



DFC 3.0 Integration Manual

New Mexico Software

Copyright © 2005 New Mexico Software, Inc.

Table of Contents

Introduction.....	1
What is DFC 3.x?.....	1
What is a DFC 3.x asset?.....	1
What is a DFC 3.x user?.....	1
What is a DFC 3.x group?.....	1
What is a DFC 3.x catalog?.....	1
What is a User-Group Relationship?.....	2
What is a Group-Catalog Relationship?.....	3
DFC 3.x Scanners, Copiers, And Document Information.....	4
Copier Devices.....	4
Separator Pages.....	5
DFC 3.x API.....	6
RemoteAW2U.php3.....	6
ID numbers.....	6
User Sessions.....	6
Session Functions.....	7
Catalog Manipulation Functions.....	7
Asset Manipulation Functions.....	8
Miscellaneous Functions.....	9
DFC 3.x API Source Code Examples.....	11
Java Examples.....	11
Python Examples.....	12

Introduction

This documentation describes the mechanisms that [DFC 3.x](#) provides to programatically retrieve and manipulate information stored in the system. In order to fully understand how to integrate [DFC 3.x](#) into your organization, an overview of the [DFC 3.x](#) system is in order.

What is DFC 3.x?

[DFC 3.x](#) is a digital filing cabinet that allows you to upload, organize, download, and tag all your digital files in one place on your network, in a manner that is conducive to your organizational strategy. [DFC 3.x](#) is not a cookie-cutter solution that you have to fit your organization; [DFC 3.x](#) can be customized to fit your organization. [DFC 3.x](#) is a server software system that manages all your digital files through a simple web interface.

A [DFC 3.x file](#) is any type of digital file that requires management and organization, including video, audio, document and image files. [DFC 3.x](#) manages these files with three fundamental concepts: [users](#), [groups](#) and [catalogs](#). [Users](#) have specific [permissions](#) to manipulate certain files, and files are organized into [catalog hierarchies](#), much like directories in a file system. But unlike a simple directory structure, [catalogs](#) in [DFC 3.x](#) can have different permissions depending on what an administrator would like those [catalogs](#) to have. [DFC 3.x](#) refers to digital files as [assets](#).

What is a DFC 3.x asset?

An [asset](#) is a [file](#) that has been tagged with [metadata](#).

What is a DFC 3.x user?

A [DFC 3.x user](#) has certain permissions within a [DFC 3.x](#) system to do certain things. A [DFC 3.x user](#) is analogous to a UNIX user, where a UNIX user can only do certain things in the operating system. Like UNIX, [DFC 3.x users](#) are part of [groups](#). [DFC 3.x users](#) are identified by their valid [e-mail address](#).

What is a DFC 3.x group?

A [DFC 3.x group](#) is a logical grouping of [DFC 3.x users](#). [Groups](#) are useful for assigning permissions to [groups](#) of [users](#) instead of to each [user](#) individually. Both [users](#) and [groups](#) have permissions with respect to [catalogs](#). [Groups](#) can be part of other [groups](#), which allows users to create [group hierarchies](#).

What is a DFC 3.x catalog?

A [DFC 3.x catalog](#) is analogous to a [folder](#) under [Microsoft Windows TM](#)¹ or a [directory](#) under [UNIX](#)^{TM2}. A [directory](#) contains files, and depending on your operating system, has permissions attached to the [directory](#). A [DFC 3.x catalog](#) has no permissions on it per se; the

1 Microsoft Windows is a trademark of Microsoft Corporation.

2 UNIX is a trademark of AT&T.

permissions exist between the [group](#) and [catalog](#) relationships as defined by an administrator. [DFC 3.x catalogs](#) can be part of other [catalogs](#); i.e. they can form [hierarchies](#).

With these fundamental concepts in mind, we can now state the following:

→ [Assets](#) are organized into [catalogs](#), and [users](#) (as part of [groups](#)) are allowed to manipulate them according to their assigned [group permissions](#).

With this foundation, you can begin to understand how [catalog hierarchies](#) and [user groups](#) can be used to manage assets within an organization. Some examples follow:

- The accounting department can create an [Accounts Payable](#) group, an [Accounts Receivable](#) group, and a [General Ledger](#) group, and assign appropriate permissions to users in those groups. For example, [Accounts Payable](#) users may have permission to utilize assets in their own group and in the [General Ledger](#) group, but may have read-only permissions in the [Accounts Receivable](#) group.
- The human resources department can store and manage resumes in a catalog called [Candidates](#), manage pension documents in a catalog called [Retirement](#), and service records in a catalog called [Employees](#). Appropriate human resources groups would be associated with these catalogs, and very limited permissions could be given to employees to access their own personal records.
- Quality assurance or quality control departments can store Standard Operating Procedures (SOP's) and test results in catalogs appropriately named for their function; testers, technicians and engineers can each have their own group respective to their function.

