you can select the 'From Airport to Airport' option to create a point-to-point mission. The 'Proximity Trigger Size' option allows you set the dimensions of the AreaRectangle used, between 10m and 5000m, and you can choose to have the waypoints marked as regular PointOfInterest items or by showing the size and direction of the entire waypoint.

For freeware users, the first page is for the starting location; that is, the airport that you want to take off from.

You can use the lists to narrow your choice down, or if you know the ICAO code for the airport you can just type it in. Because some airports can have many different starting points, such as Heathrow in London, you can narrow down the types of starting points shown using the tickboxes on the right. Basic information about the starting point you've chosen is shown at the bottom of the dialog.

Click 'Next' to go to the Destination Airport page. This starts with the same airport you just chose on the 'Starting Location' page, but you can change it to any airport you like.

The next page is the date and time for the flight to start. Simply choose the date you want and click Next again.

Now you choose the aircraft to start in. This is the same list that FSX offers.
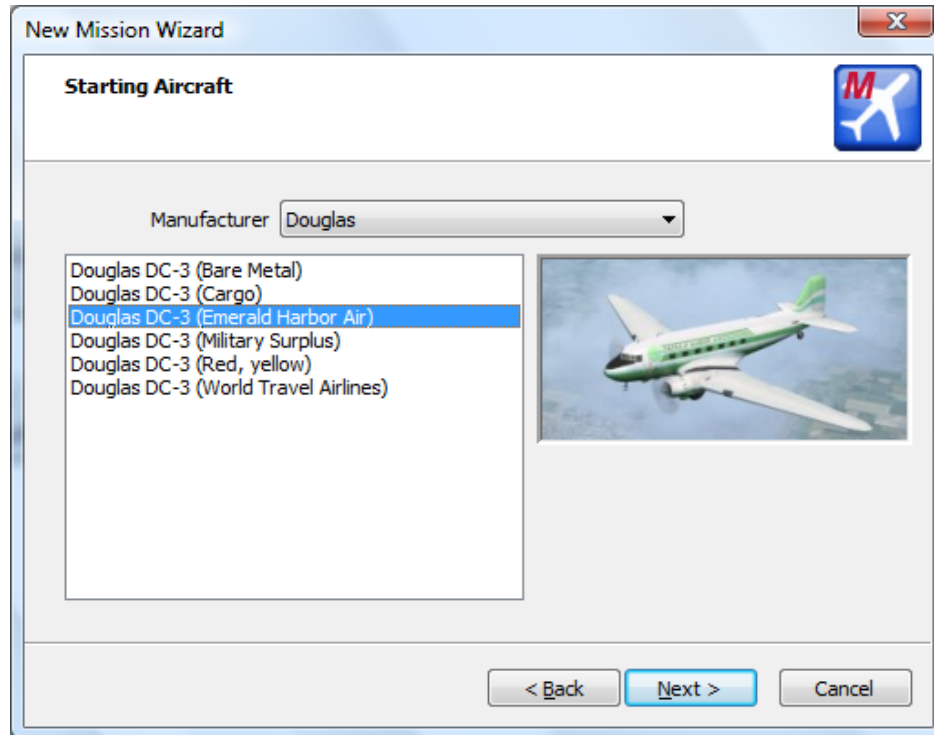
Use the Manufacturer list to narrow the choice down, or choose '_All_' to show all aircraft from every manufacturer.

Finally, you can change the mission title and description if the default ones don't suit.
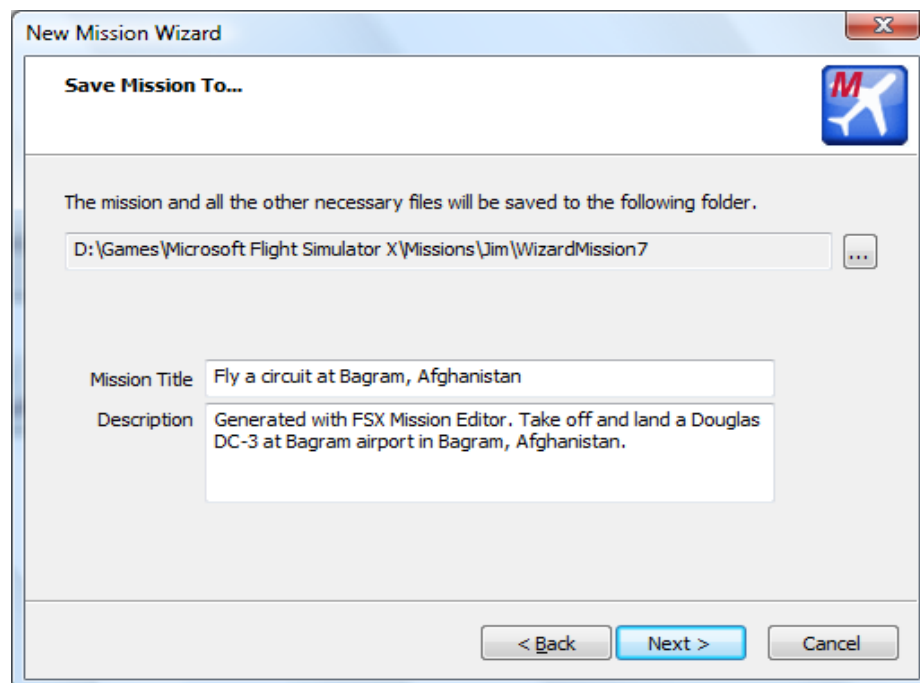
Click 'Next' then 'Finish' and your mission will be created, ready to use. FSX should recognise it the next time it is started, and the mission is already complete and ready to use. It won't be very exciting at this point though – that's up to you!
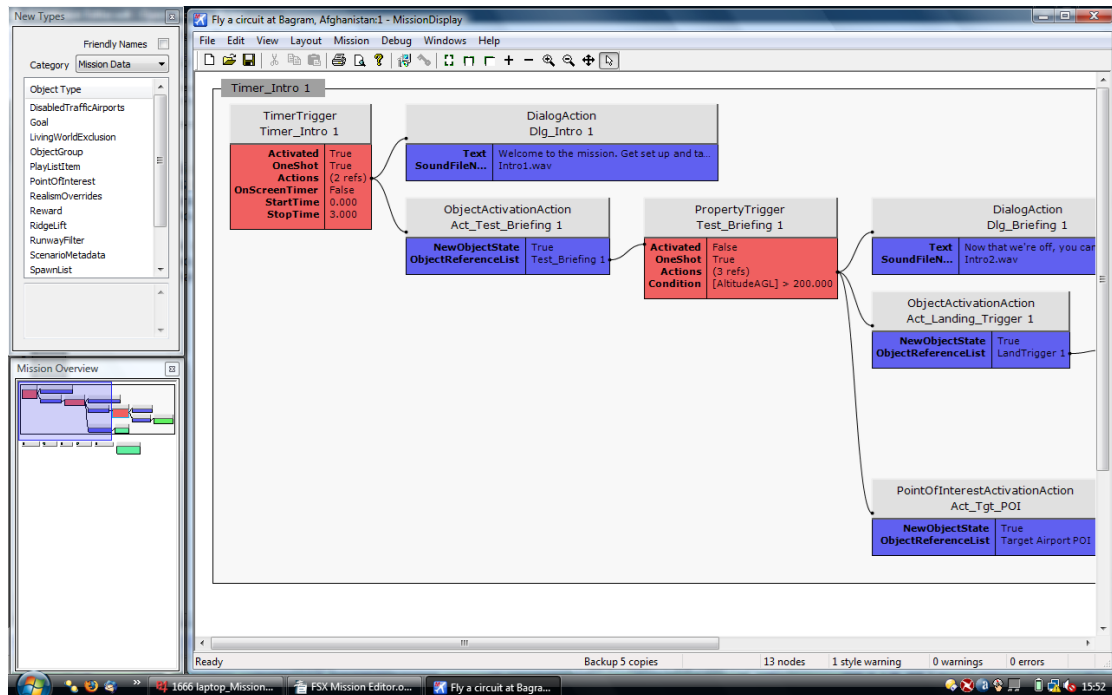
To spice things up a bit, you will need to use the other features of the program to add new triggers and actions, and link them together in a way that tells the story you have in mind for your mission.

When it's putting together the bits for your mission, the Wizard makes use of the 'Create Template Files' feature. This means that if you've customised your template files, your personal versions will be used for Wizard missions too.

GETTING AROUND

This section is a simple tour of the main features available to you. If you're the enquiring type this may be enough to get you going, ready to write or edit your own missions. Instead of using a Wizard-generated mission which only has a few different types of actions, this will use the 'Congo' mission that is supplied with the FSX SDK.

LAYOUT

There are many layout options and it can take a while to work out how you want things to be displayed. The easiest way to describe many of these is to use one of the FSX example missions.

Some layout options work globally, and others work on single nodes. A node is simply any entire instruction, whether that is a trigger, an action, a model etc.

Run FSXME by double-clicking "FSX Mission Editor.exe". The initial display should look something like this:
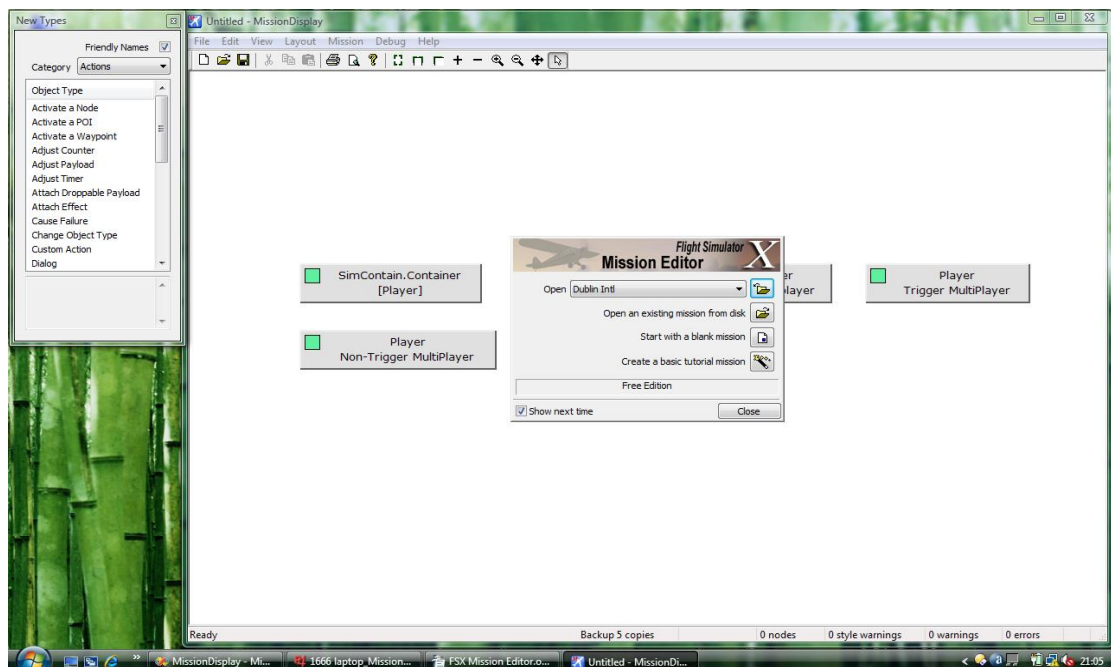
Choose the second option on the Startup window, "Open an existing mission from disk". From the FSX SDK folder, load "`Mission Creation Kit\Mission Samples\Backcountry\Congo\Congo.xml`". You could also do this by using the "File→Open" menu item, or just by dragging the XML file onto the main window. For now, close the small window called "New Types" – that's used for modifying a mission – and also open the Overview window using the "View→Overview" menu item. You can arrange these windows however you want them.
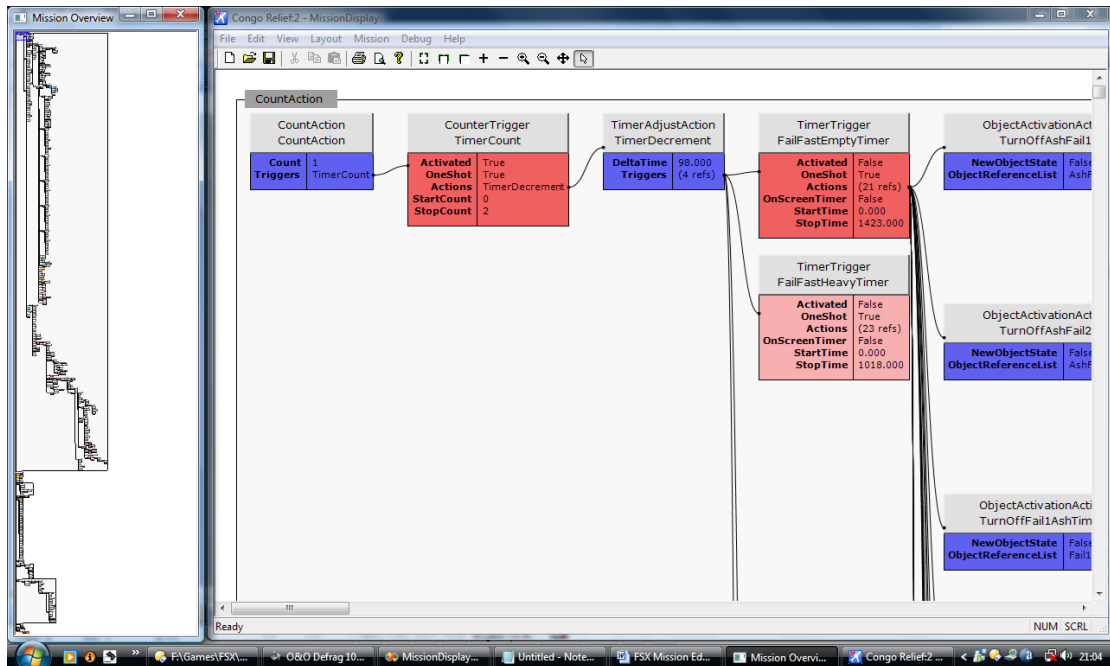
NAVIGATION

First, take a look in the Overview window, on the left in this screenshot. This can be used as a quick way of getting around the mission once it gets larger than a few nodes. At the top left is a small blue box. This shows the place in the mission that's being displayed in the main window. You can click on the overview window to change the view in the main window.

Try zooming out in the main window, either by clicking on the "-" button on the toolbar and then clicking somewhere in the window, or more easily by pressing "-" on the keyboard. The blue square in the Overview window will get larger. You can zoom in again using the "+" button, or by pressing "+" on the keyboard. You can also zoom to a particular area using the magnifying-glass button and then dragging a box round the area to zoom to, or by pressing Ctrl and scrolling the mouse wheel. Select the Arrow toolbar button to get back to normal mode from magnify mode.

There are three predefined zoom levels, each of which have buttons on the toolbar:

Zoom to entire mission.

Zoom to page width.

Zoom to 1:1.

You can re-centre the main window by clicking in the Overview window, or by simply using the scrollbars as with any Windows application. Holding down the middle mouse button and dragging will move the display around too.

Other keyboard shortcuts are "Ctrl+Home" for going to the very top of the mission and "Ctrl+End" for going to the end.

WHAT'S DISPLAYED

The overview window, on the left in the screenshots above, can't be used for anything other than getting around the mission. All of the real mission data is shown in the main window. Assuming you've still got Congo.xml loaded, reset the zoom to 1:1 and go to the top of the document.

There are lots of coloured boxes on screen, each of which is a single node. Each one has a grey header, which has the which has the node type displayed above the node name. The main part is coloured according to what type of job the node does, and contains a list of properties and their values. Which properties are shown will depend on the type of node, and some of the settings.

To try and keep the main display clean, not all properties will normally be shown. Only the ones which are most used are shown by default. You can change this using the "View→All Attributes" menu item. Choose this, and now on the node called "TimerCount" - the first red one in the display - the "Latched" and "Counter" properties are shown. To switch off the full display, select "View→All Attributes" again.

Another set of properties are also normally hidden. These are used by FSX when it's saving a mission that's in progress, to keep the current state. In a new mission they're no use. You can have these shown, if you need them, by using the "View→Private Attributes" menu.

Most of the nodes have lines linking them to others. These show the associations between nodes and should be read left-to-right, top-to-bottom. For example, the "CountAction" node is linked to the "TimerCount" node. The exact meanings of the nodes are detailed in the FSX SDK, but in this case it's saying that the "CountAction" node will add one to the "TimerCount" TimerTrigger.

It is possible for a node to be linked to more than once; for example, a trigger may be referenced by two ObjectActivation nodes, one to enable it and one to disable it. Only one copy of any node will be shown in full; that is, including any child nodes. All other copies will be shown using a paler colour, and will not have child nodes.

For example, "FailFastEmptyTimer" is shown in the screenshot below in deep red while "FailFastHeavyTimer" beneath it is lighter. This shows that "FailFastEmptyTimer" is shown in full here, while "FailFastHeavyTimer" is shown in full somewhere else. That is, the visible copy does not have any child nodes shown.

Last, in the top left corner of the window is a small grey box with the word "CountAction". Coming out of this is a paler grey box which covers a large area, which you can see on the Overview window extending about three quarters of the way to the bottom. This pale grey box surrounds a group of linked nodes and takes its name from the first one in the tree. Ideally this would be arranged so that the first node shown is the first thing that happens in the mission.

If you double-click on the group title, all the nodes in that group are selected.

