業務システム向け FAXサーバーソフト　　スターファクス サーバー エスディーケー

STARFAX® Server SDK

VC 開発向けプログラミング

はじめに

本書は、 STARFAX Server SDK を利用したユーザープログラムの作成方法をご説明しています。
なお、本書はVisual C++ 6.0 を開発ツールとしてプログラムを作成する方を対象としていますが、STARFAX Server SDK とユーザープログラムとのインターフェイスは、ファイルの操作(コピー、移動、削除)、および入出力(読み書き)が基本ですので、他の開発ツールへの応用も容易です。

本書をお読みになる前に、「STARFAX Server SDK セットアップマニュアル」をお読みいただき、STARFAX Server SDK の動作をご理解していただくようお願いいたします。

■ ご注意

本書に登場するシステム名・製品名は、一般に開発メーカーの登録商標です。

本書の構成について

本書は、次のような内容で構成されています。

- 第Ⅰ章　ユーザープログラムの開発について
- 第Ⅱ章　サンプルプログラム
- 第Ⅲ章　クライアントプログラムの開発について
- 第Ⅳ章　FAX 送信命令フォルダを共有して FAX 送信
- 第Ⅴ章　[メール de ＦＡＸ]で FAX 送信

まずは、第Ⅰ章に開発の手順等をご説明していますので、 第Ⅰ章をご覧になってから第Ⅱ章にお進み下さい。

目次

第Ⅰ章

ユーザープログラムの開発について

ユーザープログラムの開発の手順や概要についてご説明しています。

1.1　開発の手順

STARFAX Server SDK を操作するユーザープログラムは、 STARFAX Server SDK が動作していることが前提です。ユーザープログラム開発を行う前に、STARFAX Server SDK 本体のインストールを行い、その操作を簡単に理解しておく必要があります。
それらを考慮して、以下の手順でユーザープログラム開発を行うことをお奨めします。

① 「STARFAX Server SDK オペレーションマニュアル」をお読みください。

●STARFAX Server SDK 本体のインストールを行って下さい。
● 基本的な操作を理解してください。

② *1.2 ユーザープログラムの基本的な動き* をお読みください。

● STARFAX Server SDK のプログラムインターフェイスを理解してください。

③ STARFAX Server SDK で実現したいことを、第Ⅱ章　サンプルプログラム よりお選びください。

● サンプルプログラムを学習してください。

④ ユーザープログラムを作成してください。

● ①~③ を踏まえて、ユーザープログラムの作成・テストを行って下さい。

なお、STARFAX Server SDK プログラミングマニュアル(ファイル操作版)は、記述されている各種項目の詳細に関しては、「STARFAX Server SDK ファイル de FAX」にてご説明しておりますので、本マニュアルと併せてご利用ください。

1.2　ユーザープログラムの基本的な動き

STARFAX Server SDK のプログラムインターフェイスには、以下項目があります。

●STARFAX Server SDK への動作命令
●STARFAX Server SDK の情報の参照

ユーザープログラムとのやりとりは、全てファイルで行います。

補足: [CTRL FOLDER] ... 制御関連インターフェイスフォルダ

■STARFAX Server SDK への動作命令

ユーザープログラムが、特定のフォルダに指定のフォーマットのファイルを置くと、STARFAX Server SDK は一定間隔でファイルを読んで解析し、指定された内容の動作を行います。
以下の動作の命令ができます。

● **FAX 送信**

FAX 送信を行います。

送信命令ファイル ... [CTRL FOLDER]¥SENDMIS¥S*.WSM

● **FAX 情報の削除**

任意の FAX 送受信情報を削除します。

受信情報削除命令ファイル ... [CTRL FOLDER]¥INFORECV¥DELMIS¥DR*.WSM
送信情報削除命令ファイル ... [CTRL FOLDER]¥INFOSEND¥DELMIS¥DS*.WSM
未送信情報削除命令ファイル ... [CTRL FOLDER]¥INFOQUE¥DELMIS¥DQ*.WSM

● **ファイル印刷**

ファイルを印刷します。

印刷命令ファイル ... [CTRL FOLDER]¥PRTMIS¥PD*.WSM

● **メール送信**

メールを送信します。

メール送信命令ファイル　... [CTRL FOLDER]¥EMSMIS¥SD*.WSM

■STARFAX Server SDK の情報の参照

特定のフォルダに書かれたファイルを参照することで、ユーザープログラムは、STARFAX Server SDK の動作状況を知ることができます。
以下の情報を参照できます。

● **FAX 情報**

FAX の送受信情報です。

受信情報インデックスファイル　... [CTRL FOLDER]¥INFORECV¥SfCsRcv.Idx
送信情報インデックスファイル　... [CTRL FOLDER]¥INFOSEND¥SfCsSnd.Idx
未送信情報インデックスファイル　... [CTRL FOLDER]¥INFOQUE¥SfCsQue.Idx

● **FAX 情報の追加、削除状況**

STARFAX Server SDK によって、追加、削除、または、ユーザープログラムによって削除された、FAX 送受信情報(以前との差分)です。

受信情報追加済み通知ファイル　... [CTRL FOLDER]¥NOTICE¥ADDRECV¥AR*.WSM
送信情報追加済み通知ファイル　... [CTRL FOLDER]¥NOTICE¥ADDSEND¥AS*.WSM
受信情報削除済み通知ファイル　... [CTRL FOLDER]¥NOTICE¥DELRECV¥DR*.WSM
送信情報削除済み通知ファイル　... [CTRL FOLDER]¥NOTICE¥DELSEND¥DS*.WSM

● 動作情報

STARFAX Server SDK の起動状況、バージョン等です。

起動情報ファイル　... [CTRL FOLDER]¥SfCsRun.inf

第Ⅱ章

サンプルプログラム

STARFAX Server SDK を操作するサンプルプログラムです。

2.1 FAX 送信

FAX 送信プログラム【SendFax.exe】 は、FAX の送信を行うサンプルプログラムです。本 CD-ROM の以下の位置に入っています。

¥サンプル¥VC6 SP6¥SendFax.exe ... FAX 送信プログラム
¥サンプル¥VC6 SP6¥SendFax¥ ... FAX 送信プログラム 開発プロジェクト

主な仕様、および操作方法は以下の通りです。

① STARFAX Server SDK を起動します。
(STARFAX Server SDK の起動は、「STARFAX Server SDK セットアップアニュアル」P21 参照)

② FAX 送信プログラム【SendFax.exe】 を起動します。
(起動時の作業(カレント)フォルダの指定は特にありません)

③ 相手先(S)ボタンを押して、相手先を指定します。

- 最大４件まで指定できます。(これは、このサンプルプログラムの仕様です)
- 最低限、FAX 番号(A) の入力が必要です。送付状への差込が必要な場合や、送信情報に情報を残したい場合は、会社名(B)以降の項目も入力します。