This overview describes only the top-level functionality of the [DFC 3.x Server Software](#) system. To gain a complete understanding of the functionality of [DFC 3.x system software](#), refer to the [DFC 3.x User's Manual and Admin Manual](#).

The [permission](#) structure is very simple. Every [group](#), depending on the [permissions](#) granted to it, can do one or all of the following:

- View, download, upload or delete [assets](#)
- Edit [metadata](#)
- Create and/or distribute [collections](#)

If a [group](#) has [sub-groups](#), then permissions are inherited from the top of the tree down to the bottom. For example, if a root [group](#) can view assets, then all [sub-groups](#) of that [group](#) can view [assets](#) as well.

What is a User-Group Relationship?

A [user](#) is part of a [group](#), and the [group](#) has certain [permissions](#) on assigned [catalogs](#).

What is a Group-Catalog Relationship?

[Groups](#) are attached to certain [catalogs](#). [Catalogs](#) are arranged in hierarchies much like the file system of a computer's operating system; catalogs can contain other [catalogs](#); they can contain several [catalogs](#); they can be [sub-catalogs](#) of other [catalogs](#), etc.. Any sense of [catalog permissions](#) are inherited from the [groups](#) that are assigned to it.

The administration pages allow administrators to control and fine-tune all aspects of the way [DFC 3.x](#) manages [assets](#). For more information, refer to the [DFC 3.x Administrator's Manual](#).

DFC 3.x Scanners, Copiers, And Document Information

DFC 3.x is called a **Digital Filing Cabinet** because it can store, organize, and manage all types of digital files. In addition to complete management of digital files, DFC 3.x can also accept paper documents produced by many "scan-to-file" copiers that produce G3 or G4 TIFF format files. The paper documents are automatically processed to determine their text content via **Optical Character Recognition**. The documents are converted to full-text-searchable PDF files as they are processed by the DFC 3.x server. You can scan in invoices, contracts, bills, or any type of paper document. After they have been processed, you can find them quickly with a **keyword search**.

The input mechanism for the **DFC 3.x Copier Option** is very straight forward. Every DFC 3.x server has a built-in **Windows networking shared folder** called **public**. You can access this folder from any **Windows PC** that lets you send a **username** and **password** to the server. You do not need a username or a password to login; if you are prompted for a **username** and **password**, leave both blank.

You can test mounting your **DFC 3.x** public share from a **Windows 2000/XP** box by right-clicking **My Network Places** and choosing **Map Network Drive**. In the **folder** tab enter:

```
\\<ip addr>\public
```

and press **Finish**. If you are prompted for a **username** and a **password**, just hit **Enter**.

Now you can copy **G3** or **G4 TIFF** files to the **public** shared folder and they are automatically processed by **DFC 3.x** into the full-text-searchable **PDF** files. You will see the **TIFF** files disappear from the **public** folder as they are moved into an internal processing queue. From the **Administration** page, you will see the file name appear in the **Copier Queue**. It will disappear from the **Copier Queue** when it is finished processing and the **PDF** file will be accessible in the **Scan Inbox** folder (or the folder of your choice) in your **DFC 3.x** site.

Copier Devices

Different scanning devices integrate differently with the **DFC 3.x** server. Some larger copiers can scan directly to a public folder in **G3** or **G4 TIFF** file format. In this case, simply create a scan target through the copiers management interface that scans to the **DFC 3.x public** shared folder. Other copiers require special software to download the files from the copier to a **Windows** server. In this case, you will need to share the **folder** on the **Windows** server out to the **DFC 3.x** server, and have the **DFC 3.x** server mount that **folder** locally under the `/home/public` folder.

For example, create a directory under `/home/public` called **external** using the `mkdir` command at a command-line prompt: from the **DFC 3.x** server:

```
mkdir /home/public/external
```

DFC 3.x System Integration Manual

Then mount the [Windows](#) share using `smbmount` under that folder:

```
mount -t smbfs -o username=user,password=pass //win2kserver/docfolder /home/public/external
```

Any [TIFF](#) file in any subdirectory of the `/home/public` is automatically picked up and processed by the [DFC 3.x](#) server. The expected format is a multi-page [TIFF](#) encoded with [G3](#) or [G4](#) compression that contains a single document.

Separator Pages

Multi-page documents can also contain a [DFC 3.x](#) job [separator page](#). [Separator pages](#) allow an end user to input a large batch job with many pages . The [DFC 3.x](#) server splits up the batch job into smaller documents. The [separator pages](#) act as dividers between the documents. Using separator pages, a single input job can become a set of [PDF](#) files, with individual indexes for each document.

