

Synology DiskStation Manager

3rd-Party Apps Developer Guide

Synology[®]

2011-12-15



Synology Inc.
© 2011 Synology Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Synology Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Synology's copyright notice.

The Synology logo is a trademark of Synology Inc.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Synology retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Synology-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Synology is not responsible for typographical errors.

Synology Inc.
3F-3, No. 106, Chang-An W.
Rd. Taipei 103, Taiwan

Synology and the Synology logo are trademarks of Synology Inc., registered in the United States and other countries.

Marvell is registered trademarks of Marvell Semiconductor, Inc. or its subsidiaries in the United States and other countries.

Freescale is registered trademarks of Freescale

Semiconductor, Inc. or its subsidiaries in the United States and other countries.

Other products and company names mentioned herein are trademarks of their respective holders.

Even though Synology has reviewed this document, SYNOLOGY MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY. IN NO EVENT WILL SYNOLOGY BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Synology dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Table of Content

Introduction	4
Who Should Read This Document?	4
System Requirements.....	4
Compiling an Application	5
Downloading DSM Tool Chain.....	6
Compiling	7
Compiling Open Source Projects.....	8
Compiling Kernel Module.....	11
Installation	13
Put the Application in /usr/local.....	13
Run the Application When the System Boots	13
Integrating Applications into Synology DiskStation Manager 3.0	14
Startup.....	14
config.....	15
Integrating Applications into Synology DiskStation Manager 2.3	17
Startup.....	17
Application.cfg.....	18
Example	20
Adding Your Customized Applications to Desktop Items	21
Desktop.cfg	22
Integrating Synology DiskStation with web authentication	24
Creating a Synology Package	26
Package Introduction	26
Package Structure	26
INFO	27
WIZARD_UIFILES.....	31
package.tgz	32
Scripts	33
Script Environment Variables	34

Introduction

To instruct our users and system integrators on how to install 3rd -party applications on our product DiskStation developed on Linux kernel, Synology offers this developer guide to help you:

1. Compile programs to run on Synology DiskStation.
2. Integrate the applications with DiskStation's operation system, i.e. Synology DiskStation Manager (DSM.)
3. Install the application files to the recommended path to keep them intact during DSM upgrade.
4. Integrate the applications with Synology web authentication interface.
5. Create a package file for manual installation via Synology Package Center.

Who Should Read This Document?

This document is written for Synology users and system integrators interested in adding their applications to Synology DiskStation. You are advised to have some basic understanding of Linux programming before starting to read this document.

System Requirements

DSM 2.0-0636 or above is required on Synology DiskStation.

Compiling an Application

The Synology DiskStation uses an embedded SoC as CPU, which has a different architecture from x86-based PCs. There are two platforms, ARM and PowerPC, for different Synology DiskStation models. In order to run 3rd-party applications on the Synology DiskStation, it is necessary to compile the applications into an executable format for the corresponding platform.

The table below lists the CPU, architecture, Endianness, and Linux kernel version of each Synology DiskStation model. The information will help you determine which set of DSM tool chain (please refer to **Downloading DSM Tool Chain** on page 6) to download for respective models.

Model	CPU	Arch	Endianness	Linux
CS407, DS107+, DS207+, RS407	Marvell 5281	ARM	Little Endian	2.6.15
DS109, DS209, DS409, DS409slim, RS409, DS110j, DS210j, DS410j, DS211j, DS411j	Marvell 6281	ARM	Little Endian	2.6.32
DS111, DS211, DS411, DS411slim, DS211+	Marvell 6282	ARM	Little Endian	2.6.32
CS407e, DS207, DS209j	Freescale 8241	PowerPC	Big Endian	2.6.24
DS107e, DS107, DS108j, DS109j	Freescale 8241	PowerPC	Big Endian	2.4
RS408, RS408-RP, DS408, DS508	Freescale 8543	PowerPC	Big Endian	2.6.32
DS109+, DS209+, DS409+, RS409+, RS409RP+, DS509+	Freescale 8533	PowerPC	Big Endian	2.6.32
DS110+, DS210+, DS410	Freescale 8533 E	PowerPC	Big Endian	2.6.32
DS710+	Intel Atom D410	Intel x86	Little Endian	2.6.32
DS712+	Intel Atom D425	Intel x86	Little Endian	2.6.32
DS1010+, DS411+, RS810+, RS810RP+	Intel Atom D510	Intel x86	Little Endian	2.6.32

Model	CPU	Arch	Endianness	Linux
DS411+II, DS2411+, DS1511+, RS2211+, RS2211RP+	Intel Atom D525	Intel x86	Little Endian	2.6.32
DS3611xs, RS3411xs, RS3411RPxs	Intel Core i3	Intel x86	Little Endian	2.6.32

To compile an application for the Synology DiskStation, a compiler that runs on Linux PC is required in order to generate an executable file for the Synology DiskStation. This compiling procedure is called “cross compiling”, and the set of compiling tools (compiler, linker, etc) used to compile the application is called “tool chain”.

Downloading DSM Tool Chain

To download DSM tool chain, please go to <http://sourceforge.net/projects/dsgpl/files>. The table below shows the filename of tool chain for each DiskStation model:

Model	Tool Chain
CS407, DS107+, DS207+, RS407	gcc343_glibc232_88f5281.tgz
DS109, DS209, DS409, DS409slim, DS411slim, RS409	gcc421_glibc25_88f6281-GPL.tgz
DS110j, DS210j, DS410j, DS211j, DS411j, DS111, DS211, DS411, DS211+	gcc421_glibc25_88f6281-GPL.tgz
CS407e, DS207, DS209j	gcc334_glibc233_ppc_2624.tgz
DS107e, DS107, DS108j, DS109j	gcc334_glibc233_ppc_2422.tgz
DS408, DS508, RS408, RS408-RP	gcc343_glibc234_ppc854x-GPL.tgz
DS109+, DS209+, DS409+, RS409+, DS509+, RS409RP+, DS110+, DS210+, DS410	gcc343_glibc234_ppc853x-GPL.tgz
DS710+, DS712+, DS1010+, DS1511+, DS411+, RS810+, RS810RP+, DS411+II, DS2411+, RS2211+, RS2211RP+	gcc420_glibc236_x64-GPL.tgz, gcc421_glibc236_x86-GPL.tgz

Model	Tool Chain
DS3611xs, RS3411xs, RS3411RPxs	gcc421_glibc236_x86_bromolow-GPL.tgz, gcc420_glibc236_x64_bromolow-GPL.tgz

After you download the DSM tool chain, untar it into **/usr/local/** using the following command:

```
# tar xzpf gcc343_glibc232_88f5281.tgz -C /usr/local/
```

Please make sure the tool chain is located in the directory **/usr/local** to ensure proper integration.

