ABCREATOR TUTORIAL

Trademarks

Contact Information

Lone Wolf Development, Inc.
Web: www.wolflair.com
Email: support@wolflair.com
Forum: www.yahoogroups.com/group/armybuilder

Table of Contents

OVERVIEW

What is ABCreator?

ABCreator is the primary tool used to edit data files for Army Builder. This tool allows you to create and maintain data files for any game system for which you have an appropriate definition file. Full documentation for ABCreator can be found in the Construction Kit manual, which is installed as part of the Construction Kit and be found in the file “constkit.rtf”. The reader is assumed to have read the manual prior to using this tutorial, as many references in this document are made to the manual.

A Database Tool

For all intents and purposes, ABCreator is a simple database editing tool, since each Army Builder data file is its own database of units, races, etc. The role of ABCreator is to define all of the races, units, options, items, elements, links, and augmentations that will be used by Army Builder at run-time. Each of these components is treated just like a database record, having a number of fields which contain the information describing each specific component. Consequently, to keep the nomenclature simple and consistent, these components are called “records”.

When you use ABCreator, you are either creating, editing, or deleting records that are stored in a data file. Just like with many databases, every record in the data file must have a unique identifier. All of the remaining fields of each record can be duplicated in whatever way you wish. In addition, certain types of records within a data file may be connected to other records, just like records in a database can reference other records. It is the unique identifier field which is referenced by connections between records.

The single most important thing to remember when you first start to create an Army Builder data file is that all you are doing is creating records that will be shown by Army Builder. If you keep this in mind and start out with a simple data file, you will quickly become comfortable creating your own data files. After you get a little bit of practice, you can then start to explore the power of Army Builder and introduce more sophisticated behaviors into your data files.

Importance of the Definition File

The core of everything is the definition file, which is always named “datadef.*” (the file extension corresponds to the game system). A definition file must exist for every game system, and all data files for a game system make explicit references to the definition file. This tutorial does not attempt to explain how to create a definition file, yet it assumes that you are familiar with the contents of a definition file. More importantly, you are assumed to be familiar with the specific contents of the definition file for which you will be creating a data file. This understanding is critical, since many of the fields of various records will need to make explicit reference to the names and abbreviations specified within the definition file. For example, all of the composition groups to which units can be assigned are defined within a definition file. You must be familiar with what these groups are and how they are intended to be utilized, because every unit must be assigned to an appropriate composition group. This tutorial does not require that you create or even edit a definition file, but you do need to know the contents of the definition file that will be referenced by your data files.

Important Reminders

This tutorial is NOT a replacement for the Construction Kit manual. Instead, it is intended to be used AFTER you have read through that manual and have a basic understanding of the overall structures and mechanisms utilized by Army Builder. This tutorial only serves to introduce the user to the creation of a few simple records of various types. If this tutorial achieves its goal, then once it is completed, you should be sufficiently comfortable with using ABCreator that you can switch your focus to the complexities of attributes, types, and numerous other sophisticated constructs that are available within Army Builder.

Tutorial Objective

The primary objective for this collection of tutorials is to walk you through the creation of a new data file, wherein you define one or more new races, units, options, and items. Once this is accomplished, you should be able to use Army Builder to create an army of this new race, add your new units to the roster, select the new options for the unit, and give the units the new items you have defined. Additional sections are included which describe some of the more commonly used mechanisms within Army Builder. It is hoped that any user who successfully completes all of these sections will be comfortable and confident in embarking upon the creation of a new data file for any suitable game system.

REVIEW THE DEFINITION FILE

Getting Started

Before you begin to edit a data file, it is very important to be familiar with the definition file for the appropriate game system. However, if you intend to create a new data file, it is critical. For the tutorial, we will be working with the aptly named tutorial data and definition files. These files will be found in the Army Builder “data” directory with the names “data.tut” and “datadef.tut”, respectively. Both of these files are run-time files, so they are unreadable to anything except Army Builder. Since we are interested in the definition file at this point, we need to convert the run-time file into a readable text format. This is accomplished through the use of the “ABUndef” tool, which is installed as part of the Construction Kit. To convert the file, follow these steps:

- Open a Console Window. This is most easily accomplished by clicking on the “Start” menu in the lower left corner, then going to “Programs” and clicking on the “MS-DOS Prompt” menu item. This will open up a new window with the (oft-dreaded) command-line interface. (Note: If you use Windows NT, then the menu item will be labelled “Command Prompt”).
- Change your current directory to the directory where you installed Army Builder. For example, if you installed Army Builder to the default directory that is proposed, the command you would type is:

```
cd c:\armybuilder<Enter>
```

- Within the Army Builder directory is a subdirectory called “data”. It is within this “data” subdirectory that all of the data files used by Army Builder are placed. To see the list of these data files, you can enter the command:

```
dir data<Enter>
```

- Invoke the “ABUndef” utility without any parameters, as shown below. An error should be reported that indicates the wrong number of parameters were specified, plus the proper syntax for the tool is shown.

```
abundef<Enter>
```

- We can now follow the syntax shown to generate a text version of the definition file. Let’s convert the definition file to a text file named “tutor.txt”, which we will save in the “data” directory with the run-time definition file. To accomplish this conversion, enter the command below:

```
abundef data\datadef.tut data\tutor.txt
```

- If the command is successful, then no errors should be reported. You can look for the created “tutor.txt” file by issuing the command:

```
dir data\tutor.txt<Enter>
```

- Now that the text file has been created, we are finished with the Console Window and can close it. This is accomplished by entering the “exit” command, as shown below. When this is done, the Console Window will disappear.

```
exit<Enter>
```

- We now need to view the contents of the text version of the definition file that we just created. This is easily done with Notepad, although you are welcome to use whatever text editor you prefer for this purpose. Launch the appropriate program (e.g. Notepad) and then open up the newly created file. This file will be found in the path “c:\armybuilder\data\tutor.txt” (assuming that all of the defaults were used).

General Settings

Now that the definition file is open for viewing, we need to analyze each of the sections to understand the portions that are important to us. The first section of the definition file, the “General” section, lists nothing that is pertinent to a data file creator. The fields in this section only specify information that is used by Army Builder to properly format and annotate information that is presented to the user.

Stats

The second section of the definition file specifies all of the different “stats” that are involved in the game system. The names of these stats is important, as well as a few additional fields, namely the “minimum” and “maximum” stat values, which appear in the 4th and 5th columns. These fields indicate the minimum and maximum values that will be displayed for each of the stats. In some cases, an “=” will also be shown, indicating that any stat value that meets or exceeds the specified minimum/maximum will instead be displayed with the given text. For example, if the “minimum” column is given as “0=-”, then any stat values of 0 or less will actually be displayed as “-”. This is important, since some units will need to take advantage of these special boundary values.

Composition Groups

The next section lists all of the composition groups that exist for the game system. Each composition group is given a name and an abbreviation, and both are important. They are important because every unit we create must be assigned an appropriate composition group. In addition, the definition of a race must specify which composition groups are valid for that race, along with their relative sizes within a created army roster.

