PACKAGING TUTORIAL

JANUARY 2012

Putting a Mod Package together for rFactor2 is not very difficult, and if you're familiar with rF1 it's not much different. Primarily, that's who this is written for—those that have worked with rF1 in the past. But, I've written it in such a way as someone new to rFactor modding can follow along just as easily…

Now, when I say "not much different", what I mean is that it's VERY different, but not in the way you may be thinking. While vehicle and track asset creation is pretty much the same and should be familiar, how the content is packaged and delivered will probably be somewhat alien to you at first. However, my hope is that by the time you finish reading this you'll have a pretty clear understanding of what an rFactor2 MOD Package is, how to create one, and why we've chosen to do what we've done.

First, let's start with the "why". Why did we choose to package vehicle and track assets into deliverable MOD Packages? One of the primary complaints/issues players had with rF1 was online mismatches—the endless trial and error of joining an online server only to get booted because of a missing track or modified car file. The MOD package solves this issue by reigning in the modding insanity that was so rampant in rF1. Let me explain how.

A MOD file is made up of one or more components, and each of those components (be it vehicle, track, or other) is made up of one or more .MAS files. When a component is created it is signed, based on the content of the MAS files in it. The entire MOD Package contains all of the required components, and knows the signature of each. So, when you install a new MOD and go to join a server running the same we can check the data and make sure they are identical. If not, either because you changed your data, or the server changed theirs, you can't join. Another benefit of this system is that it also cures content cheating.

Now, as a modder you may be thinking, "Great, now I can't do *x*, *y*, or *z*." I'm willing to bet that you can in fact do 95% of everything you could do in rF1. The only difference is that you HAVE to be more organized in how you go about doing it. That aside, the biggest hurdles you may face are either resisting change (which no one really likes), or learning to make the Packaging system work for you. Hopefully, I can help you over these potential hurdles. So, before we go on, let's go over some definitions so we're both speaking the same language.

Definitions

MAS file, (.mas)—a file that contains some or all of the loose files that create a particular asset, be it a vehicle, track, sound pack, etc. This should be very familiar to you if you did previous modding for rF1.

COMPONENT file (.rfcomp)—this is a single element of a MOD. A component contains all the .mas files that were needed to create a single track, vehicle, etc.

MOD file, (.rfmod)—this contains all of the signed components that it requires. If a MOD consisted of 2 tracks and 3 cars then it would have at LEAST 5 components packed in it, if not more due to sound, UI or other elements.

Now that we know what we’re saying, and before we get into actually making a MOD, let’s quickly go over some common, “Can I…” questions. Hopefully that will help ease some initial worries.

Can I …

… "ship" just a single car or track? Yup, you sure can! Single components can be released and installed for use on their own, or can be included with other MOD packages later. A cool thing here is that you could put some kind of "Created By" info (let's say on the loading screen of a track) on your creation, and encrypt the mas files of your component so they can't be altered. If someone else included that in a MOD they release, people will know who made that part of the MOD. If we can stop worrying about who created what, then maybe we can start sharing our work more.

… release a small update to a MOD I've already released? Again, yes. By including all the files that have changed in an "update" MOD those files take priority over the originals. More than this, you can include all new components in an update, too. So, you could also add new cars and tracks to a current MOD, as well as update parts of the original components. We'll go over how to create these updates in more detail later….

… create "patches" to distribute to my MOD's beta testers? Yes. Again, you can do this by putting the altered files in an update. By doing this you can incrementally increase the MOD version until you've reached its' final state.

… create a MOD that includes a collection of cars and tracks for use by my friends/league members, etc? Yes. Any component you have installed can be used in the creation of a new MOD. So, if your league is running a GT series on an assortment of tracks, just create a .rfmod with these components and distribute it to your league drivers. Everyone will have the exact same data with no mismatches.

… change the files within a .mas file of a component? Absolutely not. What would happen is that when you alter the .mas file its signature will change, and while it would work for offline play, when you went to join an online server you would not be allowed to because of the data difference. This is what prevents mismatches. If you want to alter the data in a mas file you must unpack the mas, change the data, pack up a new .mas and create an entirely new component OR create an update for the original component.

… place loose files in a component directory and have those get used instead of what's in the .mas files? Again, no. Loose files are completely ignored (except when used in the "moddev" directory structure where mod asset creation occurs). Only content in the component's .mas files is used. If you want to change or alter the data in a component you must either unpack, alter, repack, and build a new component, or create an update for the existing component.

… release a mod that uses components not included with the mod itself? Yes. This is the basis for an update MOD (you're referencing components released in the original mod), as well as a "virtual" MOD—a MOD that contains no actual components, only references to them. A virtual MOD is not really meant for large distribution, but rather is for small groups—friends or league members, for example. They MUST have the components already installed that the MOD calls for, otherwise it will NOT install. If the MOD isn't installed then they can't join a server that uses it.

As you can see, while it IS a bit more work to alter a car or track, it's not impossible (nor is it difficult). And while this system may seem rather restrictive at first, it maintains data integrity so that failed connections due to mismatches will not occur.

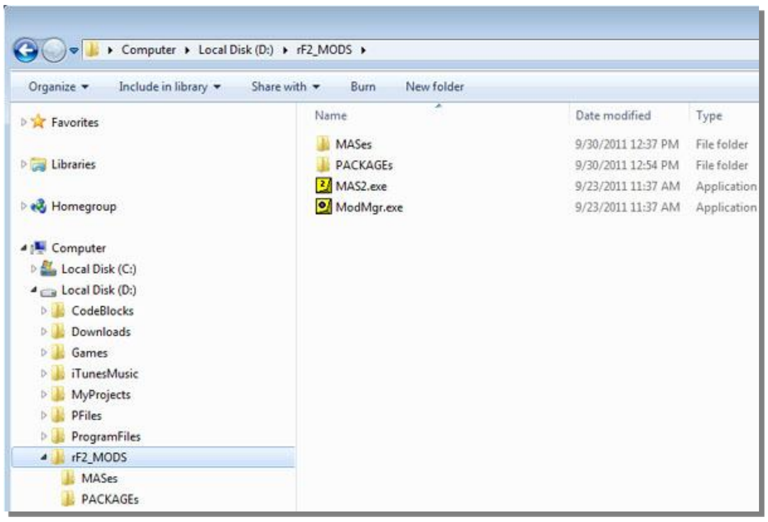
Setup

Now that we've gotten all of that out of the way, let's get ready to build a MOD Package piece by piece. We'll keep things simple and build a MOD that contains one car and one track. I'm going to assume you already know how to create content for rF2--this document isn't meant as a track or car building guide. That said it is important to note that a directory change has been made to rF2.

