優秀賞

Delphi/400による無線ハンディターミナルのデータ集約の仕組みの実装

寺西 健一 様

大阪シーリング印刷株式会社
IT 推進部
情報システム課
主査

大阪シーリング印刷株式会社
http://www.osp.co.jp/

1927 年創業以来、加工業としての本業に徹した堅実経営を貫き、主に凸版印刷を中心とした原紙製造から印刷までの一貫生産工程を軸に、全国をオンラインで結ぶ営業・生産ネットワークを活用。シール業界のリーディングカンパニーとして、印刷の枠を越えた総合パッケージメーカーとして事業を展開している。

業務課題

印刷工場において､印刷に用いる「版」の入庫を確認し、版管理データとしてIBM i 基幹システムに登録している。

従来の登録業務は、工場内の各作業場所でハンディターミナル（バッチ式）によるデータ読み取り･蓄積の後､ハンディターミナルを PC に設置した置台（通信ユニット）に置く、という 2 段階でデータ転送していたため､手間がかかっていた。

そこで、各作業場所から直接データの収集・登録を行うために、バッチ式に変えて無線ハンディターミナルを導入することとした。

技術課題

従来は 1 台の PC だけで実装された仕組みだったので、実績データのファイル名を固定させて FTP による通信を行っていた。しかし無線による通信に変わったため、任意のタイミングでデータを送信する仕組みの実装が必要となった。

具体的には、以下の 3 点を仕様として組み込む必要があった。

(a) IBM i へのデータ送信を定期的に自動処理する（注 1）
(b) ハンディターミナルからの実績データを 1 つにまとめて IBM i に送信
(c) ハンディターミナルからのファイル名競合の回避（注 2）

(注 1)　ハンディターミナルのデータは、中継用のサーバー区画（シンクライアント)を経由してIBM iに登録し､サーバー区画上で Delphi/400 アプリケーション（待ち受けアプリ）を起動する仕組みとした。中継用に、PC の代わりにシンクライアント区画を利用することにより、セキュリティ面と環境管理に関しても考慮した。【図 1】

(注 2)　複数のハンディターミナルから待ち受けアプリに同時にデータを送信する場合、Delphi/400 の待ち受けアプリ側で受信ファイル名が競合して、処理がロックする問題があった。ハンディターミナル側でも待ち状態になり、ロックが解除できなくなる。