Unit Categories

Next is the list of unit categories. All units must be assigned a category, so the names of these categories is important to us. Also take note of the abbreviations defined in the second column, as these are referenced by the special item categories, which are described in a moment.

Option Categories

After the unit categories come the option categories. You should review this list briefly, but don’t worry about memorizing anything in it. Whenever you create a new option, you will need to assign it a category, but ABCreator always shows you this list of category names, so there is no need to remember them.

Conflict Groups

The sixth section of the definition file is the list of conflict groups. This section can become relatively complex and is described pretty thoroughly within the Construction Kit manual. When creating serious data files, it is critical that you understand what conflict groups are and how they work. For your purposes as a data file creator, you will need to pay close attention to the first two columns. These columns identify the abbreviations used for each conflict group and the maximum number of options and/or items from the conflict group that may be selected at the same time for a unit. For example, the “Arm” group reflects the number of suits of armor that may be assigned to a unit. Obviously, the limit should be one.

Tweak Categories

The next section contains the list of all tweak categories. Tweaks are very similar to options, except that tweaks are essentially “options for items”, so a different set of categories is used. Whenever you create a tweak, you will need to assign it to one of these categories, which you’ll select from a list presented by ABCreator. Since the tutorial data files don’t make use of tweaks, there are no categories defined.

Stats Sets

Following tweaks categories are the definition of any stat sets used by units in the data files. If units will need to display custom stat lines, each set of custom stats must be defined in this section. This section is empty in the tutorial data files, since no stat sets are utilized.

Special Item Categories

At the very end of the definition file is the list of special item categories. As a data file creator, all of the information in this section could be of interest, depending on the level of sophistication you wish to achieve within your data files. However, for relatively non-complex data files, a cursory review of the category names is sufficient. If you delve further into the nuances of data file construction, then you will want to review the documentation on these special item categories in detail and pay close attention to their definitions.

Special Considerations

Although it is not required, you might find it convenient to leave this definition file open within the text editor as you begin to create your first data file. There may be times that you wish to make references to the data file or simply check on certain values within the file. Having it open in another window can make this very easy to accomplish, as you can quickly switch between the applications and check on values while you work on the data files.

CREATING A NEW RACE

Objective

Our goal in this first tutorial is to create a new race. The basic tutorial data file, “data.tut”, already contains four races (“Elves”, “Stunties”, “Tree Huggers”, and “Xamples”). We will call our new race the “Misfits”. This race will be comprised of three composition groups: Heroes, Troops, and Allies. The Hero group will be restricted to no more than 25% of the total points of the army, the Troops group must be at least 50% of the army, and the Allies group must be less than or equal to the Hero group. Due to their nature, our Misfits don’t work well with other races, although they can sometimes convince other races to join them. To account for this, whenever Allies are added to a Misfits roster, the cost of those allies is always 10% more than the normal cost.

Create the Data File

Before we can do anything else, we first need to create a new data file in which we will place all of our new records. While we could edit the existing “data.tut” file, this would probably be a bad idea. First of all, this makes for a large data file, and smaller files are much easier to edit (as you will discover once you start working with ABCreator). Secondly, if someone else had written and given you the “data.tut” file, then anytime they release a new update to this file, you will have to re-enter all of your changes again. By using a separate data file, your changes should almost always work with any new updates to other files (the one exception being any references you make to those files from within yours, but these are quick and simple to correct). Creating your new data file is accomplished via the following steps:

- Launch ABCreator. This typically requires clicking on the “Start” menu in the lower left corner, then selecting the “Army Builder” group, and finally clicking on the “ABCreator” menu item.
- Click on the “File” menu at the top of ABCreator, then select the “New” menu item.
- When the dialog is displayed to select the desired game system, be sure to choose the “Tutorial Game System” with the “TUT” file extension. Single-click on the proper selection and then click on “OK”.
- An empty data file now exists in memory on your computer. We will begin to create the contents of this data file in the next section.

View the Definition File Summary

All of your data files will constantly make reference to the contents of the definition file. If you forget the abbreviation you assigned to something in your definition file, then you’ll need to look it up while you’re creating your data files. To make life easy on you, we’ve included a “cheat sheet” summary of the contents of your definition file within ABCreator. To access this summary, select the “View” menu and click on the “Definition Info” menu item. A dialog will appear that lists all the pertinent information from the definition file that you might want to reference while creating your data files. To return to editing your data files, click on the “Done” button at the bottom.

Create the Race

Creating a new record of any type always follows the same sequence of steps. When the proper record type is selected in the tabs across the top, we can create a new record of that type with the “New” command.

- Across the top of the ABCreator window, you will see nine tabs. The first of these tabs will be labeled “Race”. Click on this tab to be certain that you are manipulating races.
- In the lower left corner of the ABCreator window, there should be four buttons. One of these buttons should be labeled “New”. To create a new race record, simply click on the “New” button.
- This creates an empty race record that you can enter your information into. In the list of “Existing Races” on the left, you will see a new record. The new record is shown with a name of “??????” and an abbreviation of “xx”. On the right, you will see that all of the appropriate race record fields have been properly initialized, with the name of the race and prefix fields both being initialized to the same values.

- When the new race is created, the first field, labeled "Name of Race", is automatically selected so that you can begin to enter the contents of the record. Consequently, the "??????" is highlighted and will be replaced when you start entering the name of the new race.
- Look down at the bottom of the ABCreator window. You should see a darker grey bar across the bottom. Within this bar is help text that describes what form of input is being sought for the currently selected field. Since the "Name of Race" field is now selected, the help text initially states "Name of the race or army being defined". If you are ever unsure what information belongs in a particular field, first glance at this help text, since it will often provide enough of a reminder for you to enter the correct information.

Specify the Basics

The first step in creating your race is to enter the basic information for the race. This consists of all of the fields, except for the "Composition Group Rules" and the "Special Attributes". The following steps walk you through the specification of all these fields.

- The very first field for a race is its name. Since we decided that we were going to name our race "Misfits", make sure that the first field is selected and then enter the name "Misfits".
- Switch to the next field by pressing the <Tab> key. If you would prefer, you can also click within the field you desire, although it is often quickest to simply press the <Tab> key to move to the next field for the record.
- The "Prefix" field should now be selected. This field specifies the two-character prefix that will be used to uniquely identify the new race and all units which belong to the race. Let's use the prefix "mf", so enter those letters into the field and switch to the next field.
- The third field is represented by a checkbox and is labeled "Is Shared?". This checkbox indicates whether the units for this new race should constitute their own race that the user can select or a shared race that is made available to all other races. For example, many fantasy systems have monsters that can be used by all of the different races. Such a race would have this checkbox checked to enable this sharing. For our purposes, though, we are creating an independent race, so this checkbox should be left unchecked.
- Press the <Tab> key to move to the fourth field, labeled "Allied Races". This field will specify the list of all other races that can be chosen as allies for our new race. As described at the bottom in the help text, the list must consist of the two-letter race prefixes, each separated by a comma. Let's allow our Misfits to choose both "Elves" and "Tree Huggers" as allies. The race prefix for "Elves" is "el" and the prefix for "Tree Huggers" is "th" (these are defined in the "data.tut" file). This means that our entry for this field should be "el,th". Enter this text into the field and press <Tab> to select the fifth field.
- The next field allows us to determine which composition group, if any, will be used to collect all of the points for "Allied" units. If our race was not allowed to select any allies, then the "*None*" option would be the proper selection. However, since our race does have allies, we need to choose the "Allies" composition group, which is defined in the definition file for this very purpose.
- Skip over the "Composition Group Rules" field for a moment and proceed to the "Special Item Groups" field. This field allows you to specify any special item groups that are normally not allowed to any races and should be made available to this new race as an exception. For example, in Warhammer Fantasy, the Dwarves have special runic items that are only available to Dwarves. This field is where those special groups would be identified. Our race of Misfits has no special categories, though, so this field should be left blank.