If you've built content before for the original rFactor, you are familiar with the `/GameData` directory structure, which includes sounds, vehicles, tracks, etc. In rF2, while the sub-directories are the same, the main content directory has changed from “`GameData`” to “`ModDev`”. The assets created by all of this loose content are only accessible in “Developer” game mode, not single, nor multi-player. Development mode also includes all the editors (AIW, Camera, Vehicle, etc) that you will need. Once the assets are ready for the public they are turned into .rfcomp and .rfmod files and distributed, which is what we're about to do using the rTrainer assets that were shipped with rF2 to create a “new” rTrainer MOD package.

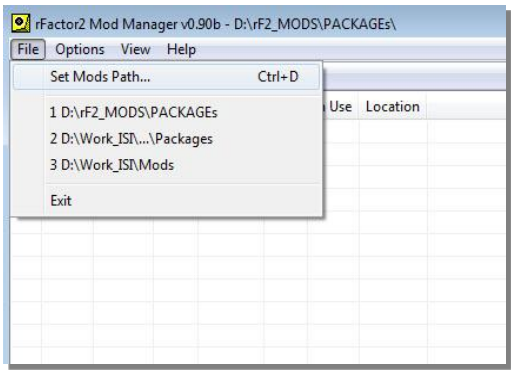
First, we need to put all of the loose files for the sounds, car and track into .mas files. We'll start with the Joesville track. Track components usually have at least five .mas files if you use the ISI standard: “_GMTs.mas”, “_MAPs.mas”, “_SPONSORMAPs.mas, “_ANIMs.mas, and a .mas file for each layout. So, let's make these for Joesville.

It's worth suggesting that you have a copy of ModMgr.exe and MAS2.exe in a separate working directory. This will keep you rF2 install clean and safe from possible corruption (accidentally overwriting current assets, etc). Here’s an example of my working directory:



Name	Date modified	Type
MASes	9/30/2011 12:37 PM	File folder
PACKAGEs	9/30/2011 12:54 PM	File folder
MAS2.exe	9/23/2011 11:37 AM	Application
ModMgr.exe	9/23/2011 11:37 AM	Application

I have a “rF2_MODS” directory, with “MASes” and “PACKAGEs” sub-directories. As I make various components I’ll place these in the proper directories to help stay organized. You don’t have to work this way of course—all down to personal preference.



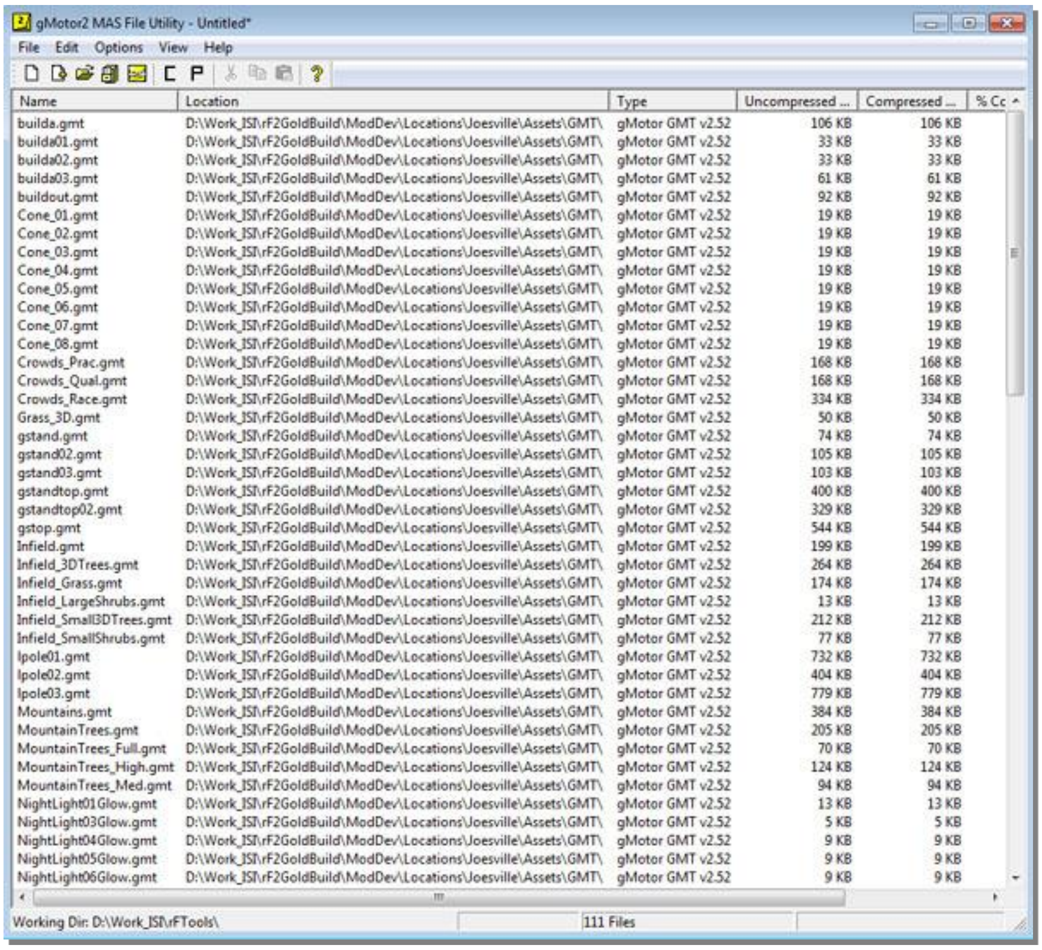
rFactor2 Mod Manager v0.90b - D:\rF2_MODS\PACKAGEs\

At this point you probably want to add your “PACKAGEs” directory to ModMgr as a “search” path. Do this by opening ModMgr.exe, clicking “File” and “Set Mods Path…” You can see in the image below I already have done this, along with 2 other paths I previously set up. ModMgr will look in the currently selected path for .rfcomp and .rfmod files to install. You can select a path simply by clicking on its listing—if installable files are found there they’ll be listed…

So, let's get started…

You've just built the greatest MOD ever—it's time to package that up and ship it out for the rFactor world to see. Go ahead and open MAS2.exe. Again, we'll be using the Developer version of the rTrainer mod as our "world's greatest MOD". Let's start with the Joesville track.