Compiling

Now you can start to compile an application. For example, the content of an application called “sysinfo.c” looks like this:

```
#include <sys/sysinfo.h>

int main()
{
    struct sysinfo info;
    int ret;

    ret = sysinfo(&info);
    if (ret != 0) {
        printf("Failed to get system information.\n");
        return -1;
    }
    printf("Total RAM: %u\n", info.totalram);
    printf("Free RAM: %u\n", info.freeram);
    return 0;
}
```

To compile the application, run the command:

```
# /usr/local/arm-marvell-linux-gnu/bin/arm-marvell-linux-gnu-gcc
sysinfo.c -o sysinfo
```

You can also write a Makefile for it:

```
EXEC= sysinfo
OBSJ= sysinfo.o

CC= /usr/local/arm-marvell-linux-gnu/bin/arm-marvell-linux-gnu-gcc
LD= /usr/local/arm-marvell-linux-gnu/bin/arm-marvell-linux-gnu-ld
CFLAGS += -I/usr/local/arm-marvell-linux-gnu/include
LDFLAGS+=-L/usr/local/arm-marvell-linux-gnu/lib

all: $(EXEC)
```

```
$(EXEC) : $(OBJS)
    $(CC) $(CFLAGS) $(OBJS) -o $@ $(LDFLAGS)
clean:
    rm -rf *.o $(PROG) *.core
```

Compiling Open Source Projects

To compile an application on most open source projects, following are the three steps that you will be asked to execute:

1. configure
2. make
3. make install

The configure script basically consists of many lines which are used to check some details about the machine on which the software is going to be installed. This script checks for lots of dependencies on your system. When you run the configure script, you would see a lot of output on the screen, each being some sort of question and a respective yes/no as the reply. If any of the major requirements are missing on your system, the configure script would exit and you cannot proceed with the installation, until you get those required things. In most cases, to compile applications on some particular target machines requires you to modify the configure script manually so as to provide the correct values.

When running the configure script to configure software packages for cross compiling, you will need to specify the CC, LD, CFLAGS, host, target, and build, etc. Examples are given as below.

Marvell 5281 platform:

```
# env
CC=/usr/local/arm-marvell-linux-gnu/bin/arm-marvell-linux-gnu-gcc \
    LD=/usr/local/arm-marvell-linux-gnu/bin/arm-marvell-linux-gnu-ld \
    RANLIB=/usr/local/arm-marvell-linux-gnu/bin/arm-marvell-linux-gnu-ranlib \
    CFLAGS="-I/usr/local/arm-marvell-linux-gnu/include" \
    LDFLAGS="-L/usr/local/arm-marvell-linux-gnu/lib" \
    ./configure \
    --host=armle-unknown-linux \
    --target=armle-unknown-linux \
    --build=i686-pc-linux \
    --prefix=/usr/local
```

Power PC8241 platform:

```
# env CC=/usr/local/powerpc-linux/bin/powerpc-linux-gcc \
LD=/usr/local/powerpc-linux/bin/powerpc-linux-ld \
RANLIB=/usr/local/powerpc-linux/bin/powerpc-linux-ranlib \
CFLAGS="-I/usr/local/powerpc-linux/include" \
LDFLAGS="-L/usr/local/powerpc-linux/lib" \
./configure \
--host=powerpc-unknown-linux \
--target=powerpc-unknown-linux \
--build=i686-pc-linux \
--prefix=/usr/local
```

For PowerPC 8544/8533 platform:

```
# env CC=/usr/local/powerpc-linux-gnuspe/bin/powerpc-linux-gnuspe-gcc \
LD=/usr/local/powerpc-linux-gnuspe/bin/powerpc-linux-gnuspe-ld \
RANLIB=/usr/local/powerpc-linux-gnuspe/bin/powerpc-linux-gnuspe-ra \
nlib \
CFLAGS="-I/usr/local/powerpc-linux-gnuspe/include -mcpu=8548 \
-mhard-float -mfloat-gprs=double" \
LDFLAGS="-L/usr/local/powerpc-linux-gnuspe/lib" \
./configure \
--host=powerpc-unknown-linux \
--target=powerpc-unknown-linux \
--build=i686-pc-linux \
--prefix=/usr/local
```

For Marvell 6281 platform:

```
# env
CC=/usr/local/arm-none-linux-gnueabi/bin/arm-none-linux-gnueabi-gcc \
LD=/usr/local/arm-none-linux-gnueabi/bin/arm-none-linux-gnueabi-ld \
RANLIB=/usr/local/arm-none-linux-gnueabi/bin/arm-none-linux-gnueabi-ranlib \
CFLAGS="-I/usr/local/arm-none-linux-gnueabi/include" \
LDFLAGS="-L/usr/local/arm-none-linux-gnueabi/lib" \
./configure \
--host=armle-unknown-linux \
--target=armle-unknown-linux \
--build=i686-pc-linux \
--prefix=/usr/local
```

For Intel X86 platform:

```
# env CC=/usr/local/i686-linux-gnu/bin/i686-linux-gnu-gcc \
LD=/usr/local/i686-linux-gnu/bin/i686-linux-gnu-ld \
RANLIB=/usr/local/i686-linux-gnu/bin/i686-linux-gnu-ranlib \
CFLAGS="-I/usr/local/i686-linux-gnu/include" \
LDFLAGS="-L/usr/local/i686-linux-gnu/lib" \
./configure \
--host=i686-linux-gnu \
--target=i686-linux-gnu \
--build=i686-pc-linux \
--prefix=/usr/local
```

Compiling Kernel Module

If you would like to compile kernel modules, you will need to obtain Synology GPL to access the kernel source code. Please refer to <http://www.synology.com/enu/gpl/> for details.