Composition Rules

The composition rules can become a somewhat complex problem. They really serve two independent roles, so it is probably best to address these roles as two separate tasks. The first function they serve is to identify which of the composition groups are utilized by the race you are creating. For the tutorial game system, there are only four of these groups to select from, but some game systems may have a dozen or more to worry about. The second task involved is to decide what the relative compositions of the selected groups are allowed to be. For example, if the Hero group cannot be any more than 30% of the total roster, then this would dictate the rule that must be defined. However, if the Hero group is required to always consist of fewer points than the regular Troops group, then this would dictate a different rule.

Suggestion: You may want to review the corresponding section of the Construction Kit manual before proceeding, as the syntax for this field is a bit complex.

- Select the "Composition Group Rules" field.
- The format of this field is a comma-separated list of the composition groups assigned to the race and the size constraint rules for each. Our first group is the "Hero" group, with an abbreviation of "hero", and its valid size range is up to 25% of the total army. So, enter the first group and rule as "hero:%max=25".
- Our second group is defined above to be the "Troops" group. This group has an abbreviation of "trps", and its valid size must be at least 50% of the total army. Since we need a comma between each composition group rule, the text we need to enter is ",trps:%min=50".
- The third composition group for our race is the Allies. This group must maintain a relationship with one of the other composition groups to remain valid. The result is that we have to define the rule for our third group a bit differently. Append the following text to the current contents of the field: ",ally:prelp<=hero*1".
- The completed contents of the composition group rules field should be as shown below. If there are any differences, please correct them before continuing:

hero:%max=25,trps:%min=50,ally:prelp<=hero*1

Attributes

Adding attributes for the race is the next step. Attributes for all record types are edited in the same way, so they all use the same interface described here. The only difference is that each type of record has a different set of attributes that can be assigned.

- Locate the "Special Attributes" region on the display.
- Since our current list of attributes is empty, our first objective is to add a new attribute to our race. To begin this process, click on the "New" button within the "Special Attributes" region.
- A new dialog appears in which all of the valid race attributes are listed, along with a description of the syntax for those attributes. The attribute that we want to use is, conveniently, the first attribute listed: "acst". Click on the "acst" attribute in the list on the left. Within the text region on the upper right, you will see the proper syntax and description for the highlighted attribute.
- Click within the empty edit region on the lower right of the dialog. This is where we will enter our attribute.
- For our new race, the cost of allies is 10% more than the normal price. According to the description of the attribute, the actual cost is multiplied by the value that we specify. In order to increase the cost by 10%, we need to multiply the cost by the value 1.1. Therefore, the attribute must be entered into the edit region with the format "acst:1.1".
- Verify that you have entered the text properly, then click on the "OK" button to accept the new attribute for the race.
- The attribute editing dialog disappears, and you now see the new attribute listed in the "Special Attributes" region. If you wanted to edit this attribute, you could click on it and then click on the "Edit" button to the right, but there is no need to do that at this time.

Author Comments

At the bottom of the window, you will see a field entitled "Author Comments". You are free to enter anything you want into this field (or nothing at all). This field is stored with your data file and is intended as a place where you can enter special notes and reminders to yourself about this record. For example, if you specified a few attributes for the race, you might want to leave yourself a reminder about why those attributes are important. This way, if you come back to these files later on to enhance them, you won't have to try and remember why those attributes are there.

The "Author Comments" field is provided on all the major record types and you are free to use this field in any way you wish. This field is not displayed to any users of your data files, as it only appears within ABCreator to make the process of creating data files easier for you. This field always appears at the bottom of each record so that it does not get confused with other facets of the various record types.

Important! The "Author Comments" field should NOT be confused with the "Comments" field that appears for some record types. The "Author Comments" field is always intended for reminder notes by the author that are not seen by the users, while the "Comments" fields are definitely used and viewed by users.

Save the New Record

Our race is now complete. Unfortunately, it hasn't been saved yet. There are two steps involved in this process: saving our changes to the new record and then saving these changes into the new file.

- Look up at the "Race" tab across the top of the window. The name of the tab is shown as "**Race**". The extra asterisks on either side of the name indicate that the record of that type is "dirty". A dirty record simply means that changes have been made to the record which haven't been committed yet.
- Our record is dirty, so let's commit our changes. This is accomplished by clicking on the "Save" button in the lower left corner of the window. As long as the record data you have entered is valid, the changes are accepted and stored in memory, and the "dirty" indicator goes away (i.e. the label reverts to "Race").
- If a field happens to contain invalid data for some reason and you attempt to save the record, a message box is displayed that warns you of the problem. It also identifies one of the fields in which an error was found, helping you to fix the problem more readily.
- Once the record changes are confirmed, then the new race record needs to be saved in the data file on your disk. Remember, saving a record only accepts the changes in memory. You still need to actually save the file itself. To do this, click on the "File" menu and then click on the "Save" menu item.
- A standard Windows dialog appears, asking you to name the file that you want to save. By default, the directory selected is the Army Builder "data" directory, so any file you create will automatically be saved in the correct place so that it is loaded when Army Builder runs. Since the location is correct, you simply need to specify the actual filename that you want used. Unless you specify an extension of your own, the file extension of the definition file is automatically used, so you only need to give a base filename. Enter the name "misfits" and then click on the "Save" button (or press <Enter>). This will save your new race in a data file named "misfits.tut".

Verify the Race

Now that the race has been saved to the new data file, it's time to make sure that our changes are working properly. This requires that we load the data files into Army Builder and give them a try. Since the new file was automatically saved in the proper location, Army Builder will immediately find it when it is next launched. There is no need to move the file into a special location.

- Launch Army Builder.
- When the dialog appears to select the game system, select the "Tutorial Game System".
- When the "Create New Roster" dialog appears, click on the arrow to show the list of races that you can create. Our new race "Misfits" should be visible.
- Create a new "Misfits" army.
- The list of available units at the top should be empty. Along the left, there should be three filter buttons shown, which correspond to the three composition groups that you specified for the race.
- Click on the arrow in the Allies combobox in the upper right corner to show all of the valid allies for the race. The list should show both the Elves and the Tree Huggers.
- Select the "Elves" as an Allied race. The list of available units should show all of the Elf units.
- Double-click on the "Archers" unit to add it to the roster. The normal cost of the Archers is 9 points per model, as shown in the list of available units. However, when the unit is added to the roster below, a unit of 5 models is shown at a cost of 50 points (10 points each). This verifies that the race attribute (which charges an extra 10% for all allied units) is working properly.
- If everything here is correct, then your new race is created correctly. Exit Army Builder and we'll create a new unit for the Misfits.

