

Synology NAS Server

3rd-Party Apps Integration Guide

Synology®

2009-02-09



Synology Inc.
© 2009 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.
6F-2, 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 to the Synology NAS Server 3rd-Party Apps Integration Guide	4
Who Should Read This Document?	4
System Requirement	4
Compiling an Application	5
Downloading toolchain.....	5
Compiling	6
Compiling open source projects	7
Compiling kernel module	8
Installation.....	10
Put the application in /usr/local.....	10
Run the application when the system boots	10
Integrating into the Synology Disk Station Manager 2.0	12
Overview	12
Application.cfg.....	13
Example	14
Integrating with Synology web authentication.....	16
Document Revision History.....	18

Introduction to the Synology NAS Server 3rd-Party Apps Integration Guide

The Synology NAS Server (Disk Station series, Cube Station series, and Rack Station series) uses Linux as its operating system. Due to frequent requests from Synology users and system integrators to install additional applications on the Synology NAS Server, Synology has put together this development guide in order to help users install 3rd-party applications on their Synology NAS Server.

With this integration guide, you will learn how to:

1. Compile applications to be run on the Synology NAS Server
2. Install the applications into proper paths in order to keep the applications intact after a firmware upgrade.
3. Integrate the applications into Synology's AJAX management UI, aka Synology Disk Station Manager 2.0.
4. Integrate the applications into the Synology web authentication interface.

Who Should Read This Document?

This document is for Synology users and system integrators interested in adding their applications to their Synology NAS Server.

This document assumes you have a basic understanding of the Linux environment. Many resources exist in print and on the web for learning about Linux application development. If you are new to this, you may want to do learn some basics before starting to use this document.

System Requirement

The Synology NAS Server must have firmware version DSM 2.0-0636 or above.

Compiling an Application

The Synology NAS Server uses an embedded SoC as the CPU, which has a different architecture from x86-based PCs. There are two platforms, ARM and PowerPC, available for the different Synology NAS Server models. In order to run applications on the Synology NAS Server, it will be necessary to first 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 NAS Server model. The information will help you determine the correct way to proceed.

Model	CPU	Arch	Endianness	Linux
CS/RS407, DS207+, DS107+	Marvell 5281	ARM	Little Endian	2.6
CS407e, DS207	Freescale 8241	PowerPC	Big Endian	2.6
DS107e, DS107, DS108j	Freescale 8241	PowerPC	Big Endian	2.4
DS508, RS408, RS408-RP	Freescale 8543	PowerPC	Big Endian	2.6
DS209+	Freescale 8533	PowerPC	Big Endian	2.6

To compile an application for the Synology NAS Server, a compiler that runs on a Linux PC is needed in order to generate an executable file for the Synology NAS Server. The process is called “cross compiling”, and the set of compiling tools (compiler, linker, etc) is called “toolchain”.

Downloading toolchain

To download toolchain, go to <http://sourceforge.net/projects/dsgpl/>. The table below shows the toolchain file name for each model:

Model	Toolchain
CS/RS407, DS207+, DS107+	gcc343_glibc232_88f5281.tar.gz

CS407e, DS207	gcc334_glibc233_ppc_2.6.tar.gz
DS107e, DS107, DS108j	gcc334_glibc233_ppc_2.4.tar.gz
DS508, RS408, RS408-RP	gcc343_glibc234_854x.tar.gz
DS209+	gcc343_glibc234_854x.tar.gz

After you download the toolchain, un-tar it into **/usr/local/** using the following command:

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

Please make sure the toolchain is put into the **/usr/local** directory to ensure proper integration.

Compiling

Now you can start to compile the application. For example, if you have an application called "sysinfo.c", with the content below:

```
#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, 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

Most open source projects need the following steps to compile the application:

5. configure
6. make
7. make install

When running “configure” to configure the software package for cross compiling, you will need to specify the CC, LD, CFLAGS, host, target, and build, etc.

For example, for 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