④ 原稿(D)ボタンを押して、原稿を指定します。

- 最大４つのファイルを指定でき、送信時に連結して送信されます。
(これは、このサンプルプログラムの仕様です)
- 最低限、１つの原稿ファイルを指定する必要があります。ただし、送付状が指定されている場合は、原稿の指定がなくても送付状のみ送信されます。
- 指定できるファイル形式は以下の通りです。
  - TIFF 形式 圧縮なし、修正 CCITT MH 圧縮、CCITT G3 MH 圧縮、CCITT G3 MR 圧縮
    PackBits 圧縮、Class F 圧縮、G4 圧縮、JPEG 圧縮
  - BMP ファイル
  - PCX ファイル
  - DCX ファイル
  - JPEG ファイル
  - テキストファイル
  - FAX ファイル
  - LNK ファイル

⑤ 必要であれば、送付状(C)ボタンを押して、送付状を指定します。
(詳細は「STARFAX Server SDK ファイル de FAX」P6「送信命令ファイルを作成し、FAX 送信を指示する」をご覧下さい)

⑥ 必要であれば、発信元(U)ボタンを押して、発信元情報を指定します。
(詳細は「STARFAX Server SDK ファイル de FAX」 P6「送信命令ファイルを作成し、FAX 送信を指示する」をご覧下さい)

⑦ 送信(G)ボタンを押して、FAX 送信を行います。
この後、FAX 送信が正常に動作していない場合は、 STARFAX ログ管理プログラム でイベントの内容を確認して下さい。

2.1.1 FAXを送信する

FAX送信する為のプログラミング例を、FAX送信プログラム 開発プロジェクトのソースファイルを元にご説明します。
FAX送信プログラム【SendFax.exe】 の仕様、および操作方法については、*2.1 FAX送信* をご覧下さい。
開発プロジェクトは、本CD-ROMの以下の位置に入っています。
¥サンプル¥VC6 SP6¥SendFax¥ ... FAX送信プログラム 開発プロジェクト

■ FAX送信

STARFAX Server SDK のFAX送信命令は､送信命令フォルダに送信命令ファイルを置くことで行います。
送信命令ファイルの詳細は**「STARFAX Server SDK ファイル de FAX」 P6「送信命令ファイルを作成し、FAX 送信を指示する」**に記述されていますので、以下と併せてご覧下さい。
送信命令ファイルを作成する部分は、**ApiSend.cpp** の **MakeSendMission()関数** に記述されています。

ここで重要なことは、直接送信命令ファイルを作成せずに、一時ファイルを作成して、それをリネームしている点です。

これは、ファイルの書き出しに **WritePrivateProfileString()【Win32API】** を使用しているので、OSによって実際のファイルへの書き出しのタイミングが異なり、STARFAX Server SDK が中途半端な状態で読み出すことを防ぐためです。

なお、ソースファイル中に **SFCSSYS_** で始まる名称の関数があります。これらの関数は、サンプルプログラムで汎用的に利用される共通関数です。詳細は、本書の ***付録*** に記述されていますのでご覧下さい。

作成したユーザープログラムが､FAX送信が正常に動作していない場合は､**STARFAX ログ管理プログラム** でイベントの内容を確認して下さい。
そして、その内容を参考にしてプログラムを見直してみてください。