Special Considerations

There are a few key points to remember when creating your own data files, or even editing existing files:

- When you make changes to the more complex fields of records, it is wise to save the record regularly. This is done easily by pressing <Ctrl-S>. The advantage of doing this is that the syntax of changes you make is automatically validated every time that you save the changes. By performing the validation frequently, you can quickly identify errors and correct them while the information is fresh in your mind.
- Validating your changes within Army Builder is an important part of the data file editing process. Consequently, it is strongly recommended that you test out your changes in Army Builder very frequently. In data files created by Lone Wolf Development, we often create/change only 2 or 3 records before testing out those changes within Army Builder. This process may seem a little bit slow at first, but it makes the process of finding and fixing problems must faster when they occur. Since you only changed a couple of things at a time, you will be able to tell what is wrong very quickly.

CREATING A NEW REGIMENTAL UNIT

Objective

Our goal in this tutorial is to create a unit of “Loonies”, an infantry unit for our Misfits race. Each instance of this unit must have at least five models in it, and the cost of each model is 11 points. The unit will not be allowed to select any special items and it will also not be given any options at this time. Later, we will return and give this unit an option that it can select. Our Misfits unit will also have a couple of special characteristics that need to be reflected in the unit’s definition. First of all, our unit’s movement will vary from turn to turn, since these Misfits are often arguing among themselves and therefore distracted from the battle. Their actual movement rate will be “d6”. Secondly, only one unit of “Loonies” can be taken in any army. Lastly, our Loonies are also completely insane and are therefore not affected by any sort of fear, nor are they able to be routed in combat.

Create the Unit

Creating a new unit uses the same sequence of steps as for races:

- Open up the “misfits.tut” data file within ABCreator if you have not already done so.
- Click on the “Unit” tab along the top. This will show a list of all of the existing units on the left – this list should be empty at this point – and all of the fields for a unit are shown on the right.
- Click on the “New” button in the lower left corner. This will create a new unit for you to edit. By default the unit is assigned a name of “??????” (just like for Races) and a unique id of “xxxxxxxx”. All of the stat fields are initialized to a value of 0, and a few other fields are initialized appropriately to default values.

Specify the Basics

Let’s get started with all of the basic facets of our new unit. The following steps take you through specifying proper values for each of the basic fields for the unit.

- Select the “Name of Unit” field to begin with, and enter the name “Loonies” in this field.
- Switch to the “Unique Id” field (you can use the <Tab> key for this). Since our unit is part of the “Misfits” race, we need to make sure that the unit’s unique id indicates this fact. This is accomplished by using the two-letter prefix for Misfits (“mf”) as the first two characters of the unit’s unique id. The remaining six characters can be whatever we want to identify this unit. Since the unit is called the “Loonies”, we’ll just use the first 6 characters of that name. The combined result is the unique id “mfloonie”. Enter the text “mfloonie” in this field.
- Switch to the “Abbrev” field, where you can assign an abbreviation for the unit. Since this abbreviation can be up to five characters in length, let’s just use the value “Loon”. Enter this text in the field.
- Switch to the “Unit Type” field. Since types are a special facility, we won’t be using them at this time (although a later tutorial shows how they work). You can simply leave this field blank.
- Switch to the first of the stats fields, which is labeled “Mv” for the tutorial game system. Each game system will have its own set of stats, appropriate to that system. The tutorial system has 10 stats. Since the stats for our tutorial game system are used solely as an example, they really are pretty meaningless. To keep things simple, just enter the same value of “4” for each stat. Although the unit’s movement is supposed to be special, we can’t specify that here, so don’t worry about entering a value of “4” at this time. We’ll make the proper adjustments later.
- Switch to the “Composition Group” field. In this drop-down list, each of the valid composition groups for the game system are listed. Since we are creating a unit of infantry, select the “Troops” group. This will ensure that all of the points for this unit are accumulated into the “Troops” composition group.
- We stated that the minimum size for this regiment must be five models, but there is no maximum size. To specify this, switch to the “Size” field and then enter the value “5”. A single number in this field always indicates the minimum number of models in the unit.

- Switch to the “Unit Category” field. This is another drop-down list, and this one contains all of the valid unit categories which are defined for the game system. Our unit is infantry, so we will categorize it as a “Regiment”.
- The next field is the “Cost” field. The cost here is per model, which is 11 points, so enter the value “11”.
- The last of the “basic” fields is the number of special items that the unit is allowed to select. Since this unit is not allowed to take any special items, simply enter a value of “0” (which is also the default value for the field).

Comments, Notes, and Report Fields

There are three different description fields for a unit. The first is the “Comments” field. The contents of this field are displayed only in the “Details” region for the unit when it is highlighted in the list of “Available Units” within Army Builder. The second is the “Notes” field. These notes are displayed both in the “Details” region within Army Builder *and* in the army roster output (e.g. printouts). The final description field is the “Report” field. The contents of this field are included only in the army roster output. Within the “Details” region in Army Builder, the “Comments” are displayed first, followed by the “Notes”. Within the roster output, the “Notes” are shown first, followed by the “Report” contents. This separation allows you to choose what information is included for your units (and this same approach is used for options).

Each of these fields can contain multiple text segments, and each segment must be separated by a semi-colon and a space (“; ”). If you use any other text, it will be treated as a single segment. You will not be able to enter any carriage returns into the memo fields, so the semi-colon and space is the convention used between segments.

- Select the “Comments” field. Since this unit will only be allowed to be added once to an army roster, we need to let the user know this. However, since there is no need to include this fact in the roster printouts, the best place to put this information is in the “Comments” field. So, within this field, enter a statement like “Maximum of one unit allowed in roster”. When displayed on-screen within Army Builder, each segment starts on a new line, just as if you had inserted a carriage return.
- Switch to the “Notes” field. There are two important things that need to be described about this unit. First of all, they are immune to the effects of fear. Secondly, they cannot be routed in combat. These are two separate entries that must be specified within the “Notes” field. The proper text to be entered is shown below. Be careful to ensure that the two segments are separated by a semi-colon and a space, as this will cause the two segments to appear on separate lines within Army Builder (our desired result).

  Immune to Fear; Cannot be routed in HtH
- Switch to the “Report” field. This field is only used to include lengthy descriptions that are too long to display within the “Details” region of Army Builder but that need to be included in the roster printouts. There is no information for Loonies that is like this, so just leave this field blank.

Attributes

Attributes for units work the same way that they do for races. The key difference is that units have two different kinds of attributes: local and external. It turns out that our Loonies require one of each kind.