```

For 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-ranlib \
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
```

Please note that most projects require that you modify the “configure” when cross compiling because it cannot run a test program to determine the capability of the target machine. For example, it cannot run a test program to get the size of pointers, whether the system has the device **/dev/random** or not. When “configure” fails, you need to modify the configuration file and provide the correct value manually.

Compiling kernel module

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

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

CPU	Configure file	ARCH	Kernel
CS/RS407, DS207+, DS107+	88f5182-config	ARM	2.6
CS407e, DS207	ppc824x-config	PowerPC	2.6

DS107e, DS107, DS108j	powerpc-config	PowerPC	2.4
DS508, RS408, RS408-RP	synoconfigs/ppc854x	PowerPC	2.6
DS209+	synoconfigs/ppc8533	PowerPC	2.6

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 want to compile, you have to set the 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 compiling the open source packages or your own application, the “make install” will not install the application on the Synology NAS Server, instead, it will install the application to your Linux PC. What you have to do is to copy the needed files to the Synology NAS Server. Details are described in the following section.

Put the application in `/usr/local`

Synology releases new firmware from time to time. It is important that you install your application in the correct directory so the software will not be deleted after firmware upgrade.

The `/usr/local` is reserved for third-party application. It is recommended that you create a directory for your application, and then put everything in it, for example, you can first create `/usr/local/myapp`, and then create a directory `bin` in it to put the utilities, `sbin` for daemons and system utilities, `etc` for configuration files, and `lib` for your libraries. Then, copy your applications into the proper directory. Please note that you might need to specify the correct prefix when running “configure”, so the application can find the correct path information upon execution.

When doing firmware upgrades, the `/usr/local` will be backed up and restored. However, because Synology might upgrade the libraries or built-in software packages, if your application depends on the built-in libraries or utilities, there is no guarantee it can still run after an upgrade.

Run the application when the system boots

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

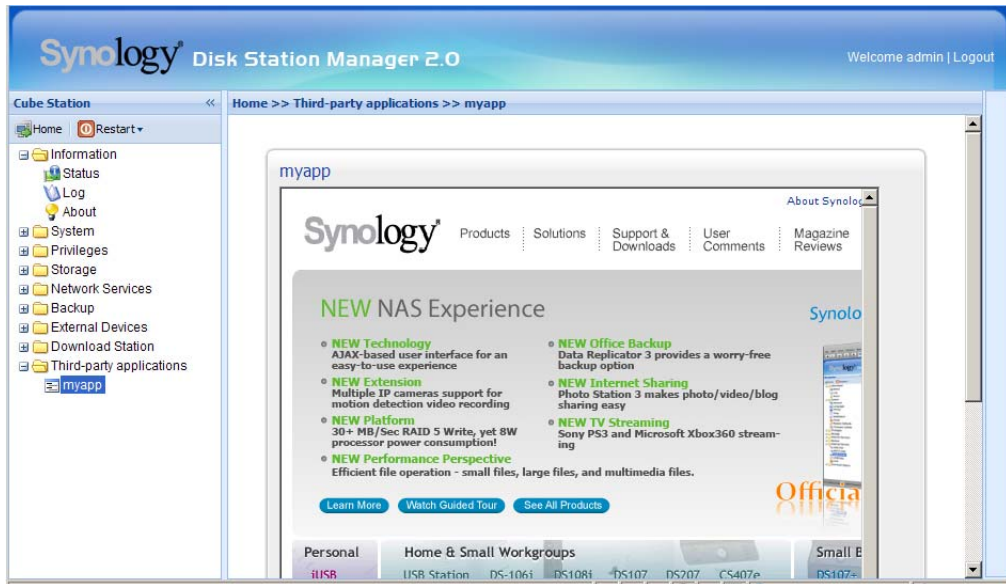
1. It must have suffix “.sh”. For example, “**myprog.sh**”.
2. The permission must be 755.
3. It must take the options “start” and “stop”. When the system boots up, it will call “**myprog.sh start**”; when system shuts down, it will call “**myprog.sh stop**”.

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

Integrating into the Synology Disk Station Manager 2.0

The Synology Disk Station Manager 2.0 includes an official mechanism to integrate third-party applications into the new AJAX management UI as a menu item.

For example, the following picture shows a third-party application named “myapp”. When clicking on it, the Synology website will be displayed in the right frame.



Overview

To integrate an application into the Synology Disk Station Manager 2.0, you have to:

1. Create a directory in the `/usr/syno/synoman/webman/3rdparty/`.
2. Put a text file named "**application.cfg**" under the created directory.

For example, you can create a directory named “myapp” in the directory:

`/usr/syno/synoman/webman/3rdparty/`

After that, you can put other UI related components, such as images, CSS, and CGI, in the path:

/usr/syno/synoman/webman/3rdparty/myapp/

Application.cfg

The “**application.cfg**” is a text file to configure the UI behavior. The content of “**application.cfg**” is in the **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
```