DFC 3.x API

DFC 3.x allows programmers to access the system internals at a lower level than the web browser interface. The [DFC 3.x Application Programming Interface \(API\)](#) is an interface that marshals requests to and responses from the internal workings of the [DFC 3.x](#) server via a rudimentary [HTTP](#) protocol interface. The [API](#) allows an application programmer to manipulate a **PREVIOUSLY INITIALIZED SYSTEM**. If the system has no [users](#), [groups](#), [catalogs](#), or [assets](#), the [API](#) does not afford the programmer any more functionality than the traditional web browser interface. It is suggested that the system be initialized via the web browser interface before creating low level applications. Future [API](#)'s will allow the programmer to initialize a [DFC 3.x](#) server via the low-level interface.

The heart of the [DFC 3.x API](#) is a file called [RemoteAW2U.php3](#). [CGI parameter calls](#) to this file determine the internal functions called and the output received. As such, any programming language or platform that can handle [URL](#) calls can work easily with the [DFC 3.x API](#).

RemoteAW2U.php3

The internal workings of the [DFC 3.x](#) system is accessed via a facade file called [RemoteAW2U.php3](#). All calls necessary to access, maintain, and manipulate a working session with a [DFC 3.x](#) server go through this interface file. A few concepts need to be explained before a programmer can work with [DFC 3.x](#) at a low level.

ID numbers

[DFC 3.x](#) identifies [catalogs](#) and [assets](#) via the use of [ID numbers](#). Each [catalog](#) has its own unique [catalog ID number](#), and each [asset](#) has its own unique [asset ID number](#). As far as the application programmer is concerned, these numbers are magic numbers and only have meaning within the [DFC 3.x](#) internal database. They should be accounted for by meaningful variables in application programs.

User Sessions

Since [DFC 3.x](#) is designed to work with a [web browser](#), certain tasks that a [web browser](#) would normally perform for free have to be explicitly performed by the application programmer. [DFC 3.x](#) keeps track of [user sessions](#) in its internal database. It passes a [cookie](#) to the [web browser](#) that requested a login, which helps [DFC 3.x](#) keep track of which [web browser](#) is associated with which [session](#).

The session function [GetSession](#) must be called before any other function, because it is tantamount to logging into the server via a [web browser](#).

PLEASE NOTE: Some of the following functions use [Session](#), and some functions use [SessionId](#). If you are having trouble accessing certain functions of the [DFC 3.x API](#), verify the session parameter you are using is the correct one.

Session Functions

It is up to the programmer to obtain a [session](#) from the [DFC 3.x](#) server. A brief description of the [GetSession](#) function follows:

GetSession	Establishes a user session with the given DFC 3.x host. Returns the name of the session. URL parameters: Cmd , Username , Password
CheckSession	Checks to see if a session is valid URL parameters: Cmd , SessionId
RefreshSession	Refreshes a users session. Rebuilds the permissions for the session in case the database has been modified significantly. URL parameters: Cmd , SessionId

The [HTTP](#) call to obtain a session is as follows:

```
http://<DFC 3.x host>/RemoteAW2U.php3?Cmd=GetSession&
      Username=<username>&Password=<password>
```

all one line, with no spaces. Notice the call to obtain a new [session](#) is the [Cmd=GetSession](#) part of the [CGI](#) parameter call. This call creates a new [session](#) in the [DFC 3.x](#) database and allows the programmer to access the other functions of the [API](#).

The [session name](#) is returned. This name should be saved for future use (it is used in all subsequent calls).

Catalog Manipulation Functions

A brief description of each function follows:

GetCurCat	returns the current catalog ID number URL parameters: Cmd , Session
SetCurCat	sets the current catalog ID number URL parameters: Cmd , CatalogId , Session
GetCatName	returns the current catalog name URL parameters: Cmd , CatalogId , Session
GetCatPath	returns the catalog path for the given catalog ID number URL parameters: Cmd , CatalogId , Session
GetCatalogChildren	returns a tab-separated list of the immediate child catalog ID

	numbers and names for the given catalog ID number. Each line is separated by a newline character. URL parameters: Cmd, CatalogId, Session
GetCatalogAssets	returns a tab-separated list of all the asset ID numbers and filenames for the given catalog ID number. Each line is separated by a newline character. URL parameters: Cmd, CatalogId, Session
CreateCatalog	creates a new catalog URL parameters: Cmd, ParentCat, CatName, SessionId
DeleteCatalog	deletes a catalog URL parameters: Cmd, CatalogId, SessionId
MoveCatalog	moves a catalog from one catalog to another URL parameters: Cmd, CatalogId, FromCat, ToCat, SessionId
RenameCatalog	renames a catalog URL parameters: Cmd, CatalogId, NewName, SessionId
GetCatalogAssetType	Returns the asset type of the given catalog URL parameters: Cmd, CatalogId
GetCatalogMetadata	Gets the metadata for a specific field for the given catalog URL parameters: Cmd, CatalogId, FieldName
SetCatalogMetadata	Sets the value of a metadata field for a catalog URL parameters: Cmd, CatalogId, FieldName, Value

Catalog manipulation functions are accessed in the same way a **session** is obtained. An example follows:

`http://<DFC 3.x host>/RemoteAW2U.php3?Cmd=GetCurCat&
Session=<session name>`

Just replace the **GetCurCat** with the function desired. The **Session** or **SessionId** parameter must accompany all **URL** calls to the server.