- Look at the “Local Attributes” region along the right. It is the upper one of the two different attributes regions. We need to create a new local attribute, so click on the “New” button within this region.
- The attribute specification dialog appears. This time, it lists all of the local unit attributes in the column along the left. Our Loonies have a special movement rate of “d6”, so we need to select the attribute which allows us to specify a text value for a stat. This happens to be the first attribute listed on the left (“attr”), so click on this attribute in the list, and the syntax for the attribute appears in the upper right description region.
- Click within the edit box in the lower right, as this is where we will enter the proper attribute text.

- Enter the attribute text shown below. This attribute indicates that the value of the "mv" stat should be replaced with the text "d6". As a result, the value of "4" that we entered earlier will be ignored when the unit is displayed within Army Builder. The use of the quotation marks is important, since we could use any characters we want for the stat (including spaces).

attr:mv="d6"

- Click on the "OK" button to save this new attribute value. You will see this attribute listed now in the list of local attributes for the unit.
- Now we need to specify the external attribute for the unit. Click on the "New" button within the "External Attributes" region, and the attribute specification dialog appears.
- The list of attributes on the left is again different, and this time we need the attribute which restricts the maximum number of times this unit can be taken within the army roster. This attribute happens to be at the bottom of the list ("umax"). Click on the "umax" attribute on the left to show its syntax at the top right.
- Click within the edit box so that we can enter the proper text for this attribute. Although the syntax for the "umax" attribute can be relatively complicated in some situations, virtually the entire syntax is optional. We only need this attribute in its most simple form, as we need to specify that there can be only one of this unit in a given roster. Enter the text "umax:1u" within the edit box.
- Click "OK" to save the new attribute. The attribute appears in the list of external attributes for the unit.

Option Links

The final section that we need to specify for this unit is the set of "Option Links". Since we haven't created any options yet, we will leave this section empty. Later in the tutorial, after we have created an option, we will return here to assign an option to this unit.

Verify the Unit

It is now time to verify that all of our changes are working. The first thing we must do is save our changes to the record, then we must save the updated file. After that, we can actually load and verify our changes.

- Look at the tabs across the top. You will see that the currently selected tab is shown as "**Unit**". This indicates that we have not saved the changes to the current record. To do this, click on the "Save" button in the lower left corner or press <Ctrl-S>. If you are told that there is an error in one of the fields, please verify that you have entered everything in the fields exactly as specified in this tutorial.
- The next thing we have to do is save our changes to the data file. Do this by clicking on the "File" menu and then clicking on the "Save" menu item. Our data file will be updated with the changes we have made.
- It is now time to verify our changes, so launch Army Builder. Alternately, you can leave Army Builder running and select "Switch Game System" from the "File" menu. By selecting the "Tutorial Game System" and clicking on "OK", the data files are reloaded, complete with all the newly added material. This is a lot easier than waiting to re-launch Army Builder every time.
- Select the tutorial game system.
- Create a "Misfits" army, specifying a size of 500 points.
- Our new unit, "Loonies", should be shown as the only available unit for this race. All of the stats and cost information should be exactly as you entered it, with the "Mv" stat being shown as "d6". In the "Details" region to the right, there should be 3 text segments shown. First, there should be the information you entered into the "Comments" field, and this should be followed by the entries in the "Notes" field.
- Add a "Loonies" unit to your roster by double-clicking on it. Within the roster, the Loonies unit should also have the "Mv" stat shown as "d6".
- The roster should be determined to be invalid by Army Builder, so the exclamation point in the lower right should be glowing. Click on this button to bring up the validation results. The problem reported should be that there are not enough points allocated to the "Troops" composition group. Since our army size is 500 and we only have 55 points of Troops in the army (the default size unit of 5 Loonies), we haven't met the minimum requirement.

- Add enough models to the current “Loonies” unit to bring the total to 250 or more points. The army should now be reported as valid, since the “Troops” composition group is now within the valid range.
- Add a second unit of “Loonies” to the roster. The army should now be reported as *invalid* again.
- Click on the exclamation point to bring up the validation results. This time, the problem should be that you can only have one unit of “Loonies” in the roster (and you have two). This means that our “umax” attribute is working correctly.
- At this point, we have verified that our new unit is working properly, so you can close down Army Builder.

Special Considerations

Here are a few additional points to remember when working with data files in ABCreator:

- As mentioned previously, save you changes to the current record at regular intervals along the way. There are many fields that require specific syntax rules to be followed. It is much easier to identify and fix the problems right away than to wait until later.
- If you attempt to save your record and an error is reported in an attribute, you will need to edit that attribute to correct it. The attributes are always shown in a list. To edit or delete one of them, you need to click on the attribute that you want to operate upon. Then, you can click either the “Edit” or “Delete” button associated with the group of attributes. If you click on “Edit”, you are presented with the attribute specification dialog, and the contents of the selected attribute are automatically placed in the edit box for you to modify.

CREATING A NEW OPTION

Objective

This tutorial is intended to walk you through the creation of a special weapon that we will ultimately give to our new Loonies unit. This new weapon is a Battle Axe, and the effects of the weapon are that it increases the strength stat of the wielding model(s) by +1. The cost of the option is 1 point.

Create the Option and Specify the Basics

These two steps have been combined now, since you should be getting the hang of how this works.

- Open up the “misfits.tut” data file within ABCreator.
- Click on the “Option” tab along the top. This will show the list of existing options on the left (none yet) and the fields for an option on the right.
- Click on the “New” button in the lower left corner to create a new option. As with new units, the new option is assigned a name of “??????”and a unique id of “xxxxxxxx”.
- Enter the name of the option as “Battle Axe”.
- Enter a unique idea which will identify this weapon. Since the weapon is unique to the Misfits race we are creating, it is probably best to assign the option the “mf” prefix (for consistency). If this option was going to be used by multiple races, then it would be best to not use any special prefix. As it is, let’s assign this option the unique id “mfbattle”.
- The next field is the abbreviation that will be shown in the “Options” column of the roster when the option is given to a unit. This abbreviation can be a maximum of three characters. Enter the abbreviation as “Btl”.
- Next we need to specify the category of option that this belongs to. The category is used to sort the options that are displayed in the Options Panel within Army Builder. Since this is a weapon, we can assign it the “Weapon” category.
- The last basic field is the cost. This was established to be 1 point, so enter a value of “1”.

Attributes

We now need to define any special attributes for this option. Since the weapon gives its wielder +1 Strength, we need to specify that fact via attributes so that Army Builder will make the proper adjustments when the option is selected.

- Click on the “New” button within the “Special Attributes” region, and the attribute specification dialog appears, listing the attributes that are specific to options.
- The attribute we need is one that adjusts the stat value for the unit which the option belongs to. That particular attribute is the “stat” attribute. Click on it in the column on the left, displaying the syntax in the upper region.
- In our case, we want the option to increase the “St” stat of the unit by 1. The proper syntax for this will be “stat:st+1”, so enter this text.
- Click on the “OK” button to save the new attribute.