Group name for all nodes linked to the first one
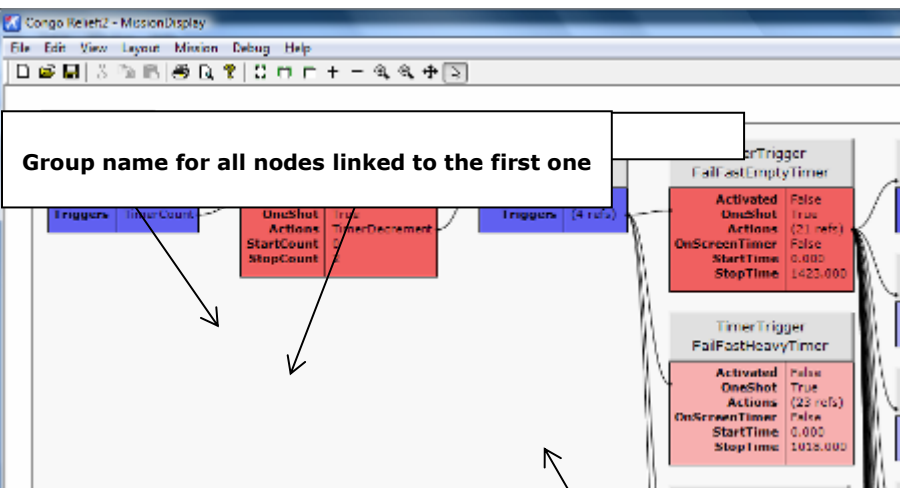
Bright node with children displayed

Pale node with no children displayed

Property names for this node **values for this node**

ERROR HIGHLIGHTING

Next let's have a look at the error highlighting. Press Ctrl+F or select the "Edit→Find" menu, and find the node called "OffAirportLand". Double-click the item in the list to centre the main window on it, then close the Find dialog. You should now see this:
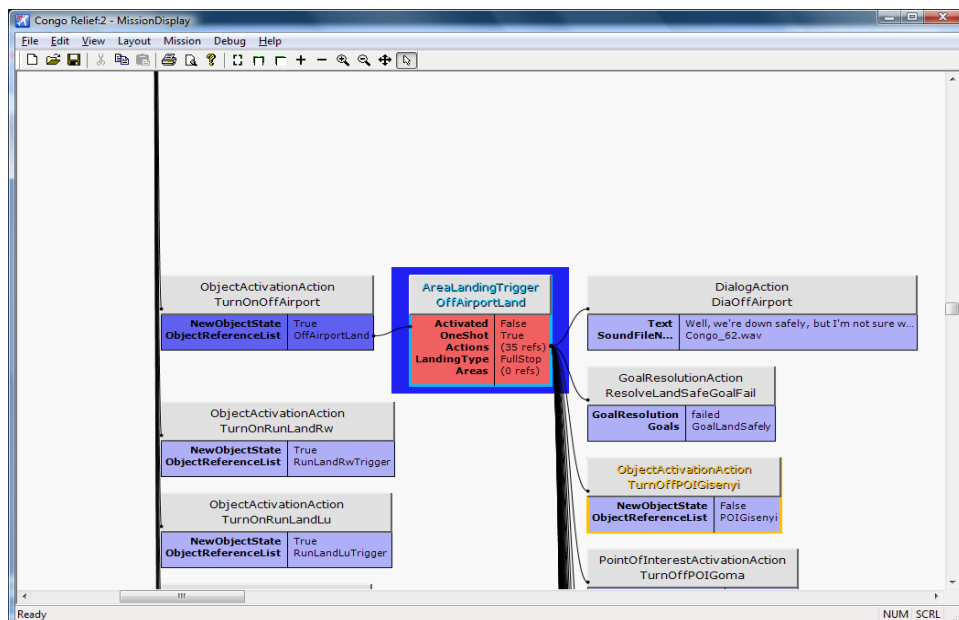
The node that you found, "OffAirportLand", is surrounded by a dark blue box. This shows that it is currently selected. However, there is also a pale blue line around that node, and the name is also shown in pale blue. These colours show that there is a problem with this node. In this case, the blue colour means it's only a "style" error, or something which may well be correct but is worth pointing out. Find out what the error is by right-clicking on the selected node and selecting "List Errors" from the context menu or using the keyboard shortcut, Shift-F6. You should get a single error:
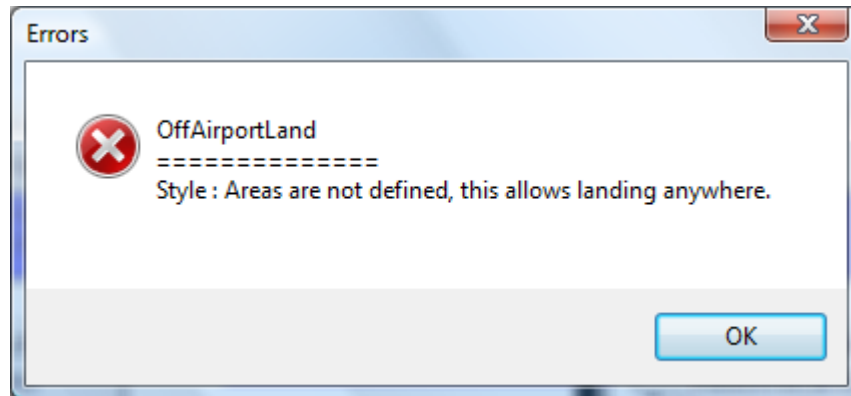
For style errors, it is up to you to decide whether or not they are a problem. Other errors may be marked as "Warning" or "Error" depending on how severe the problem is; warnings are yellow and errors are red. A warning means that something is almost certainly wrong, and an error shows that something is definitely wrong. Ignoring a warning may mean that your mission may not do what you intend. Ignoring an error may mean that it will probably not work at all, or even that FSX will refuse to load the entire mission.

List the errors for "TurnOffPOIGisenyi", just below and right of "OffAirportLand", which is marked in yellow and has a warning.
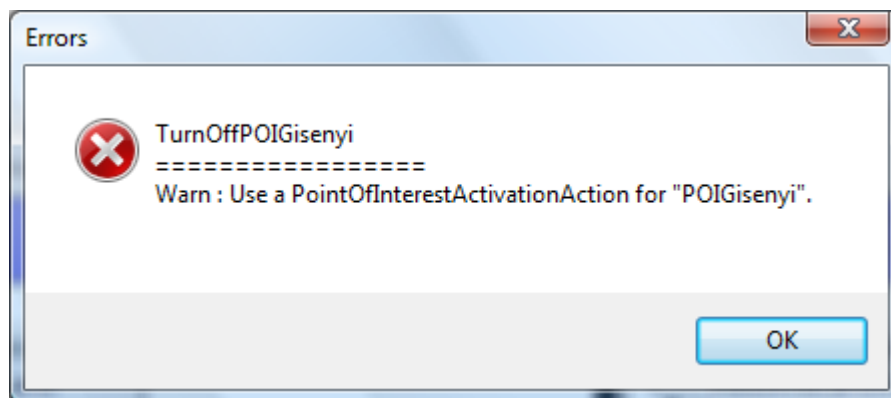
In this case, the warning is saying that a Point of Interest is being activated with a regular ObjectActivationAction; this works, but is not what is recommended in the SDK documentation.

Search for the node called "AdjustPayloadEquip" using the Find dialog (Ctrl+F). It is outlined in red, showing an error that must be fixed. In this case, 492 pounds are being added to an aircraft, but it is not specified to which aircraft the payload should be added.

CHANGING THE LAYOUT

Now that the nodes, their contents and colours, and the error highlighting are clear (I hope!) let's see how to rearrange them.

The layout has taken into account which nodes are connected to others, and which are activated by default. If there are many triggers which are active though, it can't know which ones come first. It has chosen "CountAction" as the first node, but this isn't a good choice. However, it will recalculate the layout if it is told to, based on hints you give on individual nodes.

Find the node called "KatUnloadTrigger", and right-click to bring up the context menu. Select "Allow Parent", which should be ticked, to de-select it. This means that "KatUnloadTrigger" will never be shown with a parent node, making it always be used as the first node of a group. Find it again, and it should be in its own small group.
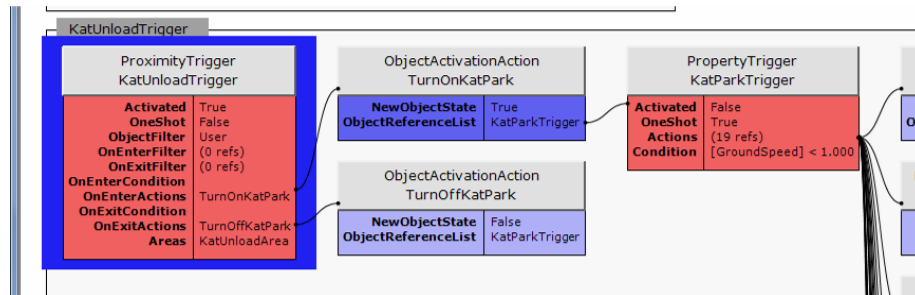
Let's separate some more areas. Search for "AshTrigger" and de-select "Allow Parent" on the context menu. Note that "AshTrigger" is activated by default – the "Activated" attribute is set to "True" – which means that if you were to fly straight to that area instead of following the instructions in the mission, it would still fire.

Do the same for "EnrouteKatTrigger", "KatApprRadTrigger" and "KatTrigger", all of which are also activated by default. Looking at the overview window now, the mission is split into two big sections and several small ones. Let's try and simplify it a bit more.

Search for "FailTimer". If you look down the list of child nodes, it starts with two GoalResolutionActions which cause their goals to fail, and then a load of ObjectActivationActions which switch things off. "FailTimer" is failing the mission after a set time – 73 minutes, or 4380 seconds – and then switching everything off. It's a simple action, but it's taking a lot of screen space. Right-click on "FailTimer" and select "Hide Children". This will collapse all the child nodes to a single, small box:

Note the small, grey box on the right with the "+" symbol. This shows that there are hidden children. We've just made things look much simpler. As a reminder though, you can add a comment to this node. Press ";" or select "Edit Comment" from the context-menu. A new dialog appears where you can type anything you like. Add something like "Fails mission and cleans up triggers" and click "Ok". You should now see this:
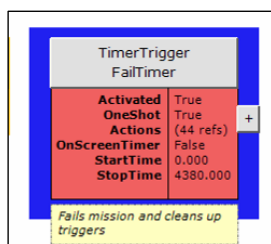
The comment has been added to the bottom of the node, in the yellow box.
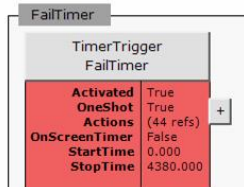
There are several other triggers which do a similar thing. Find "GomaLand" and look at its children. It enables two goals for safe landings, fails them, and marks "GoalLandSafely" as complete. Again, this completes the mission and switches off any active triggers so let's clean this one in the same way. Right-click on "GomaLand", select "Hide Children" and then add a comment like "Land safely but fail to drop cargo at Katale and land at Lubero".

"GisenyiLand", "RutAshLand", "RwinAshLand" and "LubAshLand" are all similar, so hide and comment them to make the main display even simpler. "FailSlowHeavyTimer", "FailFastHeavyTimer" and "FailSlowEmptyTimer" are similar; lots of object activations and deactivations, and some dialogs. Add a comment of "Engine 1 fails, head for Rwindi" and hide the children. Last, find "OffAirportLand" again and hide it's children. Once again, it is simply deactivating lots of objects and setting goals to finish the mission.

Now that all these changes have been made, you should save the mission. Note that the actual mission itself hasn't been changed at all, it's just the layout. This is saved in a separate file which is automatically created when

you save the main mission. You can also just save the layout by choosing the "File→Save Layout Only" menu item.
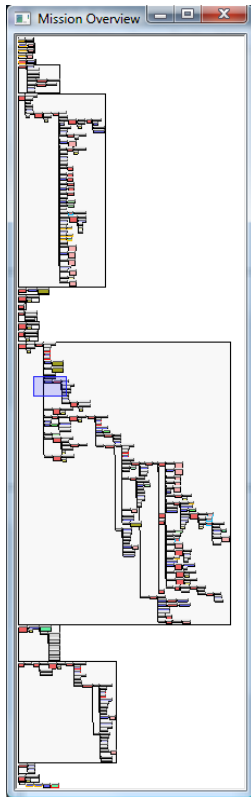
Now that a lot of the mission 'housekeeping' is hidden, the overview should look like this. You can see the overall shape of the mission fairly clearly now. The first big grey box has a lot of nodes all coming from a single parent, meaning that that parent is particularly important. The next big group has a nice clear tree of events, showing that the mission has a well-defined structure. There are also some smaller groups which aren't yet linked to anything.

CHANGING THE APPEARANCE

There are some settings that can be changed which make the entire display look different. These can be found by using the "Edit→Preferences" menu item.

GENERAL LAYOUT

The values here should mostly be self-explanatory. 'Maximum node width' controls the largest size that a single node can reach. This is important for things like Dialog nodes, where the dialog text may be quite long. Without a width limit, these nodes could get very large. "Distance between groups" sets how much space to leave between the big groups of connected nodes; this may help make the overview window clearer if it is set to larger values.

The 'Fonts' section controls the fonts that will be used in different parts of the display.
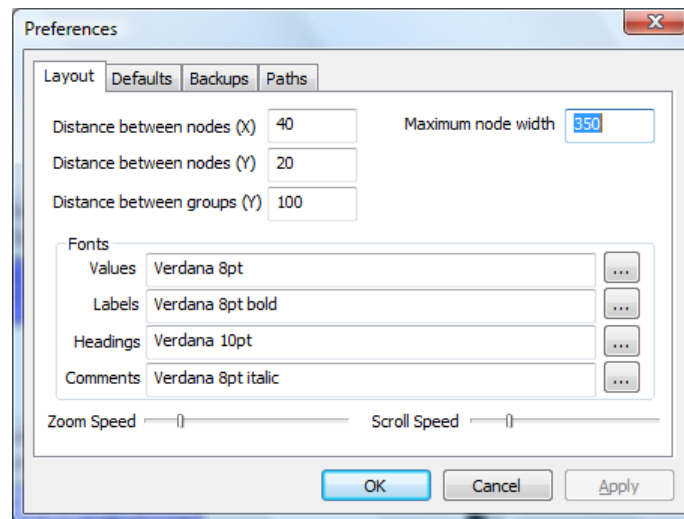
"Scroll Speed" controls how fast the mouse-wheel will scroll the main display, and "Zoom Speed" controls how fast the mouse-wheel will zoom the main display when Ctrl is pressed.

NAMES AND COLOURS

As you have seen, each type of node has its own colour to make it easier to identify. Both the colours and the 'friendly names' that can be shown in the Action Palette can be changed.

To change the name, double-click on the item in the "User Name" column. An edit box will appear, and you
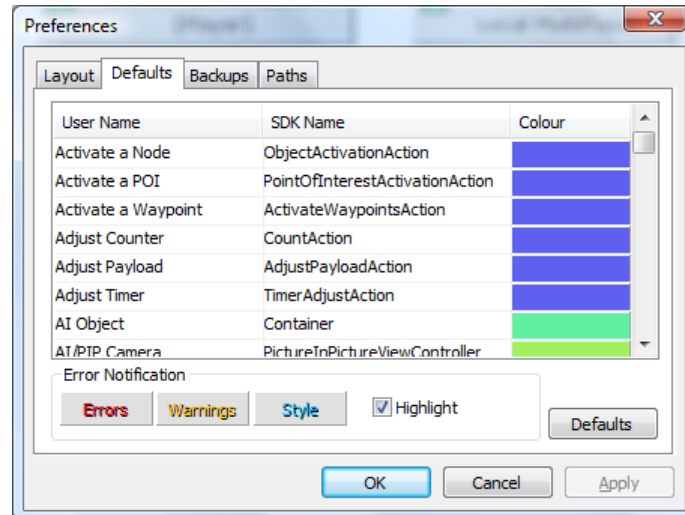
can change the name to whatever you wish. Similarly, to edit the colour double-click in the "Colour" column and pick the new colour.

You can also choose the colours that will be used to show errors, warnings and style warnings by clicking on the buttons under "Error Notification". The 'Highlight' tickbox controls the black 'shadow' text that is used to make error messages stand out from the background. If you prefer to use have the coloured text with no shadow, you can turn it off here.

These settings will be used for all missions. If you want to change back to the original names and colours, press the "Defaults" button.

EDITING

We can now start to make some small changes to this mission. For now, let's just fine-tune the structure to make it look better on screen. This may also help in performance tuning. If you have, say, ten active ProximityTriggers then FSX will need to check each of them with every pass through the mission. If they are structured so that you can't reach one without passing through a previous one it may help to switch the later ones off to begin with. I imagine it's faster to check for something being activated than it is to make the actual property check.

BASICS

Earlier we went through several triggers which were Activated by default, setting them as group headers by switching off "Allow Parent".

Search for "IntroTimer". This is active by default, and fires after three seconds. Not much else is going to happen in that time, so let's assume this is the first trigger to fire. In fact there is an earlier one but it only starts an effect, so we can ignore it.

There are some other active triggers that look as though they fire early, "V1Trigger", "100AGLTrigger" and "300AGLTrigger". You're not going to get to V1 speed in 3 seconds, so that one fires after "IntroTimer". Similarly, the "100AGLTrigger" must come after "V1Trigger", and "300AGLTrigger" after "100AGLTrigger". Let's link these to show that logic.

USING THE ACTION PALETTE

Open the Action Palette using the "View→Action Palette" menu. This window should appear:
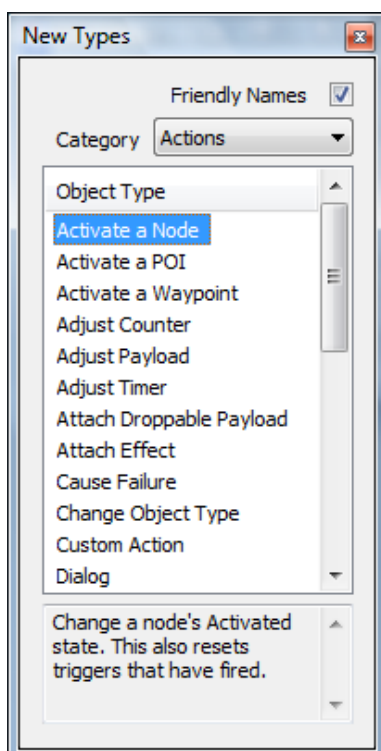
This is where you create new nodes. All of the available nodes are listed here, and split into different categories to try and make then easier to find. You can choose which category gets shown in the main list, or choose to show every available type.

Beneath the list is a short description of what this node does.

Last, there is a tickbox at the top labelled 'Friendly Names'. This switches the list from showing descriptive names for the various node types to showing the 'official' names – the ones defined in the SDK. You can edit the 'friendly' names to suit your own preferences using the 'Defaults' tab in the Preferences dialog.

Let's add a new action node. Change the Category list to "Actions" to shorten the list, and then select "Activate a Node" from the list. A description of the action's effect is shown at the bottom of the window.

On the main display, double-click in the red area of "IntroTimer" to bring up the Attribute Editor. Halfway down the list of attributes is "Actions", showing "3 refs, drop more here". These are the actions that get performed when this trigger fires.

Now, on the Action Palette, double-click "Activate a Node" to create a new action. You will get a new ObjectActivationAction. The white square in the top right tells you that this node has not been saved, and the red border and text show that there are errors. In this case, the error is simply that it doesn't activate anything.

Next, select this node and drag it over the Attribute Editor, then drop it on the box to the right of the "Actions" attribute. The display will change, showing the new node with "IntroTimer" as its parent. "IntroTimer" now also has the white box showing that it has changed since the last save. Now double-click on the coloured area of "New Activate a Node" to show it's attributes. Note that the ObjectReferenceList attribute is shown in red, indicating an error. Next, find "V1Trigger" and drop it on the "ObjectReferenceList" attribute of "New Activate a Node" to link it. The only thing left to do is switch "V1Trigger" off by default – double-click it and untick the "Activated" attribute.

You can rename "New Activate a Node" using the context menu (right-click and select Rename) or by double-clicking in the grey header area. To follow the naming convention of the rest of the mission, rename it to "TurnOnV1Trigger".

Looking at "IntroTimer", you can see that it now has four Actions to perform. The new one, "TurnOnV1Trigger", is the last one, after three dialogs. Since the dialogs will be on-screen for several seconds each, "V1Trigger" won't be activated until about ten seconds after IntroTimer has fired. In this case it's not really a problem, but sometimes that delay may be important. You can re-order the Actions by simply selecting "TurnOnV1Trigger" on the main display and dragging it to the top of the list. A black bar will appear to show you where it's being dragged to.

At this point you might want to save the changes, so – since we're modifying one of the demo missions – use the "File→Save As…" menu to give it a different name. The white 'modified' flags should disappear once it has been saved.

You can follow the same procedure for "100AGLTrigger"; create a new ObjectActivationAction, rename it to "TurnOn100AGL", link it to the "Actions" attribute of "V1Trigger", link "100AGLTrigger" to the "ObjectReferenceList" attribute of the "V1Trigger" and deactivate "100AGLTrigger". "300AGLTrigger" should then be linked to "100AGLTrigger" in the same way.

The start of the mission is beginning to come together, but then stops after passing through 300m. To find what happens next, bring up the Find dialog, change "_Any Type_" to "ProximityTrigger" and search for "Kat". There are five triggers found, four of which are Activated.

Ignoring "KatTakeoffTrigger" – you can't take off until you've landed, so that must happen last – double-click on each of the other triggers. Looking at the Areas associated with each, they are all centered on almost the same place, but with different values for Width, Length and Height. The one with the largest box must come first, in this case "EnrouteKatTrigger".

In the same way as before, link that to "300AGLTrigger" via a new ObjectActivationAction, making sure that "EnrouteKatTrigger" is changed to being inactive by default. If you want to avoid some of the interaction with the main display, you can drag the node reference straight from the Find dialog to the Attribute dialog. This time though, when "EnrouteKatTrigger" is linked to the new ObjectActivationAction, it still appears as a pale pink copy instead of as the full node. This is because it has "Allow Parent" switched off. To switch it back on, you need to go to the real copy of the node. Right-click on "EnrouteKatTrigger" and select "Referenced by→EnrouteKatTrigger". This will centre the display on the real copy of the node. Now you can re-enable "Allow Parent" - right-click, "Allow Parent" - and it will link properly.

Judging from the sizes of the AreaRefs attached to the other three triggers, the correct order is "EnrouteKatTrigger", "KatApprRadTrigger", "KatTrigger" and finally "KatUnloadTrigger". If you link these together, using new ObjectActivationActions attached to the "OnEnterActions" attribute of the previous trigger, most of the mission will be linked correctly.

Let's clean up the last few sections. Again from looking at the AreaRefs, it appears that "RutCarTrigger" must happen after "RutApprTrigger" so create a new ObjectActivationAction and link those.

