コードレビュー

Vol.20

CODE Review

USP 研究所技術研究員
written by 大内智明

外部から取得したファイルを企業内ネットワークに送信する方法を説明します。

外部から取得したファイルを企業内ネットワーク内のサーバに送信

今回は、発注業務で発生する、企業内・外のデータ IF について説明します。

発注サーバは、発注データを IF サーバを経由して、企業外にある取引先に送信します。すると納品データが、再度、IF サーバを経由して伝票サーバに配信されます。伝票サーバでは、受信した納品データと発注データを確認して検収処理を行います（図1）。

今回は、この一連の流れの中の「IF サーバを経由して伝票サーバへ配信される」という部分について解説します。図1では❺に該当します。

技術的な概要

ユニケージ内の実データファイルとセマフォファイルの関係から、未送信ファイルを相手サーバに送信する処理を行います。ファイルの生成から送信するまでの順番は図2に示す通りです。

Point

- セマフォと実データのファイル名は、同じにします。
- 全ての処理は、セマフォファイルを起点にして、送信処理、未送信から送信済みへのステータス変更、ファイル有無の確認を行います。

図1　データ IF 図

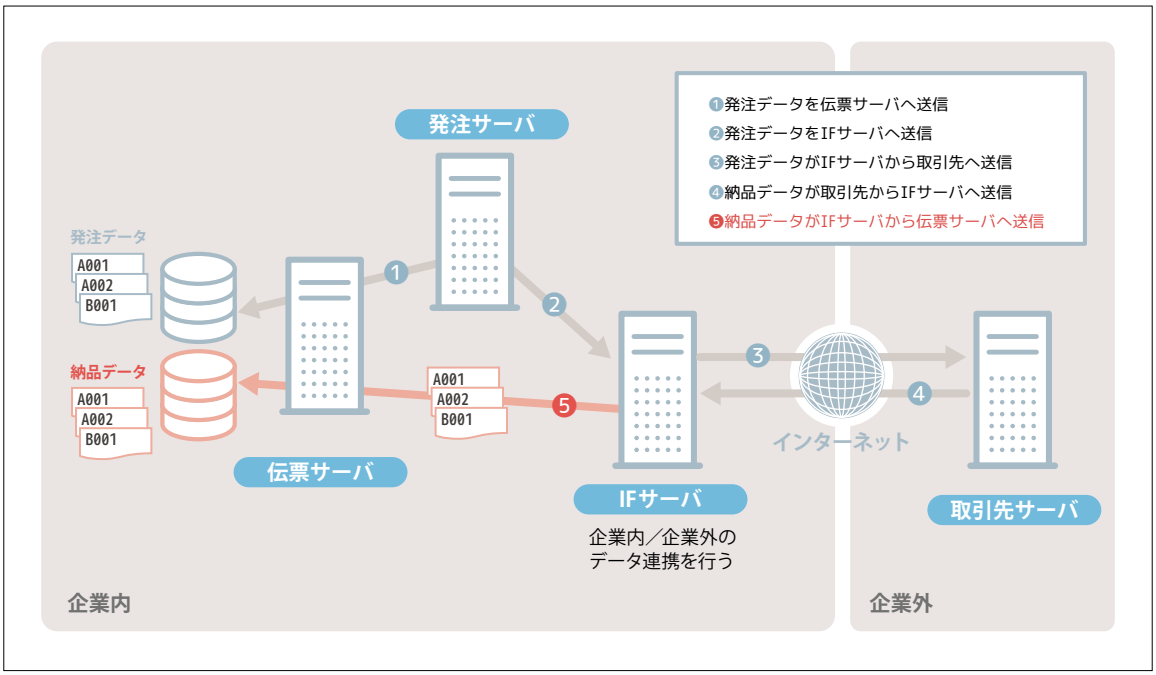
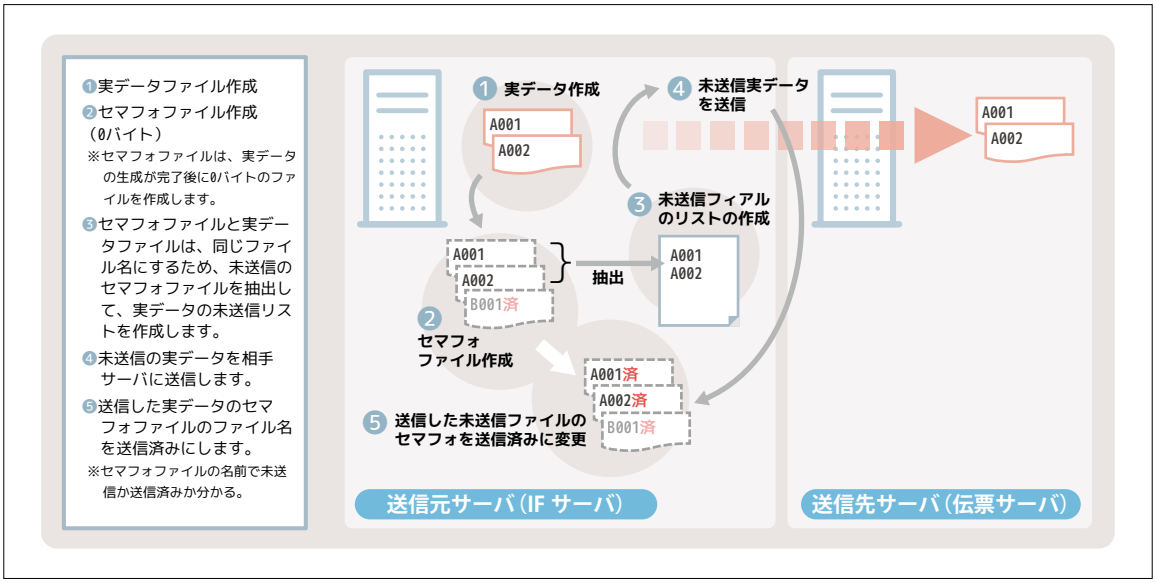