Comments, Notes, and Report Fields

As with units, there are three different description fields that can be used. They work similarly to the ones for units, hence they are named the same. The “Comments” field contains descriptions that are only displayed to the user within Army Builder – they are not included in any printouts. The “Notes” field contains descriptions that appear both within the display and in the printouts. The “Report” field contains text that is only included in roster printouts.

- Click on the “Comments” field.

- Our option is a weapon which increases the strength of the wielder by +1. This is going to be automatically handled by Army Builder within the roster printouts, so there is no reason to repeat this within the roster. Consequently, we simply need to tell the user how the option works when he is editing his roster in Army Builder. Enter a description in the "Comments" field like "Adds +1 Strength to wielder".
- Select the "Notes" field. Since all of the mechanics of this option will be handled by Army Builder and reflected in the roster printouts, there is no need for restating that information. Leave this field blank.
- Leave the "Report" field blank as well (for the same reason).

Conflict Groups

The use of conflict groups can be very important, yet it is often times somewhat confusing at first. This is especially true when an option needs to be assigned to more than one conflict group at a time. Fortunately, our Battle Axe is a simple option that doesn't need to use any of the complicated mechanisms.

- Our option is a simple weapon. So, click on the "Wep" entry within the list of conflict group abbrevations. By assigning the option to the "Wep" group, we instruct Army Builder to make sure that all options belonging to this group are properly restriced. The "Wep" group is defined (in the definition file) to allow only one option belonging to the "Wep" group to be given to a unit at one time. So, if a unit has multiple "Wep" options available to it, Army Builder will make sure that only one can ever be taken at a time. We will see how this works later in this set of tutorials.

Verify the Option

Unfortunately, there is no way to verify that an option is defined properly unless it is assign to a unit. Consequently, all we can do at this step is save the option and the file.

- Save the current option record (<Ctrl-S>).
- Save the changes to the data file ("File"->"Save").

Special Considerations

There is one issue you may find important that is particular to Options.

- Some options will need to be assigned to more than one conflict group. When this occurs, you need to select multiple groups within the listbox presented. This listbox is a standard Windows multiple-selection listbox. As such, the mechanics for selecting more than one option involve the use of the <Shift> and <Ctrl> keys. If you select one group, and then hold down the <Shift> key when you select the second group, all groups inbetween are also selected. If you select a group, and then hold down the <Ctrl> key when you select the second group, the newly selected group is selected in addition to the first group. If you want to select three or more non-adjacent groups, simply hold down the <Ctrl> key when you click on the second and subsequent groups. Clicking on a group that is already selected will deselect it when the <Ctrl> key is down.

PUTTING IT ALL TOGETHER

Objective

This tutorial will show you how to assign the new weapon option to the new unit. The weapon option will be selectable by the user from within Army Builder.

Link Nature

The way that options are attached to units is through "Links". However, before we create any links between units and options, it is important that the different types of links are understood. Basically, there are four fundamental types of links, and, in order to avoid confusion with the "types" mechanism within Army Builder they are referred to as link "natures". Each of these link natures is described within the Construction Kit manual, but they are reviewed quickly below. There are also two additional link natures, but they are actually just a convenient notation that combines a few characteristcs. The different natures are:

- Cost
  The linked option is selectable by the user and the cost of the option is charged when it is selected. This is used for units which have options that can be purchased at an extra cost.
- Free
  The linked option is selectable by the user, but there are no points charged when it is selected. This is used when a unit can choose an option for free.
- Auto
  The linked option is automatically selected for the unit, and it cannot be deselected by the user. The cost of the option is automatically charged, just as if the link nature were "Cost" and the user selected it. This is used when a unit is compelled to purchase an option and the option cost is not built into the unit cost (i.e. the option must be paid for).
- Incl
  The linked option is automatically selected for unit, and it cannot be deselected by the user. The cost of the option is waived for the unit, so the option is granted free of charge. This is used for units that come pre-equipped with options, where the option cost is built into the unit cost.
- Hide
  The linked option is treated as a standard "Cost" option. The key difference is that the option is, by default, hidden from the user, just as if the "hide" attribute was assigned to the option.
- Appl
  The linked option is treated as a standard "Incl" option. The key difference is that all adjustments by the option that are normally applied to the base stat value of the unit are not applied. This is useful when you want to show the adjusted value in the basic unit stat, but you still want the other effects of the options to be applied normally when the unit is added to the roster.

Establish the Link

It's now time to assign our new weapon to our new unit so that it can be selected and used in battle.

- Click on the "Unit" tab to show the units defined in our data file (there should be only one).
- Click on the "Loonies" unit in the list on the left. Its contents are filled into the fields on the right, and we can now edit that contents.
- Find the "Option Links" region, which is along the right side and about mid-height. Click on the "New" button in this region to create a new link to an option.
- The dialog for editing option links appears, allowing you to enter the information for the new link. There are two things that you must specify for a link. First, you must identify the option to which the link will be made. This is accomplished by specifying the unique id of the option. The second thing required is the nature of the link that you want.
- The unique id for the option we created is "mfbattle". So, enter the text "mfbattle" into the field labeled "Unique Id".

- Of the four primary link natures that we can choose from, the nature that we want for this unit is “Cost”. So, select “Cost” from the list of link natures that is presented.
- Click on the “OK” button to save this new link.

Verify the Link

We can now save our changes to the unit and then verify that the option is available and working properly within Army Builder.

- We have changed the unit, so we need to save our changes. Save the unit changes first (<Ctrl-S>).
- Now, save the changes we made to the data file (“File”->“Save”).
- Switch game systems within Army Builder again and select the tutorial game system.
- Create a new “Misfits” army.
- Double-click on the “Loonies” unit to add it to the roster. The new “Battle Axe” option should appear in the “Options Panel” on the right for the newly added unit.
- Right-click over the “Battle Axe” option on the right. A description window should appear that lists the cost of the option at 1 point and that displays the text you entered into the “Comments” field for the option.
- Click anywhere to cause the description window to disappear.
- Left-click over the “Battle Axe” option to add it to the unit. Three things should have occurred. First, the cost of the unit should have increased from 55 to 60 points. This reflects the added cost of the 1 point option for 5 models in the unit. Second, the abbreviation for the option (“Btl”) should appear in the “Options” column of the roster. Lastly, the strength of the unit should now show as “4/5”, indicating that the effects of the option have been factored into the Strength and adjusted it from 4 to 5.
- Left-click on the option to deselected it. The unit should revert to its initial state.
- Our data file is working properly. We have now created a new unit that can choose a new option.

INTER-FILE REFERENCES

Objective

The objective of this section of the tutorial is to demonstrate how you can make references between data files. Since all data files with the same file extension are loaded and merged together when Army Builder is launched, it is possible to define references between records in the two files. For example, in our race definition for the Misfits, we earlier specified that they could have Elves and Tree Huggers as allies. This was a reference to the information in another data file (these races are defined in “data.tut”). In this next example, we demonstrate that other records can reference each other, too.