At the top of the mission are a handful of actions which aren't referenced by anything. If they're not referenced, they're not any use so we can delete them. Select them – just the blue actions, not the goal or triggers – by either dragging a box round them or by using shift-click to select them individually. You should have "ResolveRutGoalFail", "TurnOffSafeNorth", "TurnOffSafeSouth", "TurnOnLubFailLand" and "TurnOnPOIKatale" selected. Now just press "Delete", or use the context menu's "Delete" item. Confirm that you want to delete the five nodes, then save the mission again.

Now your overview should look like this, showing three clear sections. It should be much easier to follow the mission storyline simply by following the flow from the start of each group.
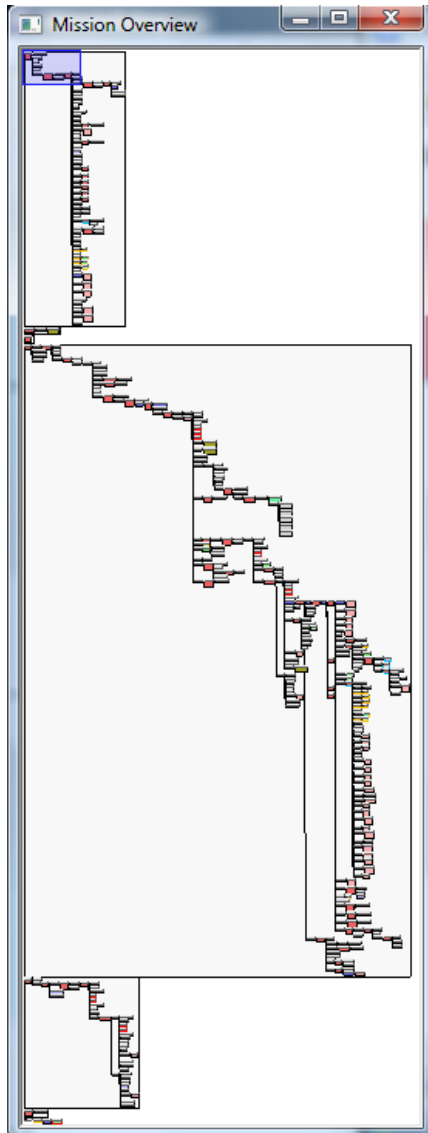
When you design your own missions, you can make your job much easier by only activating things as they are needed and then just following the flow. You can prove this by using the "Layout→Allow Parent on all Nodes" and "Layout→Clear all Preferred Parents" menu items. This clears every layout decision and allows the layout to be based purely on the mission structure. With the exception of "AshTrigger", which gets reattached, the layout is almost identical.

In addition to creating new nodes using the Action Palette, you can copy and paste nodes from the main display. The node contents, and any links between copied nodes, are also copied. You can also paste the XML for any copied nodes into a text editor.

ATTRIBUTE EDITOR

Almost all of the detailed editing is done using the Attribute Editor. This is where you change the details of a selected node to make it do what you want. This appears when you double-click on the attribute area of a node.
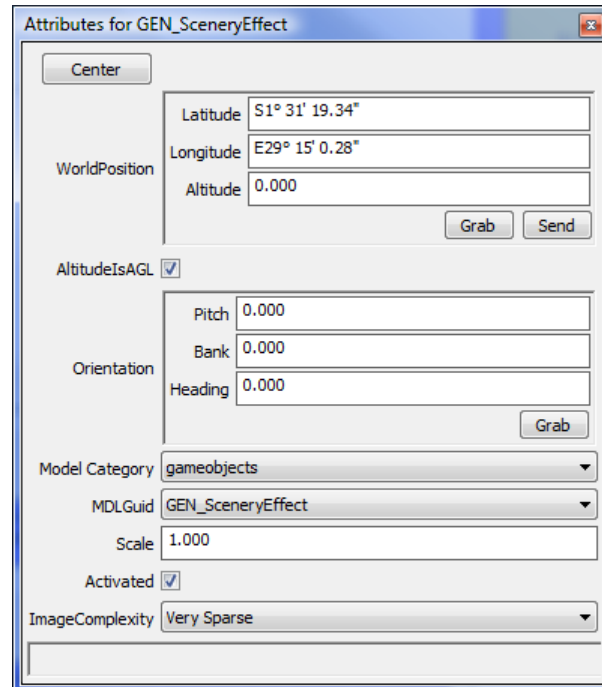
At the top of the window, the 'Centre' button will simply re-centre the node in the main display. All of the other items are different depending on what node you are editing.

The last item at the bottom will show a description of the current attribute. These are read from Microsoft's own configuration files and although most are defined, there are some which are not.

In this screenshot, the first three items (Latitude, Longitude and Altitude) are part of a group called WorldPosition. This is shown by the extra border round the group. It has two extra buttons, Grab and Send, which we'll ignore for now.

The other items mostly correspond to the attributes that are listed in the SDK for this node type, a LibraryObject. You would edit these to have whatever values you want, just as with the standard mission editor. There is an extra item here though, "Model Category", that isn't a part of a LibraryObject. These extra items are sometimes used where long lists of values are available, to narrow down the list.

The next item, MDLGuid, is part of a LibraryObject but unlike the default mission editor, there is a combo-box available. (If you have no combo-box, just a standard text field, make sure you ran "Regenerate BGL Data Lists" on the File menu). Where you would normally have to type in a long GUID, you can just pick the name of the model from the list. This does depend on having the name available though, so it may not work for all models. Most of the models provided with FSX do have names.

The Attribute Editor also helps with error checking. If one of the attributes has a problem, its name will be shown with a colour instead of being black. These are re-checked with every change, so you can see immediately if the new value has corrected the problem. If an attribute is shown with a white title, it means it is an internal attribute meant only for use by FSX when saving a mission in progress and shouldn't usually be edited.

Most importantly, this is where you link nodes together to form the structure of your mission. Many node types, especially Triggers, are only useful when they are linked to others. To create a link, open the Attribute Editor for the parent object (i.e. the trigger) and then drop the child node onto the box which reads "drop more here".
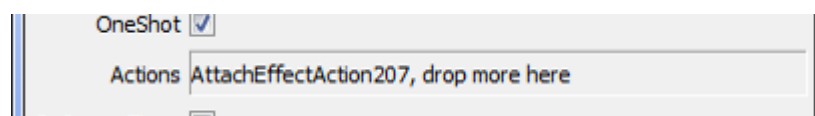
Sometimes an extra button, labelled "P+", appears to the right of a link item. This means that the link will accept the Player – that is, the plane being flown in the mission – as a target. Although you can use the main display to locate the special node for the player and drop it, as with any other, this button acts as a shortcut.

Normally the main display would be the right place to find the nodes you want to drag into the Attribute Editor to be linked. However, you can also drag them from any dialog that shows a list of nodes, such as the ones used for "Find" and "All Errors".

You can unlink nodes directly from the attribute editor, by right-clicking on the link item and selecting the node to detach. You can do the same thing from the main display, by selecting the node you want to unlink and using the 'Detach' command on the context menu.
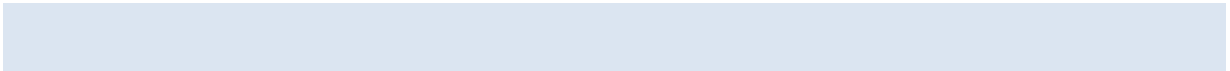
CONDITION EDITOR

Some triggers make use of Conditions, a way of checking a small subset of the things that FSX knows about both the player and other AI aircraft. For an example, find the "RutParkTrigger" node in the Congo sample mission and open the Attribute Editor. The condition is the last attribute:
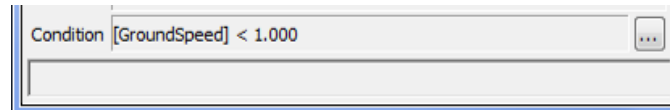
Conditions can be very simple, like this one, or much more complicated including groups of multiple conditions combined using Boolean (And, Or, Not) operators such as this one:

To open the Condition Editor, click the button labelled "…" to the right of the Condition in the Attribute Editor. The following dialog will appear.

This looks complicated! It's not as bad as it seems though. Let's split it into sections.

CONDITION LIST

At the top, the list of conditions shows all of the individual parts that make up the entire set of conditions that will be tested. The one above is a simple, single-entry condition (`[GroundSpeed] < 100`). The next example shows a more complex condition:
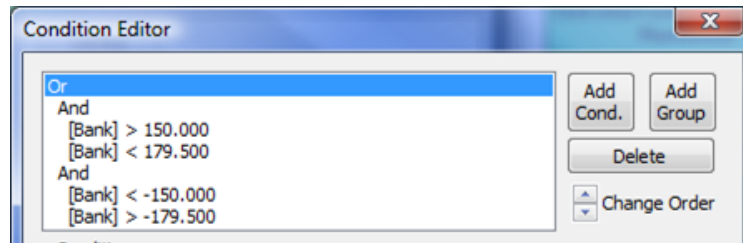
This one has several groups, shown using 'Or' and 'And' operators. The members of the groups are shown by the level of indenting in the list. In this example the first 'And' group requires Bank to be between 150 and 179.5, the second 'And' group requires Bank to be between -179.5 and -150 and the 'Or' group requires one or the other of the two 'And' groups to be true. This is summarised at the bottom of the Condition Editor:
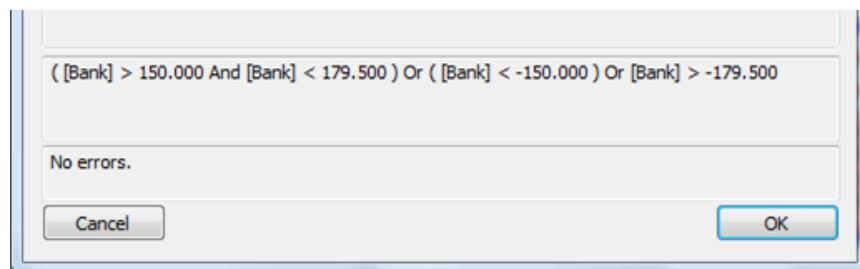
The buttons to the right of the list of conditions let you create, remove and rearrange the conditions. "Add Cond" creates a new condition, which allows you to perform some kind of test. "Add Group" creates a new Boolean group, where you can combine the results of several tests. Any new items of either type get added to the end of the list.

You can delete either a single item or an entire group of items at once. For example, if you were to select one of the "And" groups and press "Delete", both the group and it's contents would be removed.

Last, the "Change Order" control rearranges the conditions. This can take a little getting used to, because the ordering is partly shown by the position in the list and partly by the level of indenting. A condition being moved up the list may be added to a previous group, leaving it in the same place in the list but with increased (or decreased) indenting to show the changed grouping. The best way to learn this is to do it; create a complex condition and move elements around to see what happens. Remember that the summary of the overall condition is always shown at the bottom of the window.
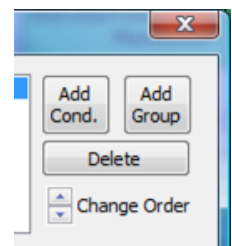
TEST EDITING

The next section of the window is the main part, where you can change the conditions being tested.
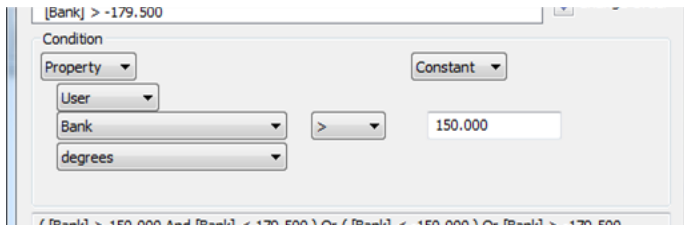
This is split into two halves, left and right. Each half describes a value which can come from an AI or user aircraft property, or a reference to an AI, or a constant value. Between the two halves you can choose what relationship to test; in this example, the left hand side must be greater than the right hand side for the condition to be true.

By changing the list boxes you can choose what to test for. There are many different combinations, but most are available under "Property" and usually you will be comparing a property with a constant, as shown.

Any errors found while you are changing the values here are shown immediately at the bottom of the Condition Editor window.

If you choose "Reference" for the property source instead of "User", an extra control appears which allows you to drop an AI object onto it in the same way that the Attribute Editor does, when linking nodes.
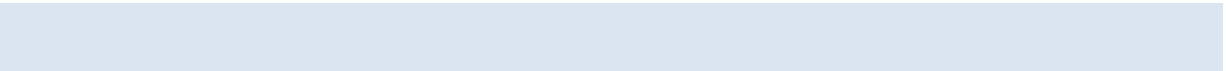
REWARD CRITERIA EDITOR

Rewards can have restrictions assigned to them to make them work with 'Free Flight' mode. When all the requirements have been achieved, FSX grants you the reward.

When you edit the attributes of a Reward, there will be an attribute called "Criteria". Clicking the small button to the right of this will open the Criteria Editor.
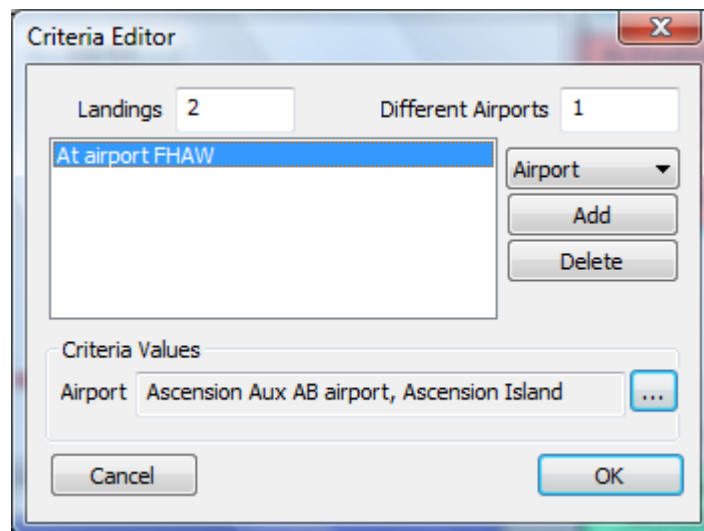
The two values at the top apply to the entire Reward. In this case, they are saying that you have to land twice, anywhere, for this reward to be granted. However, there's another restriction in the list below.

The list can have as many restrictions added as you need. They must all be satisfied for the reward to be granted. You can add restrictions based on the airports which must be visited, the number of hours that must be flown under a range of different conditions, or that another reward must have been granted first.

Rewards are saved in a separate file from the rest of the mission, because FSX requires them to be compiled into a .RWD file. Normally the editor will use a single XML file for rewards for each mission, but it is possible to make it share one XML file for multiple missions. If you are designing a mission pack, you may only want to deploy one .RWD file. You may want to grant a single reward from more than one mission.

To change this, use the "Mission→Set Reward File" menu item to choose the XML file you want this mission to store it's rewards in. When you choose an existing file, any rewards already in that file will be loaded. Any existing rewards are not changed.

ADVANCED EDITING ✕

Although the mission system is powerful, it can sometimes take a combination of several nodes to achieve a certain effect. You may also simply have a group of things that always get done together.

RECIPES

To save some time with groups like this, you can define your own 'Recipes', or collections of related nodes, which can then be created all at once. You can even customise them every time they are created.

USING A RECIPE

For an example, create a new mission (File→New), open the Action Palette if need be (View→Action Palette) and choose the 'Recipes' category. Double-click on 'Mission Intro' and a dialog will appear.
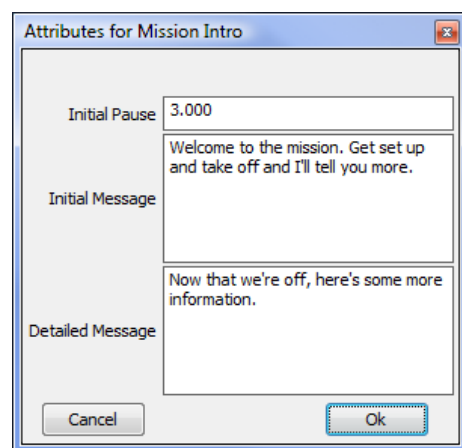
These are customisable features of the 'Mission Intro' recipe. Change the values to whatever you want and click 'Ok'. You will get a single node in the main display called "Timer_Intro", with the small grey box on the right showing that it has hidden children. Show those children using 'Right-click→Hide Children' and you will see five connected nodes.

◊

CREATING A RECIPE

Of course, you can define your own Recipes easily. In brief, you need to select all the nodes you want to include, select 'File→Save Selected as Recipe' and state which attributes you want to be customisable. Let's try it.

Load the 'Congo.xml' sample mission and find "TurnOnForgotGear". This enables three other nodes, two of which are tests that you might want to do every time the player takes off. Let's turn this into a Recipe.

We don't want 'EnrouteKatTrigger', that's specific to the Congo mission. Select 'TurnOnForgotGear' and the two PropertyTriggers (Forgot...Trigger) and their dialogs. Remember you can just drag a box around them to select them easily.
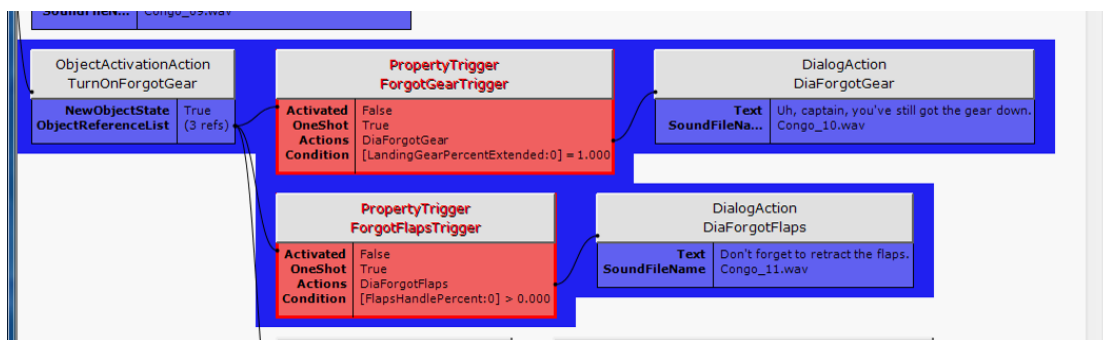
Now use the 'File→Save Selected as Recipe' menu item and a new dialog will appear.

The first two items are for display. 'Title' is what this recipe will be called in the Action Palette, and 'Description' is the more descriptive text that appears at the bottom of the Action Palette and is also set as the comment when this recipe is used. Set these to 'Takeoff Checks' and 'Check landing gear and flaps after takeoff'. The filename can be anything you like, but it must go into the default Recipes directory.

The bottom half of the dialog shows a list of attributes, and which nodes they belong to. If you tick any of the boxes, that attribute will be prompted for when the recipe is used. The attributes in bold text are those ones which are most often used.

You can also rename the attributes so that they make more sense. This works the same way as Windows Explorer; select the attribute, pause and click it again to turn it into an edit box. For this example, rename 'DiaForgotFlaps→Text' to 'Forgot Flaps', and tick the box. Do the same for 'DiaForgotGear→Text', renaming it to 'Forgot Gear'. The dialog should now look like the screenshot above.
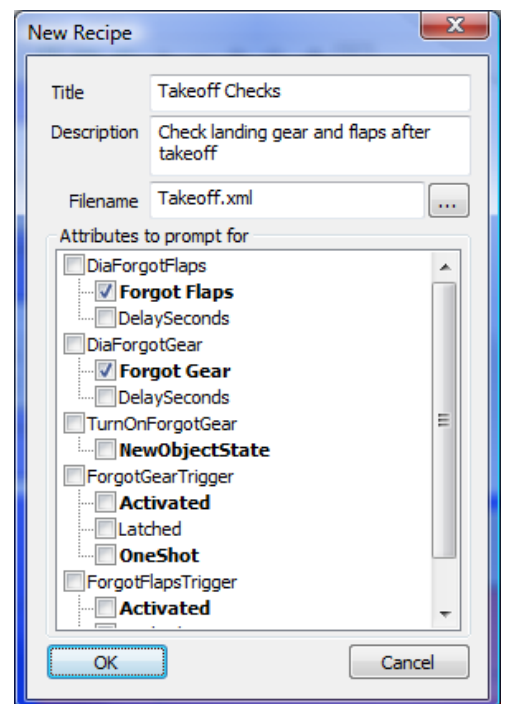
When you click 'Ok', this recipe will be saved and can then be used from the Action Palette. You can check it by clearing the mission (File→New), and then loading the 'Takeoff Checks' recipe. You will be asked for the new messages for 'Forgot Flaps' and 'Forgot Gear' and then the 'TurnOnForgotGear' node, with hidden children, will be created.