Asset Manipulation Functions

A brief description of each function follows:

GetFilename	Returns the filename associated with the given asset ID number
--------------------	--

	URL parameters: Cmd, AssetId, SessionId
CopyAsset	Creates a copy of an asset URL parameters: Cmd, AssetId, FromCat, ToCat, SessionId
MoveAsset	Moves an asset from one catalog to another catalog URL parameters: Cmd, AssetId, FromCat, ToCat, SessionId
DeleteAsset	Deletes an asset URL parameters: Cmd, AssetId, SessionId
GetMetadata	Gets the metadata for an asset URL parameters: Cmd, AssetId, FieldName, SessionId
SetMetadata	Sets the metadata for an asset for a specific field URL parameters: Cmd, AssetId, FieldName, Value, SessionId
GetAssetType	Gets the asset type of an asset. URL parameters: Cmd, AssetId
GetMetadataFields	Gets the metadata fields associated to an asset type URL parameters: Cmd, AssetType

An example follows:

`http://<DFC 3.x host>/RemoteAW2U.php3?Cmd=MoveAsset&AssetId=15&FromCat=46&ToCat=35&SessionId=<session name>`

The **Session** or **SessionId** parameter must accompany all **URL** calls to the server.

Miscellaneous Functions

A brief description of each function follows:

GetSearchResults	Returns a list of strings containing asset ID numbers, filenames, and catalog ID numbers for the given catalog ID number and keywords. ID numbers, filenames and keywords are tab-separated, and each string is separated by a newline. Keywords should be space-separated and URL encoded URL parameters: Cmd, CatalogId, Keywords, Session
GetSearchResultsByField	Performs a search on a specific field and returns the results URL parameters: Cmd, CatalogId, Keywords, FieldName, Session
GetStreamer	Returns the URL necessary to access the DFC 3.x image server

DFC 3.x System Integration Manual

	URL parameters: Cmd
IsAdmin	Returns 1 is the user is an admin, 0 otherwise URL parameters: Cmd, SessionId

An example follows:

<http://<DFC 3.x host>/RemoteAW2U.php3?Cmd=GetSearchResults&CatalogId=455&Keywords=the%20and&Session=<session name>>

DFC 3.x API Source Code Examples

The following source code examples demonstrate how to use the [DFC 3.x API](#) using [Java](#) and [Python](#). [Java](#) lends itself better to software engineering programmers; [Python](#) is more flexible and can satisfy both software engineering types and scriptwriter types. Both have built-in [URL](#) and [HTTP](#) functionality.

Java Examples

Below is an example of a [Java](#) class called [Session.java](#) that facilitates the retrieval of a [DFC 3.x](#) user session.

```
import java.net.URL;
import java.net.MalformedURLException;
import java.io.IOException;
import java.io.BufferedReader;
import java.io.InputStreamReader;

class Session {

    private String remoteURL;
    private String sessionName;

    Session(String AWHost) {
        remoteURL = AWHost + "/RemoteAW2U.php3";
    }

    String getSession(String user, String pass)
        throws MalformedURLException, IOException {

        URL host = host = new URL(remoteURL + "?Cmd=GetSession&Username=" +
            user + "&Password=" + pass);

        BufferedReader br = new BufferedReader(
            new InputStreamReader(host.openStream()));
        sessionName = br.readLine();

        return sessionName;
    }

    String getSessionName() { return sessionName; }

    String getRemoteURL() { return remoteURL; }

    public String toString() {
        return "Remote URL: " + remoteURL + "; Name: " + sessionName;
    }
}
}
```

The code to use the [Session](#) class could look like the following:

```
import java.net.MalformedURLException;
import java.io.IOException;

public Test {

    public static void main(String args[]) {

        Session userSession = new Session("http://DFC 3.x.mydomain.com");
        String temp = null;
        try {
            // argv[0] would be the user's name
```

DFC 3.x System Integration Manual

```
        // argv[1] would be the password
        temp = sess.getSession(argv[0], argv[1]);
    }
    catch (MalformedURLException m) {
        System.err.println(m.getMessage());
    }
    catch (IOException i) {
        System.err.println(i.getMessage());
    }
    if (temp.equalsIgnoreCase("")) {
        System.err.println("Login failed.");
    }
    System.out.println(userSession);
}
}
```

With these example code files under our belt, we can provide **Java** code snippets to further illustrate the use of the **DFC 3.x API**. These examples are not fully complemented with variable declarations; they are just code snippets so that the application programmer can become more familiar with the **URL** syntax of the **API** functions.