図2　実データをセマフォファイル



サーバ間をファイル転送する方法として、FTP、rsync、scp があります。サーバ間でファイルの更新を行う際には、同じファイルを何度も更新するような場合には差分更新を行う rsync が有効です。今回は、同じファイルは、一度しか送信しないので scp を使用します。

リスト　外部から取得したファイルを内部システムに転送する

```
#!/bin/bash -xv
# システム名     :ＵＳＰシステム
# サブシステム名:外部I/F
# 業務名         :外部I/Fから内部システムに転送
# 備考(Usage)    :TRANSFER.LOOPIF.EDI [YYYYMMDD]
# シェル名       :TRANSFER.LOOPIF.EDI
# 作成日         :20xx/xx/xx
# 会社名         :USP
# 作成者         :Y.Shinmi

# 走行ログの記録
logd=${HOME}/LOG
logf=${logd}/LOG.$(basename $0).$(date +%Y%m%d)_$(date +%H%M%S)_$$
echo   "${logf}" &> /dev/null
exec 2> ${logf}

#//////////////////////////////////////////////////////////////////////////////
# 初期設定
#//////////////////////////////////////////////////////////////////////////////
# パスの定義: sh対応､全てのパスを通す
export PATH=/home/UTL:/home/TOOL:/${PATH}
export LANG=ja_JP.UTF-8

tmp=/tmp/$$-$(basename $0)_$(date +%Y%m%d%H%M%S)
hostname=$(ghostname)
semd="${HOME}/SEMAPHORE"                                # セマフォディレクトリ

sday=$(date +%Y%m%d)

# 引き数の確認
[ $# -eq 1 ] && sday=$1

```

```
# エラー時の終了処理定義
ERROR_EXIT(){
  touch ${semd}/$(basename $0).${hostname}.ERROR.${sday}
  echo "${hostname} $(basename $0)_${sday} ERROR $(date +%Y%m%d%H%M%S) ${logf}" >> ${logd}/UPCNT
  exit 1
}

#---------------------------------
# TRANSFER情報テーブル
#---------------------------------
dtable="/home/FMT/TRANSFERLIST.EDI"

# 当サーバーの送信ディレクトリ
# HULFT 受信処理が出力する先
sndsemd="/home/SND/DATA/SEMAPHORE/HULFT/EDI"
sndd="/home/SND/DATA/HULFT/EDI"

# 相手サーバーの受信ディレクトリ
# 伝票サーバの受信ディレクトリ命名にあわせる
rcvsemd="/home/RCV/DATA/SEMAPHORE/HULFT"
rcvd="/home/RCV/DATA/HULFT"

# 当日起動で当日終了するループ
limittm="${sday}2358"                                    # 期限日時

# 前回セマフォの消去
rm ${semd}/$(basename $0).${hostname}.*.${sday}

# 起動時刻の記録
echo "${hostname} $(basename $0)_${sday} START $(date +%Y%m%d%H%M%S)" >> ${logd}/UPCNT
touch ${semd}/$(basename $0).${hostname}.START.${sday}

#////////////////////////////////////////////////////////////////////////////
# 処理部
#////////////////////////////////////////////////////////////////////////////

[ ! -e ${dtable} ] && ERROR_EXIT

cat ${dtable}                                            |
awk '($1!~/^#/)&&(length($0)>0)'                         |
self 1 2 -                                               |
LANG=C sort -u                                           > ${tmp}-dtable
[ $(plus ${PIPESTATUS[@]}) -eq 0 ] || ERROR_EXIT
# 1:HULFT-ID 2:送信JOBグループ


# 送信相手の受信ディレクトリを整備する
# 失敗した場合はそのまま続行
:> ${tmp}-host_list.0

# 対象ファイルリスト
# 1:セマフォファイル名にあるHULFT-ID 2:送信JOBグループ
for jobg in $(cat ${tmp}-dtable | self 2 - | LANG=C sort -u) ; do
  # 送信JOBグループ単位で処理
  for jobg in $(cat ${tmp}-dtable | self 2 | LANG=C sort -u) ; do
    msctrl -job ${jobg} -ctrl C -print host >> ${tmp}-host_list.0
    [ $(plus ${PIPESTATUS[@]}) -eq 0 ] || ERROR_EXIT
  done
```