技術課題の解決策

前述の技術課題に対して、Delphi/400 の待ち受けアプリにて、次のように解決した。

(a) IBM i へのデータ送信を定期的に反復処理

ハンディから実績データを IBM i へ送信する頻度を画面上で設定可能とした。具体的には、Timer を実行させる時間間隔を指定できるようにした。【図 2】

(b) ハンディターミナルからの実績データを 1 つにまとめて IBM i に送信

Delphi アプリケーションでは、ハンディから FTP 送信されたフォルダを、指定された時間間隔で検索（ファイル *.txt）し、ファイルがあれば、1 つのテ

図1　データフロー

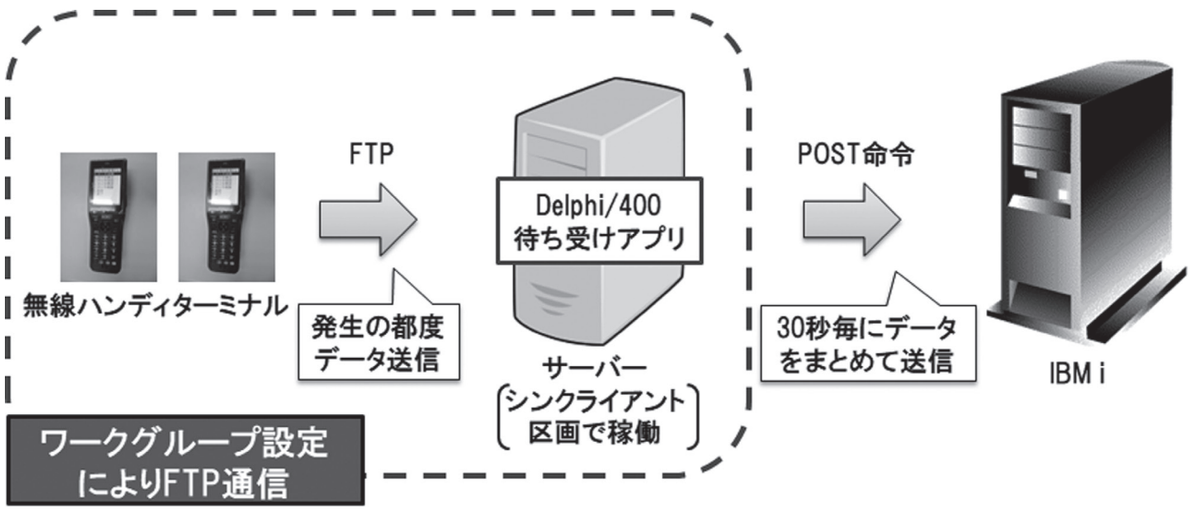
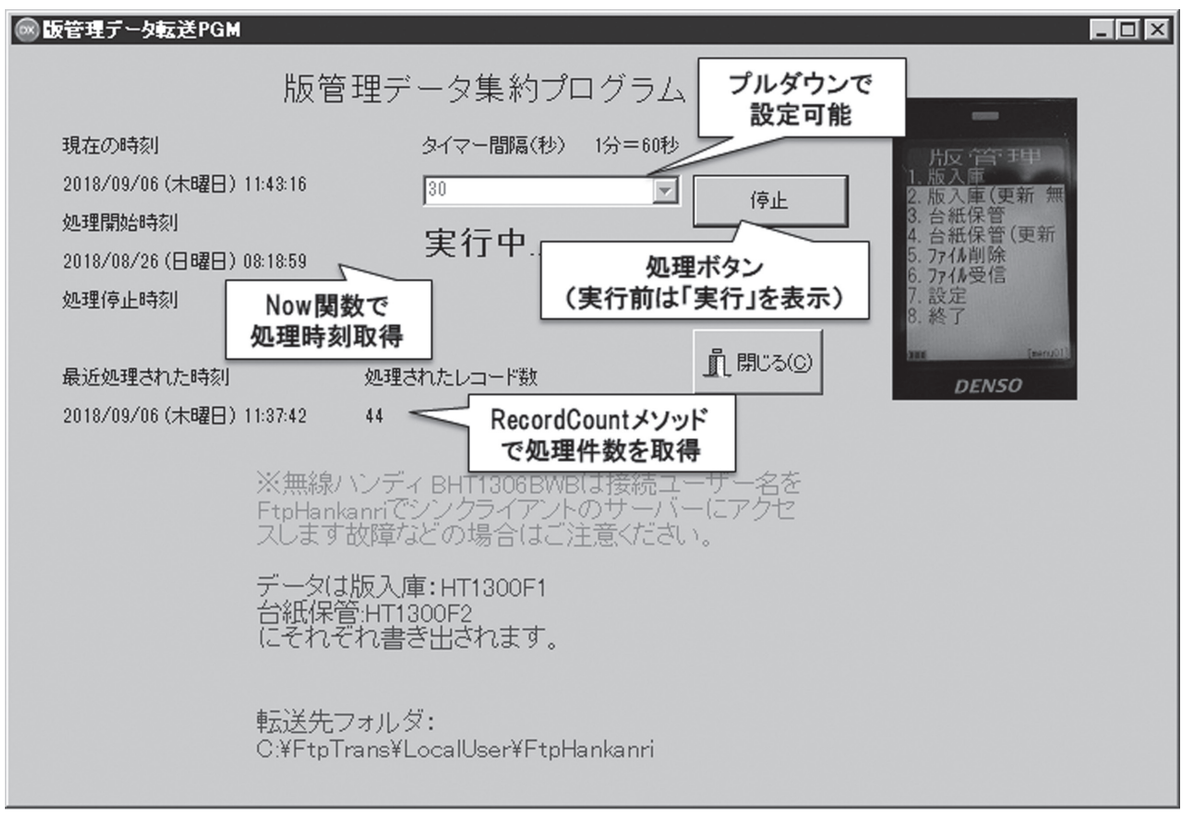


