Affymetrix®

White Paper

Using the NetAffx® SDK XML Feed for Notification of Annotation or Library File Updates

The NetAffx® Analysis Center provides an XML file which lists annotation, library and analysis configuration files for all supported catalog arrays. A date stamp and size is provided for each file, so that changes can be tracked.

Affymetrix analysis software uses this XML feed to check for updates to annotation, library and configuration files so that they can be downloaded and kept current in the analysis package. This white paper describes how the XML feed can be read through a Perl script and used to automate the downloading and updating of Affymetrix annotation and library file data for most catalog array products.

The NetAffx portal supports updates of probe set annotations, library files and configuration files for Affymetrix analysis applications such as Expression Console™, Genotyping Console™ and Tiling Array Analysis Software. Table 1 lists some of the most commonly accessed files which are supported by the SDK.

Table 1: Popular file classes supported by the NetAffx SDK XML feed.

Annotation Files	**Library Files**	**NetAffx Analysis configuration files**
▪ Annot MAGE-ML ▪ BLASTp CSV ▪ BLASTx CSV ▪ CN Annot CSV ▪ Ortholog CSV ▪ Probeset Annot CSV ▪ Transcript Cluster Annot CSV ▪ Annot CSV ▪ Annotation BED Files	▪ PPM GCC PPD CLF PGF ▪ PPW BGP CDF CIF PSI ▪ Probe FASTA ▪ Probe Tabular ▪ Control Seq FASTA ▪ Exemplar Seq FASTA ▪ Target Seq FASTA ▪ Consensus Seq FASTA ▪ Flanking Seq FASTA ▪ Transcript Cluster FASTA ▪ Design Time Annotations, Probe set, CSV ▪ Design Time Annotations, Full, GFF ▪ Transcript Cluster ID Mappings ▪ Tiling Array BPMAP	▪ Exon analysis configuration ▪ Prior estimates ▪ GC CNV analysis configuration ▪ CANARY normalization ▪ GC analysis configuration ▪ CNWORKFLOW ▪ Exp analysis configuration

Join the Affymetrix Developers' Network

The Affymetrix Developers' Network (ADN) is a forum and resource center for developers and users of expression and genotyping bioinformatics software. For a full description of the ADN and the resources offers, please visit http://www.affymetrix.com/support/developer/.

To join the ADN, you will need a license key and an affymetrix.com login name and password, which can be obtained by registering at affymetrix.com. Contact devnet@affymetrix.com to join the ADN and request a free license key.

Among the resources provided by the ADN is the NetAffx SDK library, which handles updates to the NetAffx support via the NetAffx XML feed. The SDK libraries include a set of COM interfaces which can interact with VB, C++, VC++ and ASP for Microsoft-based applications. Java classes relating to the libraries are available using Java Native Interface.

Fetching the SDK URL

The XML feed can be obtained directly via a script, which can be preferable for Unix-based bioinformatics environments. Once you have the license key, you can obtain the XML file through this URL form:

http://www.affymetrix.com/analysis/downloads/netaffxapi/GetFileList.jsp?licence=[key]&user=[username]&password=[password]

You will need to substitute in your key, login and password for affymetrix.com as indicated in the square brackets. The URL works in a web browser, but it also contains the authentication information so that a simple script can access the file without the use of cookies.

An example entry of the SDK XML Feed

The XML feed currently contains update information for 120 arrays and array sets and about 900 files.

Below is a sample XML Array entry for the GeneChip® Porcine Genome Array (some file entries are excerpted). While the XML may allow for more than one file per annotation type, in practice there is only one file per annotation type. The file type, name, size, date stamp and URL are provided for each file.

By convention, there is only one file and one URL in each file entry. The date will indicate when files are updated. The DTD for the XML file is available here: http://www.affymetrix.com/analysis/downloads/netaffxapi/NetAffxAnnotFileList.dtd.

