Q: Where should I install a new service (regular, filter or printer)?

A: Release 3 adds new flexibility in providing services. The format of the service definition remains the same, but new places are searched for services.

1) Your app wrapper can contain a service description file named **services**. (e.g. ~/Apps/MyApp.app/services) To include it in your ProjectBuilder project, just drop it into the *Other Resources* briefcase in the files viewer of ProjectBuilder.

   This is the preferred way to provide a service. Even lightweight services providers should have a small NEXTSTEP interface which at least allows the user to get information, help, and possibly set preferences for the service.

2) The service registration process also recognizes files and directories with a **.service** extension which

are installed in one of the newly established **~/Library/Services** or **/LocalLibrary/Services** directories. A file with a **.service** extension must be a valid service description file. A **.service** directory must contain a valid service description file named **services**. (e.g. the service description file would be named something like ~/Library/Services/myOwn.service or /LocalLibrary/Services/ALocalOne.service/services.) This is the least preferred way because it doesn't allow the for a NEXTSTEP interface to the service. Also, the Workspace Manager only registers these services when the user logs in, it doesn't automatically update the service cache during a working session.

The proper format for a service description file is described in the documentation. Here is an example for a fictitious application called **MyApp**. It owns the **.mine** extension and has a **mine-ascii** filter which is installed in the **MyApp.app** directory. Here is the filter service description file which is also in the **MyApp.app** directory:

Filter:
Port: NXUNIXSTDIO
Send Type: NXTypedFilenamePboardType:mine
Return Type: NXAsciiPboardType
Executable: mine-ascii

Note that you don't need to specify the full pathname of the executable if it is in the same directory as the service description file.

For backward compatibility, the services registration process continues to look for the __services section of the __ICON segment in Mach-O files that are installed in your application path.

Note that if you convert a 2.0 IB.proj file file to a 3.0 PB.project file you will have probably want to make two changes so that your services file will conform to the new preferred protocol for installing services:

- Delete the line in your Makefile.preamble which builds the services file into your Mach-O. It looks something like "LDFLAGS=-sectcreate __ICON __services services".
- Make sure that the service specification file is called **service**s and move it from the *Supporting Files* briefcase to the *Other Resources* briefcase. This will ensure that it gets copied into the app wrapper when you install.

Q: How do I register my service dynamically while I am debugging it?

A: The Workspace Manager dynamically registers or reregisters any services described in an application wrapper that is dragged into one of the directories in your application path (e.g. ~/Apps, /LocalApps). Note that the Workspace does not notice if you just drop a new **services** file into your app wrapper, you must install the whole application.

If you add the service using some other method, you can rebuild the services cache in one of two ways.

1) You can call the function **NXUpdateDynamicServices(void)** to register services which you add programmatically. Note that this function spawns a task in parallel that may take a short time to complete, so do not expect the services to be immediately available.

2) You can rebuild the services cache manually. This is useful if you are editing the services file in place and don't want to install a new copy of the application each time you make a change, or if you install a **.service** in one of the Services directories, or if you install the service using ProjectBuilder or the command line interface to UNIX. To rebuild the services cache you can run the **make_services** program from a Terminal window manually or log in and out so the Workspace will rebuild the services cache.

Q: How can I tell which services have been installed?

A: For regular services that are installed properly in an application wrapper, you can simply bring the Services menu of any application on screen. It rebuilds itself if needed before it is redisplayed. You no longer have to quit an application to take advantage of a new service. But remember to be patient, it takes a short time to reregister all of the services. So you might toggle the Services menu of an application that can take advantage of the particular data type until your new service appears.

If you are installing a filter service, it's difficult to know if it's really installed properly because there are no user interface elements which display it. But you can use the **make_services** program to explicitly update all of the services. It has a debugging mode which prints out the service information as it is processed. Run it from a Terminal window:

```
make_services debug
```

Alternately, this bit of codes prints the list of services that are available for a given filename:

```
- printTypesForFile:(const char *)filename
{
        const NXAtom       *listOfTypes;
        static Pasteboard *pb = nil;

        if (pb) pb = [pb free];
        pb = [Pasteboard newByFilteringFile:filename];
        if (!(listOfTypes = [pb types]) || !listOfTypes[0])
                return (const NXAtom *) 0;

        [self clearText];      // I've got a little method that just clears the
                                        // text object used to display the debugging info
        for (; listOfTypes && *listOfTypes; listOfTypes++) {
```

```
            [self appendText: *listOfTypes];// and a method to append to it
            [self appendText:"\n"];
        }
        return self;
}
```

Q: What if there is more than one filter service provider for a given set of send and receive types?

A: All services are installed even if duplicated, but there is no guaranteed order of registration. The first one encountered with the proper send and return types is the one that happens to be used.

Q: Any known bugs?

A:  #24667Ðregistering for more services in the middle of handling a service is dicey.

#29821Ð (fixed in 3.1) your filter service cannot expect to be launched with a full pathname if it uses the special NXUNIXSTDIO port.  For example, if a mine-ascii filter is registered for the .mine extension, it will be spawned like so:

```
execl("/LocalApps/myApp.app/mine-ascii", "mine-ascii", "somefile.mine")
```

Since the second argument doesn't include the full path, mine-ascii can't check its name from within the program to see where it was run from.  But you shouldn't need to do this anyway.

#29905Ð(fixed in 3.1) the documentation describes new features for the Workspace which don't currently work, this document documents the mechanisms that work.  Here are the bugs for your info:

- The Workspace doesn't register .service files or directories that are installed in your application path. It only understands about services files within app wrappers in your application path, and .services in the Services directories.
- The Workspace doesn't automatically update your services if you drop a new .service into one of the Services directories. You have to log out/in to get it to register if you install it there.
- The Workspace doesn't automatically update when you drop a new services file into your app wrapper, you have to drop the whole application into the Apps directory.

Q: Where else can I get information about Services, especially about the format for a service description file?

A: You can read the release notes, and the services chapter of the concepts manual.

/NextLibrary/Documentation/NextDev/Concepts/Services.rtf
/NextLibrary/Documentation/NextDev/ReleaseNotes/Appkit.rtf

QA874

Valid for 3.0