The following code snippet illustrates the use of the **GetCatName** function.

```
host = new URL(sess.getRemoteURL() +
    "?Cmd=GetCatName&CatalogId=" +
    catalogID + "&Session=" + sess.getSessionName());

BufferedReader br = new BufferedReader(
    new InputStreamReader(host.openStream()));

String catalogName = br.readLine();
```

The following code snippet illustrates a potential use of the **GetCatalogAssets** function. The information is placed into a **HashMap** for future reference.

```
import java.util.HashMap;
import java.util.StringTokenizer;
.
.
String keywords = java.net.URLEncoder.encode("the and or but not neither");

host = new URL(sess.getRemoteURL() +
    "?Cmd=GetCatalogAssets&CatalogId=" +
    catalogID + "&Session=" + sess.getSessionName());

BufferedReader br = new BufferedReader(
    new InputStreamReader(host.openStream()));

String searchResults = br.readLine();
StringTokenizer st1 = new StringTokenizer(searchResults, "\n");
StringTokenizer st2 = null;
HashMap assets = new HashMap();

while (st.hasMoreTokens()) {
    String temp = st.nextToken();
    st2 = new StringTokenizer(temp, "\t");
    while (st2.hasMoreTokens()) {
        String temp1 = st2.nextToken();
        assets.put(temp, temp1);
    }
}
```

DFC 3.x System Integration Manual

```
// Now the assets HashMap can be used to lookup filenames
// via their asset ID numbers
```

Python Examples

Below is an example of a **Python** code file called **Session.py** that prints out the name of a **DFC 3.x** user session. The example assumes a **UNIX** operating system, but can be applied under any operating system that supports **Python**.

```
#!/usr/bin/env python

import urllib

username = "DFC 3.xuser@mydomain.com"
password = "password"
host = "http://DFC 3.x.mydomain.com"

page = urllib.urlopen(host + "/RemoteAW2U.php3?Cmd=GetSession&Username=" +
    username + "&Password=" + password)

sessionName = page.read()
```

The following code snippets illustrate the syntax required for **DFC 3.x API** calls. These examples are not fully complemented with variable declarations; they are just code snippets so that the application programmer can become more familiar with the **URL** syntax of the **API** functions.

This example gets a list of the immediate children for the given **catalog ID number** and places the information in a list.

```
page = urllib.urlopen(host + "/RemoteAW2U.php3?Cmd=GetCatalogChildren&CatalogId=" +
    catalogId + "&Session=" + sessionName)

catalogChildren = page.read().split()
```

This example obtains the **asset ID numbers** and **filenames** for the given **catalog ID number** and places the information in a dictionary.

```
page = urllib.urlopen(host + "/RemoteAW2U.php3?Cmd=GetCatalogAssets&CatalogId=" +
    catalogId + "&Session=" + sessionName)

assetlines = page.read().split("\n")
for asset in assetlines:
    if asset == '':
        assetlines.remove(asset)

assets = { }
for asset in assetlines:
    temp = asset.split("\t")
    assets[temp[0]] = temp[1]

# Now the assets dictionary can be used to lookup filenames
# via their asset ID numbers
```

Visual Basic (5.0) Examples

The following example has complete VB .bas module for preparing files for upload and corresponding code to actually send the file to a server.

DFC 3.x System Integration Manual

HTTPPost.bas follows:

```
Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (lpDest As Any, _
lpSource As Any, _
ByValcbCopy As Long)\

Public Sub HTTPPostPrep(ByVal RemoteHost As String, _
ByVal Path As String, _
ByVal Filename As String, _
ByVal cookie As String, _
ByRef HeaderData() As Byte, _
ByRef HeaderSize As Long, _
ByRef FooterData() As Byte, _
ByRef FooterSize As Long)

Dim Filesize As Long
Dim Boundary As String
Dim Header As String
Dim Message1 As String
Dim Message2 As String
Dim bHeader() As Byte
Dim bFile() As Byte
Dim bMessage1() As Byte
Dim bMessage2() As Byte
Dim FileHandle As Long
Dim ContentLength As Long
Dim i As Long

' Get the file size of the Content-length header
Filesize = FileLen(Filename)

' We no longer need to read in the entire file here
' as the calling function will stream it off of disk
' and eliminate the memory overflow problem.
' Read the file into a byte array here
' ReDim bFile(FileSize) As Byte
' FileHandle = FreeFile
' Open Filename For Binary Access Read As FileHandle
' Get FileHandle, , bFile
' Close FileHandle
' Establish a MIME boundary
Boundary = "-----1862481476497"

' Build the first part of the POST header
Header = "POST " + Path + " HTTP/1.0" + vbCrLf
Header = Header + "Connection: Keep-Alive" + vbCrLf
Header = Header + "User-Agent: AssetWare Productivity API 1.0" + vbCrLf
Header = Header + "Host: " + RemoteHost + ":80" + vbCrLf
Header = Header + "Accept-Encoding: gzip" + vbCrLf
Header = Header + "Accept-Language: en" + vbCrLf
Header = Header + "Content-type: multipart/form-data; boundary=" + Boundary + vbCrLf
Header = Header + "Content-Transfer-Encoding: binary" + vbCrLf

' Build up the message before the file
Message1 = "--" + Boundary + vbCrLf
Message1 = Message1 + "Content-Disposition: form-data; name=""MAX_FILE_SIZE"" + vbCrLf
Message1 = Message1 + vbCrLf
Message1 = Message1 + "--" + Boundary + vbCrLf
Message1 = Message1 + "Content-Disposition: form-data; name=""submit"" + vbCrLf
Message1 = Message1 + vbCrLf
Message1 = Message1 + "Send File" + vbCrLf
Message1 = Message1 + "--" + Boundary + vbCrLf
Message1 = Message1 + "Content-Disposition: form-data; name=""userfile""; filename="" +
Filename + """" + vbCrLf
Message1 = Message1 + "Content-Type: application/octet-stream" + vbCrLf
Message1 = Message1 + vbCrLf
bMessage1 = StrConv(Message1, vbFromUnicode)
```