ApiSend.cpp ：

```
//*********************************************************************
// 関数名　　：MakeSendMission
// 機能　　：送信命令ファイル作成
// 呼び出し　：MakeSendMission(SENDFAX_MISSION *pInfo, int *piError )
// 入力　　：pInfo　= 送信命令ファイル作成情報ポインタ
//　　：piError = エラー時、エラー取得用ポインタ
// 出力　　：返値　TRUE　：正常終了
//　　：　　FALSE：失敗
//　　：　　*piError = エラーコード
//　　：
//　　：【エラーコード】
//　　：　　SENDFAX_ERR_NoMisFolder　　... 送信命令フォルダが存在しません
//　　：　　SENDFAX_ERR_GetMisFolder　　... 送信命令フォルダの取得に失敗しました
//　　：　　SENDFAX_ERR_GetTempFolder　... 一時フォルダの取得に失敗しました
//　　：　　SENDFAX_ERR_GetTempFile　　... 一時ファイルの取得に失敗しました
//　　：　　SENDFAX_ERR_MakeMisName　　... 送信命令ファイル名の作成に失敗しました
//　　：　　SENDFAX_ERR_PARAM_INFO　　... 関数引数エラー: 送信命令ファイル作成情報が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_SENDNUM　... 関数引数エラー: 相手先数の指定が 0 です。
//　　：　　SENDFAX_ERR_PARAM_SENDINFO　... 関数引数エラー: 相手先情報が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_FAX　　... 関数引数エラー: FAX 番号が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_DOCorCOVER ... 関数引数エラー: 送信原稿ファイル、送付状ファイルが、
//　　　　ともに指定されていません。
//　　：　　SENDFAX_ERR_PARAM_DOCNAME　... 関数引数エラー: 送信原稿ファイル名が指定されていません。
//　　：
// 特記事項　　：　送信命令ファイル作成を作成します。
// 作成者　　：
// 作成日　　：2005.07.04
//*********************************************************************
BOOL APIENTRY　MakeSendMission(SENDFAX_MISSION *pInfo, int *piError )
{
    BOOL　bRet = FALSE;　// 返値 ... 初期値は失敗

    int iErrorDummy;　// NULL 用ダミーエラーコード設定

    int i;

    BOOL　bTemp = FALSE;　　// 一時ファイルが作成されたか否か

    char　szTempFolder[MAX_PATH+1];　// 一時ファイルフォルダ用
    char　szTempFile[MAX_PATH+1];　　// 一時ファイル用
```

```
char    szSendMisFolder[MAX_PATH+1];    // 送信命令フォルダ用
char    szSendMis[MAX_PATH+1];          // 送信命令ファイル用

SENDFAX_SENDINFO    *pSendInfo; // 相手先情報用
char    *pDocName;              // 送信原稿用

char    szSection[32];  // セクション設定用
char    szKey[32];      // キー設定用
char    szWork[512];    // 作業用

if(!piError ) {
    // エラーコード取得用ポインタにダミー設定
    piError = &iErrorDummy;
}
*piError = SENDFAX_SUCCESS;     // エラーコードに初期値設定 ... 正常終了

////////////////////////////////////////////////////////////////////////////
// (1) 簡単な引数の内容チェック
//

    // 送信命令ファイル作成情報チェック
if(!pInfo ) {
    *piError = SENDFAX_ERR_PARAM_INFO;
    goto    EXIT;
}
    // 相手先数チェック
if(!pInfo->iSendNum ) {
    *piError = SENDFAX_ERR_PARAM_SENDNUM;
    goto    EXIT;
}
    // 相手先情報チェック
for(i = 0 ;  i < pInfo->iSendNum ; i++ ) {
    pSendInfo = *(pInfo->ppSend + i );
    if(!pSendInfo ) {
        *piError = SENDFAX_ERR_PARAM_SENDINFO;
        goto    EXIT;
    }
        // FAX 番号チェック
    if(!pSendInfo->szFax[0] ) {
        *piError = SENDFAX_ERR_PARAM_FAX;
        goto    EXIT;
    }
}
    // 送信原稿ファイル、送付状ファイルチェック
```

```
if(!pInfo->iDocNum && !pInfo->pCoverName ) {
    *piError = SENDFAX_ERR_PARAM_DOCorCOVER;
    goto    EXIT;
}
    // 送信原稿ファイル名チェック
if(pInfo->iDocNum ) {
    for(i = 0 ;  i < pInfo->iDocNum ; i++ ) {
        pDocName = *(pInfo->ppDocName + i );
        if(!pDocName ) {
            *piError = SENDFAX_ERR_PARAM_DOCNAME;
            goto    EXIT;
        }
    }
}

////////////////////////////////////////////////////////////////////////////
// (2)  送信命令フォルダパスのチェック
//

if(SFCSSYS_CheckSubFolder(SFCS_SMODE_SENDMIS ) != SFCS_SUCCESS ) {
    *piError = SENDFAX_ERR_NoMisFolder;
    goto    EXIT;
}

////////////////////////////////////////////////////////////////////////////
// (3)  送信命令フォルダパスを取得
//

if(!SFCSSYS_GetSubFolder(SFCS_SMODE_SENDMIS, szSendMisFolder, MAX_PATH + 1 ) ) {
    *piError = SENDFAX_ERR_GetMisFolder;
    goto    EXIT;
}

////////////////////////////////////////////////////////////////////////////
// (4)  一時ファイル名パスを取得 【重要】
//
//      送信命令ファイルの形式は、INI ファイル形式なので、ファイルの書き出しは、
//    WritePrivateProfileString() 【Win32API】 を使用します。
//    ただ、OSにより、WritePrivateProfileString() 【Win32API】 を使用しての
//    実際のファイルへの書き出しのタイミングが異なる為、STARFAX Engine が
//    中途半端な状態で読み出すおそれがあります。それを防ぐ為に、一旦、一時ファイルを
//    作成し、それをリネームするようにします。
//
if(!GetTempPath(sizeof(szTempFolder), szTempFolder ) ) {
```

```
    *piError = SENDFAX_ERR_GetTempFolder;
    goto    EXIT;
}
if(!GetTempFileName(szTempFolder, "SFCS", 0, szTempFile ) ) {
    *piError = SENDFAX_ERR_GetTempFile;
    goto    EXIT;
}

////////////////////////////////////////////////////////////////////////////
// (5)  一時ファイルへ書き込み
//

bTemp = TRUE;

// セクション名: [SendInfo]  ... 送信のための相手先情報

// ・ Acount     ... アカウント(ユーザが自由に利用できるエリア) (最大 64 バイト)
if(pInfo->szAcount[0] ) {
    WritePrivateProfileString("SendInfo", "Acount", pInfo->szAcount, szTempFile );
}

// ・ Num        ... 送信相手先数
wsprintf(szWork, "%d", pInfo->iSendNum );
WritePrivateProfileString("SendInfo", "Num", szWork, szTempFile );

// セクション名: [Send%d]    ... 送信のための相手先内容  (1～)
for(i = 0 ;  i < pInfo->iSendNum ; i++ ) {
    pSendInfo = *(pInfo->ppSend + i );

    wsprintf(szSection, "Send%d", i + 1 );

    // ・ Fax        ... FAX 番号 (最大 128 バイト)
    WritePrivateProfileString(szSection, "Fax", pSendInfo->szFax, szTempFile );
    // ・ Company    ... 会社名 (最大 128 バイト)
    if(pSendInfo->szCompany[0] ) {
        wsprintf(szWork, "¥"%s¥"", pSendInfo->szCompany );
        WritePrivateProfileString(szSection, "Company", szWork, szTempFile );
    }
    // ・ Division   ... 所属名 (最大 128 バイト)
    if(pSendInfo->szDivision[0] ) {
        wsprintf(szWork, "¥"%s¥"", pSendInfo->szDivision );
        WritePrivateProfileString(szSection, "Division", szWork, szTempFile );
    }
    // ・ Position   ... 役職名 (最大 128 バイト)
```

```
if(pSendInfo->szPosition[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szPosition );
    WritePrivateProfileString(szSection, "Position", szWork, szTempFile );
}
// ・ Name       ... 氏名 (最大128バイト)
if(pSendInfo->szName[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szName );
    WritePrivateProfileString(szSection, "Name", szWork, szTempFile );
}
// ・ Title      ... 敬称 (最大128バイト)
if(pSendInfo->szTitle[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szTitle );
    WritePrivateProfileString(szSection, "Title", szWork, szTempFile );
}
// ・ Telephone  ... 電話番号 (最大128バイト)
if(pSendInfo->szTelephone[0] ) {
    WritePrivateProfileString(szSection, "Telephone", pSendInfo->szTelephone, szTempFile );
}
// ・ ZipCode    ... 郵便番号 (最大128バイト)
if(pSendInfo->szZipCode[0] ) {
    WritePrivateProfileString(szSection, "ZipCode", pSendInfo->szZipCode, szTempFile );
}
// ・ Address1   ... 住所1 (最大128バイト)
if(pSendInfo->szAddress1[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szAddress1 );
    WritePrivateProfileString(szSection, "Address1", szWork, szTempFile );
}
// ・ Address2   ... 住所2 (最大128バイト)
if(pSendInfo->szAddress2[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szAddress2 );
    WritePrivateProfileString(szSection, "Address2", szWork, szTempFile );
}
// ・ FCode      ... Ｆコード番号 (最大20バイト)
if(pSendInfo->szFCode[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szFCode );
    WritePrivateProfileString(szSection, "FCode", szWork, szTempFile );
}
// ・ FreeArea   ... ユーザが自由に利用できるエリア (最大128バイト)
if(pSendInfo->szFreeArea[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szFreeArea );
    WritePrivateProfileString(szSection, "FreeArea", szWork, szTempFile );
}
// ・ Speed      ... 通信速度 ("0":自動, "1":高速, "2":中速, "3":低速)
if(pSendInfo->iSpeed ) {
```

```
        wsprintf(szWork, "%d", pSendInfo->iSpeed );
        WritePrivateProfileString(szSection, "Speed", szWork, szTempFile );
    }
    // ・ Comp      ... 圧縮方式 ("0":自動, "1":MH, "2":MR, "3":MMR)
    if(pSendInfo->iComp ) {
        wsprintf(szWork, "%d", pSendInfo->iComp );
        WritePrivateProfileString(szSection, "Comp", szWork, szTempFile );
    }
    // ・ Ecm       ... エラー訂正 ("0":自動, "1":利用する, "2":利用しない)
    if(pSendInfo->iEcm ) {
        wsprintf(szWork, "%d", pSendInfo->iEcm );
        WritePrivateProfileString(szSection, "Ecm", szWork, szTempFile );
    }
    // ・ Line      ... 送信回線指定 (0～)
    if(pSendInfo->bLine ) {
        wsprintf(szWork, "%d", pSendInfo->iLine );
        WritePrivateProfileString(szSection, "Line", szWork, szTempFile );
    }
    // ・ Priority  ... 優先順位 (0～15)
    if(pSendInfo->bPriority ) {
        wsprintf(szWork, "%d", pSendInfo->iPriority );
        WritePrivateProfileString(szSection, "Priority", szWork, szTempFile );
    }
    // ・ Time      ... 送信時刻 ("YYYYMMDDHHMMSS")
    if(pSendInfo->bTime ) {
        wsprintf(szWork, "%s", pSendInfo->szTime );
        WritePrivateProfileString(szSection, "Time", szWork, szTempFile );
    }
}

// セクション名: [Doc]      ... 送信原稿
// ・ Num       ... 原稿数
if(pInfo->iDocNum ) {

    wsprintf(szWork, "%d", pInfo->iDocNum );
    WritePrivateProfileString("Doc", "Num", szWork, szTempFile );

    // ・ Name%d    ... 原稿ファイルパス (1～)
    for(i = 0 ;  i < pInfo->iDocNum ; i++ ) {
        pDocName = *(pInfo->ppDocName + i );
        wsprintf(szKey, "Name%d", i + 1 );
        wsprintf(szWork, "¥"%s¥"", pDocName );
        WritePrivateProfileString("Doc", szKey, szWork, szTempFile );
    }
```

```
    // ・ Delete      ... 命令ファイル処理後、送信原稿を削除（"0":削除しない, "1":削除する）
    if(pInfo->bDocDelete ) {
        WritePrivateProfileString("Doc", "Delete", "1", szTempFile );
    }
}

// セクション名: [Cover]      ... 送付状
// ・ Name        ... 送付状ファイルパス(.txt)
if(pInfo->pCoverName ) {
    wsprintf(szWork, "¥"%s¥"", pInfo->pCoverName );
    WritePrivateProfileString("Cover", "Name", szWork, szTempFile );
}

// セクション名: [User]       ... 発信元情報
// ・ UserInfo    ... 発信元情報記録（"記録位置,記録情報"）(最大 140 バイト)
//                    ("0":記録しない, "1":原稿の内側に記録, "2":原稿の外側に記録)
if(pInfo->szUserInfo[0] ) {
    wsprintf(szWork, "¥"%s¥"", pInfo->szUserInfo );
    WritePrivateProfileString("User", "UserInfo", szWork, szTempFile );
}
// ・ UserID      ... 自局電話番号 (FAXID として、相手ファクシミリに通知されます) (最大 20 バイト)
if(pInfo->szUserID[0] ) {
    wsprintf(szWork, "¥"%s¥"", pInfo->szUserID );
    WritePrivateProfileString("User", "UserID", szWork, szTempFile );
}

////////////////////////////////////////////////////////////////////////////
// (6)  一時ファイルのフラッシュ 【重要】
//
//      OS により、WritePrivateProfileString() 【Win32API】 を使用しての
//    実際のファイルへの書き出しのタイミングが異なる為、必ずこのフラッシュの
//    ステップが必要です。
//

WritePrivateProfileString(NULL, NULL, NULL, szTempFile );

////////////////////////////////////////////////////////////////////////////
// (7)  送信命令ファイル名作成
//

// 固有のファイル名作成
if(!SFCSSYS_GetUniqFileName(szSendMisFolder, SFCS_FRM_SENDMIS, szSendMis ) ) {
    *piError = SENDFAX_ERR_MakeMisName;
```

```
        goto    EXIT;
    }

    ////////////////////////////////////////////////////////////////////////////
    // (8)  一時ファイルを送信命令ファイル名にリネーム
    //

    DeleteFile(szSendMis );    // 念のため削除
    MoveFile(szTempFile, szSendMis );  // リネーム

    bRet = TRUE;    // 正常終了

EXIT:

    // エラー時は、一時ファイルを削除
    if(bTemp && !bRet ) {
        DeleteFile(szTempFile );
    }

    return bRet;
}
```