監視した処理でエラーが発生すると、異常終了する

self はユニケージコマンド
画面1参照。

msctrl はユニケージコマンド
サーバの正副情、役割、ホスト名などの情報
が取得できる。

```
 91 done
 92
 93 cat ${tmp}-host_list.0                  |
 94 # 自ホストは除外
 95 awk '$1!="'${hostname}'"'               |
 96 LANG=C sort -u                          > ${tmp}-host_list
 97 [ $(plus ${PIPESTATUS[@]}) -eq 0 ] || ERROR_EXIT
 98
 99 #---------------------------------------------
100 # 相手側サーバに必要なディレクトリを作成する
101 #---------------------------------------------
102 for hostnm in $(cat ${tmp}-host_list) ; do
103   ssh ${hostnm} mkdir -p ${rcvd} ${rcvsemd} < /dev/null
104   [ $(plus ${PIPESTATUS[@]}) -eq 0 ] || ERROR_EXIT
105 done
106
107 ######################################
108 # 無限ループ
109 ######################################
110
111 while true ; do
112
113   #-------------------------------------------------
114   # 終了チェック
115   #-------------------------------------------------
116   # 翌日になった時点で終了
117   [ $(date +%Y%m%d%H%M) -gt ${limittm} ] && break
118   [ $(mdate ${sday}) -eq $(mdate ${sday}/+1) ] && break
119
120   # 共通の停止セマフォチェック
121   if [ -e ${semd}/$(basename $0).${hostname}.STOP.${sday} ] ; then
122     mv ${semd}/$(basename $0).${hostname}.STOP.${sday} \
123        ${semd}/$(basename $0).${hostname}.STOP_DONE.${sday}
124     break
125   fi
126
127   #-------------------------------------------------
128   # 対象ファイルを転送(未処理のセマフォが駆動軸)
129   #-------------------------------------------------
130   cat ${tmp}-dtable                         |
131   # 1:セマフォファイル名にあるHULFT-ID 2:送信JOBグループ
132   while read hulftid jobg ; do
133
134     # 未処理のセマフォ
135     echo ${sndsemd}/${hulftid}*             |
136     tarr*                                   |
137     ugrep -v 'SUMI'                         |
138     ugrep -v '\*'                           > ${tmp}-notsend
139     [ $(plus ${PIPESTATUS[@]}) -eq 0 ] || exit 1
140
141     [ ! -s ${tmp}-notsend ] && continue
142
143     # 対応するホスト名を取得
144     :> ${tmp}-host_list.0
145     for jobg in $(cat ${tmp}-dtable | self 2 | LANG=C sort -u) ; do
146       msctrl -job ${jobg} -ctrl C -print host >> ${tmp}-host_list.0
147       [ $(plus ${PIPESTATUS[@]}) -eq 0 ] || exit 1
148     done
```