Once it has been created, you can-reuse a recipe as many times as you want in the same mission. The node names and InstanceId values, both of which must be unique, will be changed for you.

Recipes can be easily copied too, so why not make some of your favourites available for others to download? They are stored in the system's "Common Application Data" folder, in a subfolder called "`FSAddon\FSXME\Recipes`". The exact location can be found on the Paths tab of the Preferences dialog (Edit→Preferences).

If you were to redefine this test recipe to include a PropertyTrigger, firing at 300ft AGL, it could be dropped into any mission in a few seconds and be useful straight away.
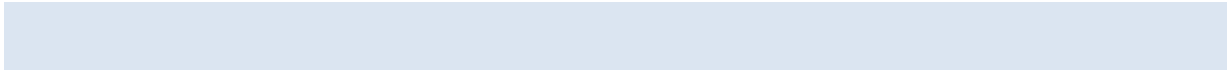
CHECKING COMPATIBILITY

With the Acceleration Pack for FSX, there are several new types of actions and triggers available for use in a mission. Of course, these won't work if they are used with standard FSX. When you create a new mission, it will have a compatibility level set to the highest supported version of FSX that it can find; if you have the FSX X-Pack installed, it will assume you want to create missions for X-Pack. You can choose to create missions for older versions of FSX by using the "Mission→Set Compatibility" menus.

Before doing this, you can see what will happen by using the "Mission→Show Compatibility" menus. This will highlight anything that will cause a problem in whatever version of FSX you choose. So, if you have developed your mission against X-Pack and want to see if it will run on SP2, set the 'Show Compatibility' level to SP2. Anything that is not available in SP2 will be shown in purple, and will be listed in the Errors dialog (F6).

A list of all the scenery models and SimObjects available in each version is included, so these will be highlighted too. Using a model which isn't available will make the mission simply not load in older versions of FSX.

If you reduce the compatibility level, it is possible for some data to be lost. For example, if you have used some of the Race triggers provided with X-Pack, or set the "TargetPlayer" attribute which has been added to several actions, and then decide to reduce the compatibility level of the mission then those triggers and attributes would no longer be available. You will see a warning that some things are going to be deleted and, if you accept, any attributes or nodes which require a newer version of FSX than the one you have chosen will be removed from the mission. Any affected nodes, or those which were parents or children of one which has been deleted, will have a comment added to show that they have been affected. This comment will be searched for automatically so that you can see immediately what has changed.

When a mission is loaded, the compatibility level is calculated from the types of nodes and attributes it uses. So, even if you had the compatibility level set to "XPack" when doing some work on a mission, it may show compatibility as being "RTM" when it is reloaded if you haven't used any XPack-specific features. You can always increase the level again by using the "Mission→Set Compatibility" menus.
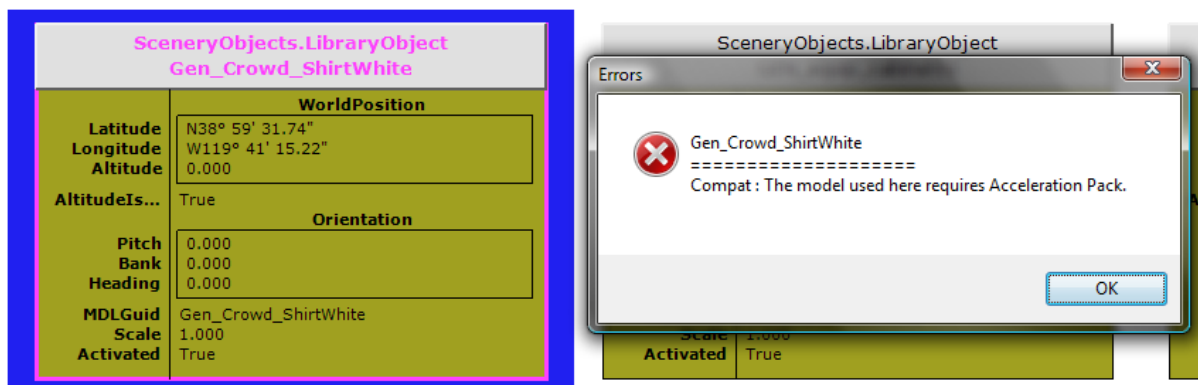
STARTING FROM SCRATCH

If you're creating an entirely new mission, there are several files you will need before FSX will recognise it. Apart from the mission file itself, you must create files for the mission's default flight, the briefing and some images. These must all be located somewhere under "`[FSX]\Missions`".

To make this easier, there are a set of template files supplied with FSXME. First you need to create an empty mission, add a ScenarioMetadata node and save it in a new folder under "`[FSX]\Missions`", for example "`[FSX]\Missions\New Missions\Demo`". You will get a warning about a .FLT file not being found. Clear this warning and select 'Mission→Create Template Files". The following files will be created for you in the same folder that you saved your mission:

- Briefing.html

- Images\Overview.jpg

- Img_complete.bmp

- Img_incomplete.bmp

- Mission.FLT

- Mission.WX

- Kneeboard.htm

These are enough to get FSX to load your mission, so all you need to do is start writing it! Of course, you will probably want to change all of these default files to suit your mission but at least these will give you somewhere to start.

This option can also create template images for any rewards you have created.

You must save your mission first so that FSXME knows where to put the template files.

CHANGING THE TEMPLATE

The first time you use the 'Mission→Create Template Files' menu, the default files will be written to the Application Data area in a folder called "`Template`". (You can find the location of this on the 'Paths' tab of the 'Edit→Preferences' dialog box). You can change any of these files to suit your own needs. The next time you use the template files to help set up a new mission, your edited files will be used instead.

SETTING THE TITLE

Your mission will need to have a title and description set, which appear in the Missions screen in FSX. The Title is also used as the package title when creating an MSI installer. You can set these values using the 'Mission→Set Description' menu item.

CREATING SYNTHETIC SPEECH

You can record a set of WAV files for your mission using the Windows speech synthesis facilities. They are obviously not as good as recording real speech, but if you are not in a position to record it yourself or you think the speech may change during development they will give you at least an idea of what the mission will be like.

The "Mission→Create Dialog Speech" menu will show this window:
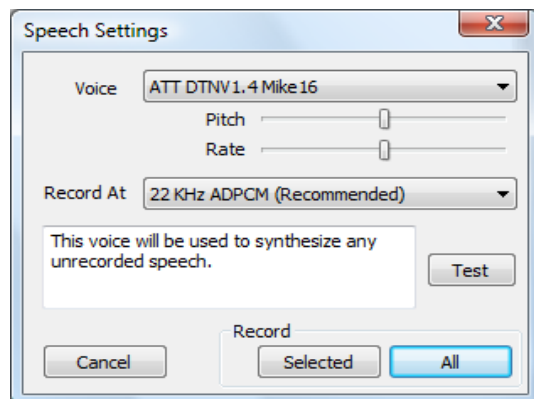
You can select the voice and change the pitch and rate of the speech. Press the 'Test' button to see what the voice sounds like.

Not all voices support changing the pitch. Also, there appears to be a problem with the default voices on Vista x64 since they do not work even from the 32-bit Control Panel.

The recording rate defaults to 22KHz ADPCM on Vista, as recommended in the Mission SDK. However, you may wish to record at other settings. The GSM610 and ADPCM modes listed are compressed; they will sound slightly worse but will take up much less disk space. On XP the default format is GSM610 instead of ADPCM, because there is a bug in the speech engine on XP where it records broken WAV files in ADPCM mode.

When you have tuned the voice, press either 'Selected' or 'All' to start recording the speech. 'Record Selected' will only cause speech for any dialogs that are currently selected to be recorded. If any of the WAV files already exist, you will be asked if you want to overwrite them. If you choose 'No', only those files which don't already exist will be created.

It can be easier to manage the recorded speech if you add a prefix to your dialog nodes, showing who is speaking. If you are using the synthetic speech, this allows you to easily choose to re-record all the speech for a single character. You can simply select all the WAV files for that character using the 'Find' dialog and then re-record them with the right voice settings.

VOICES

There are many different voices available. Vista's "Anna" is good, while XP's "Sam" is a dalek with laryngitis.

Microsoft provide some slightly more usable free voices from here.   They also list many voice engines but all seem to be aimed at companies, with corresponding price-tags. However, most of the voices are available cheaply from NextUp.com as part of their 'TextAloud' package.

Make sure that any voice you get is compatible with SAPI5, *not* SAPI4. Also, on 64-bit operating systems you will need to use the 32-bit control panel to set up the default voice. This is found under "32-bit Control Panels", under "Additional Options" in Vista x64.

SAVING THE MISSION

When you save the mission, there are several different things that happen. First of course, the mission itself is saved to disk as an XML file that FSX will load.

The mission layout is saved as a separate file. This contains all the things that relate to the visual display of the mission, and which FSX doesn't expect to see. Node colours, comments and parent/child relationships are stored in here along with other settings like the MSI details.

Last, any .FLT files that are in the same folder as the mission file are modified if necessary to have the correct title and description. They also have an entry called "ObjectFile" changed. This is how FSX determines what mission to load; the .FLT file provides the descriptions and the starting details, and points to the XML (or SPB) file that contains the mission logic.

If you are saving a new mission, there will be no .FLT file and you will get a warning message. You can create a default .FLT file (and all the other necessary files) using the 'Mission→Create Template Files' menu option.

REWARDS

Any rewards that are used will get saved to a separate file. By default this is called "`<Title>_MERewards.xml`" but this can be overridden using the "Mission→Set Reward File" menu. If the BGL compiler is available, the rewards are compiled into a .RWD file automatically.

If the rewards don't compile properly, it could be that there are no images. These need to exist before the reward can be compiled. You can create default images using the "Mission→Create Template Files" menu option.

If the compiler (`bglcomp.exe`) can't be found, you will need to compile the rewards yourself before FSX will find them.

COMPILING TO SPB

FSX will load a compiled mission faster than a non-compiled mission. However, both versions should work identically. If you want to test your mission as a compiled version, you can use the "Mission→Compile to SPB" option to do this.

This does mean that if you make further changes to the mission, which is stored in XML form, there may be some confusion about which version of your mission is being used by FSX; the SPB or the XML. To help avoid this, you will be warned if an SPB file exists when you save the mission.

If you want to include an SPB with an MSI installer package, use the 'Compile to SPB' tickbox on the MSI settings dialog instead unless you are combining many missions into a single MSI package.

BACKUPS

You can optionally back up the key mission files every time you save. This is enabled by default, but you can change the settings using the `File→Preferences' menu. The third tab on the Preferences dialog is for backups.
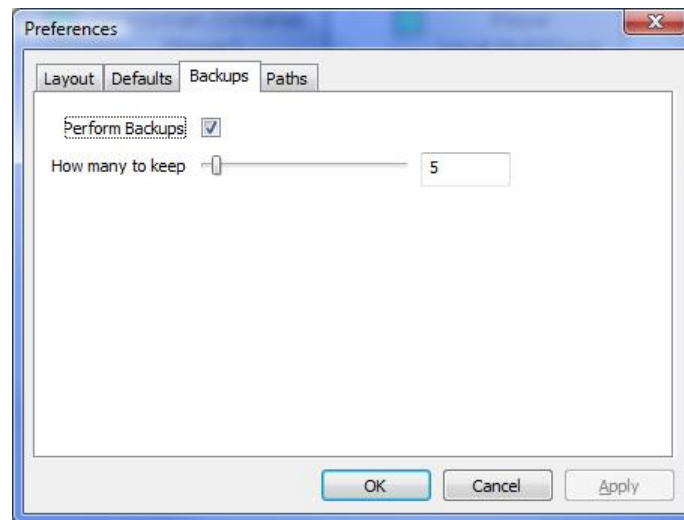
The mission itself, the definitions of any rewards you have created, the layout and the .FLT file(s) associated with the mission are all backed up into a single .CAB file when you save. Old backups are automatically deleted when there are too many; you can change the number you want to keep.

A CAB file is used instead of simple file copies to stop FSX from picking the backed-up files as a different mission. CAB files can be opened simply by double-clicking on them using Windows Explorer.

If you do end up having to remove nodes to find out why FSX isn't loading something, it may be useful to switch off the auto-backup. This will stop you from losing a 'good' backup if you're only keeping a small number of the most recent ones. When you persuade FSX to load the mission again, you can turn auto-backups back on and make the changes you need.

WORKING WITH MODELS AND AREAS

Almost every mission will need to use some scenery models, or define some Areas. The standard FSX Object Placement Tool (OPT) already works well for manipulating these, so there are very few similar features in FSXME. It was always intended to be a partner to, rather than a replacement for, the OPT.

Inside FSXME, you can of course create and edit all kinds of scenery and areas. You can also get their locations from FSX, using the 'Grab' button in the Attribute Editor. For more detailed editing though, it is easier to use the OPT as described in the SDK documentation.

The OPT uses exactly the same mission file as FSXME, and you can easily use both tools at the same time. The only thing you need to remember to do is to save the mission when you are finished using one tool, so that the other can see the changes.

FSXME will notice when anything else saves the mission, and will offer to reload it automatically. This makes it easy to swap between the two different tools. The recommended way of working is this:

1) Create as much of the mission as you can in FSXME, including any AreaDefinitions you need. Get rough placements for these by using the 'Grab' button on the Attribute Editor, which gets the plane's current position from FSX.

2) In FSX, open the OPT and load the mission. Use the interactive tools to visually fine-tune your areas, and to place any scenery objects you need.

3) Save the mission from the OPT. FSXME will offer to reload it, allowing you to continue editing.

The only thing to beware of is that while the OPT is running inside FSX, the 'Grab' button will often return incorrect values. You can keep swapping between tools as often as you like, as long as you remember to save your work in one before using the other.

INFORMATION DIALOGS

There are several dialogs which show general information about your mission. They are available from the 'Mission' menu, and all open a window like this:
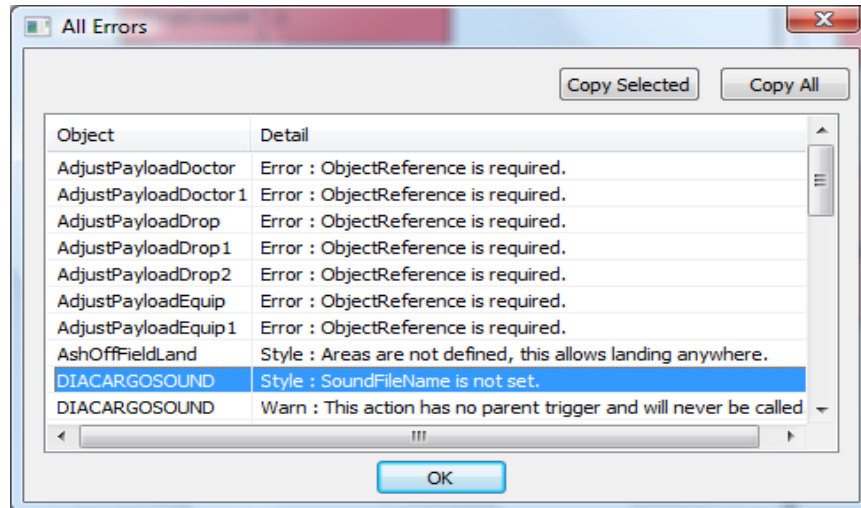
Each dialog has some common features. The 'Copy' button at the top places the main list into the clipboard, so that you can paste it into a text editor. 'Copy Selected' does the same, but only for the rows of the dialog that are currently selected. You can change the order of the items by clicking on the list headers, and you can drag nodes from the list onto the Attribute Editor to link them to other nodes. This can be useful for example in the Errors dialog, where an action has no parent. Finally, you can double-click the items in the list to centre them in the main display. If the node you double-click is not currently visible, typically because it is hidden for some reason, you will be warned and asked if you want to make sure it is displayed.

ALL ERRORS

This dialog shows all nodes which have an error, no matter what kind of error it is. You can use this to quickly identify places which may need your attention.

ERROR-CHECKING RULES

There are four classes of error; Errors, which are potentially serious, Warnings which are less serious, Style warnings which are probably OK and Compat warnings which show you that you have used a feature which will not work on a specific version of FSX. These have been based on the Mission SDK, common sense and observation of the mission system at work.

You may find, particularly when loading the sample missions, that some things are listed as Errors when the mission clearly works without any problems. The sample missions are prone to this because they were written early on, and the rules of the mission system changed slightly after their creation. One example, again from the Congo mission, is that all of the AdjustPayload nodes are marked with "Error : ObjectReference is required". Originally an empty reference was taken to mean "The Player", but this was later changed so that the player had a special, fixed GUID which could be properly linked to. Rather than writing all these special exceptions in, they are left as errors because they don't match the final rules even if the mission system will ignore them.

This means that a mission with errors may well work correctly. The error-checking is deliberately more strict than it might be, because one of the greatest frustrations of the default mission editor was that it offered no indication of what was causing a mission to go wrong. In some cases FSX would not even load the resulting mission, forcing you to slowly and painfully delete nodes one by one, reloading the mission with every change, until it started working again.

DIALOG SCRIPT

This lists the text in every DialogAction in your mission, along with the name of the WAV file that is used. You can use this to produce a script for the voice-actors who will be recording the speech. Use the 'Copy' button to create the script, and simply paste it into Notepad or an e-mail. The script will also have the node names, which they might use along with a list of node comments to get additional direction.

ALL COMMENTS

Use this to list all the comments you have assigned to nodes. Since the comments are not used for anything explicit, this can be put to any use you can imagine. As described above, you could add a comment like "Sound excited" to a DialogAction and pass this list to your voice-actors along with the script. You could add 'TODO: Fix something' comments as notes to yourself, explanations of why things are hidden or done in a particular way or anything else you can come up with.

DEBUG EVENTS

This shows the progress of your mission in real-time, as you fly it in FSX. This is covered in detail under "Debugging your Mission".

DEBUGGING YOUR MISSION ✕

For more complicated missions, it may sometimes be difficult to tell exactly what is happening when you are testing it in FSX. You could put extra Dialog messages in the mission to show you what's happening but this still isn't ideal. Unfortunately there's no way of getting access to the mission internals while it's running.

To try and work round this, FSXME can alter your mission so that every trigger reports back information on when it fires, and where the user's plane is at the time. This information can be recorded so you can see the exact sequence of events, hopefully allowing you to work out what is not happening in the way you'd expect.

ADDING TRACE ACTIONS

To make this change, select the "Debug→Patch All Triggers" menu item. Every node that can call an action will have a new action added to it which reports back to the editor when it is used. Every trigger will be monitored in this way, but RandomActions are not. Since only one item in a RandomAction's Actions list is used, adding a trace action here may change the way your mission works.

Once these trace actions have been added, the display will change so that every trigger with a trace action attached will have a small red dot. As you fly the mission in FSX, these indicators will turn purple when the trigger fires.

You can also set breakpoints on nodes. These also allow you to trace the mission, but when a breakpoint is hit, FSX will be paused. Use Shift+F9 to set a breakpoint.

Trace indicator

Remember to save your mission after adding these trace actions, otherwise they will not be present when you load the mission in FSX.

When you are done tracing, selecting "Debug→Unpatch All Triggers" will remove the trace actions. Remember to do this before you give your mission out. Leaving the trace actions in place won't stop it working, but it will be much larger than it needs to be.

If you place a trace or a break on some items, in particular MenuItems, you will get a notice when the mission is re-loaded that it has been changed. If you are not planning on using the debug features, you can ignore this warning. However, if you do intend to run the debug monitor, you need to re-save the mission before loading it into FSX. This is because for some items, the GUID that uniquely identifies it can't be written into the XML. A new GUID must be allocated every time it is loaded, and the debug actions need to be updated to match.