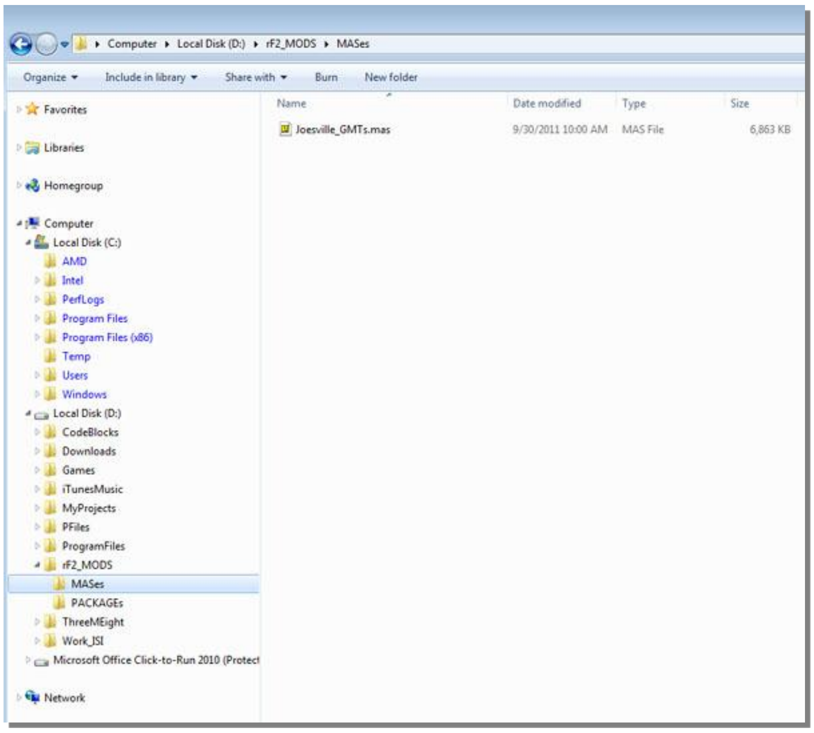
To add loose files to MAS2 you can either drag-n-drop them from an Explorer window into MAS2, or use the "Add files to MAS archive" button ("piece of paper with a '+' symbol" icon). Using the Drag-n-Drop method, open up an Explorer window and browse to your rF2 install's "moddev/locations/joesville/gmts" folder. Select all the .gmt files there and drag them into MAS2's main window. They will then be listed, ready to be saved (see image below):



Name	Location	Type	Uncompressed ...	Compressed ...	% Cc
builda.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	106 KB	106 KB	
builda01.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	33 KB	33 KB	
builda02.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	33 KB	33 KB	
builda03.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	61 KB	61 KB	
buildout.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	92 KB	92 KB	
Cone_01.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	19 KB	19 KB	
Cone_02.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	19 KB	19 KB	
Cone_03.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	19 KB	19 KB	
Cone_04.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	19 KB	19 KB	
Cone_05.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	19 KB	19 KB	
Cone_06.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	19 KB	19 KB	
Cone_07.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	19 KB	19 KB	
Cone_08.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	19 KB	19 KB	
Crowds_Prac.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	168 KB	168 KB	
Crowds_Qual.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	168 KB	168 KB	
Crowds_Race.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	334 KB	334 KB	
Grass_3D.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	50 KB	50 KB	
gstand.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	74 KB	74 KB	
gstand02.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	105 KB	105 KB	
gstand03.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	103 KB	103 KB	
gstandtop.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	400 KB	400 KB	
gstandtop02.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	329 KB	329 KB	
gstop.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	544 KB	544 KB	
Infield.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	199 KB	199 KB	
Infield_3DTrees.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	264 KB	264 KB	
Infield_Grass.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	174 KB	174 KB	
Infield_LargeShrubs.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	13 KB	13 KB	
Infield_Small3DTrees.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	212 KB	212 KB	
Infield_SmallShrubs.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	77 KB	77 KB	
lpole01.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	732 KB	732 KB	
lpole02.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	404 KB	404 KB	
lpole03.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	779 KB	779 KB	
Mountains.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	384 KB	384 KB	
MountainTrees.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	205 KB	205 KB	
MountainTrees_Full.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	70 KB	70 KB	
MountainTrees_High.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	124 KB	124 KB	
MountainTrees_Med.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	94 KB	94 KB	
NightLight01Glow.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	13 KB	13 KB	
NightLight03Glow.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	5 KB	5 KB	
NightLight04Glow.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	9 KB	9 KB	
NightLight05Glow.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	9 KB	9 KB	
NightLight06Glow.gmt	D:\Work_ISI\rF2GoldBuild\ModDev\Locations\Joesville\Assets\GMT\	gMotor GMT v2.52	9 KB	9 KB	