2.2　FAX 情報の参照、削除

FAX 情報表示プログラム【MonLog.exe】 は、FAX 情報の参照、削除を行うサンプルプログラムです。
本 CD-ROM の以下の位置に入っています。

¥サンプル¥VC6 SP6¥MonLog.exe　...　FAX 情報表示プログラム
¥サンプル¥VC6 SP6¥MonLog¥　...　FAX 情報表示プログラム 開発プロジェクト

主な仕様、および操作方法は以下の通りです。

① STARFAX Server SDK を起動します。
（STARFAX Server SDK の起動は、「STARFAX Server SDK セットアップアニュアル」P21 参照）

② FAX 情報表示プログラム【MonLog.exe】 を起動します。
（起動時の作業(カレント)フォルダの指定は特にありません）

③　ツリービューの「未送信」「送信」「受信」の各項目をクリックすることにより、リストビューにそれぞれの情報が表示されます。

④　リストビューに表示される「未送信」「送信」「受信」の情報は、それぞれ、操作(A)-最新の情報に更新(F)メニュー、または F5 キーで最新の情報に更新されます。

⑤　リストビューに、「未送信」「送信」または、「受信」の情報が表示されている状態で、レコード
⑥　を選択状態にして、編集(E)-削除(D)メニューで、選択されたレコードを削除することができます。
⑦　ただし、この段階では、 STARFAX Server SDK に対して削除命令を通知しただけです。F5 キーでリストビューを更新していると、実際に削除された状態に表示が更新されます。
送信・受信・削除等の状態については、インデックスファイルに反映されますが、サービスからの通知はありません。そのため、ユーザープログラムからの動的な情報取得が必要です。
削除が正常に動作していない場合は、 STARFAX ログ管理プログラムでイベントの内容を確認して下さい。なお、現在通信中の未送信情報を指定した場合はエラーとなり、削除できません。
エラー内容は、 STARFAX ログ管理プログラムで参照できます。

⑥ リストビューに「未送信」「送信」「または、「受信」の情報が表示されている状態で、レコードを選択状態にして、ツール(T)-ビューア(V)メニューで、選択されたレコードを STARFAX Server SDK ビューアプログラムで表示することができます。

2.2.1 送信情報を参照する

送信情報を参照するためのプログラミング例を、FAX 情報表示プログラム 開発プロジェクトのソースファイルを元にご説明します。
FAX 情報表示プログラム【MonLog.exe】 の仕様、および操作方法については、 *2.2 FAX 情報の参照、削除* をご覧下さい。
開発プロジェクトは、本 CD-ROM の以下の位置に入っています。
¥サンプル¥VC6 SP6¥MonLog¥ ... FAX 情報表示プログラム 開発プロジェクト

■ 送信情報の参照

送信情報の参照は、送信情報インデックスファイルを参照することにより行います。
送信情報インデックスファイルの詳細は**「STARFAX Server SDK ファイル de FAX」P11「ログの一覧を取得する」**に記述されていますので、以下と併せてご覧下さい。
送信情報インデックスファイルの読み込みは、**ReadIdx.cpp の RIDX_GetIndexInfo()関数** で行っています。この関数は、以下のような処理を行っています。

① ファイルをすべてメモリに読み込みます。
② 全レコードの各項目の先頭位置を構造体のテーブルにセットして、各項目の終端に NULL をセットします。

このプログラムでの送信情報インデックスファイルの参照は、このような手法を採っていますが、送信情報インデックスファイルは CSV 形式のファイルですので、ユーザープログラムの仕様に合わせて読み込み処理を作成していただいて問題ありません。