DFC 3.x System Integration Manual

```
' Build up the message after the file
Message2 = Message2 + vbCrLf
Message2 = Message2 + "--" + Boundary + vbCrLf
Message2 = Message2 + "Content-Disposition: form-data; name=""AT"" + vbCrLf
Message2 = Message2 + vbCrLf
Message2 = Message2 + "1" + vbCrLf
Message2 = Message2 + "--" + Boundary + "--" + vbCrLf
bMessage2 = StrConv(Message2, vbFromUnicode)

' Finish off the header
ContentLength = (Len(Message1) + Filesize + Len(Message2))
Header = Header + "Content-Length: " + Str(ContentLength) + vbCrLf
Header = Header + "Cookie: " + cookie + vbCrLf + vbCrLf
bHeader = StrConv(Header, vbFromUnicode)

' Return back the header and the footer.
' The file will be streamed by the calling function
HeaderSize = Len(Header) + Len(Message1)

ReDim HeaderData(HeaderSize)
For i = 0 To Len(Header) - 1
    HeaderData(i) = bHeader(i)
Next i

For i = Len(Header) To (Len(Header) + Len(Message1) - 1)
    HeaderData(i) = bMessage1(i - Len(Header))
Next i

FooterSize = Len(Message2)
ReDim FooterData(FooterSize)
For i = 0 To Len(Message2) - 1
    FooterData(i) = bMessage2(i)
Next i
End Sub
```

Perl Example

The simplest example in Perl is logging into the system. Using LWP::Simple:

```
use LWP::Simple;
$session = get("http://192.168.1.1/Command=GetSession&Username=user&Password=pass");
```

The following example uses Perl to perform an upload to the system.

```
sub build_post {
    $remote = $_[0];
    $path = $_[1];
    $filename = $_[2];
    $cookie = $_[3];
    # Get filesize for fileread
    $size = (-s $filename);
    # Binary file goes here
    open FILE, "<$filename";
    binmode FILE;
    read FILE, $file, $size;
    close FILE;
    $boundary = "-----1862481476497";
    $header = "";
    $header .= "POST $path HTTP/1.0\r\n";
    $header .= "Connection: Keep-Alive\r\n";
    $header .= "User-Agent: Mozilla/4.77 [en] (Windows NT 5.0; U)\r\n";
    $header .= "Host: $remote:80\r\n";
    $header .= "Accept-Encoding: gzip\r\n";
}
```

DFC 3.x System Integration Manual

```
$header .= "Accept-Language: en\r\n";
$header .= "Accept-Charset: iso-88591,*,utf-8\r\n";
$header .= "Content-type: multipart/form-data; boundary=$boundary\r\n";
$header .= "Content-Transfer-Encoding: binary\r\n";
$msg .=" --$boundary\r\n";
$msg .=" Content-Disposition: form-data; name=\"MAX_FILE_SIZE\"\r\n";
$msg .=" \r\n";
$msg .=" 1073741824\r\n";
$msg .=" --$boundary\r\n";
$msg .=" Content-Disposition: form-data; name=\"submit\"\r\n";
$msg .=" \r\n";
$msg .=" Send File\r\n";
$msg .=" --$boundary\r\n";
$msg .=" Content-Disposition: form-data; name=\"userfile\"; filename=\"$filename\"\r\n";
$msg .=" Content-Type: application/octet-stream\r\n";
$msg .=" \r\n";
$msg .=" $file . "\r\n";
$msg .=" --$boundary\r\n";
$msg .=" Content-Disposition: form-data; name=\"AT\"\r\n";
$msg .=" \r\n";
$msg .=" 1\r\n";
$msg .=" --$boundary--\r\n";
$header .= "Content-Length: " . length($msg) . "\r\n";
$header .= "Cookie: $cookie\r\n\r\n";
return $header . $msg;
}
```