MONITORING THE MISSION

Once you have patched and saved your mission, you can reload it in FSX and start to fly. As each trigger fires, the trace actions will call back to FSXME showing you what is happening in real-time. To enable this, select "Debug→Monitor". You would also normally open the debug dialog, using "Mission→Debug Events". As the trace events come in, this dialog and the main display will both be updated.
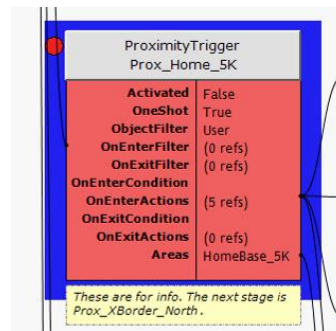
If the Monitor command is greyed out, it was not possible to load SimConnect. This will be the case if you are running on a machine which doesn't have FSX, or the SimConnect standalone package, installed.

The list of events shows you the name of the trigger that has fired, and the name of the attribute which was used. Normally this will be 'Actions', but some triggers such as ProximityTriggers have more than one list of actions. It also shows the time that the event happened, as the local time in the simulator.
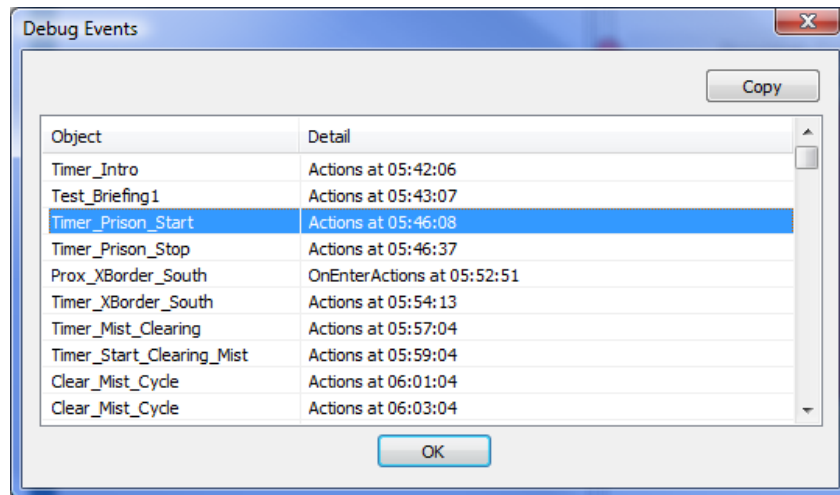
As with the other similar dialogs, you can double-click on the list to show the trigger on the main display. This allows you to follow the mission's progress in your own time, tracking the sequence of events and hopefully making it easier to correct problems in the mission's logic.

If you have placed breakpoints as well as tracepoints, FSX will pause when a breakpoint is hit. However, because of the way that SimConnect works, the sim may not pause for a short time after the breakpoint is hit. If you have several triggers which fire in a very short period of time, the mission will have continued a little way before it is paused. You should be able to use the debug event dialog, or simply follow the mission flow in the main window, to find out what events have fired.

If the mission is restarted, the list of trace events will be cleared.

TUNING THE TRACE

You may find that you have some triggers which you don't want to monitor, such as short-period recurring TimerTriggers. You can turn monitoring on and off for each trigger individually using the 'Set Trace' item on the context menu, or by pressing F9.

KEEPING THE TRACE

You can save the trace along with the mission by using the "Debug→Save Trace" menu item. It will be saved along with the main mission, by default with ".`trace.xml`" appended to the mission's filename although you can change the name if you wish. You can reload this trace later using "Debug→Load Trace", or you can view it as a single list by double-clicking on the trace file in Windows Explorer or opening it in a browser.

DEBUGGING TECHNIQUES

The debugging feature is meant to allow you to follow the mission flow through, to see which triggers are firing and then hopefully to work out what isn't happening in the way you'd expect. However, once you've flown your mission for the thirtieth time, trying to get something working that only happens after forty minutes, it can get a little repetitive.

One way round this is to use a menu at the start of the mission which allows you to skip the earlier parts, perhaps by enabling triggers which only normally get activated late in the mission. Typically, a well-designed mission will have at least a few places where all the possible story threads come together, and these places are ideal for jumping into from a debug menu.

To help with this, there is a CustomAction provided called "Debug Relocate". This works with the Debug Monitoring mode of FSXME. When this is active and this CustomAction is used, the plane will be moved to the location and orientation set in the CustomAction.

If your debug menu command enables all the necessary triggers and objects for the part of your mission you want to test, using the Debug Relocate action will put the plane in the right place to start the debugging run.

CREATING MSI PACKAGES ×

Once you have finished with your mission, you will probably want to give it to other people so they can fly it. While the more experienced users will be happy to manually unzip or copy folders around, many others will struggle or simply not bother. By creating an MSI package they will have a familiar way of installing and removing new missions.

The MSI creator included is fairly simple but should be enough for many situations. By default it will copy the entire folder, including subfolders, where your mission is kept. In other words, if your mission is saved as "`C:\My Missions\Demo.xml`" it will take everything under '`C:\My Missions`' and put it in the MSI to be installed on the users' computers.

There are some files that won't get included in any MSI. These are the layout and trace files that may be created by FSXME, and any temporary batch files. Reward files (.RWD) are also treated differently. Since these must be in a 'Rewards' folder under FSX, and not in the mission folder, they can't just be installed in the same way as other files. Any .RWD files which are found in the mission folder are put into "`[FSXHOME]\Rewards`" instead. Last, the ResourcePath value in any .FLT files is changed during installation to the relative path from FSX to the .FLT file. This assumes that the .FLT file is in the same folder as the other mission files.

After creating an MSI, the mission is automatically saved, because several of the MSI settings are stored alongside the mission. If you later need to issue an updated MSI, or a patch, the version number of the MSI must be correct. Saving automatically helps to enforce this.

If you decide to copy your mission XML to use as the basis for a different mission, it is important that you also delete and recreate the ScenarioMetadata node. This node is used as a key to the entire mission by the MSI system, and if you don't recreate it then MSI will think you are reinstalling the original mission even if you change the name of the MSI itself and the location it gets installed to. In effect, when someone installed your new mission they would either get an error or would have the original mission uninstalled. Simply doing a copy/paste of the ScenarioMetadata, then deleting the original one, is enough to solve this.

SIMPLE MISSIONS

Let's try the simple case first. Load up the 'Congo.xml' sample mission and then select the 'Mission→Create Installation' menu item. This dialog will appear:

There are lots of options here, but the default values will create a working mission installer. They are:
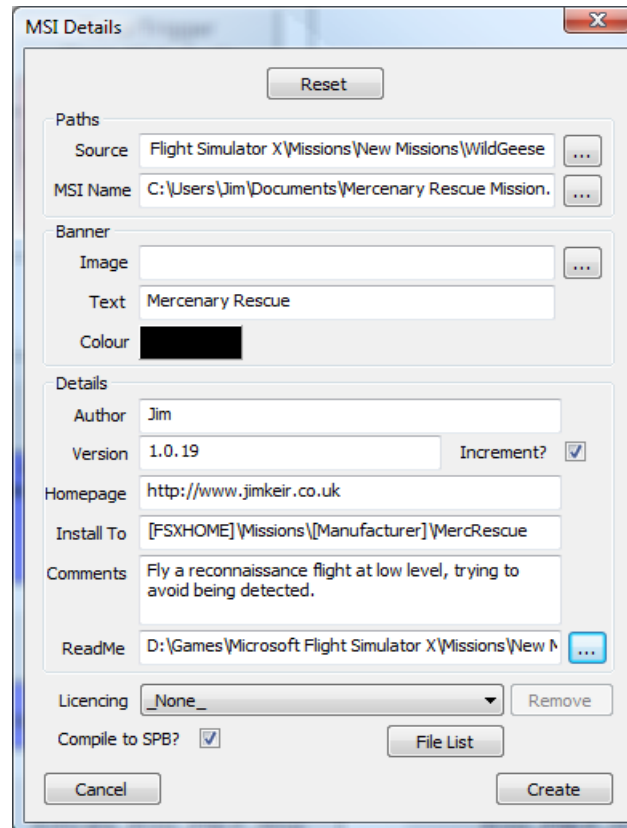
- **Source:** The folder which contains all the files you want to include with your mission. For a simple installer, the default value is correct. Whatever folder you set as the source, it must contain the currently loaded mission somewhere beneath it.

For example, if your mission is at "C:\FSX\Missions\MyMissions\Test.xml" then the default path would be "C:\FSX\Missions\MyMissions". You could also have "C:\FSX\Missions" or even "C:\FSX" because both those paths contain the mission XML. Using "C:\SomewhereElse" would not work because the mission XML is not included.

- **MSI Name:** The full path to the MSI that will be created.

- **Banner:** A 500*x*70 image file, one of .BMP, .PNG or .JPG, which contains the image that appears at the top of the installer. If you leave this blank, a default one is used.

- **Author:** This appears as the Publisher in the Windows Add/Remove Programs control panel. By default it is also used as the folder name to store your mission under.

- **Title:** The name of the item that appears in the Add/Remove Programs control panel.

- **Version:** Also appears in Add/Remove Programs. If the 'Increment' tickbox is set, the last digit of this version number will be increased after the new installer is created. It is important that you increment the version number, otherwise if you release updates to your mission they will be more complicated to install.

- **Homepage:** Only appears in Add/Remove Programs.

- **Install To:** This is the default location to install the mission to. The special value '`[FSXHOME]`' will be set to the FSX installation folder on the user's PC. The other special values, '`[Manufacturer]`' and '`[ProductName]`' are set to the values for 'Author' and the mission's title. The title can be set using the 'Mission→Set Description' menu.

- **Licencing:** You can optionally decide to release your mission with a simple key-based licencing scheme. One is provided – shortkey – which will allow you to generate licence keys which are unique to this mission. Once you have switched licencing on, however, it should not be switched off.

- **Comments:** These will appear in the Add/Remove Programs window.

- **ReadMe:** The name of a file to show at the start of the installation. This can be any text you want to show, usually a readme or a licence agreement. It must be in RTF format; this can be created best with Wordpad. Create your text file and when you save it, change the 'Save as type' selection from 'Text Document' to 'Rich Text Format'. MS Word and OpenOffice can also produce RTF documents but they are much larger - up to ten times the size - than those produced by Wordpad.

- **Compile to SPB:** Once you're finished with the mission you can produce a compiled version which will load faster and may stop people from reading your code quite so easily. Tick this box to compile your mission before packaging it.

If you are compiling more than one mission into a single MSI (see Advanced Missions) then only the current mission will be compiled. If you want all your missions to be compiled, you will need to compile them separately using the 'Mission→Create SPB' menu item.

- **File List:** This button allows you to include or exclude specific files. By default all files are included except for ones that are known to be used only for development; backups, batch files etc.
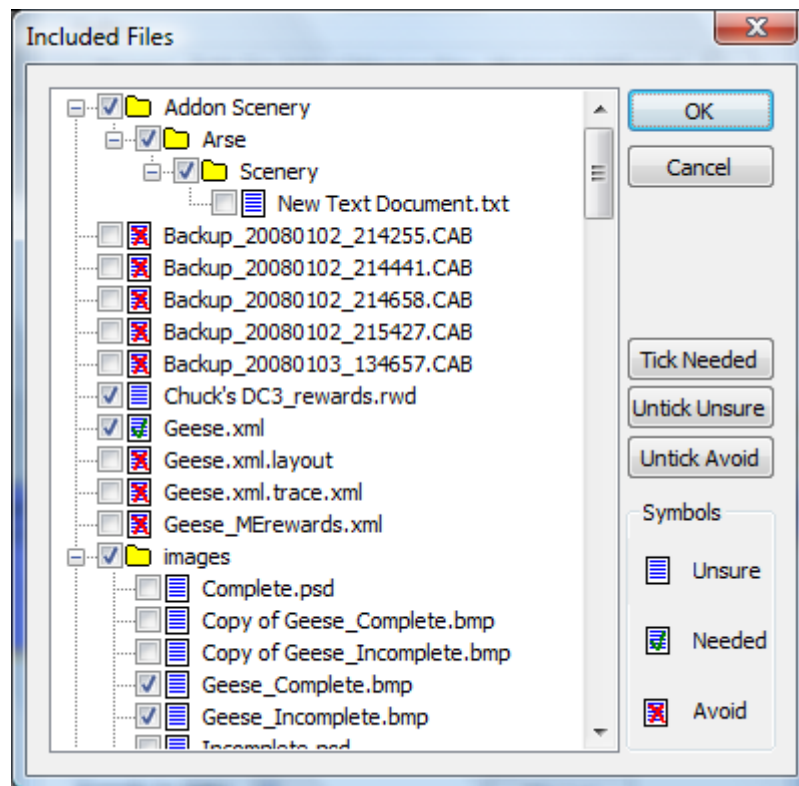
In this dialog, files which are known to be required are shown as 'Needed', with a green tick. Files which are known to be *not* required are marked as 'Avoid', with a red cross.

You can tick or untick individual files to include or exclude them from the MSI. Note that if you use the 'Untick Unsure' button, it is likely that any images you have used in your briefing or overview HTML files will be unticked, so you need to find and enable them.

'Needed' files are those which are explicitly referenced by the mission, such as audio (WAV), the mission briefing and bitmaps. 'Avoid' files are those which match these patterns:

```
•   *.bat
•   *.xml.layout
•   *.trace.xml
•   *.msi
•   *.rewards.xml
•   missiontrace.xsl
•   backup_*
•   *.tmp
```

All others are marked as 'Unsure'. These are enabled by default; if you disable them all using the 'Untick Unsure' button, you need to make sure that you include any other files your mission needs. These will include any images used by the mission briefing HTML, and additional scenery, effects or aircraft etc.

Back in the main MSI dialog if you now click 'Create', you will get an error message:
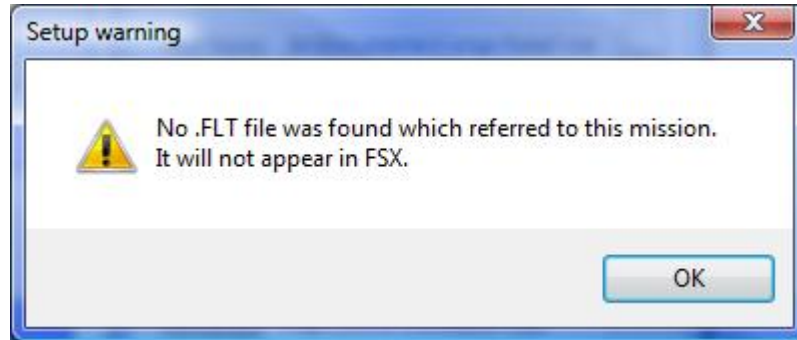
FSX will not load a mission without a corresponding .FLT file. Since this is a copy of one of the default missions, you can just copy that one. Go to '`[FSX]\Missions\Backcountry\Congo`" and copy the entire folder, except for '`Congo.spb`', to the folder with your edited version of '`Congo.xml`'. You should now be able to create the MSI, and it can be run just like any other installation.
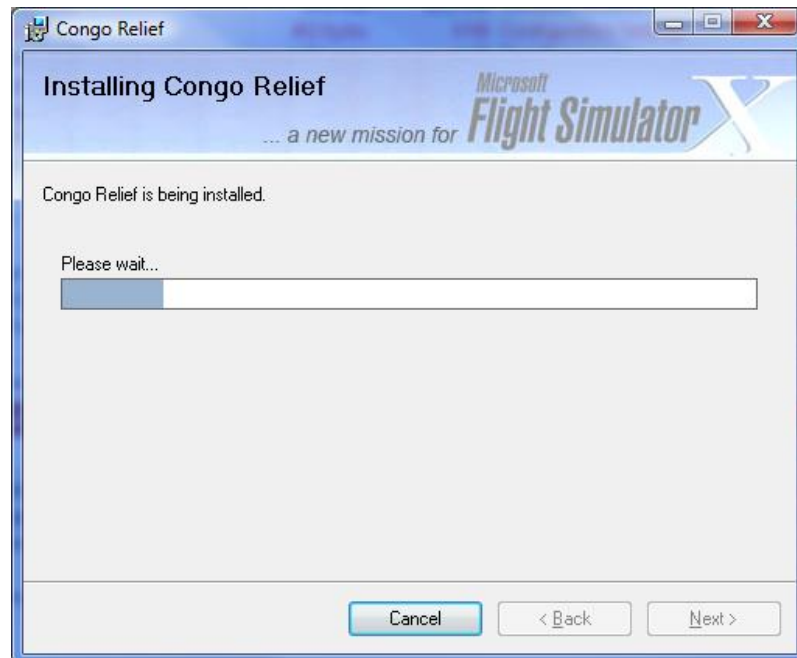
EXTRA MSI SERVICES

If the MSI were just deploying files there wouldn't be much benefit over using a simple zip file. In fact it tries to complete as much configuration editing as it can.

If your mission uses a new category this will be created when the package is installed. It is not removed if the package is un-installed, in case it's in use by other missions. After all the files have been installed, the FSX category files are checked to see if they contain all the categories that are used by this mission. If not, a new category file called [Manufacturer].xml is created, containing the new categories. The associated bitmaps are copied into place too.

Any scenery areas found are added to the scenery database automatically, and removed if the mission is uninstalled. A scenery area is any folder included in the MSI called "scenery" which has a parent folder. If you create an advanced MSI and put your scenery in "[FSX]\Addon Scenery\scenery", this is not removed from the scenery database if the mission is uninstalled.

If you include any .dll or .exe files they are assumed to be FSX extensions. If they don't already exist, they will be entered into the dll.xml or exe.xml files as appropriate, so that they are started automatically with FSX. They are removed only when the last mission that uses them is uninstalled. The dll or exe file itself is copied into "[FSX]\Extensions".

Any .FLT files, which are the starting point for FSX to load a mission, are edited on install. The ResourcePath value which tells FSX where to look for the mission files is changed to it's actual location. Also, the location of any flightplan file in the GPS_Engine section of the FLT file is changed. The flightplan is assumed to be in the same folder as the FLT file itself.

New categories and their banner images, and rewards are all left in place if your mission is uninstalled. This is because other missions may be using them.

ADVANCED MISSIONS

The simple, default behaviour of the MSI creator can be changed. You can choose both the source and destination folders so, if you have other files to copy such as effects or models, you can include those. Be careful though, because any file that you install *will be removed when the MSI is uninstalled*! This means that if you copy a new version of '`[FSX]\Categories\FSCategories.xml`', for example, it will be deleted if your mission is uninstalled – breaking FSX.

You shouldn't need to include any of the default files anyway; the installer should handle setting up categories, scenery, extensions and editing the FLT file.

To put together an advanced installer, you will need to prepare a folder to act as a staging area, containing all the files you want to deploy. However, for the advanced installer you would include all the files you need to copy, typically starting from the FSX home folder. You may have extra scenery, models or effects to include. The mission itself and all the other associated files should be put in the appropriate location, something like "`[Mission Source]\Missions\adventure\MyNewMission\MyNewMission.xml`".

In other words, you are preparing a folder that reflects the FSX file structure but with only the new files you need for your mission. This is often how missions are distributed, except that they will be packaged as a zip-file instead of an MSI installer and may still need manual editing of categories and scenery.

On the "MSI Details" dialog, you would change the "Mission Source" to the folder that contains your complete set of files, and "Install To" to `[FSXHOME]`. Note that in this advanced mode, any Reward files will not be moved and so must be in the correct place (i.e. `[Mission Source]\Rewards`).

The "Mission Source" you choose must include the mission file that is loaded. Without this, there is no way to check the current mission's required files are include, or to compile the XML to SPB if required.

CHECKING THE INSTALL

Just because the MSI installs correctly doesn't mean it's correct! There are some things you should check for before posting or selling your missions.

- Install it on a clean FSX installation to check for missing scenery or models.

- Fly the entire mission or, better, get someone who hasn't seen it before to fly it.

- Uninstall the mission and check that FSX still works. If you've included any FSX configuration files in your install you'll probably destroy the users' FSX installation when your mission is uninstalled.

- Reinstall, just to make sure it can be done.

LICENCING YOUR MISSION

You've finished your mission and decided it's a masterpiece, ready to go on sale to the world. You probably don't want people to be able to copy it around too easily. This works in much the same way as most software today; the installer asks for a licence key and won't install if it isn't given a valid one. For any given mission, you create the lock once when the mission MSI is created, and the keys as often as you need.