なお、ソースファイル中に **SFCSSYS_** で始まる名称の関数があります。これらの関数は、サンプルプログラムで汎用的に利用される共通関数です。詳細は本書の ***付録*** に記述されていますのでご覧下さい。

● **送信情報インデックスファイルの読み込み**

MonLog.cpp : CMonLogApp::InitInstance()

```
～

////////////////////////////////////////////////////////////////////////////
// 制御関連インターフェイスフォルダ 取得
SFCSSYS_GetCtrlFolder(m_szCTRL, MAX_PATH+1 );

～
```

MonLog.cpp : CMonLogApp::LoadLogAll()

```
～

wsprintf(m_szSend, "%s¥¥%s¥¥%s", m_szCTRL, SFCS_FLD_SENDINFO, SFCS_FILE_SFCSSND );

～

m_bSend = LoadIndex(&m_infoSend, m_szSend );

～
```

MonLog.cpp : CMonLogApp::LoadIndex()

```
～

// インデックスファイル読み込み初期設定
bRet = RIDX_GetIndexInfo(pInfo, pFileName, &iError, &m_CallBack );

～
```

● プログラム終了時の終了処理

MonLog.cpp : CMonLogApp::ReleaseLogAll()

～

```
    ReleaseIndex(&m_infoSend );
```

～

MonLog.cpp : CMonLogApp::ReleaseIndex()

～

```
// インデックスファイル読み込み終了設定
RIDX_GetIndexTerminate(pInfo );
```

～

2.2.2 受信情報を参照する

受信情報を参照するためのプログラミング例を、FAX 情報表示プログラム 開発プロジェクトのソースファイルを元にご説明します。
FAX 情報表示プログラム【MonLog.exe】 の仕様、および操作方法については、 *2.2 FAX 情報の参照、削除* をご覧下さい。
開発プロジェクトは、本 CD-ROM の以下の位置に入っています。
¥サンプル¥VC6 SP6¥MonLog¥ ... FAX 情報表示プログラム 開発プロジェクト

■ 受信情報の参照

受信情報の参照は、受信情報インデックスファイルを参照することにより行います。受信情報インデックスファイルの詳細は**「STARFAX Server SDK ファイル de FAX」P11「ログの一覧を取得する」**に記述されていますので、以下と併せてご覧下さい。
受信情報インデックスファイルの読み込みは、**ReadIdx.cpp の RIDX_GetIndexInfo()関数** で行っています。この関数は、以下のような処理を行っています。

① ファイルをすべてメモリに読み込みます。
② 全レコードの各項目の先頭位置を構造体のテーブルにセットして、各項目の終端に NULL をセットします。

このプログラムでの受信情報インデックスファイルの参照は、このような手法を採っていますが、受信情報インデックスファイルは CSV 形式のファイルですので、ユーザープログラムの仕様に合わせて、読み込み処理を作成していただいて問題ありません。

なお、ソースファイル中に **SFCSSYS_** で始まる名称の関数があります。これらの関数は、サンプルプログラムで汎用的に利用される共通関数です。詳細は本書の ***付録*** に記述されていますのでご覧下さい。

● **受信情報インデックスファイルの読み込み**

MonLog.cpp : CMonLogApp::InitInstance()

```
～

////////////////////////////////////////////////////////////////////////////
// 制御関連インターフェイスフォルダ 取得
SFCSSYS_GetCtrlFolder(m_szCTRL, MAX_PATH+1 );

～
```

MonLog.cpp : CMonLogApp::LoadLogAll()

```
～

wsprintf(m_szRecv, "%s¥¥%s¥¥%s", m_szCTRL, SFCS_FLD_SENDINFO, SFCS_FILE_SFCSSND );

～

m_bSend = LoadIndex(&m_infoRecv, m_szRecv );

～
```

MonLog.cpp : CMonLogApp::LoadIndex()

```
～

// インデックスファイル読み込み初期設定
bRet = RIDX_GetIndexInfo(pInfo, pFileName, &iError, &m_CallBack );

～
```

● プログラム終了時の終了処理

MonLog.cpp : CMonLogApp::ReleaseLogAll()

～

```
    ReleaseIndex(&m_infoRecv );
```

～

MonLog.cpp : CMonLogApp::ReleaseIndex()

～

```
// インデックスファイル読み込み終了設定
RIDX_GetIndexTerminate(pInfo );
```

～

2.2.3 未送信情報を参照する

未送信情報を参照するためのプログラミング例を、FAX 情報表示プログラム 開発プロジェクトのソースファイルを元にご説明します。
FAX 情報表示プログラム【MonLog.exe】 の仕様、および操作方法については、 *2.2 FAX 情報の参照、削除* をご覧下さい。
開発プロジェクトは、本 CD-ROM の以下の位置に入っています。
¥サンプル¥VC6 SP6¥MonLog¥ ... FAX 情報表示プログラム 開発プロジェクト

■ 未送信情報の参照

未送信情報の参照は、未送信情報インデックスファイルを参照することにより行います。
未送信情報インデックスファイルの詳細は**「STARFAX Server SDK ファイル de FAX」P11「ログの一覧を取得する」**に記述されていますので、以下と併せてご覧下さい。
未送信情報インデックスファイルの読み込みは、**ReadIdx.cpp の RIDX_GetIndexInfo()関数** で行っています。この関数は、以下のような処理を行っています。

① ファイルをすべてメモリに読み込みます。
② 全レコードの各項目の先頭位置を構造体のテーブルにセットして、各項目の終端に NULL をセットします。

このプログラムでの未送信情報インデックスファイルの参照は、このような手法を採っていますが、未送信情報インデックスファイルは CSV 形式のファイルですので、ユーザープログラムの仕様に合わせて、読み込み処理を作成していただいて問題ありません。
なお、ソースファイル中に **SFCSSYS_** で始まる名称の関数があります。これらの関数は、サンプルプログラムで汎用的に利用される共通関数です。詳細は本書の ***付録*** に記述されていますのでご覧下さい。

● **未送信情報インデックスファイルの読み込み**

MonLog.cpp : CMonLogApp::InitInstance()

```
～

////////////////////////////////////////////////////////////////////////////
// 制御関連インターフェイスフォルダ 取得
SFCSSYS_GetCtrlFolder(m_szCTRL, MAX_PATH+1 );

～
```

MonLog.cpp : CMonLogApp::LoadLogAll()

```
～

wsprintf(m_szQue, "%s¥¥%s¥¥%s", m_szCTRL, SFCS_FLD_SENDINFO, SFCS_FILE_SFCSSND );

～

m_bSend = LoadIndex(&m_infoQue, m_szQue );

～
```

MonLog.cpp : CMonLogApp::LoadIndex()

```
～

// インデックスファイル読み込み初期設定
bRet = RIDX_GetIndexInfo(pInfo, pFileName, &iError, &m_CallBack );

～
```

● プログラム終了時の終了処理

MonLog.cpp : CMonLogApp::ReleaseLogAll()

```
～

    ReleaseIndex(&m_infoQue );

～
```