With some basic tools, it is fairly easy to use the XML feed to make sure you are regularly notified of changes in the annotation or library files used for expression and genotyping analysis. In the next section, we give some example implementation using Perl and the Unix crontab application, both of which are widely available through training or on the internet.

```
<Array name="Porcine">
   <Annotation type="Annot CSV" description="Annotations, CSV format">
     <File name="Porcine.na26.annot.csv" size="27289196" date="Jul 8,
2008">
       <URL compression="application/zip"
size="3610028">http://www.affymetrix.
com/analysis/downloads/na26/ivt/Porcine.na26.annot.csv.zip</URL>
     </File>
   </Annotation>
 . . .
   <Annotation type="PSI" description="PSI Library File">
     <File name="Porcine.psi" size="660765" date="Dec 7, 2004">
       <URL compression="application/zip"
size="140669">http://www.affymetrix.com/analysis/downloads/lf/Porcine_ps
i.zip</URL>
     </File>
   </Annotation>
. . .
   <Annotation type="Probe Tabular" description="Probe Sequences, tabular
format">
     <File name="Porcine_probe_tab" size="17400015" date="Feb 16,
2005">
       <URL compression="application/zip"
size="4564792">http://www.affymetrix.com/analysis/downloads/data/Porcin
e_probe_tab.zip</URL>
     </File>
   </Annotation>
</Array>
```

Figure 1: Sample NetAffx XML feed entry for the GeneChip® Porcine Genome Array, showing file entries for the annotation CSV files, PSI library file and the tabular probe sequence file.

Demonstration Perl script to fetch the SDK XML file and print information

Below is a Perl script that illustrates the basics of how to fetch the NetAffx SDK XML file using some common Perl Libraries. The script may be run on a Unix or Windows machine which has internet access with Perl installed. The script extracts and displays current information about Affymetrix library and annotation files. It can be easily modified to compare current file information to data stored in a file or a database, or download newer files than those currently available on your system.

The XML file is authenticated through the URL to make it simple to download via the Unix crontab application.

While library files such as probe tab and FASTA files are almost never changed, the SDK can serve as a notification service for the release of new Affymetrix arrays or the update of annotation files due to a new NetAffx data release.

```
#!/usr/bin/env perl

## For usage information, run this with a "-h" argument.

=head1 NAME

netaffxSDK-simple.pl - Get info about Affymetrix data files via the NetAffx SDK

=head1 SYNOPSIS

  netaffxSDK-simple.pl -user USERNAME -pass PASSWORD -lic LICENSE

=head1 DESCRIPTION

This script demonstrates how to access the NetAffx SDK XML file and
extract information about Affymetrix data files from it.
It takes user credentials via the three required command-line arguments.

For a more advanced script with additional functionality, including
file downloading and filtering, see the appendix of the whitepaper.

For more information about the NetAffx SDK:
http://www.affymetrix.com/support/developer/netaffx_sdk/netaffx_sdk_overview.aff
x

Use of this software requires an Affymetrix Developers Network license key.
To get one, contact devnet@affymetrix.com

=head1 AUTHORS

 Ron Shigeta <ron_shigeta@affymetrix.com>
 Steve Chervitz <steve_chervitz@affymetrix.com>

=head1 COPYRIGHT

Copyright 2008. Affymetrix, Inc. All rights reserved.

This software is covered by the terms of use or license located at
http://www.affymetrix.com/site/terms.affx

Revision: $Id: netaffxSDK-simple.pl,v 1.2 2008/09/24 23:34:01 steve Exp $

=cut

use strict;
use XML::LibXML;
use LWP::Simple;
use URI::URL;
use Getopt::Long;
use Pod::Usage;

my $license;
```

```
my $username;
my $password;
my $help;

GetOptions(
          "l|lic:s" => \$license
        , "u|user:s" => \$username
        , "p|pass:s" => \$password
        , "h|help" => \$help
);

pod2usage(-exit=>0, -verbose=>2) if $help;

unless (defined $license and defined $username and defined $password) {
   pod2usage(-message =>"\nFATAL: Missing required arguments",
          -exit=>1, -verbose=>1);
}

# Get the SDK XML file.
my $sdkurl =
"http://www.affymetrix.com/analysis/downloads/netaffxapi/GetFileList.jsp?licence=$
{license}&user=${username}&password=${password}";

# Grab the content of the SDK XML file in memory
my $url = url($sdkurl);
$url->query_form('license'  => $license,
             'user'     => $username,
             'password' => $password);
my $sdk_content = get($url) or die "FATAL: Can't get SDK content: $!\n";

# Parse it into a DOM structure
my $parser = XML::LibXML->new();
my $tree = $parser->parse_string($sdk_content) or
   die "\nFATAL: SDK XML data did not parse successfully. Possibly incomplete
download.\n";

my $root = $tree->getDocumentElement;
my @GeneArrays = $root->getElementsByTagName('Array');

# Collects information about all files, printing out key information.
# The zip_size attribute can be used to verify that the file
# has been completely downloaded, comparing against the downloaded file.

foreach my $array_id (@GeneArrays) {
   my $array_name = $array_id->getAttribute('name');
   my @Ann_node  = $array_id->getElementsByTagName('Annotation');
   foreach my $ann (@Ann_node) {
      my $type = $ann->getAttribute('type');
      my @Files = $ann->getElementsByTagName('File');
      foreach my $file (@Files) {
         my $fname = $file->getAttribute('name');
         my $date = $file->getAttribute('date');
```

```
        my $size = $file->getAttribute('size');
        my @urls = $file->getElementsByTagName('URL');
        my $zip_size=$urls[0]->getAttribute('size');
        my $zip_comp=$urls[0]->getAttribute('compression');
        my $file_url = $urls[0]->textContent;

        print "$array_name\t$type\t$fname\t$date\t$size\t$zip_size\t$file_url\n";
    }
  }
}
```