In the kernel source code, there are different configuration files for different platforms. The configuration files used in each model are listed below:

Model	Configuration File	Arch	Linux
CS407, DS107+, DS207+, RS407	88f5281-config	ARM	2.6.15
DS109, DS209, DS409, DS409slim, DS411slim, RS409, DS110j, DS210j, DS410j, DS211j, DS411j	synoconfigs/88f6281	ARM	2.6.32
DS111, DS211, DS411, DS211+	synoconfigs/88f6281	ARM	2.6.32
DS209j	synoconfigs/ppc824x	PowerPC	2.6.24
DS108j, DS109j	powerpc-config	PowerPC	2.4
DS408, DS508, RS408, RS408-RP	synoconfigs/ppc854x	PowerPC	2.6.32
DS109+, DS209+, DS409+, RS409+, RS409RP+, DS509+	synoconfigs/ppc8533	PowerPC	2.6.32
DS110+, DS210+, DS410	synoconfigs/ppc8533	PowerPC	2.6.32
DS710+, DS712+, DS1010+, DS1511+, DS411+, RS810+, RS810RP+, DS411+II, DS2411+, RS2211+, RS2211RP+	synoconfigs/x86_64	x86	2.6.32
DS3611xs, RS3411xs, RS3411RPxs	synoconfigs/bromolow	x86	2.6.32

Please copy the proper configuration file to “.config”, and run “make oldconfig” and “make menuconfig” to select your kernel modules. Depending on the platform you would like to compile, you have to set proper ARCH and CROSS_COMPILE into environment variables.

```
For example, to compile 2.6 kernel modules for 5281 platform:

# cd linux-2.6.15
# cp 88f5182-config .config

# make ARCH=arm \
CROSS_COMPILE=/usr/local/arm-marvell-linux-gnu/bin/arm-marvell-linux-
-gnu- oldconfig

# make ARCH=arm \
CROSS_COMPILE=/usr/local/arm-marvell-linux-gnu/bin/arm-marvell-linux-
-gnu- menuconfig

# make ARCH=arm \
CROSS_COMPILE=/usr/local/arm-marvell-linux-gnu/bin/arm-marvell-linux-
-gnu- modules
```

For 2.4 kernel:

```
# cd uclinux2422/linux-2.4.x
# cp powerpc-config .config
# cp Makefile.powerpc Makefile

# make oldconfig

# make menuconfig (and choose your modules)

# make dep

# make modules
```

Installation

After you compile the open source package or your own application, the make install script will not install the application on the Synology DiskStation. Instead, it will install the application to your Linux PC. What you have to do is to copy the required files to the Synology DiskStation. Details are given in the following section.

Put the Application in /usr/local

Synology releases new DSM from time to time. It is important that you install your application in the correct directory so that it will not be deleted when DSM upgrades.

The directory **/usr/local** is reserved for 3rd-party application. It is recommended that you create a directory for your application and put related files in it. For example, you can create **/usr/local/myapp** and then create a directory **bin** in it for utilities, **sbin** for daemons and system utilities, **etc** for configuration files, and **lib** for your libraries. Once it is finished, you can copy the related files of your application to the respective directories. Please note that you may need to specify the correct prefix when running configure script, so that the application can find the correct path information upon execution.

When DSM is upgrading, the directory **/usr/local** will be backed up and restored automatically. However, some library files or built-in software packages may be modified during the upgrade procedure, so if your application is dependent on these files, it may not run afterward.

Run the Application When the System Boots

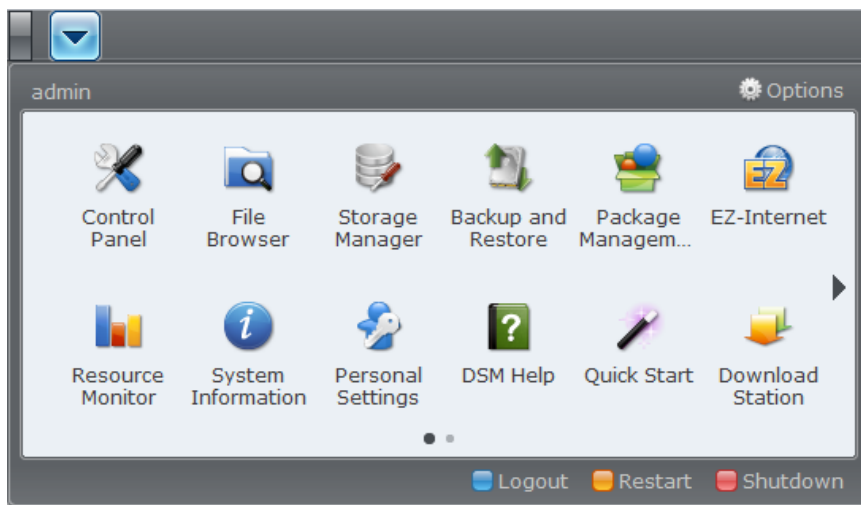
If you would like to run an application when the system boots up, you have to write a startup script and put it in **/usr/local/etc/rc.d/**. Following are some rules for the startup script:

1. It must have the suffix **“.sh”**. For example, **“myprog.sh”**.
2. The permission must be **755**.
3. It must have the options **“start”** and **“stop”**. When the system boots up, it will call **“myprog.sh start”**; when it shuts down, it will call **“myprog.sh stop”**.

You can refer to the scripts in **/usr/syno/etc/rc.d/**. They are scripts for Synology default services.

Integrating Applications into Synology DiskStation Manager 3.0

In Synology DSM 3.0, integrating 3rd-party applications into your DiskStation is even easier than before. When an application is integrated, its icon will automatically appear in the Main Menu of DSM 3.0. Users can also use customized icons for these applications. To place the icon on the desktop, you only have to drag it from the Main Menu to the desktop on DSM.



Startup

To integrate an application into Synology DiskStation Manager 3.0, follow the steps below:

1. Create a directory under the directory **`/usr/syno/synoman/webman/3rdparty/`**.
2. Create a text file named "**config**" under the directory.
3. Under the same directory, you can create a sub-directory named by your application, like **`/usr/syno/synoman/webman/3rdparty/myapp/`**, for example. You can put all UI related components, such as images, CSS, and CGI in the directory.