Then sending the data is as easy as using:

```
sub post_data {
    use IO::Socket;
    $remote = $_[0];
    $request = $_[1];
    my $server = IO::Socket::INET->new(
        PeerAddr => $remote,
        PeerPort => 80,
        Proto => 'tcp'
    ) or die "Can't create client socket: $!";
    binmode $server;
    print $server $request;
    while(<$server>) {
        print $_;
    }
    close $server;
}
```

JavaScript Form Editing Example

Forms inside the DFC can have an arbitrary layout, along with JavaScript validation. The supporting library that allows you to do this is called RemoteJS.php3. The form editor will include it for you so you can immediately use the function defined below in your script. For example here is a validating form that check 3 different metadata fields and calculates a total:

```
<SCRIPT TYPE="text/javascript">
```

```
function UpdateTotal() {
    var obj;
    if (aid == -1) {
        obj = typeinfo;
    } else {
        obj = info;
    }

    var subtot = 0;
```

DFC 3.x System Integration Manual

```
for (var i = 0; i < 5; i++) {
    var qtyObj = GetField(obj, "Qty" + i);
    var priceObj = GetField(obj, "Price" + i);

    var qtyForm = eval("document.metaform.MD" + qtyObj.id + ".value");
    var priceForm = eval("document.metaform.MD" + priceObj.id + ".value");

    var qtyVal = parseInt(qtyForm);
    var priceVal = parseFloat(priceForm);

    if (qtyVal > 0 && priceVal > 0) {
        subtot += qtyVal * priceVal;
    }
}

var subObj = GetField(obj, "Subtotal");
var subForm = eval("document.metaform.MD" + subObj.id);
var subVal = new ToFmt(subtot);
subForm.value = subVal.fmtF(8,2);

var taxObj = GetField(obj, "Tax");
var taxForm = eval("document.metaform.MD" + taxObj.id + ".value");
var tax = parseFloat(taxForm) / 100;

var taxamt = subtot * tax;

var totObj = GetField(obj, "Total");
var totForm = eval("document.metaform.MD" + totObj.id);
var totVal = new ToFmt(subtot + taxamt);
totForm.value = totVal.fmtF(8,2);
}

function GetDemoMain(obj) {
    var html = "<table style=\"border:1px solid black;padding:10px;background-color:white;\">";
    html += "<tr><th>Description</th><th>Qty.</th><th>Price</th></tr>";

    for (var i = 0; i < 5; i++) {
        var desc = GetField(obj, "Desc" + i);
        var qty = GetField(obj, "Qty" + i);
        var price = GetField(obj, "Price" + i);

        html += "<tr>";

        html += "<td>";
        html += "<input type=text name='MD" + desc.id + "' size=25 value='" + desc.value + "'>";
        html += "</td>";

        html += "<td>";
        html += "<input type=text name='MD" + qty.id + "' size=2 value='" + qty.value + "'";
        html += "onChange='UpdateTotal()'>";
        html += "</td>";

        html += "<td>";
        html += "<input type=text name='MD" + price.id + "' size=8 value='" + price.value + "'";
        html += "onChange='UpdateTotal()'>";
        html += "</td>";

        html += "</tr>";
    }

    var sub = GetField(obj, "Subtotal");
    var tax = GetField(obj, "Tax");
    var tot = GetField(obj, "Total");

    html += "<tr>";
    html += "<td colspan=2 align=right>Subtotal</td><td><input type=text size=8 name='MD" +
```

DFC 3.x System Integration Manual

```
sub.id + "' value='" + sub.value + "'></td>";
html += "</tr>"

html += "<tr>"
html += "<td colspan=2 align=right>Tax</td><td><input type=text size=8 name='MD' + tax.id
+ "' value='" + tax.value + "'></td>";
html += "</tr>"

html += "<tr>"
html += "<td colspan=2 align=right>Total</td><td><input type=text size=8 name='MD' +
tot.id + "' value='" + tot.value + "'></td>";
html += "</tr>"

html += "</table>";
return html;
}

function RenderDemoFormCreator(ele) {
    var html = GetDemoMain(typeinfo);

    var now = new Date();
    var invoicenum = Math.ceil(now.getTime() / 100000);

    var salesman = GetField(typeinfo, "Salesman");
    var inv = GetField(typeinfo, "InvoiceNum");

    html += '<input type=hidden name="MD' + salesman.id + "' value="' + user.first + " " +
user.last + "'>';
    html += '<input type=hidden name="MD' + inv.id + "' value="' + invoicenum + "'>';
    html += '<input type=hidden name="SetName" value="DemoInvoice-' + invoicenum + "'>';

    html += '<br><input type=submit name="Submit" value="Save">&nbsp;&nbsp;&nbsp;<input type=button
name="Cancel" value="Cancel"
onClick="document.location=\'/LightboxRoot.php3?Reload=N&Versions=No\'>';

    ele.innerHTML = html;
}

function RenderDemoMetadataEditor(ele) {
    var html = GetDemoMain(info);

    var atid = GetTypeId();
    var salesman = GetField(info, "Salesman");
    var invoicenum = GetField(info, "InvoiceNum");

    html += "<input type=hidden name='MD' +salesman.id+ "' value='" + salesman.value + "'>";
    html += "<input type=hidden name='MD' +invoicenum.id+ "' value='" + invoicenum.value +
"'>";
    html += "<input type=hidden name='SetName' value='" + GetField(info, "Filename").value +
"'>";
    html += "<input type=hidden name='SetType' value='" + atid + "'>";
    html += '<br><input type=submit name="Submit" value="Save Changes">';

    ele.innerHTML = html;
}

</SCRIPT>

<SCRIPT TYPE="text/javascript">
window.onload = function() {
    /* setup needed hidden form variables */
    document.forms[0].AID.value = aid;
    document.forms[0].CID.value = cid;
    document.forms[0].Versions.value = versions;

    var msg = document.getElementById("message");
    msg.innerHTML = message;
}
```