Appendix: Perl script to fetch the SDK XML file and download files

This is a more advanced version of the above demo script that will fetch the XML feed and get array and file version information. It will also optionally download files and provides the ability to filter based on the array or file types of interest.

```
#!/usr/bin/env perl

## For usage information, run this with a "-h" argument
## For man page style docs, run this with a "-m" argument

## $Id: netaffxSDK.pl,v 1.4 2008/09/24 23:15:55 steve Exp $

use strict;
use vars qw($VERSION);
$VERSION = 1.0;   ## Current version of this script
require  5.005;   ## requires this Perl version or later

=head1 NAME

netaffxSDK.pl - Download and get info about Affymetrix data files via the NetAffx
SDK

=head1 SYNOPSIS

 netaffxSDK.pl -user USERNAME -pass PASSWORD -lic LICENSE [Options]

 To print documentation:
    netaffxSDK.pl -h    (synopsis only)
    netaffxSDK.pl -o    (synopsis + options)
    netaffxSDK.pl -m    (full docs)

 Example command-line options:
    All commands must specify values for -user, -pass, and -lic
    These options are omitted from the following examples.

 -list -save                 List all files, not downloading, but save XML
 -list -array U133           List only files for array names with 'U133'
 -ls -a ht_hg -f csv         List HT human array files in CSV data format
 -ls -a HuEx -t trans -f csv    List Human Exon array transcript csv file
```

```
  -ls -a MoGene -f fasta          List all fasta files for the Mouse gene array
  -ls -dl -a bovine -t probe      List and download only Bovine array probe files
  -dl -a Hs_Prom -f bed           Download human promoter array bed file
  -dl -a Ce -f bpmap -dir tiling  Download C. elegans bpmap files to 'tiling/' dir

  -v -ls -dl -a U133_Plus_2 -dir /home/me/nasdk/ > files.txt 2> nasdk.log
                        List and download HG-U133_Plus_2 files into /home/me/nasdk
                        Save file list to files.txt, script log to nasdk.log


=head1 OPTIONS

B<netaffxSDK.pl> has three required arguments for the user login
credentials. The rest of the arguments are optional, though at a
minimum, either the C<-list> or C<-download> arguments or both would
normally be used.

=over 8

=item B<-user> or B<-u>

Username for affymetrix.com login (required)

=item B<-pass> or B<-p>

Password for affymetrix.com login (required)

=item B<-lic> or B<-l>

License key for NetAffx SDK provided by Affymetrix (required)

Use of this software requires a license.
To get one, contact devnet@affymetrix.com

=item B<-array> <array_name> or B<-a> <array_name>

Array name filter (case-insensitive)  (optional)

Can be just a portion of an array name. Example: C<U133>

Files with names matching a supplied value to the C<-array> argument
will be processed by this script. All other files will be ignored.

=item B<-format> <format_string> or B<-f> <format_string>

File data format filter (case-insensitive) (optional).

Available formats: (subject to change)

    bar bed bgn bp2 bpmap cdf cif clf csv extras fa fasta
    gcc gff mapping mps pgf ppd ppm ppw ps psi tab xml
```