Typically, inter-file references should be used very carefully. It is very helpful if at least one core data file is defined for a given game system. This file (or files) should contain all of the core information, such as standard weapons, monsters, spells, magic items, etc. This core file should not make any references to non-core files. Then, each new race should have its own data file, making references to the contents of the core file(s). This approach is described in detail witin the Construction Kit manual. Our new “misfits.tut” data file works this way, and we will now make additional references to the core data file for the tutorial game system.

Link Core Options to Local Units

In this example, we will give our Loonies unit an additional weapon option: the two-handed weapon. This new option is defined in the main data file for the tutorial game system (“data.tut”).

- Open up the “data.tut” data file from within ABCreator.
- Click on the “Option” tab to view all of the options that are defined in this file.
- Click on the “Two-Handed Weapon” option in the list on the left. This will show all of the details of this weapon option. Make a note that the unique id of this option is “twohand”.
- Open the “misfits.tut” data file within ABCreator. This automatically closes and discards the “data.tut” file.
- Click on the “Unit” tab.
- Click on the “Loonies” unit in the list on the left.
- We now need to create a new option link, so click on the “New” button within the “Option Links” region on the right.
- Within the dialog that appears, enter the unique id of the two-handed weapon option that we found in the main data file. If you don’t remember it, the unique id is “twohand”.
- Specify a link nature of “Cost” and click on the “OK” button to save the new link.
- Save the changes to the record (<Ctrl-S>).
- Save the changes to the data file (“File”->“Save”).

Verify the Link

Use Army Builder to verify that the new option is available to our unit.

- Switch game systems within Army Builder again and select the tutorial game system.
- Create a new “Misfits” army.
- Double-click on the “Loonies” unit to add it to the roster.
- Verify that the “Two-Handed Weapon” option appears in the “Options Panel” on the right for the newly added unit.
- Make sure that the conflict groups are working properly by selecting one of the two weapon options. When one option is selected, the other one should be disabled.

Creating a New Hero Unit

Special Note!

From this point on, the tutorial sections will become more terse. It is assumed that you are starting to get a good grasp of how ABCreator works, so the tutorials will state their objectives more tersely and then walk you quickly through the sequence to achieve those objectives.

Objective

This tutorial will walk you through the creation of another unit, but this time we will be creating a Hero to lead the Misfits into battle. There are many similarities between a Hero unit and a regimental unit, but there are a few notable differences, and they will be pointed out along the way. The key distinctions for a Hero are that the unit size can only ever be one model and the unit is capable of selecting up to two special (a.k.a. magic) items. In the case of our Misfit Hero, though, he will be unable to select Enchanted Items. In addition, our army must have at least one unit of Loonies in it for us to take our hero. The hero's official designation will be "Learless Feader".

Create the Unit

The following steps will allow you to create the new Hero unit that is described above. It is assumed that you already have the "misfits.tut" data file opened within ABCreator.

- Click on the "Unit" tab and then create a new unit (<Ctrl-N>).
- Specify the unit name of "Learless Feader".
- Assign a unique id of "mffeader".
- Use the abbreviation "Fead".
- Leave the unit type blank.
- Assign a value to all stats. For simplicity, you might just set all the values to "4".
- Select the composition group of "Hero".
- Specify the unit size to be a minimum of 1 model and a maximum of 1 model. This requires the notation of "1:1".
- Select the unit category of "Hero".
- Assign a cost of "50" points.
- Our hero can select up to 2 items, so enter a value of "2" in the #Items field.
- Our hero is unable to take any Enchanted Items, so we might want to warn the user of this. Enter an appropriate message into the comments field, such as: "Unable to take Enchanted Items".
- Our hero is similar to the Loonies we created earlier with respect to game mechanics, so we'll enter the same text for the notes field that we used for the Loonies:

  Immune to Fear; Cannot be routed in HtH
- There is nothing special to enter for the report field.
- By default, any unit belonging to the "Hero" unit category will be able to select Enchanted Items. This hero is unable to select Enchanted Items, so we need to enforce that fact. This is done by specifying a local attribute. Click on the "New" button in the local attributes region.
- The attribute we need is the "spec" attribute, which can be used to either add or remove a special item category for a unit. Since the abbreviation for the Enchanted Item category is "ench" (per the definition file), our attribute syntax is "spec:-ench". Enter and save the attribute.
- Our hero requires that the army roster also include at least one unit of Loonies. This requirement is specified via an external attribute. So, click on the "New" button within the external attributes region.

- The attribute we need is the “reqd” attribute. Since we only need to have a single unit of Loonies in the army to enable any number of Learless Feaders, we don’t need to use any of the sophisticated syntax of the attribute. Enter the text “reqd:unit=mfloonie” for the attribute and save it.
- The last thing we need to do is give our hero all of the options that he is allowed to take. For each option listed below, click on the “New” button within the option links region and add the proper link (all of these options are defined in “data.tut”):
    - twohand @ cost
    - 2ndweap @ cost
    - mfbattle @ cost
    - shield @ cost
    - ltarmor @ cost
    - hvyarmor @ cost
- Save the changes to the record and then save the changes to the data file.

Verify the Unit

We can now verify that our new hero works properly.

- Switch game systems within Army Builder again, select the tutorial game system, and create a new Misfits army. You should see the new Learless Feader unit shown at the top, and the details region should show the text that you entered in both the comments and notes fields.
- Add a Learless Feader unit to the roster. There should be a single model added to the roster, and all of the options you assigned in the data file should be listed as available for the unit. In addition, both of the size adjustment buttons (“+” and “–”) should be disabled, since the size of the unit can never be changed from one model. Lastly, the points for our Learless Feader should be assigned to the Hero composition group, so this should be properly reflected in the composition breakdown view in the lower right corner.
- Click on the button to give the unit special items. The unit should be allowed to take up to two items (shown in the upper right corner). Also, only the Weapons and Armor categories should be available for selection (check the drop-down list of categories for this), as the Enchanted Item category should be denied to this unit. Exit the special items screen.
- The roster should be shown as invalid, so click on the exclamation point button to bring up the validation results. There should be two entries listed, the important one being that the Learless Feader unit requires a unit of Loonies. Close this report.
- Add a unit of Loonies to the roster.
- Click again on the exclamation point button to bring up the validation results. This time, the dependency on the Loonies unit should be gone, since that requirement is now satisfied.
- Our new hero unit is working properly.

Creating a "Similar" Unit

Objective

In this tutorial, we will create another regimental unit that is very similar to the Loonies unit we already created. To do this, we'll copy the Loonies unit and only make the few changes that are needed. Our new unit will be the "Bonzai Boys", and the key differences will be that their movement is not random, they come equipped with two weapons, the cost is 12 points per model, and there is no limit on the number of units that can be taken in an army.

Create the Unit

As before, you are assumed to already have the "misfits.tut" data file loaded into ABCreator when these steps begin.