CREATING THE LOCK

In the dialog that creates the MSI, you can set the 'Licencing' box to switch on a licencing key system. Currently there's only one method available, but FSXME is designed to allow other systems to be plugged in.

With a licence provider switched on, whenever the resulting MSI is run the user will be asked to enter a licence. Without this entered correctly, the MSI will refuse to install any files.

There are many ways to work round this system! Simply giving the MSI to a friend along with the licence key, as with any software, will work every time. A little persistence and technical knowledge of the MSI structure will allow someone to extract the files anyway. However, it does give people the feeling that they've got a personalised product and are perhaps a little more likely to keep it to themselves.

Once you have selected a particular licence provider, you must be very careful about changing it. The licence details are stored in the .layout file that accompanies the mission, and if these are changed or removed then there is no way to generate any more licences. You would need to create a new lock, which would of course mean that any existing keys would no longer work.

When you click 'Create', the lock for this mission is created. It is the details of the lock that get put into the MSI; there is no need to rebuild the MSI every time you create additional keys.

If you do need to remove the lock, you can use the 'Remove' button next to the list of security providers. Only do this if you want all the existing keys for your mission to become useless in a new MSI installer!

CREATING KEYS - GUI

Once your mission is locked, you will need to generate keys to give to people you want to be able to install it. With the "shortkey" security provider, these are partly random and so many keys can be created for a single

mission. After creating a locked MSI, the 'Create Licence Keys' item on the 'Mission' menu will become available. Select this, and click 'Generate' on the dialog. A new key will be created. You can paste this into the MSI installer for your mission, or into an email to send to someone else.
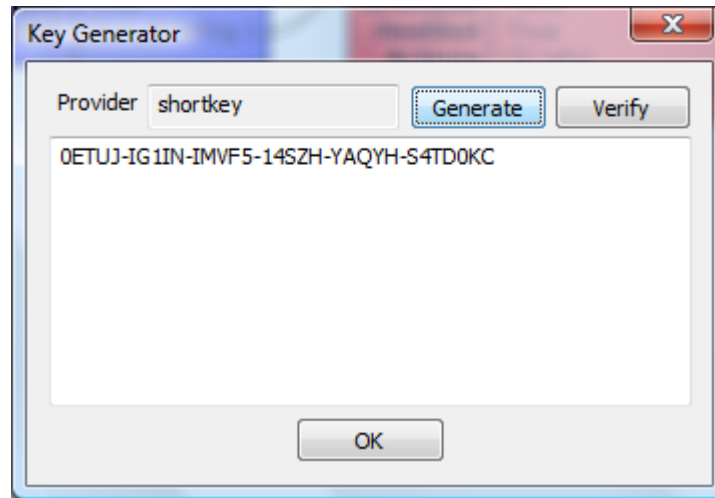
You can check keys using the same dialog; paste a key in and click 'Verify' and you will be told whether the key is valid or not.

CREATING KEYS – COMMAND-LINE

You can also create keys on the command-line. This is particularly useful if you have to provide a list of valid keys to an online retailer who is selling your mission pack.

In the same folder as the main editor program, there is another program called "Mission_Keygen.exe" . This has no icon on the Start menu because it is a DOS program and expects to be run from a DOS prompt.

It can be copied and run from anywhere, or you can open a DOS prompt and change directory to where it is installed to:

```
C:\Users\Jim>cd "c:\program files\fsaddon\FSX Mission Editor"

c:\Program Files\FSAddon\FSX Mission Editor>Mission_KeyGen.exe

Mission_KeyGen.exe -n Provider ConfigFile.xml
=> Generate a new lock, storing the values in ConfigFile.xml

Mission_KeyGen.exe -g ConfigFile.xml [Count]
=> Generate a new key, using the values in ConfigFile.xml
   The optional Count parameter generates multiple keys in one pass.

Mission_KeyGen.exe -b ConfigFile.xml [Count]
=> Generate a _bad_ key, using the values in ConfigFile.xml
   This is for false-positive testing. A real key is generated, then
   between 1 and 4 characters are changed at random.

Mission_KeyGen.exe -v ConfigFile.xml XXXX-XXXX-XXXX-XXXX-XXXX-XXXX
=> Validate a key using the values in the ConfigFile.xml
   ConfigFile.xml must exist, having been created using -n or FSXME.

Mission_KeyGen.exe -v ConfigFile.xml -f Filename.txt
=> Validate a list of keys contained in Filename.txt .
```

Normally you will create the lock using the Mission Editor; however, the "-n" switch will create a lock and embed it in any XML file. The editor will only use a lock embedded into the .layout file.

The other switches are explained in the help message, shown above. In brief, the most common command would be:

```
mission_keygen -g D:\MyMission\MyMission.xml.layout
```

This would generate a single key for your mission. To generate a list of valid keys to send to a retailer:

```
mission_keygen -g D:\MyMission\MyMission.xml.layout 1000 >KeyList.txt
```

PATCHING YOUR MISSION

Even with all the testing you can get, the sad fact is that you - or more likely other people - will find problems with your mission after it's been released. Instead of re-issuing the entire mission MSI which may contain large elements like sound files or scenery, you can issue a patch containing only the differences.

When you create a patch, the previous MSI will be used as a baseline. It must be in the location it was originally created in, and all the settings from the previous MSI build will be re-used. The version number is particularly important for MSI to work properly.

Selecting "Mission→Create Patch" will show this dialog.
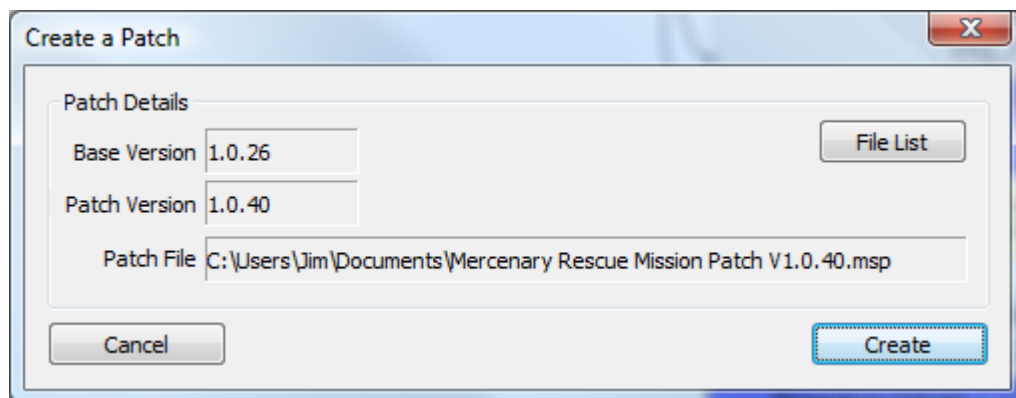
The "File List" button works in the same way as for a full MSI. It should show the entire list of files that you want to be on the users' machines, **not** just the files you want included in this patch. Any files which were included in the original MSI but which are excluded from the file list for the patch will be deleted when the patch is applied. In other words, only untick files which you want to be deleted.

If you are running Vista, there may also be a UAC shield on the "Create" button if the baseline MSI requires privilege elevation to install. This is because the first stage of creating a patch is to extract the files from the original MSI.

Next it will create a new MSI using the existing settings, so that a patch can then be created containing only the differences between the two. This is why the file list must contain all the files, not just the modified ones. Finally it will use a Microsoft tool to create the patch itself.

Unfortunately that tool is not allowed to be included with this editor. It is however a free, if awkward, download.

GETTING THE PATCH CREATION TOOL

The patch is built using a Microsoft-provided DLL called "`patchwiz.dll`". However, this is not included here because unfortunately the licence states that it can not be distributed. This is the procedure for obtaining this DLL.

1) Visit Microsoft's "Windows Platform SDK" site at

    http://go.microsoft.com/fwlink?LinkID=55774
Although this is the SDK for Vista, it is still valid for XP. It is a different thing to the FSX SDK.

2) Validate your copy of Windows and click "Download".

3) Once "Setup.exe" (400Kb) has downloaded, run it. An installer window will appear. Click "Next" on the first three pages until you reach the component selection page. De-select all items except "Win32 Development Tools" as shown here:
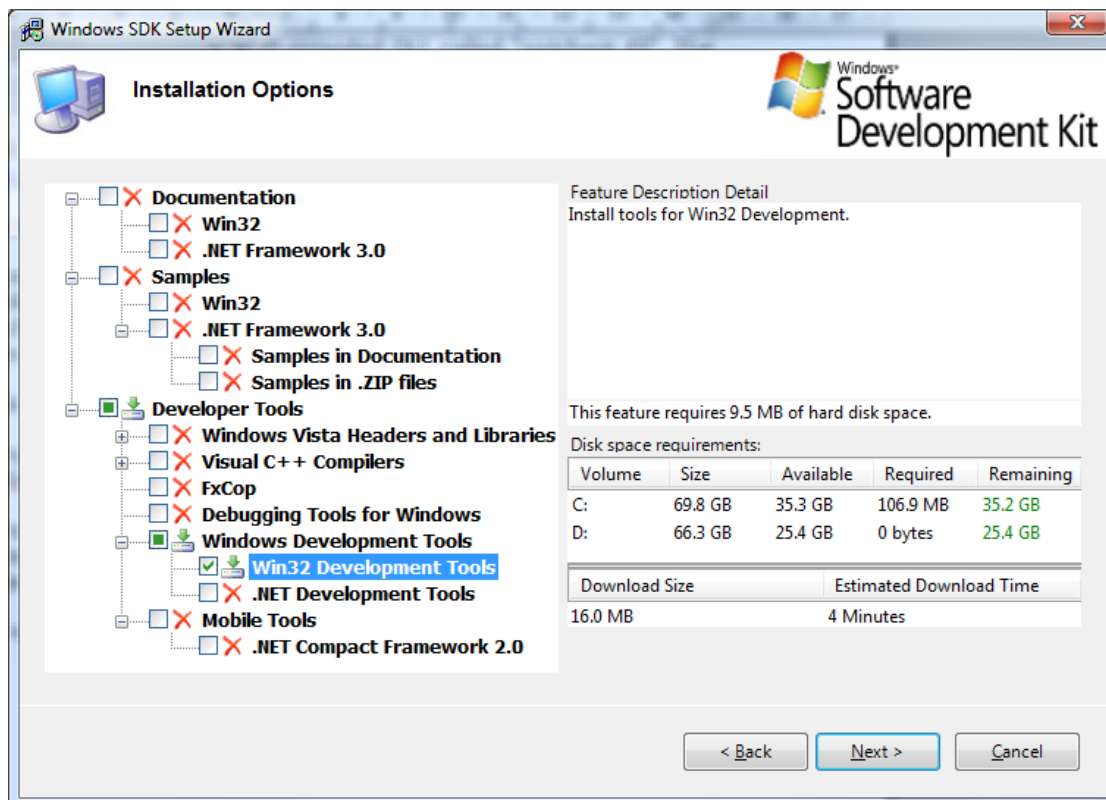
4) Click "Next" another two times and wait for it to finish.

5) Using Windows Explorer, open "`C:\Program Files\Microsoft SDKs\Windows\v6.0\Bin`".

Page 65

6) Copy "`patchwiz.dll`" and "`mspatchc.dll`" to the same folder you installed FSXME to; it should contain "MissionDisplay.exe". By default, this is "`C:\Program Files\FSAddon\FSX Mission Editor`". Again depending on whether you run Vista, you may get more UAC requests here.
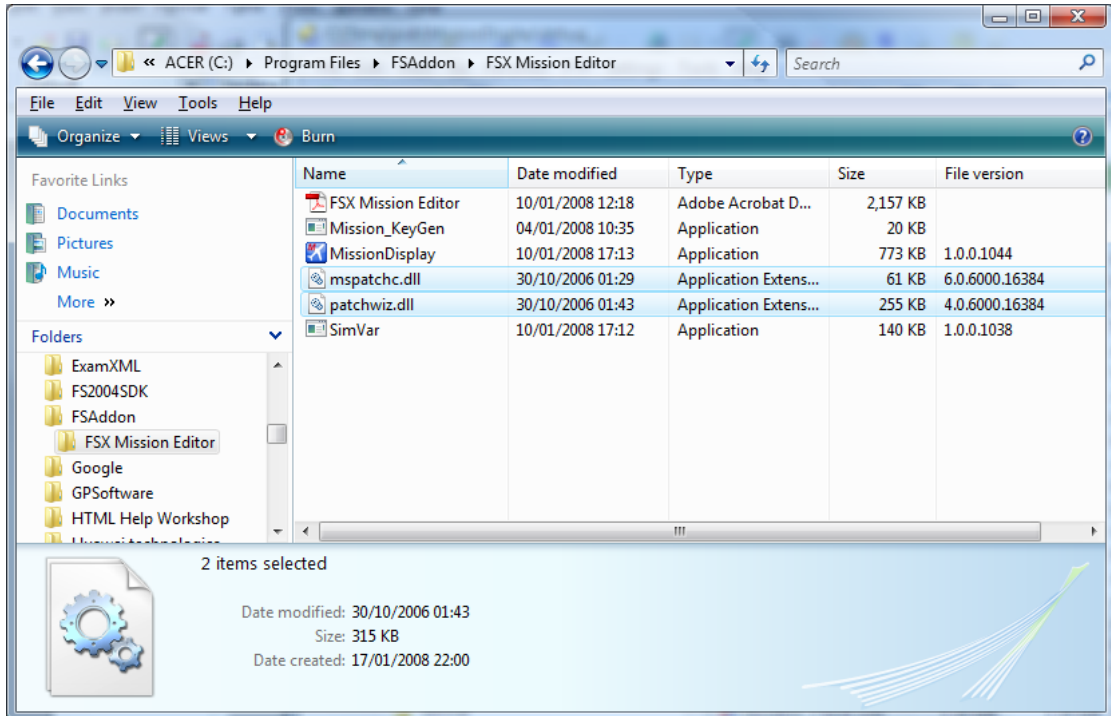
7) Re-run "Setup.exe" and remove the SDK.

The file details are as follows; older versions will not work.

– **patchwiz.dll**, 260,608 bytes, V4.0.6000.16384, 30/10/2006 01:43

– **mspatchc.dll**, 61,952 bytes, V6.0.6000.16384, 30/10/2006 01:29

ADVANCED CONFIGURATION ✕

One of the most potentially useful commands available in a mission is the "CustomAction". This allows external code, working with FSX using SimConnect, to do almost anything. It only has two attributes, one of which is the one that tells the SimConnect extension code what to do. This attribute, the PayloadString, can be set to anything.

If you have written a SimConnect extension, you may well be able to create a template for your new mission commands to allow them to be added to the mission as though they were one of the predefined commands. This will make the command easier to use for other mission writers.

Template-based CustomActions appear in the "New Types" palette as "Special Actions". To add a new one, you need to create an XML file describing your new command. For example:

```
<Mission.CustomDef Name="Debug Relocate">
  <Description>
    When debugging with the Mission Editor, move the aircraft.
  </Description>

  <Properties>
    <Constant Value="Debug:RELOCATE"/>
    <Property Display="Position" Type="LLA"
      Default="N60° 0' 0.00,E130° 0' 0.00,+000100.00"
      HelpText="Position to relocate to."/>
    <Property Display="Orientation" Type="PBH"
      HelpText="Orientation of the plane."/>
  </Properties>

</Mission.CustomDef>
```

The header information is straightforward. The "Name" attribute of the "Mission.CustomDef" node is the name that appears in the "New Types" palette, and is used for new nodes created using your template. The "Description" node is the description of the command that appears at the bottom of the palette.

DEFINING THE PROPERTIES

The next section, 'Properties', defines what fields you want to appear in the Attribute Editor. Each of these fields is concatenated to form the PayloadString, with a space between each entry. Note that, as with all XML, these values are case-sensitive.

Within Properties, you can define "Property" and "Constant" nodes. The "Constant" is a bit of text that will get copied into the PayloadString without being altered. The other type, "Property", is where you define the fields that the user will see as editable.

A Property description contains several attributes:

- **Display** is the label that appears to the left of the editable control in the Attribute Editor.

- **Type** defines the datatype to expect (see below).

Optional attributes are:

- **Default** sets a default value.

- **Helptext** defines the text that appears at the bottom of the Attribute Editor for this field.

Page 67

- **DisplayOnly** should be "True" or "False". If "True", this field is not added to the PayloadString; the default is "False".

- **Enabled** can be set to the name of a Bool Property, to dynamically enable or disable this control depending on the state of the Bool property's tickbox. This is especially useful with DisplayOnly Bools.

- **Prefix** specifies a bit of text that will be automatically put at the start of this property when it is added to the PayloadString. This, and Suffix, can only be used with Text and Named List items.

- **Suffix** is the same, but the text is added to the end of the property.

Several datatypes are supported:

- **Float** is a floating-point number, and will show a standard edit control.

- **Int** is an integer (whole) number, also showing a standard edit control.

- **Bool** is a true/false field and will be shown as a tickbox.

- **Text** is free text.

- **LLA** is Latitude/Longitude/Altitude. This gets added to the PayloadString as a single quoted string.

- **PBH** is Pitch/Bank/Heading. This gets added to the PayloadString as a single string, with the three values separated by commas.

- **ObjectReference** is a box that will accept a dropped item, and puts the GUID of that item into the PayloadString.

- A named list, shown using a drop-down list. See Named Lists, below. If the list can't be found, a text-box is displayed instead.

DYNAMIC CONTROLS

You can change which controls are available dynamically by adding a Bool property. This would normally be set to "DisplayOnly", meaning that the value of the Bool isn't added to the PayloadString. To make other controls only appear if the Bool property is true, add the name of the Bool property in the Enabled attribute. For example:

```
<Property Display="Else?" Type="Bool"
     Default="False" DisplayOnly="True"/>
<Constant Value="ELSE" Enabled="Else?"/>
<Property Display="Else Do" Enabled="Else?" Type="ObjectReference"/>
```

This creates a Bool property called "Else?". The following two entries, the constant "ELSE" and the Property "Else Do", are only used if the tickbox for "Else?" is ticked.

You can use this to disable controls by prefixing the Bool control's name with "!". For example:

```
<Property Display="Use Option 1?" Type="Bool"
  Default="False" DisplayOnly="True"/>
<Property Display="Option 1" Type="Text" Default="One"
  Enabled="Use Option 1?"/>
<Property Display="Option 2" Type="Text" Default="Two"
  Enabled="!Use Option 1?"/>
```

When the "Use Option 1?" tickbox is ticked, "Option 1" will be shown. When it is not ticked, "Option 2" will be shown.

You can also use the value of one Named List control to choose the name of a list in another Named List control. This is usually used to narrow down large lists, with the best example being the list of SimObjects. These are split into categories, and all of these are read by default and are already available as lists. You can, of course, define your own.

This code would create a Named List control asking for the category name, and another which depends on the value of the first. Enclosing the first control's name in {braces} in the second control's Type attribute makes it dynamically set. If for example the data value selected in the first list was "aircraft", the second list would show the contents of the list "SimObjects_aircraft".

```
<Property Display="Object Type" Type="SimObjectTypes"
   DisplayOnly="True"
   HelpText="Select the type of object to list."/>
<Property Display="SimObject" Type="SimObjects_{Object Type}"
   HelpText="List of objects of the selected type."/>
```

NAMED LISTS

If one of your fields is expected to be one of a set of fixed values, you can use a listbox control to show these rather than have the user type one in manually. By setting the "Type" attribute of a Property to be the name of a list, that list is used instead of having a free-text box. There are many lists already defined, mostly coming from the files under "propdefs" in the FSX directory. You can define your own lists too, so you aren't limited to using the predefined ones.

A list is defined using a description of how to read it from an XML file. These instructions need to be in the CustomDefs file you are creating to contain your command template, but the data it uses can come from any XML file. It can even come from more than one XML file. You will need at least a little knowledge of using XPath pattern matching to create these instructions. The list builder is very flexible, and so can be fairly complex. However, if you're writing your own SimConnect extensions this shouldn't be too taxing!