図2　待ち受けアプリ画面



キストデータにまとめる。まとめたデータはTtableコンポーネントを経由して、IBM iへ送信する。【図3】

(c) ハンディターミナルからのファイル名競合の回避

ハンディターミナル側のアプリケーションでは、待ち受けアプリに送信する実績データを、「端末ID＋年月日.txt」という形式にして可変のファイル名とすることで、同一ファイル名が競合することを回避した。

業務課題解決と効果

従来、置台式にて転送処理を行っていたため、置台を設定しているPCが変わるたびに転送を設定する必要があったが、無線化により環境整備の手間がなくなった。

また、現場の担当者からも置台まで行く動作が省略できてよい、との評価が寄せられた。具体的には、1日当たり30分／人の工数短縮という大きな効果を挙げることができた。

M

図3　待ち受けアプリソースコード

```
procedure TForm1.Timer2Timer(Sender: TObject);
var
slCSV:    TstringList;
slLine:   TStringList;
slGCSV:   TstringList;
i:        Integer;
SR:       TSearchRec;
//FND : Integer ;

begin
    slCSV :=TStringList.Create;
    slLine :=TStringList.Create;
    slGCSV :=TStringList.Create;
```

FindFirst命令により、*.txtの内容のファイルをすべて読みながら、一つのファイルへ集約

```
 if FindFirst('C:¥FtpTrans¥LocalUser¥FtpSiage¥*.txt',FaAnyFile,SR) =0 then
  begin
    repeat
//datからの読み込み
    try
        slCSV.LoadFromFile('C:¥FtpTrans¥LocalUser¥FtpSiage¥'+ SR.Name);  //
    except
         on e0: Exception do
         begin
          Abort; // サイレント例外
         end;
    end;
    //レコードの行数分(Count-1)処理を繰り返す。
      for i:=0 to slCSV.Count - 1 do
  begin
      slLine.CommaText:=(slCSV[i]);
       if slLine.Count >1 then
  end;
```

```
//ファイルを保存(Basho.txt)
//       ShowMessage('Basho.txtファイルを保管');
      slGCSV.SaveToFile('c:¥Siage¥Siage.txt');
//処理した件数を画面へ表示
      Label9.Caption:=IntToStr(slGCSV.Count);
//処理した時刻を画面へ表示
       stClock2.Caption := FormatDateTime('yyyy/mm/dd (aaaa) hh:nn:ss',Now());
//ファイルがなくなったら探すのを止める。
    until FindNext(SR) <> 0;
     FindClose(SR);
end;
```

TSearchRecを検索用に定義し、ファイルをすべて読み終わるとClose命令により閉じる

```
 try
   // CSVファイルをオープン
  AssignFile(CSVFile, 'C:¥Siage¥Siage.txt');
  Reset(CSVFile);
   try
     while (eof(CSVFile) = false) do
    begin
       // CSVの読み込み
       ReadLn(CSVFile, str);          // CSVの１行を読み込む
       st.CommaText := str;           // 文字列を分割する
       Number := StrToIntDef(st.Strings[1],0);
       case Number of
         //仕上げチェックの場合
         1 : begin
         //文字列をテーブルに書き込む
         //Table1.Append;                 // レコードの追加
             j := 0;
             TblSGCHKF1.Insert;                 // レコードの挿入
         //for～to～doは初めのパラメータが2つ目のパラメータの
         //値になるまで実行。
         //この場合、ファイルの件数分Table1(I5)にデータを
         //追加する。
           for j := 0 To st.Count-1 do
            TblSGCHKF1.Fields[j].AsString := st.Strings[j];
            TblSGCHKF1.Post;
```

まとめたファイルはTtableコンポーネント経由でPost命令でデータを直接書き出し