Next we will show the details of UI configuration.

config

The “**config**” is a text file to configure UI behavior. The content of “**config**” should be in **JSON** format. For example:

```
{
  ".url": {
    "com.company.App1": {
      "type": "url",
      "allUsers": true,
      "title": "Test App1",
      "desc": "Description",
      "icon": "images/app_{0}.png",
      "url": "http://www.yahoo.com"
    },
    "com.company.App2": {
      "type": "legacy",
      "allUsers": true,
      "title": "Test App2",
      "desc": "Description 2",
      "icon": "images/app2_{0}.png",
      "url": "http://www.synology.com"
    }
  }
}
```

Here are the details of the content of application.cfg:

Property	Description
com.company.App1 com.company.App2	In “.url”, each object should have a unique property name.
title (Required)	“title” represents the application name that will be displayed in the main menu.
desc	“desc” describes more details about this application upon mouse-over.

Property	Description
icon (Required)	<p>"icon" describes the icon for the application. It is a template string. The "{0}" can be replaced by "16", "24", "32", "48", depending on different pixels of the image file for the icon.</p> <p>The icon must be put under /usr/syno/synoman/webman/3rdparty/xxx/ where xxx is the directory name of your application.</p> <p>For example, if you create a directory named "images" and put the icon image file "icon.png" in it, the full path for the icon will be:</p> <p>/usr/syno/synoman/webman/3rdparty/xxx/images/icon_16.png /usr/syno/synoman/webman/3rdparty/xxx/images/icon_24.png /usr/syno/synoman/webman/3rdparty/xxx/images/icon_32.png /usr/syno/synoman/webman/3rdparty/xxx/images/icon_48.png</p> <p>And the icon value should be set as "images/icon_{0}.png".</p>
type (Required)	<p>When you click the menu item, the address you use to connect to the DSM management UI will be shown in the right frame of the management UI. However, you can customize the address as you wish.</p> <p>The "type" value can be "url" or "legacy". "url" means when you click the application icon, the URL will be opened in a pop-up window, while "legacy" implies that the URL will be opened in an iframe window application.</p> <p>You can follow the descriptions below to set up your customized URL.</p>
url (Required)	<p>This is an example of value setting for your URL of the application:</p> <pre data-bbox="730 1223 1206 1301">"url": "http://www.synology.com/" "url": "3rdparty/xxx/index.html"</pre>
allUsers	<p>This key determines whether the menu items can be seen by users when they log in with admin account. If you would like to have all users see the menu items, please set the key value as below:</p> <pre data-bbox="730 1458 962 1480">"allUsers": true</pre> <p>The default setting is that only the admin can find the application.</p>

Integrating Applications into Synology DiskStation Manager 2.3

Synology DiskStation Manager 2.3 can integrate 3rd-party applications into its new AJAX management UI as a menu item. You can also customize its icon and put it on the management desktop.

For example, the following picture shows a 3rd-party application named "MyApp". When it is clicked, Synology official website will be displayed in the right frame.



Startup

To integrate an application into Synology DiskStation Manager 2.3, follow the steps below:

1. Create a directory under the directory `/usr/syno/synoman/webman/3rdparty/`.
2. Create a text file named "**application.cfg**" under the directory. (Details will be given in the next section.)
3. Under the same directory, you can create a sub-directory named by your application, like `/usr/syno/synoman/webman/3rdparty/myapp/`, for example. You can put all UI related components, such as images, CSS, and CGI in the directory.

Next we will show the details of UI configuration.

Application.cfg

“**application.cfg**” is a text file for configuring UI behavior. Please note the encoding of “**application.cfg**” should be in UTF-8 format so that its descriptions can be shown correctly on the web interface.

The content of “**application.cfg**” should be in **key=value** format. For example:

```
text = My app
description = This is my menu item pointed to Synology WebSite
icon_16 = images/icon16.png
icon_32 = images/icon32.png
type = embedded
protocol = http
address = www.synology.com
port = 80
path = /index.php
```

Below lists more information about the contents of **application.cfg**:

Key	Description
text (required)	<p>“text” describes the menu item names shown in the menu tree.</p> <p>If you would like to localize the menu items, you can add a language abbreviation suffix to the file.</p> <p>For example:</p> <p style="padding-left: 40px;"><code>text_cht</code> → Menu item name in Traditional Chinese</p> <p style="padding-left: 40px;"><code>text_fre</code> → Menu item name in French</p> <p>The abbreviations for other languages (e.g. jpn, sve, spn...) can be found at /usr/syno/synoman/webman/texts/. If user uses a language that is not available here, "text" will be used by default. It is not recommended to create language abbreviations not included in the list because they may not be recognized by the system.</p>
description (required)	<p>“description” describes the details of the menu item. It will show upon mouse-over and in the complete function list. This string can also be localized by adding language abbreviation suffix to the file.</p> <p>For example:</p> <p style="padding-left: 40px;"><code>description_cht</code></p> <p style="padding-left: 40px;"><code>description_fre</code></p>

Key	Description
icon_16	<p>“icon_16” describes menu item icon at the size of 16x16 pixels and in .png format.</p> <p>The icon must be put under /usr/syno/synoman/webman/3rdparty/MyApp/.</p> <p>For example, if you would like to use the image “icon.png” as the icon of your application, you should set “icon_16” as "images/icon.png" in the application.cfg file.</p> <p>If you create a directory named "images" and put all related image files of the application in it, the full path for your icon file "icon.png" should be:</p> <p>/usr/syno/synoman/webman/3rdparty/xxx/images/icon.png</p>
icon_32	<p>This is the key for large icon at the size of 32x32 pixels in .png format. This icon is used in complete function list mode. Its value settings are the same as icon_16.</p>
type	<p>When you click the menu item, the address you use to connect to the DSM management UI will be shown in the right frame of the management UI. However, you can customize the address as you wish.</p> <p>“type” describes how the URL is shown. Its value can be "embedded" or "popup". "popup" means when you click the menu item, the URL will be opened in a pop-up window, while "embedded" implies that the URL will be opened in the right management UI frame.</p> <p>If the “type” is not specified, "popup" will be the default type.</p> <p>You can follow the descriptions below to set up your customized URL.</p>
protocol	<p>The value of protocol can be either "http" or "https", depending on the URL protocol you choose. If this value is not set, the current connection protocol will be used.</p> <p>protocol = http</p>
address	<p>This key describes the address of the URL you choose to link when you click the menu item. If it is left empty, the address will be the one users use to connect to the DSM management UI.</p>
port	<p>This is the port number of the URL. If this is not specified, the port users use to connect to the management UI will be used.</p>
path	<p>This describes the path where you put the URL files. The value should always start with a “/”.</p>
adminonly	<p>This key determines whether the menu items can be seen by users when they log in with admin account. If you would like to have all users see the menu items, please set the key value as below:</p> <p>adminonly = false</p> <p>The default setting is that only the admin can see the menu item.</p>

