[How can I use UAD for a special archive file ?]

(1) Introduction

To work with an archive file, UAD needs an expander/decompresser to decompress/open it before reading its contents. UAD already included many of common expanders in its module by default, whose codes are customized and optimized to the whole structure of UAD. For example, unzip for zip and unrar for rar codes are optimized and embeded in UAD.

Therefore, you can use UAD without any addtional work if your archive file is one of the embeded archive file type in UAD. Currently, the embedded expanders in UAD are as follows, and we have a plan to include more new expanders in UAD, such as 7-zip,...;

tar (on), Z (on), gz (on), bz2 (on), zip (on), TNEF (on), Hqx8 (on), SIT (on), RAR (on), CAB (on)

As you can easily guess, including/embedding ALL (100%) the expanders in UAD is almost impossible, because there are more than hundreds of archive types in IT market in the world, and the number is continuously increasing now. If UAD is trying to include all those expanders's codes in UAD, the size of UAD will be exploded and we'll have some serious performance issue. Furthermore, there are some legal license issues with some archive-expander-vendors to use their codes in UAD.

Therefore, we took a flexible approach for this issue and we provided the interface for the special archive file types. In other words, we allowed customers to use special expanders with UAD IF THEY HAVE THE PROPER EXPANDER (execution file) using UAD's -e (or --external) option.
For example, if you have cpio archive, you can use UAD IF YOU HAVE an cpio. Same with StuffIt or bzip2, etc.

(2) Supplemetary scripts for the special archive file type

We provides three types of supplementary example scripts (uad-helper.{su,chroot,chroot+setuidgid}) for customers to handle the special archive file types with UAD. All the information is included in VSTK (opt/vstk/example_scripts/uad_external) and you can use/modify/customize the codes according to your situation. Please read the README file before using the scripts.

Also, please note that those scripts, programs, and patches are only examples, and must be modified to suit your requirements before using them. CyberSoft, Inc. is not responsible for any damage, be it physical or mental, directly or indirectly caused by these examples.

In section (3) below, you can see the brief description and code of the "uad-helper.su" which is one of three supplementary example scripts we provide for this issue.

(3) uad-helper.su

uad-helper.su is a simple example which uses "su ${NOBODY} -c ..." (if run as root) to run external archive extractors. It is not very secure: archives containing absolute paths (starting with "/") or paths containing "../" may cause system or user files to be overwritten. (-> If you need more secure approach, please use uad-helper.chroot or uad-helper.chroot+setuidgid)