MonLog.cpp : CMonLogApp::ReleaseIndex()

```
～

// インデックスファイル読み込み終了設定
RIDX_GetIndexTerminate(pInfo );

～
```

2.2.4 送信情報を削除する

送信情報を削除するためのプログラミング例を、FAX 情報表示プログラム 開発プロジェクトのソースファイルを元にご説明します。
FAX 情報表示プログラム【MonLog.exe】 の仕様、及び、操作方法については、 *2.2 FAX 情報の参照、削除* をご覧下さい。
開発プロジェクトは、本 CD-ROM の以下の位置に入っています。
¥サンプル¥VC6 SP6¥MonLog¥ ... FAX 情報表示プログラム 開発プロジェクト

■ 送信情報の削除

送信情報の削除は、送信情報削除命令フォルダに送信情報削除命令ファイルを置いて、STARFAX Server SDK に対して命令することで行います。(直接、送信情報インデックスファイル (**「STARFAX Server SDK ファイル de FAX」 P11「ログの一覧を取得する」** 参照)のレコードを削除しても、 STARFAX Server SDK の内部データが更新されないため、削除されません)

送信情報削除命令ファイルの詳細は**「STARFAX Server SDK ファイル de FAX」 P13「ログを削除する」**に記述されていますので、以下と併せてご覧下さい。
送信情報削除命令ファイルを作成する部分は、**MonLog.cpp の DeleteSend()関数** に記述されています。

ここで重要なことは、直接送信情報削除命令ファイルを作成せずに、一時ファイルを作成して、それをリネームしている点です。これは、ファイルの書き出しに **WritePrivateProfileString() 【Win32API】** を使用しているので、OS によって実際のファイルへの書き出しのタイミングが異なり、STARFAX Server SDK が中途半端な状態で読み出すことを防ぐためです。
なお、ソースファイル中に **SFCSSYS_** で始まる名称の関数があります。
これらの関数は、サンプルプログラムで汎用的に利用される共通関数です。詳細は本書の ***付録*** に記述されていますのでご覧下さい。

作成したユーザープログラムで、送信情報の削除が正常に動作していない場合は、 **STARFAX ログ管理プログラム** でイベントの内容を確認して下さい。そして、その内容を参考にしてプログラムを見直してみてください。

MonLog.cpp ：

```
////////////////////////////////////////////////////////////////////////////
// 送信情報削除命令ファイル作成

BOOL CMonLogApp::DeleteSend(CPtrArray *pDelList )
{
    BOOL    bRet = FALSE;   // 返値 ... 初期値は失敗

    char    *pJobID = (char *)NULL;
    int iJobID;

    BOOL    bTemp = FALSE;      // 一時ファイルが作成されたか否か

    char    szTempFolder[MAX_PATH+1];   // 一時ファイルフォルダ用
    char    szTempFile[MAX_PATH+1];     // 一時ファイル用

    char    szSendDelMisFolder[MAX_PATH+1]; // 送信情報削除命令フォルダ用
    char    szSendDelMis[MAX_PATH+1];       // 送信情報削除命令ファイル用

    int i;

    iJobID = pDelList->GetSize();

    if(iJobID ) {

        char    szKey[32];
        char    szWork[512];

        ////////////////////////////////////////////////////////////////////////
        // (1)  送信情報削除命令フォルダパスを取得
        //

        if(!SFCSSYS_GetSubFolder(SFCS_SMODE_SENDDELMIS, szSendDelMisFolder, MAX_PATH + 1 ) ) {
            goto    EXIT;
        }

        ////////////////////////////////////////////////////////////////////////
        // (2)  一時ファイル名パスを取得 【重要】
        //
        //      送信情報削除命令ファイルの形式は、INI ファイル形式なので、ファイルの書き出しは、
        //    WritePrivateProfileString() 【Win32API】 を使用します。
        //    ただ、OSにより、WritePrivateProfileString() 【Win32API】 を使用しての
```

```
//    実際のファイルへの書き出しのタイミングが異なる為、STARFAX Engine が
//    中途半端な状態で読み出すおそれがあります。それを防ぐ為に、一旦、一時ファイルを
//    作成し、それをリネームするようにします。
//

if(!GetTempPath(sizeof(szTempFolder), szTempFolder ) ) {
    goto    EXIT;
}
if(!GetTempFileName(szTempFolder, "SFCS", 0, szTempFile ) ) {
    goto    EXIT;
}

////////////////////////////////////////////////////////////////////////////
// (3)  一時ファイルへ書き込み
//

bTemp = TRUE;

wsprintf(szWork, "%d", iJobID );
WritePrivateProfileString ("DelInfo", "Num", szWork, szTempFile );

for(i = 0 ; i < iJobID ; i++ ) {
    wsprintf(szKey, "Name%d", i + 1 );
    if((pJobID =  (char *)pDelList->GetAt(i ) ) != (char *)NULL ) {
        WritePrivateProfileString ("DelInfo", szKey, pJobID, szTempFile );
    }
    else {
        goto    EXIT;
    }
}

////////////////////////////////////////////////////////////////////////////
// (4)  一時ファイルのフラッシュ 【重要】
//
//      OSにより、WritePrivateProfileString() 【Win32API】 を使用しての
//    実際のファイルへの書き出しのタイミングが異なる為、必ずこのフラッシュの
//    ステップが必要です。
//

WritePrivateProfileString(NULL, NULL, NULL, szTempFile );

////////////////////////////////////////////////////////////////////////////
```

```
        // (5)  送信情報削除命令ファイル名作成
        //

        // 固有のファイル名作成
        if(!SFCSSYS_GetUniqFileName(szSendDelMisFolder, SFCS_FRM_SENDDELMIS, szSendDelMis ) ) {
            goto    EXIT;
        }

        ////////////////////////////////////////////////////////////////////////////
        // (6)  一時ファイルを送信情報削除命令ファイル名にリネーム
        //

        DeleteFile(szSendDelMis ); // 念のため削除
        MoveFile(szTempFile, szSendDelMis );   // リネーム

        bRet = TRUE;    // 正常終了

    }

EXIT:

    // エラー時は、一時ファイルを削除
    if(bTemp && !bRet ) {
        DeleteFile(szTempFile );
    }

    return bRet;
}
```

2.2.5 受信情報を削除する

受信情報を削除するためのプログラミング例を、FAX 情報表示プログラム 開発プロジェクトのソースファイルを元にご説明します。
FAX 情報表示プログラム【MonLog.exe】 の仕様、及び、操作方法については、 *2.2 FAX 情報の参照、削除* をご覧下さい。
開発プロジェクトは、本 CD-ROM の以下の位置に入っています。
¥サンプル¥VC6 SP6¥MonLog¥ ... FAX 情報表示プログラム 開発プロジェクト