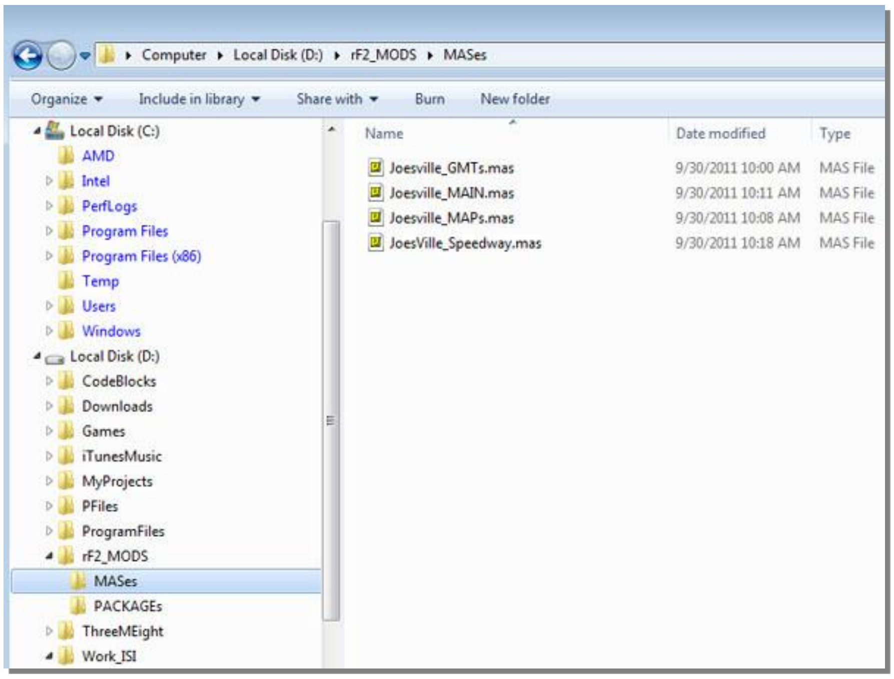
Working Dir: D:\Work_ISI\rFTools\ 111 Files

Click file/save as, browse to a location where you want to save the .mas file (it's best to put all the mas files for a component in the same location) and give it a name. The ISI standard is *trackname_type*.mas. So for this you'd call it "Joesville_GMTs.mas". You can see here that I've done this and saved to my "rF2_MODs\MASes" directory:



Name	Date modified	Type	Size
Joesville_GMTs.mas	9/30/2011 10:00 AM	MAS File	6,863 KB

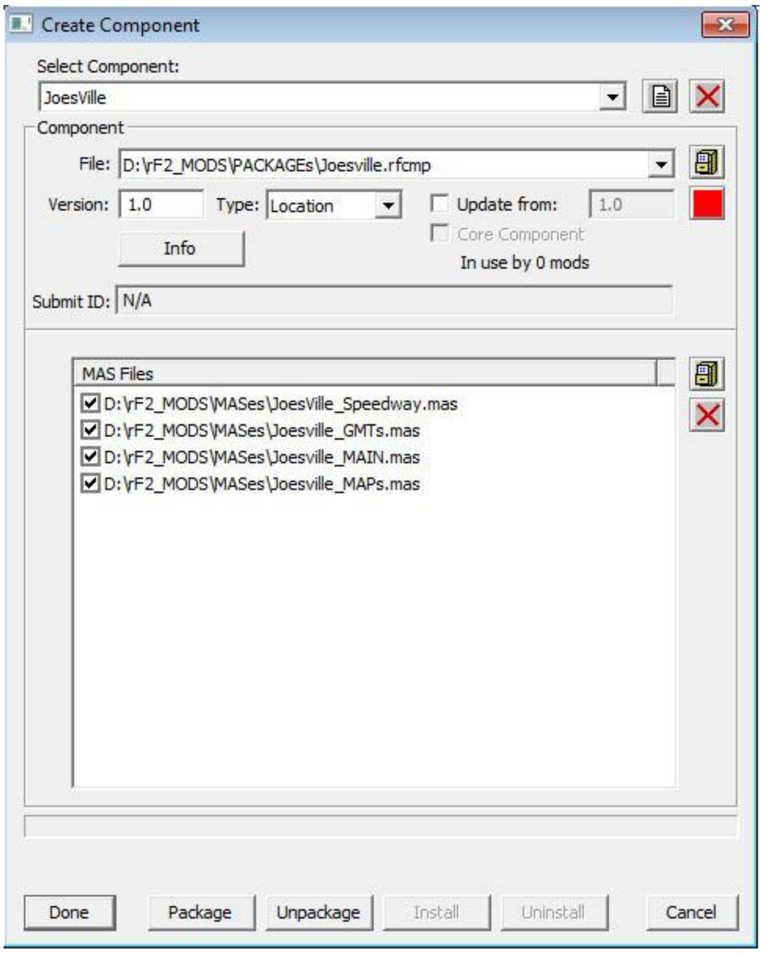
After it's saved click the "Create a new MAS file" button ("piece of paper" icon) —the list is cleared and ready for new files. So, using the above instructions go ahead and create a .mas for the assets in the "Maps" folder. You'll also need two more mas files--"MAIN" and "Layout". The "Main" mas file holds the primary track files, usually only the track's .tdf and ".dds" (UI icon), which are located in the main track folder (you'll find Joesville's in "Locations\Joesville". Drag those over into an empty MAS2 list and create a "Joesville_MAIN.mas" file. The "Layout" mas file holds all the specific info for each layout of a track. Files from the "layout" mas are loaded first, so special .gmts, maps, etc., that replace or add to files in the other mas files are located here along with the layout's usual files (.gdb, .cam, .scn, etc.). So, look in "Locations\Joesville\JoesVille_Speedway" and drag all the files into MAS2 and create a "JoesVille_Speedway.mas" file. If you've done everything correctly, you should have something like this:



Name	Date modified	Type
Joesville_GMTs.mas	9/30/2011 10:00 AM	MAS File
Joesville_MAIN.mas	9/30/2011 10:11 AM	MAS File
Joesville_MAPs.mas	9/30/2011 10:08 AM	MAS File
JoesVille_Speedway.mas	9/30/2011 10:18 AM	MAS File

If you do then you're ready to make your first rF2 component. In MAS2 you'll find a button with a "C" on it—the Component packager. Click this button and a new window opens up. Click the "Select Component:" drop-down and select "Add New Component". Now, click the button next to the "file:" drop-down and choose a location for the new component file and give it a name. This component will be version 1.0, so keep the version number the same (although if you were still in development you could assign a 0.1, etc. version number to keep track of development iterations). Don't worry about the "Update From" checkbox—we'll discuss updates a bit later. The "Info" button will allow you to add information to the component including author, date created and description.