3.1 Sample script for cpio, rar, stuffit, and bzip2 formats.
(Note. lower version of rar is embeded in UAD)

```
#! /bin/sh
#
# Version: Tue Aug 28 10:39:14 EDT 2001
#
# Example external uad-helper script which handles
# cpio, rar, stuffit, and bzip2 formats.
#
### Uses `su' and does not use `chroot'
#
# Usage (invoked by uad with -e or --external option):
#
#    uad -e path_to_uad-helper ...
#
# will cause uad to run uad-helper with three arguments:
#
#    uad-helper infile outfile infiletype
#
# for any file or expanded component file which appears to be an archive
# but can not be extracted internally by uad, or for which the file type
# is unknown to uad.  outfile will exist as an initially empty file for
# uad-helper to write its expansion of infile.  Upon return, if outfile
# is still empty, then uad assumes that uad-helper could not extract infile,
```

```
# and no further processing of infile will be performed; if outfile is
# not empty, then it will be processed for potential further extractions.
# To handle archives containing multiple files, uad-helper can, for example,
# use `tar' to write to outfile.
#
# Please check the SECURITY WARNING, CAUTION, and CONFIGURATION sections below,
# and modify this script to suit your requirements before trying to use it.
# CyberSoft, Inc. is not responsible for any damage, be it physical or mental,
# caused or indirectly caused by this example script.
#
# Copyright (c) August 2001 by CyberSoft, Incorporated.
###
##
# SECURITY WARNING:
#
# Archives can contain files with absolute pathnames (starting with /)
# or containing "../".  Extracting the archive contents may cause user
# or system files to be overwritten.
#
# If uad and this script are run as root, then "su ${NOBODY} -c ..." is
# used here when executing the external helper programs.  ${NOBODY} should be
# a userid (such as nobody) which has no privilege to overwrite important
files.
# A more secure but more complicated solution would be to perform the archive
# extraction in a chroot environment.
#
# If uad and this script are not run as root, then they should be run
# as some userid like nobody to prevent overwriting existing files.
###
##
# CAUTION:
#
# Note that if we write anything to outfile here,
# then this script may be reinvoked (potential infinite loop)
# if the outfile type is not handled by uad internally.
###
##
# REFERENCES:
#
#   cpio    - ftp://prep.ai.mit.edu/pub/gnu/cpio/
#   bzip2   - http://sourceware.cygnus.com/bzip
#   unrar   - ftp://ftp.elf.stuba.sk/pub/pc/pack/unrar250.zip
```

```
#   unstuff - http://www.stuffit.com/download_info.html
##
###

PATH="/usr/bin"
export PATH

###
##
# CONFIGURATION:
#
# userid for extraction if running as root:
#
NOBODY="frodo"
#
# paths and options for cpio, bzip2, unrar, and unstuff:
#
CPIO="/usr/bin/cpio -ivdm"
BZIP2="/usr/local/bin/bzip2 -d"
UNRAR="/usr/local/bin/unrar e -o- -r"
UNSTUFF="/usr/local/stuffit/bin/unstuff -d=. -q"
#
# comment this out to enable the script:
#
echo "uad-helper: $@" 1>&2
echo "$0: not configured." 1>&2
exit 0
##
###

# We must not write anything to stdout, since that could interfere with
# uad smartscan output, so we redirect stdout to /dev/null
#
exec >/dev/null

# check for root and set umask
#
ROOT=`id | grep "uid=0("`

umask 077

# debug:
```

```
#
# echo "  $0:"            1>&2
# echo "    args: $@"     1>&2
# echo "    ROOT = ${ROOT}" 1>&2

# set current pwd
#
P=`pwd`

# set infile path from first argument
#
case "$1" in
  /*) infile="$1";;
   *) infile="$P/$1";;
esac

# set outfile path from second argument
#
case "$2" in
  /*) outfile="$2";;
   *) outfile="$P/$2";;
esac

# set infiletype from third argument
#
infiletype="$3"

# debug:
#
# echo "    infile     = \"${infile}\""   1>&2
# echo "    outfile    = \"${outfile}\""  1>&2
# echo "    infiletype = \"${infiletype}\"" 1>&2

# set helper tmp dir
#
UADHTMPDIR="/tmp/uad-helper.$$"

###
# function to remove the helper tmp dir
#
rmtmpdir ()
{
```

```
  cd /tmp && rm -rf "${UADHTMPDIR}"
}
#
###

###
# catch signals and normal exit
#
trap 'rmtmpdir; exit' 0 1 2 3 4 5 6 7 8 10 12 13 14 15
#
###

# extract the files, based on the input file type
#
case "${infiletype}" in

  # Trust uad's infiletype to detect cpio
  #
  # We should probably check for byte-swapped, etc.
  #
  # Here are the possible ways uad could report "cpio":
  #
  #   "cpio archive"
  #   "byte-swapped cpio archive"
  #   "ASCII cpio archive (pre-SVR4 or odc)"
  #   "ASCII cpio archive (SVR4 with no CRC)"
  #   "ASCII cpio archive (SVR4 with CRC)"
  #
  *cpio*)

        echo "uad-helper: running cpio..." 1>&2

        mkdir -m 0700 "${UADHTMPDIR}" || exit 1

        cd "${UADHTMPDIR}" || exit 1