Files with names or annotation type information matching a supplied
value to the C<-format> argument will be processed by this script.
All other files will be ignored.

=item B<-type> <type_string> or B<-t> <type_string>

Data type filter (case-insensitive) (optional)

Available types: (subject to change)

    annot blast ortholog probe consensus control target
    probeset transcript readme design-annot full core extended

Files with names or annotation type information matching a supplied
value to the C<-type> argument will be processed by this script.
All other files will be ignored.

=item B<-download> or B<-dl>

Download files passing filter(s), if any. (optional)

Use the C<-dir> argument to specify a directory location where
downloaded files should be saved (or files will be downloaded into the
current working directory from where the script is executed).

For files that have previously been downloaded into the specified
directory (or current working dir), only files that have been updated
on the server will be downloaded. To determine which files have been
updated and downloaded, use the C<-verbose> command-line option.

=item B<-dir> <directory>

Local directory where to save any downloaded files.
If this is not specified, any downloading files will be
saved in the current working directory.

Also specifies location for the SDK XML file when using the C<-save>
option.

=item B<-save> or B<-s>

Saves the NetAffx SDK XML file locally (by default it is
not saved locally). It will be saved in a file named
C<netAffxSDK.xml> in the directory specified by B<-dir>
or the current working directory if -dir is not specified.