■ 受信情報の削除

受信情報の削除は、受信情報削除命令フォルダに受信情報削除命令ファイルを置いて、STARFAX Server SDK に対して命令することで行います。(直接、受信情報インデックスファイル(**「STARFAX Server SDK ファイル de FAX」 P11「ログの一覧を取得する」**参照)のレコードを削除しても、 STARFAX Server SDK の内部データが更新されないため、削除されません)
受信情報削除命令ファイルの詳細は**「STARFAX Server SDK ファイル de FAX」 P13「ログを削除する」**に記述されていますので、以下と併せてご覧下さい。
受信情報削除命令ファイルを作成する部分は、**MonLog.cpp の DeleteRecv()関数** に記述されています。**ここで重要なことは、直接受信情報削除命令ファイルを作成せずに、一時ファイルを作成して、それをリネームしている点です。**これは、ファイルの書き出しに WritePrivateProfileString() **【Win32API】** を使用しているので、OS によって実際のファイルへの書き出しのタイミングが異なり、STARFAX Server SDK が中途半端な状態で読み出すことを防ぐためです。
なお、ソースファイル中に **SFCSSYS_** で始まる名称の関数があります。これらの関数は、サンプルプログラムで汎用的に利用される共通関数です。詳細は本書の ***付録*** に記述されていますのでご覧下さい。

作成したユーザープログラムで、受信情報の削除が正常に動作していない場合は、 **STARFAX ログ管理プログラム** でイベントの内容を確認して下さい。そして、その内容を参考にしてプログラムを見直してみてください。

MonLog.cpp ：

```
////////////////////////////////////////////////////////////////////////////
// 受信情報削除命令ファイル作成

BOOL CMonLogApp::DeleteRecv(CPtrArray *pDelList )
{
    BOOL    bRet = FALSE;  // 返値 ... 初期値は失敗

    char    *pJobID = (char *)NULL;
    int iJobID;

    BOOL    bTemp = FALSE;      // 一時ファイルが作成されたか否か

    char    szTempFolder[MAX_PATH+1];   // 一時ファイルフォルダ用
    char    szTempFile[MAX_PATH+1];     // 一時ファイル用

    char    szRecvDelMisFolder[MAX_PATH+1]; // 受信情報削除命令フォルダ用
    char    szRecvDelMis[MAX_PATH+1];       // 受信情報削除命令ファイル用

    int i;

    iJobID = pDelList->GetSize();

    if(iJobID ) {

        char    szKey[32];
        char    szWork[512];

        ////////////////////////////////////////////////////////////////////////
        // (1)  受信情報削除命令フォルダパスを取得
        //

        if(!SFCSSYS_GetSubFolder(SFCS_SMODE_RECVDELMIS, szRecvDelMisFolder, MAX_PATH + 1 ) ) {
            goto    EXIT;
        }

        ////////////////////////////////////////////////////////////////////////
        // (2)  一時ファイル名パスを取得 【重要】
        //
        //      受信情報削除命令ファイルの形式は、INI ファイル形式なので、ファイルの書き出しは、
        //    WritePrivateProfileString() 【Win32API】 を使用します。
        //    ただ、OS により、WritePrivateProfileString() 【Win32API】 を使用しての
```

```
//    実際のファイルへの書き出しのタイミングが異なる為、STARFAX Engine が
//    中途半端な状態で読み出すおそれがあります。それを防ぐ為に、一旦、一時ファイルを
//    作成し、それをリネームするようにします。
//

if(!GetTempPath(sizeof(szTempFolder), szTempFolder ) ) {
    goto    EXIT;
}
if(!GetTempFileName(szTempFolder, "SFCS", 0, szTempFile ) ) {
    goto    EXIT;
}

////////////////////////////////////////////////////////////////////////////
// (3)  一時ファイルへ書き込み
//

bTemp = TRUE;

wsprintf(szWork, "%d", iJobID );
WritePrivateProfileString ("DelInfo", "Num", szWork, szTempFile );

for(i = 0 ; i < iJobID ; i++ ) {
    wsprintf(szKey, "Name%d", i + 1 );
    if((pJobID =  (char *)pDelList->GetAt(i ) ) != (char *)NULL ) {
        WritePrivateProfileString ("DelInfo", szKey, pJobID, szTempFile );
    }
    else {
        goto    EXIT;
    }
}

////////////////////////////////////////////////////////////////////////////
// (4)  一時ファイルのフラッシュ 【重要】
//
//      OSにより、WritePrivateProfileString() 【Win32API】 を使用しての
//    実際のファイルへの書き出しのタイミングが異なる為、必ずこのフラッシュの
//    ステップが必要です。
//

WritePrivateProfileString(NULL, NULL, NULL, szTempFile );

////////////////////////////////////////////////////////////////////////////
```

```
        // (5)  受信情報削除命令ファイル名作成
        //

        // 固有のファイル名作成
        if(!SFCSSYS_GetUniqFileName(szRecvDelMisFolder, SFCS_FRM_RECVDELMIS, szRecvDelMis ) ) {
            goto    EXIT;
        }

        ////////////////////////////////////////////////////////////////////////////
        // (6)  一時ファイルを受信情報削除命令ファイル名にリネーム
        //

        DeleteFile(szRecvDelMis ); // 念のため削除
        MoveFile(szTempFile, szRecvDelMis );   // リネーム

        bRet = TRUE;    // 正常終了

    }

EXIT:

    // エラー時は、一時ファイルを削除
    if(bTemp && !bRet ) {
        DeleteFile(szTempFile );
    }

    return bRet;
}
```

2.2.6 未送信情報を削除する

未送信情報を削除するためのプログラミング例を、FAX 情報表示プログラム 開発プロジェクトのソースファイルを元にご説明します。
FAX 情報表示プログラム【MonLog.exe】 の仕様、および操作方法については、 *2.2 FAX 情報の参照、削除* をご覧下さい。
開発プロジェクトは、本 CD-ROM の以下の位置に入っています。
¥サンプル¥VC6 SP6¥MonLog¥ ... FAX 情報表示プログラム 開発プロジェクト

■ 未送信情報の削除

未送信情報の削除は、未送信情報削除命令フォルダに未送信情報削除命令ファイルを置いて、STARFAX Server SDK に対して命令することで行います。(直接、未送信情報インデックスファイル(**「STARFAX Server SDK ファイル de FAX」P11「ログの一覧を取得する」** 参照)のレコードを削除しても、 STARFAX Server SDK の内部データが更新されないため、削除されません)
未送信情報削除命令ファイルの詳細は**「STARFAX Server SDK ファイル de FAX」P13「ログを削除する」**に記述されていますので、以下と併せてご覧下さい。
未送信情報削除命令ファイルを作成する部分は、**MonLog.cpp の DeleteQue()関数** に記述されています。**ここで重要なことは、直接未送信情報削除命令ファイルを作成せずに、一時ファイルを作成して、それをリネームしている点です。**これは、ファイルの書き出しに **WritePrivateProfileString()【Win32API】** を使用しているので、OS によって実際のファイルへの書き出しのタイミングが異なり、STARFAX Server SDK が中途半端な状態で読み出すことを防ぐためです。
なお、ソースファイル中に **SFCSSYS_** で始まる名称の関数があります。これらの関数は、サンプルプログラムで汎用的に利用される共通関数です。詳細は本書の ***付録*** に記述されていますのでご覧下さい。

作成したユーザープログラムで、未送信情報の削除が正常に動作していない場合は、**STARFAX Server SDK グ管理プログラム** でイベントの内容を確認して下さい。そして、その内容を参考にしてプログラムを見直してみてください。