Example

Example 1: If you would like to add a menu item named "My Menu Item" and have a new window of <http://www.synology.com/index.php> popped up by clicking the item, follow the steps below:

1. Create the directory `/usr/syno/synoman/webman/3rdparty/my_menu_item`.
2. Put the icon file in `/usr/syno/synoman/webman/3rdparty/my_menu_item/images/`.
3. Put the "application.cfg" in `/usr/syno/synoman/webman/3rdparty/`. The contents of "application.cfg" should be like this:

```
text = My Menu Item
description = This is my menu item pointed to Synology Website
icon_16 = images/icon16.png
icon_32 = images/icon32.png
type = popup
protocol = http
address = www.synology.com
port = 80
path = /index.php
```

Example 2: If you would like to add a menu item named "Second Menu Item" and have DiskStation's Web Station (port 80) displayed by clicking the item, follow the steps below:

1. Create a directory `/usr/syno/synoman/webman/3rdparty/`.
2. Put the file "application.cfg" in it. The "application.cfg" should be like this:

```
text = Second Menu Item
description = Description of second menu item
type = embedded
protocol = http
port = 80
```

Here the "address" is not set, which means that the default URL (Synology official website) will be shown as embedded (in the right management UI frame). The icon is not specified either, so the default icon will be used instead.

Adding Your Customized Applications to Desktop Items

The Desktop page is the new feature introduced in Synology DSM 2.3. It allows users to access applications with just one click. It even allows users to add their customized 3rd-party application icons to it.



To add an application icon on the Desktop of Synology DSM 2.3, follow the steps below:

1. Create a directory for your application under the directory **/usr/syno/synoman/webman/3rdparty/**. For example, you can create a directory named "SqueezeCenter" for the application. The path should be like this:
/usr/syno/synoman/webman/3rdparty/SqueezeCenter
2. To add the icon of the application on the desktop, you should put a text file

named "**desktop.cfg**" and an image file of the icon under the directory. The icon will be thus customized.

Below describes the detailed contents of the **desktop.cfg** file.

Desktop.cfg

The "**desktop.cfg**" is a text file for configuring the desktop icon and application link. Please note the encoding of "**desktop.cfg**" should be in UTF-8 format so that its descriptions can be shown correctly on the web interface.

The content of "**desktop.cfg**" should be in **key=value** format. For example:

```
adminonly=false
text=SqueezeCenter
description=SqueezeCenter Music Player
protocol=http
path=/
port=9001
icon=images/desktop.png
icon_alt=images/desktop_alt.png
```

Here are the details of the content of application.cfg:

Key	Description
text (required)	<p>"text" describes the menu item names shown in the menu tree. It usually shows under the icon.</p> <p>If you would like to localize the menu items, you can add a language abbreviation suffix to the file.</p> <p>For example:</p> <p><code>text_cht</code> → Menu item name in Traditional Chinese</p> <p><code>text_fre</code> → Menu item name in French</p> <p>The abbreviations for other languages (e.g. jpn, sve, spn...) can be found at /usr/syno/synoman/webman/texts/. If user uses a language that is not available here, "text" will be used by default. It is not recommended to create language abbreviations not included in the list because they may not be recognized by the system.</p>
description (required)	<p>"description" describes the details of the application. It will show upon mouse-over and in the complete function list. This string can also be localized by adding language abbreviation suffix to the file.</p> <p>For example:</p> <p><code>description_cht</code></p> <p><code>description_fre</code></p>

Key	Description
icon (required)	<p>“icon” describes the application icon on the desktop. It should be an image file at the size of 117x87 pixels and in .png format.</p> <p>The icon must be put under <code>/usr/syno/synoman/webman/3rdparty/MyApp/</code>. You can customize MyApp as you wish.</p> <p>For example, if you would like to use the image “icon.png” as the icon of your application, you should set “icon_16” value as “images/icon.png” in the application.cfg file.</p> <p>If you create a directory named “images” and put all related image files of the application in it, the full path for your icon file “icon.png” should be:</p> <p><code>/usr/syno/synoman/webman/3rdparty/xxx/images/icon.png</code></p>
icon_alt	This key describes the alternative icon when mouse-over appears on the desktop icon. Its value settings are the same as “icon”.
protocol	<p>When you click the menu item, the address you use to connect to the DSM management UI will be shown in the right frame of the management UI. However, you can customize the address as you wish.</p> <p>The value of protocol can be either “http” or “https”, depending on the URL protocol you choose. If this value is not set, the current connection protocol will be used.</p> <p><code>protocol=http</code></p>
address	This key describes the address of the URL you choose to link when you click the application icon. If it is left empty, the address will be the address users use to connect to the DSM management UI.
port	This is the port number of the URL. If this is not specified, the port users use to connect to the management UI will be used.
path	This describes the path where you put the URL files. The value should always start with a “/”.
adminonly	<p>This key determines whether the desktop icons can be seen by users when they login with admin account. If you would like to have all users see the menu items, please set the key value as below:</p> <p><code>adminonly = false</code></p> <p>The default setting is that only the admin can see the menu item.</p>

Integrating Synology DiskStation with web authentication

After integrating your application into Synology DSM, you may want to perform an authentication check to ensure only logged-in users can access the page.

To check whether a user has logged in, run the command below in the CGI:

```
/usr/syno/synoman/webman/modules/authenticate.cgi
```

The “**authenticate.cgi**” will output the user name if the user has logged in. There will be no output if the user has not been authenticated.