There are several good XPath references available online, including:

- http://en.wikipedia.org/wiki/XPath

- http://msdn2.microsoft.com/en-us/library/ms256471.aspx

This is a summary of the method that's used to read lists:

```
Create a list called {Name}
For each file that matches {FilePattern}
  Find the node which matches {Parent}
  For each node that matches {ForEach} under {Parent}
    Add the attributes or node values for {DataValue} and {DisplayValue}
                  to the list
  End For
End For
```

A list is defined using an XML node called "Enum". This contains various attributes which describe how to read the list:

- **Name** sets the name of the list. This is what you would use as the "Type" attribute of any Property that uses this list.

- **FilePattern** is the filename that contains the list data. This can be a single filename, or a pattern e.g. "list*.xml", or the special value "." which means "this file". The current directory is the FSX installation directory. Using absolute paths is usually not a good idea, since the path may well be different on other people's PCs.

- **Parent** is an XPath filter that selects a single node which contains all of the items in the list.

- **ForEach** is the XPath filter that will select each of the list items under the Parent.

- **DataValue** sets the value that is used when building the PayloadString. If it is prefixed with "@", it is assumed to be the name of an XML attribute. If prefixed with "." or "/" it is assumed to be an XPath pattern, and if the prefix is "$", or there is no prefix, it is assumed to be a constant.

- **DisplayValue** works the same way, but defines the text to be visible in the list. This may, of course, be the same as the DataValue.

- **Default** is an XPath filter that defines the default value, again beneath the Parent. It uses the same prefixes as DataValue and DisplayValue, so you can set the default to be a constant, or the contents of a node or an attribute. If you are reading from more than one file, the Default value must be a constant. If no Default is set, the first list item becomes the default.

- **Description** sets a textual description of this list. This is only used when displaying the 'List of Lists' inside the editor.

NAMED LIST EXAMPLES

Let's run through a specific example and see if it becomes any clearer!

```
<Enum Name="SV_WPT_Approach"
    FilePattern="."
    Parent="//Enum[@Name='SV_WPT_Approach']"
    ForEach="./Value"
    DataValue="@data"
    DisplayValue="@display"
    Default="./Value[@default='True']/@display"
    Description="Set once or updated once per second"
>
  <Value data="INTERCEPT" display="Maximum speed"/>
  <Value data="STATION" display="Maintain position" default="True"/>
</Enum>
```

The FilePattern is set to " . ", so the list data is expected to be in the same XML file as the list definition.

Parent is "`//Enum[@Name='SV_WPT_Approach']`". This selects the first node, anywhere in the XML, which is called "`Enum`" and has a "`Name`" attribute set to "`SV_WPT_Approach`". ForEach is set to "`./Value`", which in this case selects all "`Values`" nodes under the list definition itself. Each "`Value`" node is expected to have a "`data`" attribute for the DataValue, and a "`display`" attribute for the DisplayValue.

The Default value is also an XPath filter, because it starts with " . /". It finds the first "Value" node under the list's Parent node which has an attribute called "`default`", which is set to True. The value of that node's "`display`" attribute is used as the default value for the list. In this case, the second list item has a "`default`" attribute set to "`True`", so the list's default value is set to "`Maintain position`".

So, from this XML a list would be created called "SV_WPT_Approach" and would contain two items:

- "Maximum speed", for which the text "INTERCEPT" would be added to the PayloadString

- "Maintain position", for which the text "STATION" would be added to the PayloadString. This is the default value for the list.

Here's another example:

```
<!-- Category Descriptions -->
<Enum Name="CategoryDesc"
  FilePattern="Categories\*.xml"
  Parent="/SimBase.Document"
  ForEach="./SimMissionUI.ScenarioCategory"
  DataValue="@id"
  DisplayValue="./Descr"
/>
```

This one creates a list called "CategoryDesc", which reads all XML files under "Categories". Inside each of those files, it will use every node called "SimMissionUI.ScenarioCategory" under a Parent node of

"SimBase.Document" as a list item. The "id" attribute will be the data item, and the "Descr" subnode will be the displayed text. Here's a modified extract from one of the category files that this list definition will read:

```
<SimBase.Document>
    <SimMissionUI.ScenarioCategory
            id="{93290E1B-CDE3-4263-B7AE-B8F87F99D901}">
        <Descr>Pick up where you left off in Free Flight.</Descr>
        <Title>My Saved Flights</Title>
        <PreviewImage>banner_allcategories.bmp</PreviewImage>
    </SimMissionUI.ScenarioCategory>
    <SimMissionUI.ScenarioCategory
            id="{89A043AF-EB9E-4248-8A3E-D645309542B9}">
        <Descr>Fun and games as you master the basics.</Descr>
        <Title>Tutorial</Title>
        <PreviewImage>banner_tutorials.bmp</PreviewImage>
    </SimMissionUI.ScenarioCategory>
  ...
```

So, this list definition would create a list of all mission categories that FSX knows about with the category name used in the dropdown list box, and the category ID put into the PayloadString.

This method of list building should let you gather list items from any XML file. If you are defining your own lists, putting them in the same place as the list definition will keep things simpler, with only one file needed. Most lists defined by FSX are already available, so check the List of Lists before defining your own.

LIST OF LISTS

As mentioned above, there are many lists which are already defined and are used as part of the editor. You can use any of these in a CustomAction template without having to redefine them yourself. To see the existing lists, press F12 or use the "Mission→List of Lists" menu item.
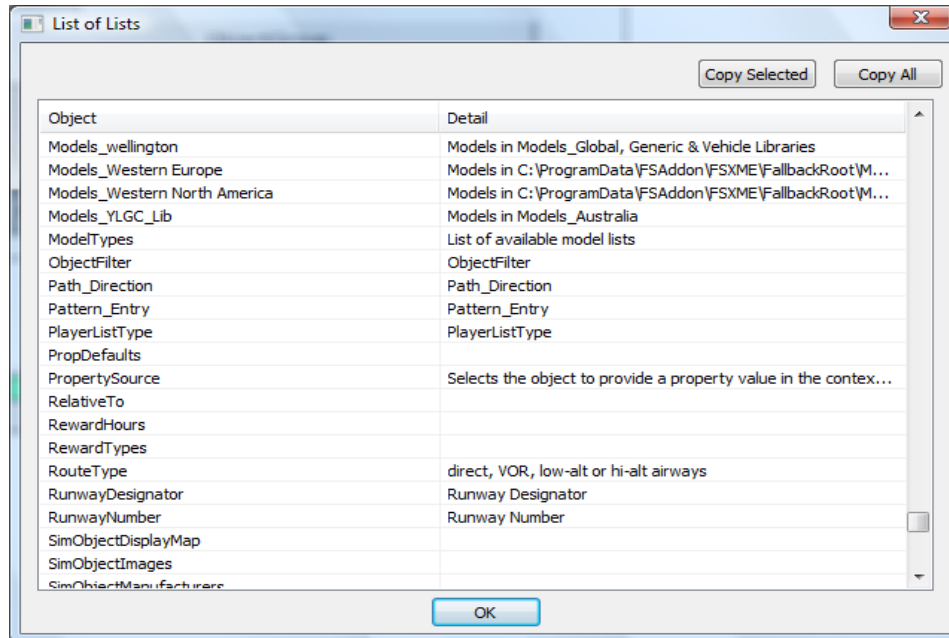
Double-clicking on any list here will pop up another dialog containing the items in that list, so you can use these to check exactly what is available for your CustomAction templates.

Menu Quick Reference

This simply lists all the menu items along with a very short description of what they do. Why not print these pages to use as quick-reference cards? Acrobat Reader will do this using Ctrl+P and then enter "71 - 75" in the Page Range section.

File

New	Create a new mission document
New Wizard Mission	Clear the current mission and run the Wizard
Open…	Load an existing mission from disk
Save	Save this mission using the same name
Save As…	Save this mission using a different name
Save layout only	Save only the layout file, but not the main mission data
Save selected as recipe	Create a Recipe from the nodes that are currently selected
Print…	Print this mission using the current settings
Print Preview	Show on-screen what would be printed
Print Setup	Show the standard Windows printing dialog
Page Layout	Set the scaling rules for printing this mission
Regenerate BGL data lists	Scan your FSX installation and create XML lists of all the models, AI objects and their attachpoints

Edit

Copy	Copy any selected nodes and their XML to the clipboard
Paste	Create copies of copied nodes to either paste into the mission as a new node, or to paste into a text editor as XML
Find	Search for text in all nodes in this mission
Preferences	Change the way the mission is displayed

VIEW

LOGIC NODES ONLY	Choose whether to show just logic or all nodes
UNATTACHED MODELS	Choose whether to show visible objects like scenery which aren't referenced by anything
AREAREFS	Choose whether to show area definitions
PLAYER REFERENCES	Choose whether to show the special 'Player' node
TITLES ONLY	Choose whether to show any attributes at all
ALL ATTRIBUTES	Choose whether to show all attributes or only those that are most commonly used
UNDEFINED VALUES	Choose whether to show undefined attributes
LEGEND	Choose whether to show the mapping of colours to node types
COMMENTS	Choose whether to show comments beneath nodes
PRIVATE ATTRIBUTES	Choose whether to show attributes which should only be set by FSX when saving a mission in progress. Off by default.

WINDOWS

TOOLBAR	Show or hide the toobar
STATUS BAR	Show or hide the status bar
ACTION PALETTE	Show or hide the window that allows creation of new nodes
OVERVIEW	Show or hide the window that shows the entire mission at once
LEGEND	Show or hide a list of the node types in use, in a separate window
ALL ERRORS	Show a dialog containing every error and warning in this mission
DIALOG SCRIPT	Show a dialog containing the text from every DialogAction
ALL COMMENTS	Show a dialog containing all comments in this mission

DEBUG EVENTS	Show a dialog containing all debug events sent from a FSX while FSXME is monitoring
XML FOR SELECTION	Show the XML for any selected nodes
LIST OF LISTS	Show a dialog containing all lists which can be used in CustomAction templates

LAYOUT

ALLOW PARENT ON ALL NODES	Reset all nodes which have had 'Allow Parent' switched off
CLEAR ALL PREFERRED PARENTS	Reset all nodes which have had a preferred layout parent set
UNHIDE ALL NODES	Reset all nodes which have been explicitly hidden, or which have had their children explicitly hidden
HIDE CONSECUTIVE DIALOGS	Where more than one DialogAction has been used by a single parent node, hide all but the first
HIDE WAYPOINTS	Hide all visible waypoint nodes

MISSION

IGNORE MISSING FILES	Don't raise errors for files such as images which are referred to but can't be found
IGNORE MISSING REWARDS	Don't raise errors for rewards which are used by GrantRewardActions but which can't be found
IGNORE STYLE WARNINGS	Ignore the least serious type of errors
CREATE INSTALLATION	Create an MSI installation package for this mission
CREATE PATCH	Create a patch containing only the differences from the most recent full MSI
COMPILE TO SPB	Produce a compiled version of the mission, if you have the FSX SDK installed.
CREATE LICENCE KEYS	Show a dialog which allows you to create or check licence keys for a protected MSI
CREATE TEMPLATE FILES	Create a set of default FLT, WX and HTM files
CREATE DIALOG SPEECH	Create a set of WAV files containing synthesized speech for all of the Dialogs
SET DESCRIPTION	Change the Title and Description values, which are used on the Mission screen in FSX

SET COMPATIBILITY	Change which version of FSX you want this mission to be for; one of RTM (original), SP1 (ServicePack 1), SP2 or Acceleration.
SHOW COMPATIBILITY	Change which version of FSX you want this mission to be checked against.
SET REWARDS FILE	Choose the XML file in which to load and save reward definitions.

DEBUG

MONITOR	Start waiting for debug messages to come from FSX when flying a debug-enabled mission
SIMCONNECT PORT	Choose which method to use when connecting to FSX via SimConnect
SAVE TRACE	Save any debug data already collected to an XML file
LOAD TRACE	Reload an XML-based debug trace file for this mission
CLEAR TRACE	Remove any debug data already collected for this mission
PATCH ALL TRIGGERS	Modify this mission so that every trigger reports debug data
UNPATCH ALL TRIGGERS	Remove any debug reporting actions from this mission

HELP

ENTER LICENSE…	Enter or change your license key
SHOW STARTUP OPTIONS	Show the Welcome window on startup
OPEN MANUAL (PDF)	Open the PDF manual
ABOUT…	Show the version number and some statistics for this mission
CHECK OPT SETUP	Check and modify the OPT setup in dll.xml

CONTEXT MENU

REFERENCED BY →	List all nodes that refer to this one
REFERENCES →	List all nodes that this one references
PREFERRED PARENT →	Choose which of this node's parent(s) is used for the main display, rather than the 'shadow' display which doesn't show this node's children
ALLOW PARENT	When unticked, this node will never be shown with a parent node and will therefore always start a new group
HIDE CHILDREN	Hide all child nodes of this node
HIDE THIS	Hide this node
CUSTOM COLOUR	Change a node's background colour
LIST ERRORS	Show errors for this node only
EDIT COMMENT	Edit this node's comment text
DETACH	Separate this node from it's visible parent
EDIT ATTRIBUTES	Show the attribute editor for this node
REVERT CHANGES	If attributes have been changed but not yet saved or viewed, reset them
RENAME	Change this node's display name
DELETE	Remove this node. If it is linked to more than once, only this one copy is removed.
SET TRACE	Add a CustomAction to this trigger which will report back when monitoring for debug data
SET PAUSE	Add a CustomAction to this trigger which will pause FSX and then report back when monitoring for debug data

KEYBOARD SHORTCUTS

These shortcuts are in addition to the normal, Windows ones.

-	Zoom out
+	Zoom in
HOME	Move main display to far left
END	Move main display to far right
CTRL+HOME	Move main display to top left
CTRL+END	Move main display to bottom left
CTRL+LEFT	Select the parent node
CTRL+RIGHT	Select the first child node
CTRL+UP	Select the previous sibling node
CTRL+DOWN	Select the next sibling node
PAGE UP	Main display moves one page up
PAGE DOWN	Main display moves one page down
DEL	Delete selected node(s)
SHIFT+D	Detach selected node(s)
ENTER	Edit attributes of selected node(s)
;	Edit comments of selected node(s)
CTRL+F	Find text in all nodes
H	Show/Hide Children
F2	Rename selected node(s)
F5	Refresh main display

F6	Show all errors in this mission
Sʜɪꜰᴛ+**F6**	Show errors for a selected node
F7	Show all dialog text in this mission
F8	Show all comments in this mission
F9	Set a breakpoint on this node
F10	Show all debug events in this mission
F11	Show XML for selected node(s)
F12	Show the List of Lists

SIMVAR MISSION EXTENSION

Even with the flexibility of the existing commands, there are places where achieving certain things is not currently possible. Many of these areas are available, but only to programmers via SimConnect.

Some of these things have been made available using this Mission command extension. The new features are integrated into the editor, and can be used in any mission provided the extension program is installed into FSX on the users' PCs.

The extension provides:

– Read/write access to all internal variables, as described in the SimConnect SDK

– Create mission-specific variables and store them between missions

– Extended 'IF', supporting both SimVars and internal variables

– Can create weather conditions (although FSX's weather system is buggy)

– Send keypresses to the sim in response to mission events

– More configurable fuel-leak

– AI can have waypoints set relative to the user, or other AI

INSTALLING

The simvar.exe program can be put anywhere. To make it start automatically with FSX, the file "exe.xml" will need to be modified to include the full path, as described in the FSX SDK. The new section should look like this:

```
<Launch.Addon>
  <Name>Mission Extension for FSX</Name>
  <Disabled>False</Disabled>
  <ManualLoad>False</ManualLoad>
  <Path>D:\Games\Microsoft Flight Simulator X\Extensions\SimVar.exe</Path>
</Launch.Addon>
```

Alternatively you can run simvar.exe manually after FSX has started. Add "/debug" to the command to enable debug messages by default.

If you include any extension .exe or .dll in an MSI, it will be automatically registered when the MSI is installed and removed when the last mission that uses it is uninstalled.

COMMAND DETAILS

The new commands are all implemented as CustomActions, with the specific command and its arguments being passed in the PayloadString. The editor's support for adding CustomAction types is used to support the new commands in the editor. In the PayloadString, all commands start with "SimVar" (case-sensitive). For example, a complete PayloadString might be:

```
SimVar PROFILE MyVariable MUL 3
```

VARIABLES

Both SimVars and internal variables are referenced by prefixing their name with a dollar sign. SimVars which have spaces in their names should be surrounded with double quote marks. For example, an internal variable could be referenced as "$Laps"; a SimVar could be "$LIGHT NAV ON". You can also specify in what units to retrieve the SimVar by adding "IN <Units>", for example "$ALTITUDE IN feet" or "$TOTAL WEIGHT IN kg". Note that the "IN units" part shouldn't be enclosed in quotes.

Anywhere in the following command descriptions where a parameter is needed, you can use a constant, an internal variable or a SimVar. If you need to concatenate a variable value with a constant string, you can end the variable name with a single dot to show where the variable name ends and the string starts. For example:

```
MESSAGE "Your plane currently weighs $TOTAL WEIGHT. Pounds"
```

COMMAND SUMMARY

IF	Test any SimVar
SET	Set any writeable SimVar
DEBUG	Switch the extension's debug messages on or off
PROFILE	Set an internal variable
SAVEONFAIL	Save the profile even if the mission fails
FUELLEAK	Start or stop a controlled fuel leak
MESSAGE	Display a small text message on screen
FXTRACK	Attach a visual effect to the player or an AI
METAR	Set local or global weather
WXSTATION	Create a new weather station
EVENT	Send a simulator event
PROFILENAME	Set the name of the file to store the mission profile in
WPT	Set dynamic waypoints relative to the player or an AI
SIMRATE	Set the simulation speed
SIMTIME	Set the time and optionally the day of the year
TARGETCAM	Force a cinematic camera to show two aircraft in the same view

IF COMMAND

This is an extension to the existing PropertyTrigger command. It supports testing of both SimVars and internal variables, and also has an optional 'ELSE' clause.

```
IF <Value> [<|>|<=|>=|=|!=] <Value> THEN {GUID} [ ELSE {GUID} ]
```

Example:

```
IF "$ENGINE TYPE" != 5 THEN {GUID}
```

... where the GUID refers to a DialogAction, reporting "You're not flying a Turboprop! Cheat!". In the editor, the GUIDs can be set using the normal drag-n-drop method for linking nodes.

SET COMMAND

This allows you to set Sim variables. Since many of the Sim variables are specific to one aircraft, you need to specify an aircraft ID; use an empty string ("") for 'the player'. Note that when setting a variable, you shouldn't use the dollar symbol as a prefix.

Only those SimVars which are marked as "Settable" in the SimConnect SDK documentation can be written.

```
SET <TailNumber> <Name> [Units] <Value>
```

Examples:

```
SET "" "KOHLSMAN SETTING MB" millibars 1012
```

Set the player's altimeter pressure setting to 1012mb .

```
SET N-NICD "CANOPY OPEN" percent 100
```

Open N-NICD's main exit door.

DEBUG COMMAND

This switches some debugging messages on or off. When enabled, these are written to the console window and so are only visible if you have manually started the extension code from a DOS box. Only useful for mission designers.

They typically show the messages received from, and sent to, FSX via SimConnect.

```
DEBUG <TRUE|FALSE>
```

PROFILE COMMAND

Similar to the SET command, but this sets internal variables. By default, all PROFILE (internal) variables are saved to disk when the mission is completed. You can use any variable name you like as long as it contains no special characters.

```
PROFILE <TagName> [SET|ADD|SUB|MUL|DIV|MIN|MAX|ERASE|ROUND] <Value>
```

The third parameter states what to do to the named variable. The ERASE option unsets it; this would usually be used to stop variables from being written to the stored profile when the mission completes.

Examples:

```
PROFILE LapsToGo SET 0
```

Set LapsToGo to 0.

```
PROFILE LapsPenalty ADD 1
```

Add one to LapsPenalty.

```
PROFILE LapsTotal SET $LapsToGo
PROFILE LapsTotal ADD $LapsPenalty
```

Calculate the total number of laps to complete and store the result in LapsTotal. This would need to be done in two separate CustomActions, called from the same Trigger node.