Now, you need to select the type of component this is—in our case a location. So, click the "Type:" drop-down and select "Location". Take a moment and look at some of the other types—the most commonly used will be location, vehicle, sounds, and talent. Finally, click the "add files" button next to the "MAS Files" list. Navigate to the location where you saved the mas files for Joesville. You can add them one at a time, or group-select them all at once. If you add a file you don't need you can remove it from the list by selecting the file from the list and clicking the "X" button. If you've set up correctly you should have something like this:


Create Component

Select Component: JoesVille

Component

File: D:\rF2_MODS\PACKAGEs\Joesville.rfcmp

Version: 1.0 Type: Location ☐ Update from: 1.0

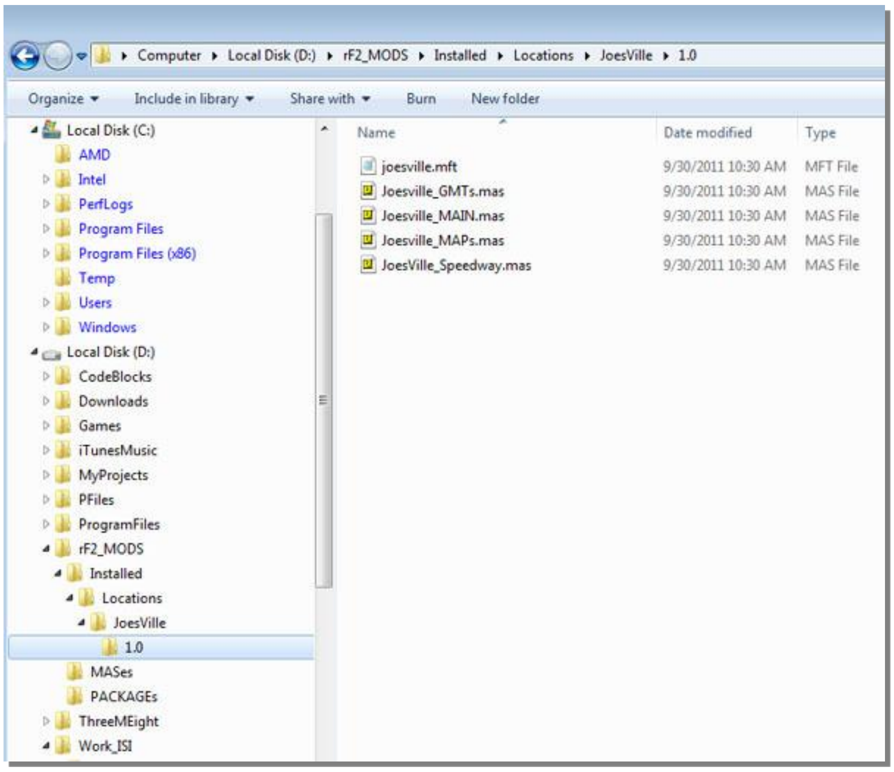
☐ Core Component

In use by 0 mods

Submit ID: N/A

MAS Files
- ☑ D:\rF2_MODS\MASes\JoesVille_Speedway.mas
- ☑ D:\rF2_MODS\MASes\Joesville_GMTs.mas
- ☑ D:\rF2_MODS\MASes\Joesville_MAIN.mas
- ☑ D:\rF2_MODS\MASes\Joesville_MAPs.mas

Now, look to the bottom of the Component window and click the "Package" button and the new component will be built. After that, you need to "install" the component so that it can be used in a MOD file, so click the "Install" button. Once installed, in the directory where your MAS2 utility is you'll find an "installed" sub-directory, with a "locations" folder and a new "Joesville" track folder within that.



Name	Date modified	Type
joesville.mft	9/30/2011 10:30 AM	MFT File
Joesville_GMTs.mas	9/30/2011 10:30 AM	MAS File
Joesville_MAIN.mas	9/30/2011 10:30 AM	MAS File
Joesville_MAPs.mas	9/30/2011 10:30 AM	MAS File
JoesVille_Speedway.mas	9/30/2011 10:30 AM	MAS File

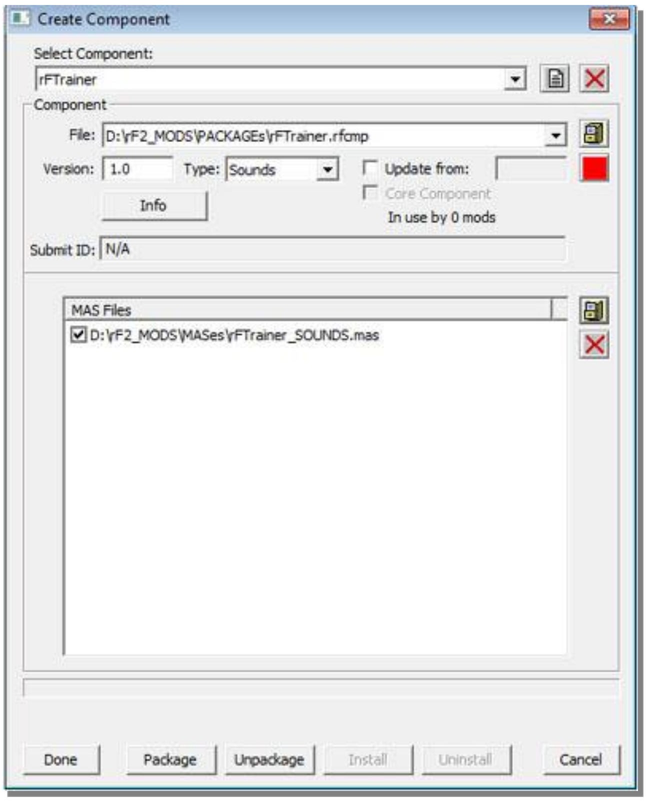
Don't be daunted by this procedure. It seems a bit tedious at first, but once you've done it a few times it becomes second nature and takes no time at all.