- Click on the Unit tab and then click on the Loonies unit.
- Our new unit is very similar to the Loonies, so there is no point in re-entering all the same data. Instead, we'll make a copy of the Loonies unit. This is done by clicking on the "Edit" menu and then selecting the "Copy Record" menu item.
- A confirmation dialog appears, asking if you are certain that you want to copy the Loonies unit. Click on the "Yes" response.
- We now have a new unit that has everything exactly the same as the Loonies unit, with one exception. The unique id for our new unit has been changed to "xxxxxxxx". Since this field must be unique between all units (hence the name), we can't create a copy that has the exact same value.
- Change the name of the unit to the "Bonzai Boys".
- Specify an appropriate unique id for this unit, such as "mfbonzai".
- Give the unit its own abbreviation, such as "Bonz".
- Since this unit does not have the random movement of the Loonies, we need to eliminate the attribute that overwrites the movement stat with "d6". Click on the local attribute "attr:mv=d6", then click on the "Delete" button immediately to its right. You should be asked to confirm deletion of the attribute, to which you should respond "Yes".
- None of the information in the comments field applies to this unit, so delete all of the text in the comments field.
- Our new unit does not have the restriction of only appearing once in a roster, so we also need to delete the "umax" attribute in the external attributes region. Click on the attribute, then click on the "Delete" button to its right. Confirm the deletion.
- The cost of the Bonzai Boys is slightly higher than the Loonies, so we need to change the cost to 12. Make the change.
- The last change we need to make is to the option links. Our Bonzai Boys do not have the ability to select either of the weapons assigned to the Loonies, so we need to delete them. This works just like deleting attributes, so click on the option link in the list, then click on the "Delete" button to the right. When asked to confirm the deletion, respond "Yes". Do this for both of the existing option links.
- Our new unit currently has no options assigned to it, so we still need to give them their second weapon option, but this time the option is included in the cost of the unit. Click on the "New" option link button, then enter the option unique id "2ndweap" and select a link nature of "Incl" (this option is defined in "data.tut"). Save the new option link.
- Save the changes to the record, then save the changes in the file.

Verify the Unit

Now that the unit is created, we need to make sure that everything was specified properly.

- Switch game systems within Army Builder, select the tutorial game system, and create a new Misfits army. You should see the new Bonzai Boys unit in the list.
- Click on the new unit. The cost should now be 12, the movement rate should be 4, and the details region should show the edited descriptions.
- Add the new unit to the roster. It should have a single option available and that option should be both checked and grayed out, since it is included within the cost of the unit.
- Add another Bonzai Boys unit to the roster.
- Click on the exclamation point button to check on the validation results. The only message should be that there are not enough points in the Troops group, which means that the restriction of only having one unit in the roster does not apply to this new unit.
- It looks like our new unit is working properly.

CREATING A NEW ITEM

Objective

It's now time to create a special magic weapon that is only available to our new Misfits race. This magic weapon will give its wielder a +2 bonus to Strength and will be called the "Force Blade". Its cost is 45 points.

Create the Item

Make sure you have the "misfits.tut" data file loaded into ABCreator and then follow these steps:

- Click on the Item tab and then click on the "New" button to create a new item. You will notice that items actually look a lot like options with respect to the fields that are shown.
- Enter the name of the new item as "Force Blade".
- Specify a unique id for this item. As with other records, since this item is specific to the Misfits race, it is probably best to include an "mf" prefix for clarity, although this is not required. For this tutorial, enter the unique id "mfforce".
- This item can only be taken once within an army roster, so we need to specify a value of "1" for the #Avail field.
- Select the item category "Weapon" from the available list, since this is a magic weapon.
- Specify the item cost of 45 points.
- Our weapon can only be used by the Misfits race, so we first need to specify the attribute which enforces this restriction. Click on the "New" button in the attributes region.
- The attribute we want is the "itst" attribute, so click on it for the details. Since we want to have this weapon be usable by the Misfits, whose prefix is "mf", the proper syntax for this attribute should be "itst:(race=mf)". Enter this attribute and save it.
- We now need to define the effects of this weapon, namely the +2 Strength adjustment. This is done via another attribute, so click on the "New" button for attributes again.
- This time we want the "stat" attribute, since we need to modify the wielder's Strength stat. The actual syntax that we need to specify is "stat:st+2", so enter the attribute and save it.
- The comments field for items is the text that is displayed within the roster while the user is editing his army. Consequently, this text must be short, since it must fit within a very small area. Use the text "Strength +2".
- The notes field is the text that is displayed within the description region to the right of the item selection list. When the user is choosing items for a unit, single-clicking on an item shows a description of what that item does, and this field specifies the text that is shown. For this example, use the text "Increases wielders Strength by +2 (included in stat adjustment)".
- The report field is the text that is actually included in roster output (e.g. printouts). In our case with this item, the text that goes in the roster printout might as well be the same as the text for the notes field. Instead of copying this text all over again, ABCreator lets you simply reference the text in the notes field from within the report field. This keeps the resulting data files a bit smaller than they would otherwise need to be. So, we can specify that the report field should contain the same text as the notes field, which is accomplished by simply inserting the character "^". For the report field, enter the text "^".
- The last thing we need to specify for our item is any conflict group(s) that it belongs to. In our case, this is a magical weapon, so we need to select the "MWp" conflict group. Simply click on this group to select it.
- Save the changes to the record and then save the changes to the data file.

Verify the Item

Verifying items can require a little more effort than verifying units, simply because there are more steps involved in modifying the special items for units. Here is how to verify that our new item is working properly.

- Switch game systems within Army Builder, select the tutorial game system, and create a new Misfits army.

- Add a Learless Feader to the roster.
- Click on the button to give this hero special items.
- In the list of magic weapons that is shown, you should see the new Force Blade that has been added. Single-click on the new item. This should display the text you entered into the notes field for the item.
- Add the item to the hero by double-clicking on it.
- Exit the special items screen.
- The roster should be updated to show the Force Blade listed beneath the Learless Feader. The description shown should be the text that you entered into the comments field for the item. In addition, the Hero's "St" stat should be shown as "4/6", reflecting the +2 adjustment due to the weapon.
- Delete the Learless Feader from the roster.
- Go to the Allies list in the upper right corner and select Elves.
- Add an Elf Hero to the roster.
- Click on the button to add special items to this hero.
- In the list of magic weapons, the new Force Blade weapon should not be shown. This indicates that the racial restriction of the item (only available to Misfits units) is being properly enforced.
- Our new item appears to be working properly.

Special Considerations

When working with items, there are certain issues that are important to be aware of:

- The "propogation" mechanism for the item descriptions will cascade. This means that you can enter text into the comments field, then include that text within the notes field by inserting the character "^". You can also propogate that exact same text into the report field by using "^" within the report field. You can also include additional text at each level. For example, could enter the text "comments" within the comments field, then enter the text "^ notes" within the notes field, and then enter the text "^ report" within the report field. The net result would be that the notes field is displayed as "comments notes" and the report field is displayed as "comments notes report". This can be extremely useful for items which require lengthy descriptions, as each level can simply be a truncated version of the next level up. This propogation feature also applies to Elements, but it does NOT apply to Units or Options.
- The mechanics of elements work very similarly to items, with only a few minor differences. Consequently, there are no examples given in this tutorial. Now that you know how to create items, you should be able to create your own elements with little trouble after reading about them in the Construction Kit manual.