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

Key	Description
Text (required)	<p>“Text” is the menu item name that will be shown in the menu tree.</p> <p>If you would like to do language localization, you can add a language abbreviation as a suffix. For example,</p> <pre>text_cht = Menu item name for Chinese language text_fre = Menu item name for French</pre> <p>The naming of other abbreviations (e.g. jpn, sve, spn, ...) can be found at /usr/syno/synoman/webman/texts/. If user uses a language that is not available here, the "text" will be used by default. Therefore, it is recommended that you set the "text" as default for other languages.</p>
Description (required)	<p>“Description” is the description of this menu item that will show upon mouse-over events, and in the complete function list. This string can be localized, too.</p> <p>For example:</p> <pre>description_cht = ... description_fre = ...</pre>
icon_16	<p>“icon_16” is the icon path for the menu item icon. It is a 16x16 (pixel) .PNG file.</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 "icon.png" in it, the full path of the icon would be:</p> <pre>/usr/syno/synoman/webman/3rdparty/xxx/images/icon.png</pre>

	And the icon_16 value should be set to " images/icon.png " (omit the part /usr/syno/synoman/webman/3rdparty/xxx/).
icon_32	The large icon path. It is a 32x32 .PNG file. This icon will be used in complete function list mode. The rule for icon_16 also applies for icon_32.
type	The "type" value can be " embedded " or " popup ". " popup " means that when clicking on the menu item, a new window will pop up, while " embedded " means that the content of the URL is opened in the right management UI frame just like other Synology functions do. If the "type" is not specified, the default will be " popup ".
protocol	The "protocol", "address", "port", and "path" are used to set the URL of the menu item: (protocol://address:port/path). For example: protocol=http The value of protocol can be "http" or "https". This is the URL protocol you would like to use. If this value is not set, the current connection protocol will be set.
address	This is the address of the URL (protocol://address:port/path). If it is empty, the current URL will be used, which means the address will be the same as the address users use to connect to the management UI.
port	This is the port number of the URL (protocol://address:port/path). If this is not set, the port that users use to connect to the management UI will be used.
path	This is the path part of the URL. (protocol://address:port/path)
adminonly	This decides if the tree menu items are only shown when the admin logs in. If you would like normal users to see the menu item, please add: adminonly=false The default is that only the admin can see the menu item.

Please note the encoding of "**application.cfg**" should be in UTF-8 so the text and description can be showed correctly on the web interface.

Example

Example 1: If you want to add a menu item named "My Menu Item". When clicking on it, it pops up a new window to <http://www.synology.com/index.php>.

You need to do these steps:

1. Create the directory `/usr/syno/synoman/webman/3rdparty/my_menu_item`
2. Put the image 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” will look like:

```
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 want to add a menu item named "Second Menu Item". When clicking on the menu item, it shows the Synology Web Station (port 80) of the Synology NAS Server.

For this case, you need to:

1. Create a directory /usr/syno/synoman/webman/3rdparty/
2. Put the file “**application.cfg**” in it. The “**application.cfg**” will look like:

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

Note that the “address” is not set, and the icon is not specified so it will show the default icons.

Integrating with Synology web authentication

After integrating your application into the Synology Disk Station Manager 2.0, you might want to do an authentication check to ensure only logged in users can access the page.

To check whether a user has logged in, you can run the command

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

in the CGI. The “**authenticate.cgi**” will output the user name if the user has logged in. However, there will not be any output if the user has not been authenticated.

Below is an example CGI:

```
#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;
    }
}
```

```
    fclose(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

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

to check whether a user is logged in. Then it will print out the user name if the user has logged in.

Document Revision History

This table describes the changes to the *Synology NAS Server 3rd-Party Apps Integration Guide*.

Date	Note
2008-06-16	Document originally released
2009-02-09	Add Freescale 8533 toolchain information.