        if [ -n "${ROOT}" ]; then

            chown "${NOBODY}" .
            su "${NOBODY}" -c "${CPIO}" <"${infile}" 2>&1

        else
```

```
            ${CPIO} <"${infile}" 2>&1

        fi | egrep '^/|"/|../' |

           sed "s;^;uad-helper: WARNING - bad path in cpio archive ${infile}:
;" 1>&2

        # debug:
        #
        # echo "    running tar..." 1>&2

        # create tar archive for further processing by uad
        #
        tar cf "${outfile}" . || exit 1

        ;;

  # Trust uad's infiletype to detect StuffIt
  #
  StuffIt*)

        echo "uad-helper: running unstuff..." 1>&2

        mkdir -m 0700 "${UADHTMPDIR}" || exit 1

        cd "${UADHTMPDIR}" || exit 1

        # unstuff seems to ignore leading / in archive names,
        # so we'll refer to ${infile} using a symlink
        #
        LINK="unstuff-link.$$"

        # ${out} will capture unstuff stdout and stderr
        #
        out="unstuff-stdout-stderr.$$"

        if [ -n "${ROOT}" ]; then

            chown "${NOBODY}" .

            COPY=""
```

```
        # must insure that ${infile} is readable by ${NOBODY},
        # probably have to make a copy of it...
        #
        su "${NOBODY}" -c "test -r ${infile}"

        if [ "$?" != 0 ]; then
            COPY="unstuff-copy.$$"
            cp "${infile}" "${COPY}"
            chmod 644 "${COPY}"
            infile="${COPY}"
        fi

        # debug:
        #
        # echo "    COPY = ${COPY}" 1>&2

        ln -s "${infile}" "${LINK}"

        su "${NOBODY}" -c "${UNSTUFF} ${LINK}" >"${out}" 2>&1

        if [ -n "${COPY}" ]; then
            rm "${COPY}"
        fi

    else

        ln -s "${infile}" "${LINK}"

        ${UNSTUFF} "${LINK}" >"${out}" 2>&1

    fi

    rm "${LINK}"

    # How to tell if unstuff failed?  Just have the one ${out} file?
    #
    count=`ls | wc -l`

    if [ "${count}" -eq 1 ]; then
        echo "uad-helper: unstuff failed:" 1>&2
        cat "${out}" 1>&2
```

```
            exit 1
        fi

        # debug:
        #
        # echo "    running tar..." 1>&2

        # create tar archive for further processing by uad
        #
        tar cf "${outfile}" . || exit 1

        ;;

    # for other infiletypes, check the file header signature ourselves
    #
    *)

        # bzip2 has a 3-byte header signature
        # rar has a 4 or 7 byte signature
        #
        header=`dd if="${infile}" ibs=1 count=4 2>/dev/null`

        case "${header}" in

            # bzip2
            #
            BZh*)

                # debug:
                #
                echo "uad-helper: running bzip2..." 1>&2

                if [ -n "${ROOT}" ]; then

                    cd /tmp

                    su "${NOBODY}" -c "${BZIP2}" <"${infile}" >"${outfile}" ||
exit 1

                else

                    ${BZIP2} <"${infile}" >"${outfile}" || exit 1
```

```
        fi

        ;;

    # rar
    #
    RE~\^|Rar!)

        # debug:
        #
        echo "uad-helper: running unrar..." 1>&2

        mkdir -m 0700 "${UADHTMPDIR}" || exit 1

        cd "${UADHTMPDIR}" || exit 1

        # ${out} will capture rar archive comments
        #
        out="unrar-stdout-stderr.$$"

        if [ -n "${ROOT}" ]; then

            chown "${NOBODY}" .

            COPY=""

            # must insure that ${infile} is readable by ${NOBODY},
            # probably have to make a copy of it...
            #
            su "${NOBODY}" -c "test -r ${infile}"

            if [ "$?" != 0 ]; then
                COPY="unrar-copy.$$"
                cp "${infile}" "${COPY}"
                chmod 644 "${COPY}"
                infile="${COPY}"
            fi

            # debug:
            #
            # echo "    COPY = ${COPY}" 1>&2
```

```
                    su "${NOBODY}" -c "${UNRAR} ${infile}" >"${out}" 2>&1

                    if [ -n "${COPY}" ]; then
                        rm "${COPY}"
                    fi

                else

                    ${UNRAR} "${infile}" >"${out}" 2>&1

                fi

                # How to tell if unrar failed?  Just have the one ${out} file?
                #
                count=`ls | wc -l`

                if [ "${count}" -eq 1 ]; then
                    echo "uad-helper: unrar failed:" 1>&2
                    cat "${out}" 1>&2
                    exit 1
                fi

                # debug:
                #
                # echo "    running tar..." 1>&2

                # create tar archive for further processing by uad
                #
                tar cf "${outfile}" . 1>&2 || exit 1

                ;;

        esac


        ;;

esac

exit 0
```