Before we continue, and as promised, let's talk about component versions and updates. If we had been making a Joesville version 1.1 component (or 1.2, 1.3, etc) we would have typed in a new version number, and we'd need to decide if this is an all-new version of the component, or an update that relies on previous data. If it is an all-new version you would have updated all the original mas files with the new data, updated the version number, and kept the “Update from:” checkbox **UNCHECKED**. This tells rF2 that the new version does not rely on any of the data located in the original mas files. And, if no installed MOD relies on the old component version it will actually be removed from rF2 in favor of the new. You would use this type of update if so much of the original data had changed there'd be no point in re-using any old data.

However, if you only change one or two files then you'd want to create a true update—this helps keep download sizes to a minimum. To do this you'd **CHECK** the “Update from:” checkbox and then enter the version number of the version you're upgrading from (note: the previous version MUST be installed). Now, add the mas files that contain the updated files. Let's say you updated all of the track maps—you'd make a new Maps.mas file and just add that to the “MAS Files” list. You could also just add all the changed files, be they gmts, maps, or whatever, to a “*trackname*_updateV#.mas” file and add just that. Files in a component update take priority over the originals. There is only one restriction to this—if any files change that would be located in a “Layout” mas then the mas file MUST be recreated with the new data. You CAN NOT update single files that are specific to a layout…

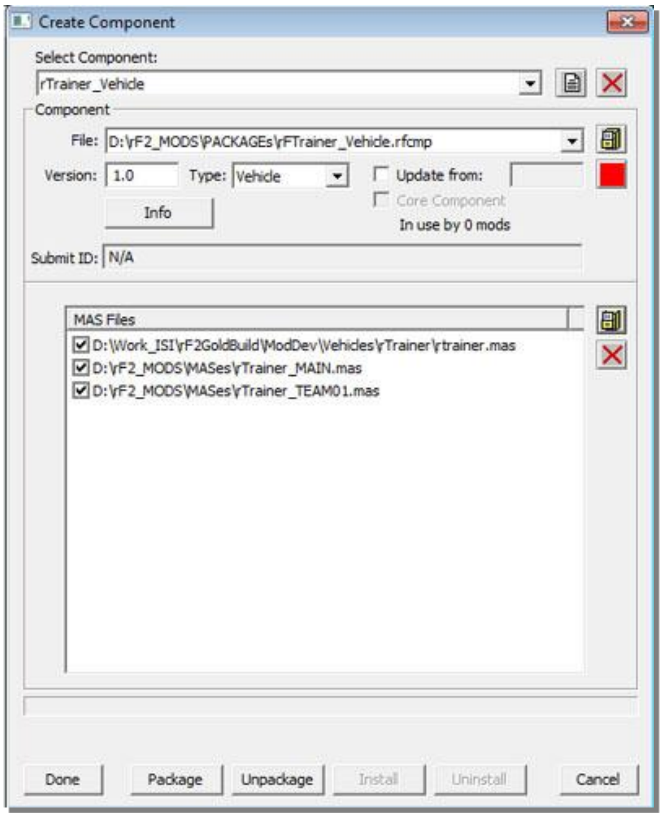
Now, let's make the sound component our rTrainer MOD. This is easier as you only need one mas file in which you'll include all the sounds the rTrainer needs. So, navigate to the moddev/sounds/rFtrainer folder, and make a new mas file—you could name it something like “rFtrainer_SOUNDS.mas”. When finished, open the Component Packager, add a new component, and name it. **NOTE: The name of the sound component MUST be the same as the folder name the**

sounds were located in! Again, this will be version 1.0, set "Sounds" as the type, add the mas file, package and install it. You should see something like this:


Create Component
Select Component: rFTrainer
File: D:\rF2_MODS\PACKAGEs\rFTrainer.rfcmp
Version: 1.0 Type: Sounds
Update from:
Core Component
In use by 0 mods
Submit ID: N/A
MAS Files
☑ D:\rF2_MODS\MASes\rFTrainer_SOUNDS.mas

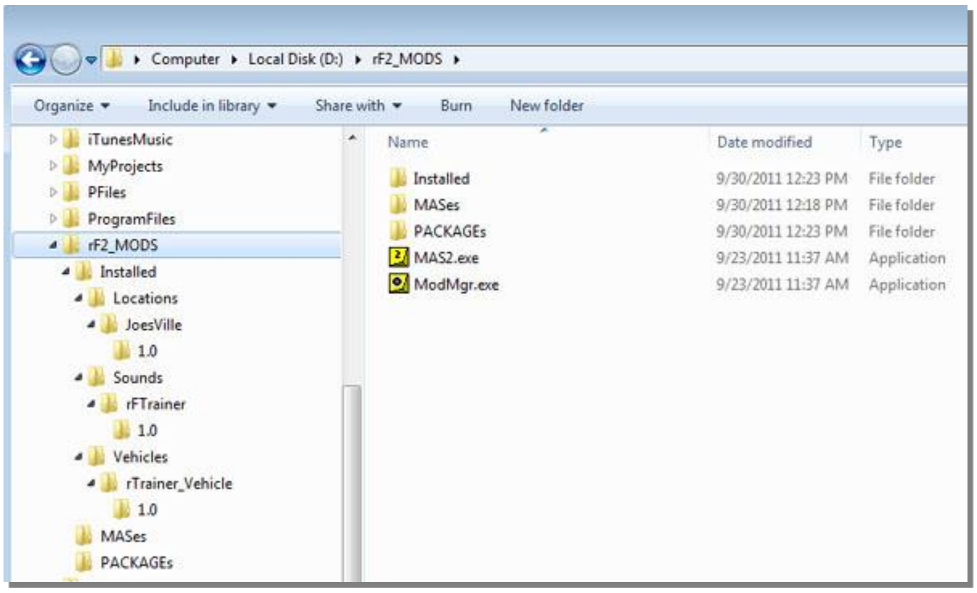
Go ahead and package, then install, the component.