Below is an example:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h>

/**
 * Check whether user is logged in.
 *
 * If user has logged in, put the username into "user".
 *
 * @param user   The buffer for get username
 * @param bufsize The buffer size of user
 *
 * @return 0: User not logged in or error
 *         1: User logged in. The user name is written to given "user"
 */
int IsUserLogin(char *user, int bufsize)
{
    FILE *fp = NULL;
    char buf[1024];
    int login = 0;

    bzero(user, bufsize);

    fp = popen("/usr/syno/synoman/webman/modules/authenticate.cgi",
"r");
    if (!fp) {
        return 0;
    }
    bzero(buf, sizeof(buf));
    fread(buf, 1024, 1, fp);

    if (strlen(buf) > 0) {
        snprintf(user, bufsize, "%s", buf);
        login = 1;
    }
    pclose(fp);
}
```

```
        return login;
    }

int main(int argc, char **argv)
{
    char user[256];

    printf("Content-type: text/html\r\n\r\n");
    if (IsUserLogin(user, sizeof(user)) == 1) {
        printf("User is authenticated. Name: %s\n", user);
    } else {
        printf("User is not authenticated.\n");
    }
    return 0;
}
```

This CGI runs the command below to check whether a user has logged in. After checking up, it will print out the user's name if the user has logged in.

```
/usr/syno/synoman/webman/modules/authenticate.cgi
```

Creating a Synology Package

Package Introduction

Synology Package Center in DSM automates the process of installing, upgrading, configuring, and uninstalling packages in DSM. In Synology Package Center, the developer defines some scripts and metadata to control the installation, uninstalling, and upgrading processes as well as to communicate with DSM.

Package Structure

The Synology package is a .spk file in tar format, containing metadata and files as the following:

File/Folder Name	Description	File/Folder Type
INFO	This file contains the information displayed in Package Center.	File
WIZARD_UIFILES	Optional. This folder contains files in which descriptions of UI components are shown during the installation, uninstalling, and upgrading processes.	Folder (Contains install_uifile, upgrade, uifile, uninstall_uifile)
package.tgz	This is a compressed file in .tgz format containing all the files that are required, such as executable binary, library, or UI files.	.tgz File
scripts	This folder contains shell scripts which are executed during the installation, uninstalling, upgrading, start, and stop processes.	Folder (Contains preinst, postinst, preuninst, postunist, preupgrade, postupgrade, start-stop-status)
LICENSE	Optional. This file is shown in the installation process, and must be less than 1MB.	File

File/Folder Name	Description	File/Folder Type
PACKAGE_ICON.PNG	Optional. 72 x 72 .png image is shown in Package Center	.png file

Note:

1. All words are case sensitive.
2. You can use "tar -cvf package.spk [files]" to create the package.

INFO

INFO file is used to describe the information of the package. Package Center will search for information to control the flow of installation, upgrading, uninstalling, start, stop processes and listing in Package Center. For example, if you would like the installation package to be dependent on some services, you can define the key as **install_dep_services**; if you would like to restart some services after the installation, you can define the key as **instuninst_restart_services**.

There are some keys configured in INFO:

Key	Description	Value	Default Value
package	Package name. No more than one version of a package can exist at the same time; therefore, the name is unique. If displayname key is empty, Package Center will show "package='xxx'" where xxx is the name of the package. For example, package="Time Backup". Note: This key cannot contain any of these special characters: , /, >, < or =.	String	(Empty)
version	Package version. End users can identify the package version, e.g. version="7.2.1".		(Empty)
description	Package Center shows a short description of the package, e.g. description = "Time Backup is an innovative solution that backs up DiskStation data in multiple versions. You could intuitively browse among versions and easily restore data to any specific time."	String	(Empty)
description_[DSM language]	Optional. Package Center shows a short description in the DSM language set by the end-user.	String	description
displayname	Optional. Package Center shows the name of the package.	String	package name

Key	Description	Value	Default Value
displayname_[DSM language]	Optional. Package Center shows the name in the DSM language set by the end-user.	String	package name
maintainer	Package Center shows the developer of the package, e.g. maintainer="Synology Inc."	String	(Empty)
arch	List the CPU architectures which can be used to install the package, e.g. arch="noarch" or arch = "88f5281 powerpc ppc824x". Reference: http://forum.synology.com/wiki/index.php/What_kind_of_CPU_does_my_NAS_have	88f5281, powerpc, ppc824x, ppc854x, ixp420, ppc853x, 88f6281, 88f6282, x86, bromolow, noarch (Separated with a space)	noarch
checksum	Optional. Contain MD5 string to verify the package.tgz.	String	(Empty)
adminport	Optional. A package listens on a specific port to display its own management UI. If the package is defined by a port, e.g. adminport="9002", a link will be opened when the package is activated, e.g. adminprotocol://ip:adminport/adminurl .	0~65536	80
adminurl	Optional. If a package is installed and has an "administration" webpage, a link will be opened when the package is activated, e.g. adminprotocol://ip:adminport/adminurl .	String	(Empty)
adminprotocol	Optional. For example: adminprotocol://ip:adminport/adminurl	http / https (Separated with a space)	http
firmware	Optional. Minimum version of DSM firmware that is required to run the package.	X.Y-Z DSM major number, DSM minor number, DSM build number	(Empty)