=item B<-list> or <-ls>

Print a tab-delimited list of information about each file from
the SDK XML, or just those files passing any filters, if defined.
Output is sent to STDOUT.

```
Contents of the listing is as follows: in the listing are:

 Column   Contents
 ------  ------------------------
   1      Array Name
   2      Data Type
   3      Uncompressed File Name
   4      Uncompressed File Date
   5      Uncompressed File Size
   6      Compressed File Size
   7      Compressed File URL

=item B<-verbose> or B<-v>

Print a detailed log for the script operation.
Output is sent to STDERR.

=item B<-help> or B<-h>

Print a brief help message and exits.

=item B<-man> or B<-m>

Prints the manual page documentation and exits.

=back

=head1 DESCRIPTION

This is a fancier version of the first NetAffx SDK processing Perl script
presented in the white paper. This version will optionally download files
which have been updated on the Affymetrix server compared to any local copy.
Also, command-line options are provided which enable filtering for just the
file of interest.

This script accesses and extracts information about Affymetrix data
files using the NetAffx SDK, optionally downloading files from
affymetrix.com. It can be used as a command-line tool to maintain
up-to-date local versions of these data files.

It takes user credentials via command-line arguments and optionally
filters the files of interest based on array name and/or file
format. Files passing any filters are listed if the C<-list> option is
supplied and downloaded if the C<-download> option is supplied.

For more information about the NetAffx SDK:
http://www.affymetrix.com/support/developer/netaffx_sdk/netaffx_sdk_overview.aff
x

Use of this software requires an Affymetrix Developers Network license key.
To get one, contact devnet@affymetrix.com
```

```
=head1 DEPENDENCIES

The following Perl modules are required by this script:

   XML::LibXML
   LWP::Simple
   URI::URL
   IO::File
   Getopt::Long
   Pod::Usage

=head1 AUTHORS

 Steve Chervitz <steve_chervitz@affymetrix.com>
 Ron Shigeta <ron_shigeta@affymetrix.com>

=head1 REVISION

 $Id: netaffxSDK.pl,v 1.4 2008/09/24 23:15:55 steve Exp $

=head1 COPYRIGHT

Copyright 2008. Affymetrix, Inc. All rights reserved.

This software is covered by the terms of use or license located at
http://www.affymetrix.com/site/terms.affx

=cut

use XML::LibXML;
use LWP::Simple;
use URI::URL;
use IO::File;
use Getopt::Long;
use Pod::Usage;

my $license;
my $username;
my $password;
my $array_filt;  # array name filter
my $format_filt; # file format filter
my $type_filt;   # data type filter
my $filtering;   # boolean: are any filters defined?
my $save_sdk;    # should the SDK XML be saved to local file?
my $sdkfile = "NetAffxSDK.xml";  # name of SDK XML, if saving.
my $download;    # should files be downloaded?
my $local_dir;   # where to save downloaded files and SDK.
my $list;       # list all files from SDK XML, after any filtering
my $help;
my $opt;
my $man;
my $verbose;
```

```
select(STDOUT); $|=1;

GetOptions(
          "l|lic:s" => \$license
        , "u|user:s" => \$username
        , "p|pass:s" => \$password
        , "a|array:s" => \$array_filt
        , "f|format:s" => \$format_filt
        , "t|type:s" => \$type_filt
        , "dl|download" => \$download
        , "ls|list" => \$list
        , "d|dir:s" => \$local_dir
        , "s|save" => \$save_sdk
        , "v|verbose" => \$verbose,
        , "h|help" => \$help
        , "o|opt" => \$opt
        , "m|man" => \$man
);

pod2usage(-exit=>0, -verbose=>0) if $help;
pod2usage(-exit=>0, -verbose=>1) if $opt;
pod2usage(-exit=>0, -verbose=>2) if $man;

unless (defined $license and defined $username and defined $password) {
   pod2usage(-message =>"\nFATAL: Missing required arguments",
          -exit=>1, -verbose=>1);
}

$local_dir = '.' unless defined $local_dir;

unless (-d $local_dir) {
   mkdir $local_dir or
      die "\nFATAL: Unable to create local directory $local_dir: $!\n";
}

# Get the SDK XML file.
# Always retrieve it, even if there may be a local copy previously downloaded.
my $sdkurl =
"http://www.affymetrix.com/analysis/downloads/netaffxapi/GetFileList.jsp?licence=$
{license}&user=${username}&password=${password}";

# Grab the content of the SDK XML file in memory
my $url = url($sdkurl);
$url->query_form('license'  => $license,
             'user'     => $username,
             'password' => $password);

my $sdk_content;
unless (defined ($sdk_content = get($url) ))  {
   die "\nFATAL: Could not get content for SDK XML from URL=$url ($!))\n";
}
```