Finally, we'll make our vehicle component. Basically, you'll follow the same principles for this as you did for the track component. You'll have a "_MAIN" mas file for the core elements of the vehicles (phsyics, sound FX files, etc), as well as "_TEAM*name*" mas files for each of the team folders (alternate team skins also go in the Team mas). Team mas files act just like a track's Layout mas files. And, finally, you'll have one or more mas files for the maps and gmts that the vehicle relies on. So, go ahead and create those mas files now, and prep the component. When done you should have something close to the image shown on the right:


Create Component
Select Component: rTrainer_Vehicle
File: D:\rF2_MODS\PACKAGEs\rFTrainer_Vehicle.rfcmp
Version: 1.0 Type: Vehicle
Update from:
Core Component
In use by 0 mods
Submit ID: N/A
MAS Files
☑ D:\Work_ISI\rF2GoldBuild\ModDev\Vehicles\rTrainer\rtrainer.mas
☑ D:\rF2_MODS\MASes\rTrainer_MAIN.mas
☑ D:\rF2_MODS\MASes\rTrainer_TEAM01.mas

Take notice to the rTrainer.mas file—it was already located in “Vehicles\rTrainer”. Mas files **CAN NOT** be included within other mas files, so this had to be listed separately. In this case it contains all the vehicle art. Again, go ahead and package/install the component.

If all has gone well you should now have 3 components “installed” in your working directory, like this:



Name	Date modified	Type
Installed	9/30/2011 12:23 PM	File folder
MASes	9/30/2011 12:18 PM	File folder
PACKAGEs	9/30/2011 12:23 PM	File folder
MAS2.exe	9/23/2011 11:37 AM	Application
ModMgr.exe	9/23/2011 11:37 AM	Application

At this point, you could also make a component for talent files, or even for a new UI. We could release these components now in stand-alone form for other people to create their own MODs with, or use them separately. Most vehicles and tracks will be released this way. However, you could go a step further and release a .rfmod package. This is what we’re going to do with our rTrainer components we’ve just made. But, before we begin, we have one final mas file to make.

This mas will contain the three files for the MOD that can be found in the “modDev/RFM” folder. These are the series’ .rfm, as well as the two icon.dds files for the UI. For our rTrainer MOD you’ll want the ones named “TestingMOD…”. Add them to MAS2 and create a “rTrainer_RFM.mas” file. Although the vehicle filters won't be used for the mod, the series rules are still taken from the .rfm, as well as the series’ name. You don’t need to create a component for this—it’s strictly for use with the “Packager”, which we talk about next.

Creating a MOD Package

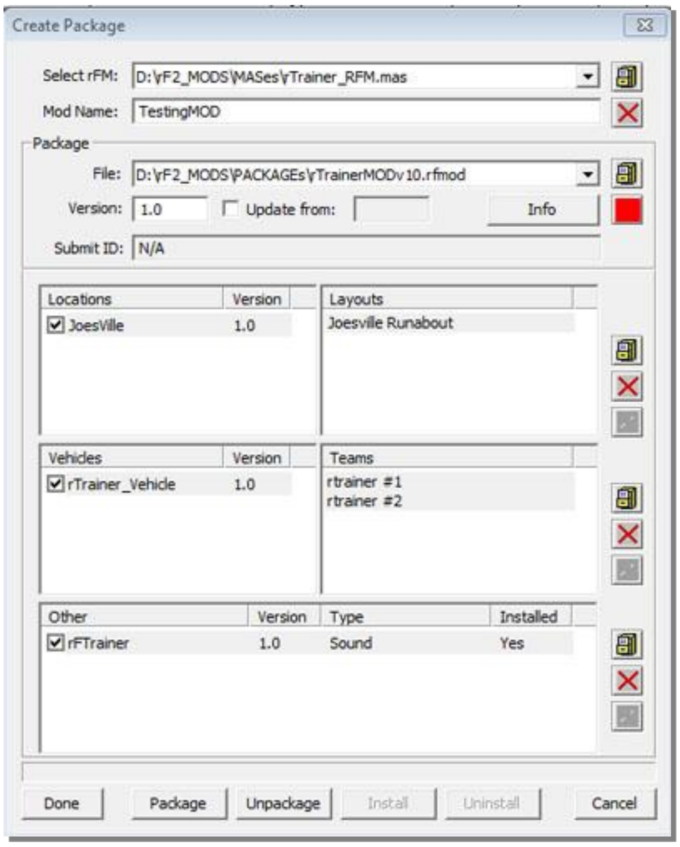
Now that we have our rfm .mas file, as well as all the required components packaged and installed, it’s time to create our full blown MOD package. In MAS2, click the “P” button (“Packager”). The first thing we need to do is click the “Add file” button located next to the “Select RFM” drop-down, and browse to the rfm.mas you just made. Again, the “Mod Name” is taken from the name entered in the .rfm. Although you can, it’s best not to change the name. If you want something different you should alter it in the .rfm.

Next create a file name for the Package file we're going to create. We’re making a 1.0 version, but again you could start with a lower sequence for development purposes (and again, we'll go over MOD updates in a minute). As with components, you can add additional information by clicking “Info”, and in fact you have to enter at least one bit of info—the ModID.

The ModID can be requested using the rF2 Launcher. Go to the “Purchase” page and select “Request New Mod ID”. Here, you'll enter the mod information and click “Submit”. Assuming everything is entered correctly you'll be given a unique ID number that you enter in the Information page. We don’t have to do that for our test MOD, but for any MOD that you want to release to the public and have it show up on the ISI matchmaker you MUST do this. That said, after we finish creating our .rfmod file we'll discuss what to do with the “Submit ID” number.

Continuing on, look a bit lower and you'll see sections for Locations, Vehicles and Other components, as well as buttons located next to each to add components. So, click the “Add Component” button located next to the “Locations” section and add the Joesville component. Then do the same for our rTrainer vehicle. Sound, Talent, and other components are added in the “Others” section, so go there and add our rTrainer sound component. All the added components will have been checked—keep them that way. This means we actually want to package these .rfComp files in our .rfMod file.