dsmuidir	Optional. DSM UI folder name in package.tgz. The folder in /volumeX/@appstore/[package name] / [dsmuidir] will be automatically linked to /usr/syno/synoman/webman/3rdparty/[package name] after the start-stop-status script with the start argument is run and the file mode bits is changed to 777. To remove the link, run the start-stop-status script with the stop argument. Note: This key cannot contain : or /.	String	(Empty)

Key	Description	Value	Default Value
checkport	Optional. Check if there is any conflict between the adminport and the ports which are reserved or are listening on DSM except web-service ports (e.g. 80, 443) and DSM ports (e.g. 5000, 5001).	"yes"/"no"	"yes"
startable	Optional. When no program in the package provides the end-user with the options to enable or disable its function, this key is set to "no" and the end-user cannot start or stop the package in Package Center. Note: If "startable" is set to "no", the start-stop-status script is still required.	"yes"/"no"	"yes"
helpurl	Optional. If a package is installed and has a "help" webpage, Package Center will show you the link, e.g. http://www.synology.com/enu/apps/3rd-party_application_integration.php .	String	(Empty)
report_url	Optional. If a package is installed and has a "report" webpage, Package Center will show you the link. This package is considered the first beta version, and the beta information will be shown in Package Center.	String	(Empty)
package_icon	72x72 png image data is encoded by Base64. Note: 1. This value will be replaced when a PACKAGE_ICON.PNG file is stored in the [package name].spk. 2. If the value is not defined and no PACKAGE_ICON.PNG file is in the [package name].spk, the package icon is a default one.		a default icon data
install_reboot	Optional. Reboot DS after installing or upgrading the package. Note: If the value is set to "yes", the value of instuninst_restart_services is ignored.	"yes"/"no"	"no"
install_dep_packages	Optional. Before a package is installed or upgraded, these packages must be installed first. The format consists of the package name followed by one of the special characters =, <, >, and firmware version which is composed by number and periods, e.g. install_dep_packages = "packageA>2.2.2". However, if more than one package is required of the format, the package name of the package(s) succeeding the firmware version will be separated with a colon, e.g. install_dep_packages = "packageA>2.2.2:packageB".	Package names (Separated with a colon)	(Empty)

Key	Description	Value	Default Value
instuninst_restart_services	Optional. Restart services after installing, upgrading and uninstalling the package. Note: 1. If the service is not enabled or started by the end-user, it cannot be restarted. 2. If the install_reboot is set to “yes”, this value is ignored in the installation process.	apache-sys, apache-web, mdns, samba, db, applenetwork, cron, nfs, firewall (Separated with a space; see Note 4)	(Empty)
startstop_restart_services	Optional. Restart services after starting and stopping the package. Note: 1. If the service is not enabled or started by the end-user, it cannot be restarted. 2. If startable is set to “no”, the value is ignored.	apache-sys, apache-web, mdns , samba, db , applenetwork, cron, nfs, firewall (Separated with a space; see Note 4)	(Empty)
install_dep_services	Optional. Before the package is installed or upgraded, these services must be started or enabled by the end-user.	apache-web, mysql, php_disable_safe_exe_dir (Separated with a space)	(Empty)
start_dep_services	Optional. Before the package is started, these services must be started or enabled by the end-user. If startable is set to “no”, this value is ignored.	apache-web, mysql, php_disable_safe_exe_dir (Separated with a space)	(Empty)
dsmappname	Optional. Before the package is upgraded or uninstalled, the app windows will be closed.	(Separated with a space)	(Empty)

Note:

- [DSM language]: DSM supports the following languages --- enu, cht, chs, krn, ger, fre, ita, spn, jpn, dan, nor, sve, nld, rus, plk, ptb, ptg, hun, trk, csy.
- All words are case insensitive.
- apache-sys** is an apache daemon listening on DSM ports (e.g. 5000 or 5001).
apache-web is an apache daemon listening on Web Station ports (e.g. 80 or 443).
- Code words of value:
 - mdns = Multicast DNS Service Discovery
 - db = MySQL and PostgreSQL
 - apple network = Apple Network
 - nfs = NFS

WIZARD_UIFILES

install_uifile, **upgrade_uifile**, and **uninstall_uifile** are the files which describe UI components in JSON format, and are stored in the “WIZARD_UIFILES” folder. During the installation, upgrading and uninstalling processes, these UI components will appear in the wizard. Once these components are selected, their keys will be set in the script environment variables, and their values are “true”, “false”, or text values.

These files can be regarded as user settings or can be used to control the flow of script execution.

- **install_uifile**: Describe some UI components for the installation process. During the running of the **preinst** and **postinst** scripts, these component keys and values can be found in the environment variables.
- **upgrade_uifile**: Describe some UI components for the upgrading process. During the running of the **preupgrade**, **postupgrade**, **preinst** and **postinst** scripts, these component keys and values can be found in the environment variables.
- **uninstall_uifile**: Describe some UI components for the uninstalling process. During the running of the **preuninstall** and **postuninstall** scripts, these component keys and values can be found in the environment variables.

Note:

If you would like to localize the descriptions of UI components, you can add a language abbreviation suffix to the file “install_uifile_[DSM language]”, “upgrade_uifile_[DSM language]” or “uninstall_uifile_[DSM language]” in this folder. For example, in order to perform installation in traditional Chinese, [DSM language] should be replaced with “cht” like “install_uifile_cht”.

Example for the file in JSON format:

```
[{
  "step_title": "Step1",
  "items": [{
    "type": "singleselect",
    "desc": "a radio group",
    "subitems": [{
      "key": "radio1",
      "desc": "Radio button 1"
    }, {
      "key": "radio2",
      "desc": "Radio button 2"
    }
  ]
}], {
  "step_title": "Step2",
  "items": [{
    "type": "multiselect",
    "desc": "a check group",
    "subitems": [{
      "key": "check1",
      "desc": "Check button 1"
    }, {
      "key": "check2",
      "desc": "Check button 2"
    }
  ]
}], {
  "type": "textfield",
  "desc": "textfield",
  "subitems": [{
```

```

        "key": "textfield1",
        "desc": "textfield 1"
    }
  }
}

```

Here are the details of file content in JSON format:

Property	Description
step_title	Optional. Describe the title of the step currently performed in wizard.
items	Describe an array containing the components of "singleselect", "multiselect", "textfield", or "password" type.
type	<p>Must be "singleselect", "multiselect", "textfield" or "password".</p> <ul style="list-style-type: none"> ▪ "singleselect" type represents the components in the subitems which are all radio buttons. End-users can select only one radio box with a unique key. ▪ "multiselect" type represents the components in the subitems which are all checkboxes. End-users can check more than one checkboxes. ▪ "textfiled" type represents the components in the subitems which are all text fields. End-users can type texts. ▪ "password" type represents the components in the subitems which are all password fields. End-users can type password.
desc	Optional. Describe a component in the label text.
subitems	Describe an array containing radio buttons, checkboxes, text fields or password components.
key	A unique component key value represents a UI component. If a component is selected by the end-user, this key will be set in the script environment variables (the string value of the selected checkbox or radio button is always "true").

Note: All words are case sensitive.

package.tgz

This is a compressed file containing all files such as executable files, library, and UI files, and will be saved in the directory "@appstore" in the installed volume once uncompressed. A package can be created with the command: "tar czf package.tgz [files]."

Scripts

There are seven script files stored in the “scripts” folder.

- **start-stop-status:** This script is used to start and stop a package, detect running status, and generate the log file. It will be copied to `/usr/local/etc/rc.d/[package name].sh` once the package is started. Parameters used by the script are listed below:

1. **start:** When the user clicks the button "Run" to run the package, after the package is installed, or when the DS is turned on, the Package Center program will call this script with "start" parameter, and a returned value will be acquired along the way.

For nonzero returned values, you can compile error messages in the **SYNOPKG_TEMP_LOGFILE** file to prompt the user. You can also write messages in the **SYNOPKG_TEMP_LOGFILE** file for zero returned values which represent that the process was successful.

2. **stop:** When the user clicks the button "Stop" to stop the running package, before the package is uninstalled, or when the DS is turned off, the Package Center program will call this script with "stop" parameter, and a returned value will be acquired along the way.

For nonzero returned values, you can compile error messages in the **SYNOPKG_TEMP_LOGFILE** file to prompt the user. You can also write messages in the **SYNOPKG_TEMP_LOGFILE** file for zero returned values which represent that the process was successful.

3. **status:** When Package Center AP is opened to check package status, the Center will send a request to ask the status of the package with this parameter. If an error occurs or the package is stopped, a nonzero value whose error message can be compiled in the **SYNOPKG_TEMP_LOGFILE** file to prompt the user will return.

4. **log:** When a log page is opened in Package Center, the Center will send a request to ask the log of the package with this parameter. When the log filename is sent to STDOUT, the content of the log file will be displayed.

- **preinst:** This script is run before the package files are transferred to @appstore. You can check if the installation requirements meet the DSM or package version, or if some services are enabled in this script.

For nonzero returned values, you can compile error messages in the **SYNOPKG_TEMP_LOGFILE** file to prompt the user. You can also write messages in the **SYNOPKG_TEMP_LOGFILE** file for zero returned values which represent that the process was successful.

- **postinst:** This script is run after the package files are transferred to @appstore. You can change the file permission and ownership in this script.

For nonzero returned values, you can compile error messages in the **SYNOPKG_TEMP_LOGFILE** file to prompt the user. You can also write messages in the **SYNOPKG_TEMP_LOGFILE** file for zero returned values which represent that the process was successful.

- **preuninst:** This script is run before the package is removed.
For nonzero returned values, you can compile error messages in the **SYNOPKG_TEMP_LOGFILE** file to prompt the user. You can also write messages in the **SYNOPKG_TEMP_LOGFILE** file for zero returned values which represent that the process was successful.
- **postuninst:** This script is run after the package is removed from the system.
For nonzero returned values, you can compile error messages in the **SYNOPKG_TEMP_LOGFILE** file to prompt the user. You can also write messages in the **SYNOPKG_TEMP_LOGFILE** file for zero returned values which represent that the process was successful.
- **preupgrade:** When you upgrade a package, the Package Center program calls this script before uninstalling the old one.
For nonzero returned values, you can compile error messages in the **SYNOPKG_TEMP_LOGFILE** file to prompt the user. You can also write messages in the **SYNOPKG_TEMP_LOGFILE** file for zero returned values which represent that the process was successful.
- **postupgrade:** When you upgrade a package, the Package Center program calls this script after installing the new one.
For nonzero returned values, you can compile error messages in the **SYNOPKG_TEMP_LOGFILE** file to prompt the user. You can also write messages in the **SYNOPKG_TEMP_LOGFILE** file for zero returned values which represent that the process was successful.

Script Environment Variables

Several variables are exported by Package Center and can be used in the scripts. Descriptions of the variables are given as below:

- **SYNOPKG_PKGNAME:** Package name which is defined in INFO.
- **SYNOPKG_PKGVER:** Package version which is defined in INFO.
- **SYNOPKG_PKGDEST:** Target directory in which the package is stored.
- **SYNOPKG_PKGPORT:** Administrator port which is defined in INFO. Packages listen on a specific port to use the management UI.
- **SYNOPKG_PKGINST_TEMP_DIR:** Packages are extracted to a temporary directory whose path is described by this variable.
- **SYNOPKG_TEMP_LOGFILE:** Package Center randomly generates a filename for a script to log the information or error messages.
- **SYNOPKG_DSM_LANGUAGE:** End-user's DSM language
- **SYNOPKG_DSM_VERSION_MAJOR:** End-user's major number of DSM version which is formatted in [DSM major number].[DSM minor number]-[DSM build number].
- **SYNOPKG_DSM_VERSION_MINOR:** End-user's minor number of DSM version which is formatted in [DSM major number].[DSM minor number]-[DSM build number].

- **SYNOPKG_DSM_VERSION_BUILD**: End-user's DSM build number of DSM version which is formatted in [DSM major number].[DSM minor number]-[DSM build number].
- **SYNOPKG_DSM_ARCH**: End-user's DSM CPU architecture. Reference: http://forum.synology.com/wiki/index.php/What_kind_of_CPU_does_my_NAS_have
- **SYNOPKG_PKG_STATUS**: Three different statuses of package processing are represented by these values: INSTALL, UPGRADE, UNINSTALL.
- **SYNOPKG_OLD_PKGVER**: Existing package version which is defined in INFO (only in **preupgrade** script).

Once the end-user enters or selects some values of the UI components which are configured in **install_uifile/upgrade_uifile/uninstall_uifile** (please refer to **WIZARD_UIFILES** on pages 31-32), the names and values of the components will be set in the environment variables, but please note that the names of these components cannot be the same as those of the environment variables.

Document Revision History

This table describes the changes to the Synology DiskStation Manager 3rd-Party Apps Developer Guide.

Date	Note
2008-06-16	1. Document originally released
2009-02-09	1. Add Freescale 8533 tool chain information
2009-03-	1. Add Marvell 6281 tool chain information 2. Add Desktop Icon chapter
2010-05-20	1. Add related information for 10 models 2. Change all DSM 2.0 & 2.1 to DSM 2.3 3. New screenshots for desktop icon & DSM application 4. Change configuration files and tool chains 5. DS1010+ uses Intel Atom D510
2010-05-28	1. Fix typo 2. Revise Tool Chain description 3. Revise 88f5182-config to 88f5281-config in Kernel Module 4. Revise path description of application.cfg / desktop.cfg
2010-11-29	1. Rename the document to “Synology DiskStation Manager 3rd-Party Apps Developer Guide” 2. Add DSM 3.0 Integration section
2011-10-31	1. Add Synology package creation section 2. Update DSM tool chain information
2011-12-15	1. Typo fixes