DFC 3.x System Integration Manual

```
var tb = document.getElementById("titlebar");
if (aid == -1) {
    tb.innerHTML = "<b>Add Form: " + typeinfo.assettype + "</b>";
} else {
    if (info.form) {
        tb.innerHTML = "<b>Edit Form</b>";
    } else {
        tb.innerHTML = "<b>Edit Metadata</b>";
    }
}

var mb = document.getElementById("metadatabox");
var fb = document.getElementById("filedbox");
if (aid == -1) {
    RenderDemoFormCreator(mb);
    RenderCatalog(fb, "To be Filed in");
} else {
    RenderDemoMetadataEditor(mb);

    var thmb = document.getElementById("thumbnail");
    RenderThumbnail(thmb);

    var ib = document.getElementById("infobox");
    RenderInfoBox(ib);

    var ab = document.getElementById("actionsbox");
    RenderActionsBox(ab);

    RenderCatalog(fb, "Filed in");
}
}
</SCRIPT>

<FORM NAME="metaform" ENCTYPE="multipart/form-data" METHOD="POST">
<INPUT TYPE="HIDDEN" NAME="CID" VALUE="">
<INPUT TYPE="HIDDEN" NAME="AID" VALUE="">
<INPUT TYPE="HIDDEN" NAME="Versions" VALUE="">
<INPUT TYPE="HIDDEN" NAME="R" VALUE="">

<SPAN ID="message" STYLE="color:red;"></SPAN>
<div style="padding:20px;"><H1>Wally's Widgets Invoice</H1></div>

<TABLE WIDTH="100%">

<TR>
    <TD ROWSPAN="2" WIDTH="50%" VALIGN="TOP" ID="metadatabox"></TD>
    <TD ID="thumbnail"></TD>
</TR>
<TR>
    <TD WIDTH="50%">
        <DIV ID="infobox"></DIV><BR>
        <DIV ID="actionsbox"></DIV><BR>
        <DIV ID="filedbox"></DIV>
    </TD>
</TR>
</TABLE>
</FORM>
```

Alphabetical Index

API		SetMetadata	9
CGI parameter calls	6	ID numbers	6
Concepts		RemoteAW2U.php3	6
session name	7	User Sessions	6
Examples		asset	1
Java	11	assets	2, 6
Perl	15	ID number	6, 8, 13
Python	13	catalog	1
Visual Basic	13	catalog hierarchies	1, 2
Functions		catalogs	3, 6
Asset	8	ID number	6, 7, 9, 13
Catalog	7	sub-catalogs	3
CheckSession	7	CGI parameter calls	6
CopyAsset	9	collections	2
CreateCatalog	8	Copier Queue	4
DeleteAsset	9	DFC 3.x	
DeleteCatalog	8	Concepts	
GetAssetType	9	asset	1
GetCatalogAssets	8	catalog	1
GetCatalogAssetType	8	group	1
GetCatalogChildren	7	users	1
GetCatalogMetadata	8	What is a Group-Catalog Relationship?	3
GetCatName	7	What is a User-Group Relationship?	2
GetCatPath	7	What it is	1
GetCurCat	7	file	1
GetFilename	8	groups	1, 2
GetMetadata	9	groups	
GetMetadataFields	9	sub-groups	2
GetSearchResults	9	ID numbers	6
GetSearchResultsByField	9	metadata	2
GetSession	7	Optical Character Recognition	4
GetStreamer	9	PDF files	4
IsAdmin	10	permissions	1
Miscellaneous	9	public, folder	4
MoveAsset	9	RemoteAW2U.php3	6
MoveCatalog	8	Scan Inbox, folder	4
RefreshSession	7	separator page	5
RenameCatalog	8	Session.java	11
Session	7	Session.py	13
SetCatalogMetadata	8	smbmount	5
SetCurCat	7		

DFC 3.x System Integration Manual

TIFF files 4, 5
user groups

2

users 1
session

6