SAVEONFAIL COMMAND

This simply tells the extension whether you want to save any PROFILE variables when the mission fails. Normally they are only written when a mission is completed successfully.

```
SAVEONFAIL <TRUE|FALSE>
```

SAVEONCHANGE COMMAND

Similarly to the previous command, this one tells SimVar to write the profile to disk after every change. This would allow you to maintain the mission state even if the simulator were closed down.

```
SAVEONCHANGE <TRUE|FALSE>
```

UNIQUEID COMMAND

Normally, when the mission state is saved to disk there is a signature included with it. This uniquely identifies the mission from which the PROFILE settings were generated, and the contents of the file itself. This prevents people from manually editing the profile contents and reloading a saved mission.

However, this prevents PROFILE values from one mission being loaded into another, which would be the case if you had written a series of missions as a single continuous story. By setting the UNIQUEID for both missions to be the same value, PROFILE settings can be shared between them.

```
UNIQUEID <Free text, 256 characters maximum>
```

FUELLEAK COMMAND

Although there is a fuel-leak command provided by FSX, this one is provided as an alternative to that. The difference is that you specify how much fuel you want remaining after a fixed period of time, instead of specifying the leak rate. This allows you to more accurately script your out-of-fuel locations.

```
FUELLEAK <percent_remaining> AFTER <seconds>
```

Setting <percent_remaining> to anything less than 0 will stop a current leak, as will the "STOP" command.

Examples:

```
FUELLEAK 3 AFTER 600
```

Exactly ten minutes (600 seconds) from now, you will have 3% fuel remaining.

```
FUELLEAK -1 AFTER 100
```

Stop leaking now. The time value is ignored if percent_remaining is < 0.

```
FUELLEAK STOP
```

Also means stop leaking now.

```
PROFILE LkTime SET 90
IF $Level = 1 THEN {GUID} --> PROFILE LkTime SET 60
IF $Level = 2 THEN {GUID} --> PROFILE LkTime SET 45
FUELLEAK 0 AFTER $LkTime
```

Assuming "Level" has been set by a previous menu choice to 0, 1 or 2, set LkTime to one of 90, 60 or 45. Then, later, run out of fuel after a number of seconds that corresponds to the difficulty level the user chose. This example would need to be split over six separate CustomAction nodes.

MESSAGE COMMAND

In FSX ServicePack 1 and above, it is possible to add messages to the display. The text for these, like most other values, can be partly or entirely based on SimVars or internal variables. The message should be surrounded with double quotes if it contains spaces.

```
MESSAGE <Text> [Duration]
```

Example:

```
MESSAGE "You are flying a $TITLE"
```

This shows a text message like "You are flying a Cessna 182", depending on the player's chosen aircraft for five seconds (the default).

```
MESSAGE "This is a longer message" 8.4
```

This shows the text "This is a longer message" for 8.4 seconds.

FXTRACK Command

This moves a specified AI object to a fixed location relative to an aircraft, and updates that position every frame. It is meant as a hack to work around the inability to attach effects to AI or player aircraft.

Although all the aircraft have attachpoints, all of these are fixed to being used with a single effect. FSX SP1 allows any effect to be attached to the user, but this is broken in both RTM and Acceleration/SP2. This makes the "AttachEffectAction" useless for anything but static scenery or a small number of visual models.

To try and work around this, I have provided ten new SimObjects. These have no visual model, but do have a single, unused attachpoint. They also have unique names, which allows them to be identified by a SimConnect client. You would attach your effect to one of these objects, and then tell the extension to slave that object's position to another aircraft.

The new SimObjects are called "AI_Tracker_Object_1" through "AI_Tracker_Object_10". To attach an effect to the player, you would create an additional AI with it's model set as "AI_Tracker_Object_1" and then attach the effect to that instead. You would also need to tell the extension to slave "AI_Tracker_Object_1" to the player's location.
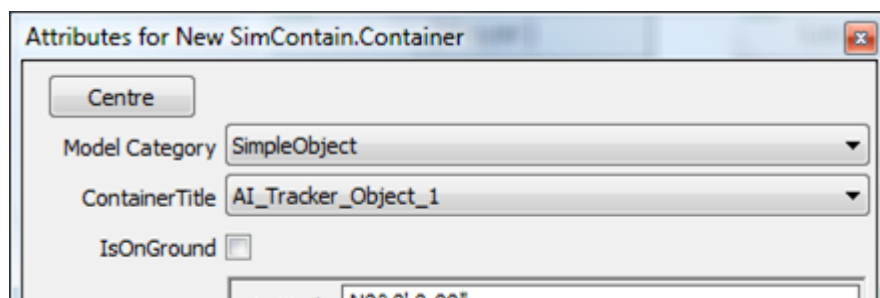

```
FXTRACK <TrackID> <MasterID> dX dY dZ
```

"TrackID" is the ContainerTitle of the AI you want to control, in this case "AI_Tracker_Object_1". "MasterID" should be the tailnumber of the aircraft you want to slave the effect-carrier AI to. As usual, use "" to mean "The player".

dX, dY and dZ are offsets in meters from the target plane's center to add to the slave AI's position. Using these you can make effects appear at different places relative to the target plane.

```
FXTRACK <TrackID> STOP
```

This stops controlling the effect AI. However, it is not possible to restore control to FSX so it will remain static.

METAR Command

This command relays the METAR string you provide to the weather system of FSX. This allows you to change weather conditions.

The METAR string is documented in the SimConnect SDK.

```
METAR <Station> <METAR text>
```

The Station ID should be the unique ID of the weather station you want to change, or one of the fake IDs "GLOB" (which sets weather globally) or "PLYR" which forces a local change (see below).

The actual weather conditions at any given point are the result of a calculation inside FSX involving the conditions of some, but not all, of the nearby weather stations; which ones are chosen depend on the player's current location and can change rapidly. Setting a thunderstorm, for example, over a particular weather station may not have any effect at all if that station is currently not being used as one of the active ones. There is no way to tell which stations are actively used at any given time.

You can set weather globally but since all weather changes apart from cloud layers occur immediately, this can cause abrupt visual changes.

To try and work round these problems, there is an additional fake ID – "PLYR" – which tries to force a weather change by creating three new weather stations very close to the player, and then setting identical weather at those.

Note that setting weather requires that you specify *all* the weather you need, you can't just set the temperature and have everything else stay unchanged. Unfortunately, sometimes the positions of clouds get reset even if the cloud settings are unchanged. Also, most weather changes become visible immediately; only cloud layer changes are graded from one setting to another. Changing visibility, for example, happens immediately.

As with most other values, you can use variables (SimVars or Profile variables) as all or part of the METAR string.

WXSTATION Command

Another way of creating local weather is to create your own, location-specific weather stations which are then taken into account by the FSX weather generation system. This can give smoother transitions between different weather conditions at different locations than forcing global weather changes.

```
WXSTATION ADD <ICAO> "<Latitude> <Longitude> <Altitude>" "<Name>"
WXSTATION REMOVE <ICAO>
```

You must give the position of the new weather station as latitude and longitude values, either the ICAO code of an existing airport with no weather station or a new, unique code, and a description.

Examples:

```
WXSTATION ADD MYWX "S17° 42' 22.94 ,E168° 19' 58.11 ,+000079.67" "MyWxStation"
```

EVENT COMMAND

This allows you to send any of the events listed in the SimConnect SDK to FSX. These usually correspond to keypresses, meaning that you can have almost any action the sim supports being triggered by missions.

```
EVENT <Callsign> "Event Name" [Event Data]
```

Examples:

```
EVENT "" "CAPTURE SCREENSHOT"
```

Capture a screenshot.

```
EVENT "" ENGINE_AUTO_START
```

Start all engines.

```
EVENT "" AUTOPILOT_ON
EVENT "" HEADING_BUG_SET $"PLANE HEADING DEGREES MAGNETIC" IN DEGREES
EVENT "" AP_PANEL_HEADING_SET 1
```

As three separate EVENT CustomActions: Switch on the autopilot, set the heading bug to the current heading and enable heading-hold mode.

Some events need to be targeted at an aircraft, and others need to be targeted at the system. For these, specify a callsign of "#0".

PROFILENAME COMMAND

This simply changes the filename that the profile variables will be saved to at the end of the mission. It also loads the contents of the file, if it already exists. This is to allow the same profile file to be shared between multiple missions. The filename is relative to the mission directory.

```
PROFILENAME <Path_To_XML>
```

Examples:

```
PROFILENAME "..\all_my_missions.xml"
```

Load the profile from the file called "all_my_missions.xml", one directory up from the current mission's location.

WHEN COMMAND

Use this command to map an input event to a mission action. Whenever the keypress or joystick event is sent, the mission action will be called. You can stop listening for input events by telling it to STOP. See the SDK documentation for "SimConnect_MapInputEventToClientEvent" for the description of the input events.

```
WHEN <input_event> THEN <GUID to call>
WHEN <input_event> STOP
```

Examples:

```
WHEN "shift+k" THEN {a GUID}
```

Call the action identified by GUID when shift and k are pressed.

WPT Command

This is the most complex of the added commands. It allows you to create dynamic waypoints for AI aircraft, based on any other aircraft including the player's. You can also use it to immediately move AI aircraft.

Unfortunately there is a bug or misfeature within FSX that limits this command. When you create an AI aircraft from a mission, the IdentificationNumber parameter is ignored. This means that the actual ID may change depending on other AI traffic, and whether the user has altered any of their aircraft.cfg files.

There are several forms of this command.

```
WPT SET <Callsign> "<Latitude>, <Longitude>, <Altitude>"
```

Immediately move the AI aircraft with the tailnumber <Callsign> to the location given. Altitude is in meters. This is the same format as many of the latitude/longitude/altitude entries in the mission XML. Positive and negative numbers are also acceptable i.e. north and east are positive, south and west are negative.

```
WPT [INTERCEPT|STATION] <Callsign> RELATIVE <Callsign> dX dY dZ
```

Tell an AI aircraft to fly to the current location of another aircraft, offset by a given number of meters. The first parameter, INTERCEPT or STATION, tells the system to either fly as fast as possible to exactly the location given (INTERCEPT) or to try and match the target plane's course and speed when nearby (STATION). Once the point is reached, control of the AI is returned to FSX.

The first callsign is the tail ID of the AI aircraft which is to be controlled; the second callsign is the tail ID of the aircraft which is to be used as the base location to fly to. As with other commands, use "" to refer to the player.

The last three parameters can be used to offset the waypoint location from the second aircraft's location; they are distances in meters.

```
WPT [INTERCEPT|STATION] <Callsign> FOLLOW <Callsign> dX dY dZ
```

The second form is the same as the previous command, except that the waypoint's location is updated once per second. This means that you can have an AI aircraft flying wingman with, or performing a real intercept on, the player. The previous command would have the AI fly to the player's location when the command was issued which would usually mean that the user is no longer there when the AI arrives.

Examples:

```
WPT INTERCEPT AI-ONE RELATIVE "" 0 0 0
```

Tell aircraft AI-ONE to fly as fast as possible to the player's current location and then just keep going. That is, act as a one-shot missile (although the player will have moved on by the time AI-ONE gets to the player's *current* location, so it won't collide).

```
WPT INTERCEPT AI-ONE FOLLOW "" 0 0 0
```

Tell AI-ONE to keep trying to get to the player's exact location, updated once per second. That is, act as a targetted missile.

```
WPT STATION AI-ONE FOLLOW "" -100 0 0
```

Tell AI-ONE to follow the player but once it gets close, maintain the same course and speed, 100 meters to the left. That is, fly formation with the player. It's not quite up to Red Arrows standard, but it should be good enough for most purposes.

```
WPT STOP <Callsign>
```

Stop creating realtime waypoints for the given AI Aircraft which has previously been told to FOLLOW.

SIMRATE COMMAND

For some reason there appears to be no way to directly set the simulation rate either from a mission or from SimConnect. This can be important, particularly if you've written a mission which has some 'transit' sections where not much happens. It's very tempting to speed those up a bit but, when something does happen, it's annoying when lots of mission events fire without giving you time to react. This command simply allows you to set the simulation rate.

```
SIMRATE <Rate>
```

Examples:

```
SIMRATE 1
```

Reset the simulation rate to normal.

```
SIMRATE 0.25
```

Set the rate to ¼ speed.

```
SIMRATE 3
```

Set the rate to 2*x* normal speed. Because FSX only supports multiples of 2, the nearest it can get without going over is 2*x*.

SIMTIME COMMAND

A convenience command to set the day and time. This can be done individually using events, but this command does it all at once.

```
SIMTIME <Time> (Date)
```

Examples:

```
SIMTIME 18:37
```

Set the local time to 18:37 .

```
SIMTIME 06:10 210
```

Set the time to 06:10 on day 210 (July 29th).

TARGETCAM COMMAND

The existing 'Cinematic Camera' mission object will allow you to create a view based on the locations of two aircraft. You need to specify an offset from the 'near' aircraft saying how far away the camera should be. This offset is based on the aircraft's orientation; that is, if you place your camera 20 meters to the left of the aircraft, that's where it stays. If either of the aircraft move in relation to each other, one of them may go out of view.

This command modifies the Cinematic Camera view so that the offset from the near aircraft to the camera is set in relation to a line between the two aircraft instead of a line along the axis of the near aircraft. This means that both aircraft say in view.

Normally you would set the OffsetXYZ attributes of the CinematicCamera to 0 and allow the extension to control the camera location. You can use both sets of offsets to achieve different effects though; for example, if you were to set the X offset of the CinematicCamera to -10, then the extension would position the camera relative to a point on a line from the far aircraft through a point 10m to the left of the near aircraft.

```
TARGETCAM <dX> <dY> <dZ>
```

Force the current view to show both the near and far aircraft at the same time, and keep their current orientation. To restore the view correctly you should issue "TARGETCAM 0 0 0" after deactivating the CinematicCamera.

Examples:
```
TARGETCAM -4 3 -20
```

The current cinematic view will keep both the "Reference" and "Relative To" locations of the CinematicCamera in view. It will be 4m to the left, 3m above and 20m behind the "Relative To" location, along a line drawn between that and the "Reference" location. Note that you still need to Activate a CinematicCamera before using this command.

GETWX COMMAND

This command is entirely intended for mission writers. It returns a correctly-formatted METAR string for a named weather station. This gives you a simpler way of building correct METAR strings – you can use the in-built "Weather" dialogs in the simulator and then use SimVar to get the string that corresponds to that weather.

```
GETWX <Station>
```

INTERCEPT COMMAND

This will calculate the intercept details from one aircraft to another, or to a fixed latitude/longitude point. Where the target is an aircraft, it's current speed and heading are taken into account.

If SimVar calculates that the "From" aircraft can never reach the target aircraft, usually in the case where the target is moving faster than your current speed, the _TIME and _DISTANCE values are set to 0 and the _HDG, _LAT and _LONG values are calculated based on the target's current location.

The values are not updated in real-time; if the target is an aircraft and it changes it's speed or heading after you have calculated the intercept point, you will need to recalculate the values again.

```
INTERCEPT <TagBase> <From Tail#> <To Tail#>
INTERCEPT <TagBase> <From Tail#> "LAT LONG ALT"
```

Typically the "From Tail#" would be the user's aircraft, and can therefore be left empty. The results of the calculation are put into five PROFILE variables, named according to "TagBase", the first argument to this command. You can then use these variables in any way you wish, such as in "MESSAGE" text or as input to an EVENT command to set the autopilot.

Examples:

```
INTERCEPT FOLLOW "" "N-A1234"
```

Set the variables as follows:

FOLLOW_TIME : Time, in seconds, until the intercept point.

FOLLOW_HDG : Heading to fly to reach the intercept point.

FOLLOW_LAT : Latitude of the intercept point.

FOLLOW_LONG : Longitude of the intercept point.

FOLLOW_DISTANCE : Distance in NM to the intercept point.

VARIABLE PERSISTENCE

All of the variables defined using the PROFILE command are written out to a file when the mission ends. Normally this is only written when the mission is successful, but you can choose to write it on failure too using the PROFILENAME command.

Since they are written to an XML file, it is possible to use the variables in this file as part of the mission briefing. This gives a nice element of feedback to the user. You can also share these profiles between missions, allowing to put together a package of related missions which change their behaviour depending on the outcomes of other missions.

DYNAMIC BRIEFINGS

To change your briefing text according to the previous results of a mission, you simply add an IFRAME element to the briefing HTML which references the mission's stored profile. You also need to edit an XSL file to format the profile in the way you want.

For example, in Briefing.htm, add this line:

```
<IFRAME SCROLLING="No" FRAMEBORDER="0" SRC="MyMission_profile.xml"
WIDTH="80%"></IFRAME>
```

The filename will of course need to be changed to match your mission's name. This will then include the profile data into the briefing text. Every profile XML file refers to an XML stylesheet called "MissionProfile.xsl". By default this just lists the variables and their values, but you can change it. Editing XSL is beyond the scope of this manual, but there are many online guides available.

CREDITS

As with any complex FS add-on there is no success, not even a product, without many hours of testing by dedicated and enthusiastic volunteers.

FSX Mission Editor is no exception to that rule and we'd like to thank our main testers here:

Paul Donnelly

Lars P. Hammer

Jaap van Hees

Ruud Faber

Darryl Wightman

Scot Gridley

... and apologies to those I may have forgotten !

ABOUT FSADDON PUBLISHING

Since you've come this far, you must be a die-hard simmer, or at least an avid reader. Congratulations on your perseverance, not many people read manuals at all ☺ To reward you, let me tell you something about this company then.

FSAddon Publishing was founded by **François Dumas** and **Miguel Blaufuks** by the end of 2003, with the main purpose of publishing new and existing software aXXXdd-ons for the Microsoft Flight Simulator range of products.

But not just any add-ons! Our aim is to provide **additional immersion** in using the simulation by providing **high-quality complete packages** that do more than just add an airfield, a utility or an aircraft. We are aiming to provide 'reality kits' that are a combination of additional FS software AND other things such as a story line, navigational information and tools, and even community access via the Internet.

The company was initially based upon the already existing **simMarket On-line shop** and the associated **simFlight Network**. simMarket currently is the **world's largest on-line distributor** of flight simulator add-ons with more than **90,000 registered customers** and **well over 1000 products** in the catalogue. But up to 2003 all products were sold under their own name and without any (or much) guidance from the distributor.

The simFlight Network consists of a multitude of flight simulator **news sites**, covering **many languages** and areas around the world, and also hosting a large number of flight simulator user communities via forums and other means. The combined news sites alone have over **one million page hits each month!**

The two organizations together formed a very powerful base for the newly founded FSAddon Publishing, providing all the basic infrastructure, skills and contacts needed to design, develop and publish flight simulator add-ons.

FSAddon Publishing has since evolved into a **full-fledged add-on publishing company** and is pretty much a separate entity, and has rapidly built a name for **high-quality software** and services, on par with the best companies in the world, and formally recognized by Microsoft as such!

The company's aim is 3-tiered:

1. To substantially expand the possibilities for beginning simpilots to use their flight simulator

2. To provide high-quality, extensive add-ons to the more experienced sim-pilots

3. To lower the thresholds for communicating and flying together using flight simulators and the internet.

We hope you'll enjoy our products, and above all, the pleasure of sharing this hobby with us and the hundreds of thousands of like-minded enthusiasts all over the world. If we can add just a little value to it, then we have achieved our goal.

LOOKING FOR DEVELOPERS/DESIGNERS

If you feel **you could add value to this concept** in any way, then we urge you to **contact us**. There are many opportunities for different skills and talents here. Scenery designers, yes, but also script writers, tool builders, story tellers, aircraft designers, web designers and even people that just have a good idea are welcome.

FSAddon Publishing will **initiate projects** and get people together in virtual teams as needed, help them to define, design and **produce a professional product** under the **FSAddon** and **FScene Design** brand names and **market it for them**.

This is a great chance to not only make some money with your hobby and skills, but also to add fun and enjoyment to the lives of so many others that are less talented; our customers and fellow flight-simmers!

Remember, one of our main aims is to bring Flight Simulator and its realism closer to the beginning and average user, in any which way we can!

See you in the (virtual) skies!

OTHER FSADDON PRODUCTS

If you like this product, then you will want to visit FSAddon's website (http://www.fsaddon.com and http://www.fsaddon.eu) from time to time, because we are working on a whole range of similar and other products, from very well-known authors and designers, but also from very talented new people in the flight simulator industry.