Now, for each component you need to select which elements from each you want to use. For example, if we made a track with 4 layouts we could decide to only use one, two, or all of the layouts. For Joesville, it only has one layout so select that. From our vehicle component we want to use all of the teams available, so Ctrl-click all of the listed teams so they all get packaged up. The sound component is what it is, so no other action is required. If everything is as expected, you should see something similar to the image on the right:


Create Package

Select rFM: D:\rF2_MODS\MASes\rTrainer_RFM.mas
Mod Name: TestingMOD
Package
File: D:\rF2_MODS\PACKAGEs\rTrainerMODv10.rfmod
Version: 1.0 ☐ Update from: Info
Submit ID: N/A

Locations	Version	Layouts
☑ JoesVille	1.0	Joesville Runabout

Vehicles	Version	Teams
☑ rTrainer_Vehicle	1.0	rtrainer #1 rtrainer #2

Other	Version	Type	Installed
☑ rFTrainer	1.0	Sound	Yes

Click “Package”. If there are any obvious errors the packager will let you know about it, but hopefully all is well and the .rfMod was created successfully. You don't NEED to install the package into your working location, but if you want to update it later, and have your Submit ID displayed, you'll HAVE to install it, so go ahead and click the “Install” button. A LONG number will be displayed in the “Submit ID” field (you may need to click “Done”, and then re-open the Packager to see it).

If this were a MOD you were going to release you would highlight and copy this number. You’d then go back to the rF2 Launcher, click “Purchase”, and “Request Mod ID”. Now though, you're going to look lower and, in the Submit” area, enter the Mod ID you were originally given, and paste the “Submit ID” number into the Submit ID field, and then click “Submit”. Now, an explanation: A unique Mod ID is requested and given to each MOD that is created, and the Submit ID will be used to check the MOD during multi-player. If either of the data is incorrect a user will not be allowed to join a server running the mod. This helps to prevent both content mismatches and content cheating as any change to any of the data will alter the SubmitID. Each component also has ID that they are checked against for an additional level of security.

Note: Again, although you're not required to get a Mod ID, or submit a “Submit ID”, your MOD will not be displayed in ISI's matchmaker without it.

That’s it—you’re all done. Now, the created .rfMod file can be added to your rF2 install's “Package” folder, where you can either install the MOD from in-game, or go to the rF2Launcer, click the “Manage Mods” button, and install from there.

So yes, while you as the mod maker has had to do a bit more organized work to get your creation ready for the public, you can be assured that your MOD is easily downloaded and installed, used in the manner it was intended, and will be free of mismatches for both clients and servers. Is it a bit more restrictive and tedious? Perhaps a little, yes. But, in the long run we feel it is a FAR better system than the loose file free-for-all that made online game play so problematic in rF1. And, with the possibility of update mods, both version updates as well as specialized updates for league play are easily possible. So, for our final subject, we will discuss updating your currently existing MOD.

As with a component, you can either do a full-content update, or just update bits and pieces of your MOD. The simplest, and largest, type is a full update. This is where every component of a MOD has changed and your new version uses none of the original elements. For this, you'd make updated components; make sure the MOD name hasn't changed (ie. Make sure the series name hasn't been altered in the .rfm file), update the MOD version number and keep “Update From:” **unchecked**. Add the updated components, make sure they're all checked and all the elements of each component selected. Click “Package” and install and you're ready to go.

Note 1: Make sure you use the original ModID for the mod. When you create the update you’ll be given a new SubmitID after you install the mod update. Using the same procedure as before, submit the new ID to ISI. The only difference this time is that you’ll enter your new version number. We track the Signature ID of each version so that new versions retain the same ModID but do not invalidate older versions of the mod.

Note 2: The ONLY person that can update a Mod is the person that first requested the ModID for the original Mod.

The next type is a partial update. This is where the new MOD version uses both new and original elements. Here, you'd enter a new version number, check “Update From:”, and enter the version number of the previous version. Again, be sure to use the original ModID for the new version and enter that in the info box. Now, add the original components you need, but make sure they are unchecked. This will tell the packager to reference the original elements for use, but not include them in the package, which makes the download size smaller. After you've done this, add the updated components you need and keep these checked so that they are included with the package file. Do NOT include both the original AND updated version of the same component. If you want to use an updated version of a component just select it alone and make sure it is checked. Again, make sure you select all the elements required from each component and click “Package”. “Install” it, and again send the new “SubmitID” to ISI using the new version number.

There is another type of MOD that we call a “Virtual” MOD. This is a MOD that does not include ANY component data—it only references already created and installed components. To create this MOD you need to have all of the components it requires already installed in the “Installed” directories in your working area, and anyone who uses this mod MUST have the required components already installed in rF2, or else the Virtual Mod **WILL NOT** be installed. To make a mod of this type, simply create the files required for the RFM mas, make that mas file, and build the MOD from that. Select the components required and uncheck them all. This type of MOD would be useful if a group of friends want to race a certain car on certain tracks, or if a league intends to run a series using cars and tracks of various MODs. Of course, you can include all the components so that you were sure everyone had the proper data, but this is an option if you know everyone will already have everything they need. Again, a ModID and SubmitID will be required for the MOD to be listed on ISI's matchmaker.

A couple last things to note here: if a MOD contains a component that a player already has installed a second version of the component **WILL NOT** be installed. Of course there’s no reason to have the same component installed twice. Also, if a player doesn’t have a component required by a virtual mod the mod can not be installed, and the player will be shown in the mod manager which component(s) is missing.

And, this is MOD packaging in a rather large nutshell. There are special-case situations we are still discovering and figuring out how to handle, and of course there may be specialized situations we need to add support for in the future. We fully intend to continue updating the Package system to increase its functionality. In its current iteration though, our new system should handle 90% of all the needs of mod makers, leagues and casual players. Again, we understand that many people have an adversity to change and may resist the new system at first. Our goal though is to get as many drivers into online race situations as easily as possible with no fuss or headaches, and KEEP THEM THERE rather than getting booted because they may be missing a car, track, or the odd updated file. We want people spending their valuable time racing, not missing race starts while they endlessly search the internet for a specific version of a car or track just so they can join a server. We also want modders to be able to start sharing their work more openly without worrying about who created what. Our goal is to have rF2 an open and friendly environment from which to create, while keeping the system clean and straight-forward for the many sim-racers out there…