❶

無限ループ処理になるため、終了条件になるとシェルが正常終了している。

終了条件1
・実行開始日付の翌日になると終了します。

終了条件2
・停止フラグが存在すると終了。

tarr はユニケージコマンド
横型のデータを縦型に展開する。

ugrep はユニケージコマンド
GNU grep のラッパー。

❷

```
149
150     cat ${tmp}-host_list.0                  |
151     # 自ホストは除外
152     awk '$1!="'${hostname}'"'               |
153     LANG=C sort -u                          > ${tmp}-host_list
154     [ $(plus ${PIPESTATUS[@]}) -eq 0 ] || exit 1
155
156     [ ! -s ${tmp}-host_list ] && continue
157
158     # 送信対象ホスト分のループ
159     for hostnm in $(cat ${tmp}-host_list) ; do
160       # セマフォのループ
161       for sndsem in $(cat ${tmp}-notsend) ; do
162         # ファイル転送，セマフォと同名の実ファイル
163         scp -p  ${sndd}/$(basename ${sndsem}) ${hostnm}:${rcvd}/
164         [ $(plus ${PIPESTATUS[@]}) -eq 0 ] || exit 1
165
166         # ファイル転送
167         scp -p ${sndsem} ${hostnm}:${rcvsemd}/
168         [ $(plus ${PIPESTATUS[@]}) -eq 0 ] || exit 1
169       done
170     done
171
172     #-------------------------------
173     # 済みにする
174     #-------------------------------
175     for semf in $(cat ${tmp}-notsend) ; do
176       mv ${semf} ${semf}.${sday}.SUMI
177       [ $? -eq 0 ] || exit 1
178
179       mv ${sndd}/$(basename ${semf}) ${sndd}/$(basename ${semf}).${sday}
180     done
181   done   #------ 対象ファイルリストのscpループ
182
183   # ループ待ち時間
184   sleep 180
185 :;
186 done
187 [ $? -ne 0 ] && ERROR_EXIT
188
189 #-----------------------------------------
190 # 古いセマフォと実ファイル削除処理
191 #-----------------------------------------
192
193 cat ${tmp}-dtable                       |
194 # 1:セマフォファイル名にあるHULFT-ID 2:送信JOBグループ
195 while read hulftid jobg ; do
196   # 古いセマフォを削除
197   echo ${sndsemd}/${hulftid}*.SUMI        |
198   tarr                                    |
199   ugrep -v '\*'                           |
200   # 本ファイルはサーバ間連動間につかうもの
201   # 同じ内容のHULFT受信ふぁいるは別途保存しているため
202   # 長期間の保持は不要
203   xargs -I% find % -type f -ctime +7      > ${tmp}-rmsemlist
204   [ $(plus ${PIPESTATUS[@]}) -eq 0 ] || exit 1
205
206   for rmsemf in $(cat ${tmp}-rmsemlist) ;do
```

```
207     rm -f ${rmsemf}
208
209     # 末尾.SUMI削除して実ファイル名を取得
210     rmfile=${sndd}/$(basename ${rmsemf} | sed 's/\.SUMI//g')
211     # 実ファイル削除
212     rm -f ${rmfile}
213   done
214 :;
215 done
216 [ $? -eq 0 ] || ERROR_EXIT
217
218 #////////////////////////////////////////////////////////////////////////
219 # 終了処理
220 #////////////////////////////////////////////////////////////////////////
221
222 # 終了時刻の記録
223 echo "${hostname} $(basename $0)_${sday} END $(date +%Y%m%d%H%M%S)" >> ${logd}/UPCNT
224 touch ${semd}/$(basename $0).${hostname}.END.${sday}
225
226 # 終了
227 set +xv
228 rm -f ${tmp}-*
229 echo "$(basename $0) exit 0"
230 set -xv
231 exit 0
```

❸

画面 1　self

```
指定したフィールドのデータを取り出す
$ cat data
0000000 浜地______ 50 F 91 59 20 76 54
0000001 鈴田______ 50 F 46 39 8  5  21
0000003 杉山______ 26 F 30 50 71 36 30
0000004 白土______ 40 M 58 71 20 10 6
0000005 崎村______ 50 F 82 79 16 21 80
$ self 4 2 data
F 浜地______
F 鈴田______
F 杉山______
M 白土______
F 崎村______
```

コードの見どころ

[1] 相手サーバを準備します（❶71 ～105行目まで）。
送信ファイルリストを元に送信先のディレクトリを作成します。

[2] 送信処理を行います（❷127 ～170行目まで）。
送信のリストを作成して、未送信対象サーバに未送信ファイルを送信します。

[3] 送信完了後の処理（❸172 ～216行目まで）。
信が終了したセマフォファイルは、全て送信済みのファイル名に変更します。7日以上前の送信終了したファイルは、消去します。

まとめ

前回号では、サーバ間でファイルの更新を行う場合、更新したファイルだけ配信される rsync コマンドが有効というお話をしました。今回のように更新するファイルはなく、全て新規作成して、データを取り込む場合には、未送信／送信済みのステータスを明確にして、未送信分だけを scp コマンドで実データファイル→セマフォファイルの順に送信するので、セマフォファイルの存在を確認すれば、受信側においても作成途中のデータファイルを取込むことはありません。

ファイルを相手サーバに送信（反映）する方法は、いろいろありますので、使用方法により送信（反映）手段を分けて使う必要があります。