```
# Begin verbose output:
if ($verbose) {
   print STDERR "$0, $VERSION\n";
   printf STDERR "Started at: %s\n\n", scalar(localtime(time));
}

if ($save_sdk) {
   my $sdkpath= "$local_dir/$sdkfile";
   my $fh = new IO::File(">$sdkpath") or
      die "\nFATAL: Can't create file to save SDK to $sdkpath: $!\n";
   print $fh $sdk_content;
   print STDERR "Saved NetAffx SDK XML to $sdkpath\n" if $verbose;
   $fh->close;
}

my $parser = XML::LibXML->new();
my $tree = $parser->parse_string($sdk_content) or
   die "\nFATAL: SDK XML data did not parse successfully. Possibly incomplete
download.";

my $root = $tree->getDocumentElement;
my @GeneArrays = $root->getElementsByTagName('Array');

$filtering=1 if defined $array_filt or defined $format_filt or defined $type_filt;

##
## Main loop
##

# Collects information about the file, printing out
# information if using verbose mode (-v).
#
# This info that can be checked against a previously downloaded
# version of the file to determine if the file should be downloaded.
# When using the -download option, files should only download if
# they have changed on the server.
#
# The size attribute can be used to verify that the file
# has been completely downloaded.

my $tcount  = 0;  # Count total files in SDK XML
my $fcount  = 0;  # Count total files in SDK XML passing any filters
my $dlcount = 0;  # Count download files
my $mcount  = 0;  # Count mirrored files

# Print out column headers when in -list mode:
if ($list) {
   printf qq{"%s"\t}x7, 'Array Name'
                      , 'Data Type'
                      , 'Uncompressed File Name'
                      , 'Uncompressed File Date'
```

```
                , 'Uncompressed File Size'
                , 'Compressed File Size'
                , 'Compressed File URL';
   print "\n";
}

foreach my $array_id (@GeneArrays) {
   my $array_name = $array_id->getAttribute('name');
   my @Ann_node  = $array_id->getElementsByTagName('Annotation');
   foreach my $ann (@Ann_node) {
      $tcount++;
      my $type = $ann->getAttribute('type');
      my @Files = $ann->getElementsByTagName('File');
      foreach my $file (@Files) {
         my $fname = $file->getAttribute('name');
         my $date = $file->getAttribute('date');
         my $size = $file->getAttribute('size');
         my @urls = $file->getElementsByTagName('URL');
         my $zip_size=$urls[0]->getAttribute('size');
         my $zip_comp=$urls[0]->getAttribute('compression');
         my $file_url = $urls[0]->textContent;

         # Check the file name against any defined filters
         my $pass = filter_entry($fname, $type);
         next unless $pass;
         $fcount++;
         if ($list) {
            print
"$array_name\t$type\t$fname\t$date\t$size\t$zip_size\t$file_url\n";
         }

         next unless $download;
         download_file($file_url);
      }
   }
}

if ($verbose) {
   printf STDERR "\nSummary:\n";
   printf STDERR "%5d total files defined in the NetAffx SDK XML\n", $tcount;
   printf STDERR "%5d files passed filters\n", $fcount if $filtering;
   printf STDERR "%5d downloaded files to $local_dir\n", $dlcount if $download;
   printf STDERR "%5d mirrored files to $local_dir\n", $mcount if $download;
}

printf STDERR "\nDone at %s\n", scalar(localtime(time)) if $verbose;

##
## Subroutines
##

# Examines the file name against any defined filters
```

```
sub filter_entry {
   my ($name, $type) = @_;
   my $pass = 1;
   my $fail = 0;

   if (defined $array_filt) {
      $pass = 0 unless $name =~ /$array_filt/oi;
      $fail = 1 unless $pass;
   }
   if (defined $format_filt and ! $fail) {
      $pass = 0 unless $name =~ /$format_filt/oi or $type =~ /$format_filt/oi;
      $fail = 1 unless $pass;
   }
   if (defined $type_filt and ! $fail) {
      $pass = 0 unless $name =~ /$type_filt/oi or $type =~ /$type_filt/oi;
      $fail = 1 unless $pass;
   }
   return $pass;
}

sub download_file {
   my ($url) = @_;
   my $name;

   if ($url =~ m|/([^/]+)$|) {
      $name = $1;
   } else {
      die "FATAL: Cannot parse file name from URL: $url\n";
   }

   # Filenames are the unzipped
   my $local_path = "$local_dir/$name";

   print STDERR "\nRetrieving $name to dir=$local_path ..." if $verbose;
   my $mirror_result = mirror($url, $local_path);

   unless (-s "$local_path") {
      die "\nFATAL: Failed to save file locally to $local_path\n";
   }

   if ( $mirror_result == RC_NOT_MODIFIED) {
      print STDERR " up-to-date (not downloaded)\n" if $verbose;
      $mcount++;
   }
   else {
      print STDERR " downloaded\n" if $verbose;
      $dlcount++;
   }
}
```