MonLog.cpp ：

```
////////////////////////////////////////////////////////////////////////////
// 未送信情報削除命令ファイル作成

BOOL CMonLogApp::DeleteQue(CPtrArray *pDelList )
{
    BOOL    bRet = FALSE;  // 返値 ... 初期値は失敗

    char    *pJobID = (char *)NULL;
    int iJobID;

    BOOL    bTemp = FALSE;      // 一時ファイルが作成されたか否か

    char    szTempFolder[MAX_PATH+1];   // 一時ファイルフォルダ用
    char    szTempFile[MAX_PATH+1];     // 一時ファイル用

    char    szQueDelMisFolder[MAX_PATH+1];  // 未送信情報削除命令フォルダ用
    char    szQueDelMis[MAX_PATH+1];        // 未送信情報削除命令ファイル用

    int i;

    iJobID = pDelList->GetSize();

    if(iJobID ) {

        char    szKey[32];
        char    szWork[512];

        ////////////////////////////////////////////////////////////////////////
        // (1)  未送信情報削除命令フォルダパスを取得
        //

        if(!SFCSSYS_GetSubFolder(SFCS_SMODE_QUEDELMIS, szQueDelMisFolder, MAX_PATH + 1 ) ) {
            goto    EXIT;
        }

        ////////////////////////////////////////////////////////////////////////
        // (2)  一時ファイル名パスを取得 【重要】
        //
        //      未送信情報削除命令ファイルの形式は、INI ファイル形式なので、ファイルの書き出しは、
        //    WritePrivateProfileString() 【Win32API】 を使用します。
        //    ただ、OS により、WritePrivateProfileString() 【Win32API】 を使用しての
        //    実際のファイルへの書き出しのタイミングが異なる為、STARFAX Engine が
        //    中途半端な状態で読み出すおそれがあります。それを防ぐ為に、一旦、一時ファイルを
```

```
//    作成し、それをリネームするようにします。
//

if(!GetTempPath(sizeof(szTempFolder), szTempFolder ) ) {
    goto    EXIT;
}
if(!GetTempFileName(szTempFolder, "SFCS", 0, szTempFile ) ) {
    goto    EXIT;
}

////////////////////////////////////////////////////////////////////////////
// (3)  一時ファイルへ書き込み
//

bTemp = TRUE;

wsprintf(szWork, "%d", iJobID );
WritePrivateProfileString ("DelInfo", "Num", szWork, szTempFile );

for(i = 0 ; i < iJobID ; i++ ) {
    wsprintf(szKey, "Name%d", i + 1 );

    if((pJobID =  (char *)pDelList->GetAt(i ) ) != (char *)NULL ) {
        WritePrivateProfileString ("DelInfo", szKey, pJobID, szTempFile );
    }
    else {
        goto    EXIT;
    }
}

////////////////////////////////////////////////////////////////////////////
// (4)  一時ファイルのフラッシュ 【重要】
//
//      OSにより、WritePrivateProfileString() 【Win32API】 を使用しての
//    実際のファイルへの書き出しのタイミングが異なる為、必ずこのフラッシュの
//    ステップが必要です。
//

WritePrivateProfileString(NULL, NULL, NULL, szTempFile );
```

```
        ////////////////////////////////////////////////////////////////////////////
        // (5)  未送信情報削除命令ファイル名作成
        //

        // 固有のファイル名作成
        if(!SFCSSYS_GetUniqFileName(szQueDelMisFolder, SFCS_FRM_QUEDELMIS, szQueDelMis ) ) {
            goto    EXIT;
        }

        ////////////////////////////////////////////////////////////////////////////
        // (6)  一時ファイルを未送信情報削除命令ファイル名にリネーム
        //

        DeleteFile(szQueDelMis );  // 念のため削除
        MoveFile(szTempFile, szQueDelMis );    // リネーム

        bRet = TRUE;    // 正常終了

    }

EXIT:

    // エラー時は、一時ファイルを削除
    if(bTemp && !bRet ) {
        DeleteFile(szTempFile );
    }

    return bRet;
}
```

2.3　FAX 情報の追加、削除状況の取得

FAX 情報の追加、削除状況の取得プログラム【MonDiff.exe】 は、FAX 情報の追加、削除状況の表示を行うサンプルプログラムです。このプログラムは、コンソールプログラムです。本 CD-ROM の以下の位置に入っています。

¥サンプル¥VC6 SP6¥MonDiff.exe ... FAX 情報の追加、削除状況の取得プログラム
¥サンプル¥VC6 SP6¥MonDiff¥　... FAX 情報の追加、削除状況の取得プログラム 開発プロジェクト

主な仕様、および操作方法は以下の通りです。

① STARFAX サービスマネージャ[環境設定]-「通知」-「通知ファイルを作成する」項目で作成したい通知ファイルにチェックをいれます。

② STARFAX Server SDK を起動します。
(STARFAX Server SDK の起動は、「STARFAX Server SDK セットアップアニュアル」P21 参照)

③ FAX 情報の追加、削除状況の取得プログラム【MonDiff.exe】 を起動します。
(起動時の作業(カレント)フォルダの指定は特にありません)

④ 約５秒間隔で以下の通知の存在をチェックして、標準出力に内容が出力されます。

- 送信情報追加済み通知
- 受信情報追加済み通知
- 送信情報削除済み通知
- 受信情報削除済み通知

ただし、①で設定されていない項目は通知されません。

2.3　FAX 情報の追加、削除状況の取得

2.3.1　送信情報が追加された事を知る

送信情報が追加された事を知るためのプログラミング例を、FAX 情報の追加、削除状況の取得プログラム 開発プロジェクトのソースファイルを元にご説明します。
FAX 情報の追加、削除状況の取得プログラム【MonDiff.exe】 の仕様、および操作方法については、*2.3 FAX 情報の追加、削除状況の取得* をご覧下さい。
開発プロジェクトは、本 CD-ROM の以下の位置に入っています。
¥サンプル¥VC6 SP6¥MonDiff¥ ... FAX 情報の追加、削除状況の取得プログラム 開発プロジェクト

■　送信情報が追加された事を知る

送信情報が追加された事を知るには、送信情報追加済み通知ファイルを参照することにより行います。
送信情報追加済み通知ファイルの詳細は**「STARFAX Server SDK ファイル de FAX」P14「追加されたログ情報だけを取得する」**に記述されていますので、以下と併せてご覧下さい。

ただし、 **STARFAX サービスマネージャ[環境設定]-「通知」-「通知ファイルを作成する」の「通信完了通知ファイル」で「送信」にチェックをいれていない場合は、**送信情報追加済み通知ファイルは作成されません。
送信情報追加済み通知ファイルの読み込みは、**MonDiff.c の SendAddNotiProc()関数** で行っています。この関数は、以下のような処理を行っています。

① 送信情報追加済み通知フォルダ内の、送信情報追加済み通知ファイルを探します。
② 送信情報追加済み通知ファイルがあれば、標準出力に出力して削除します。

ここで重要な事は、送信情報追加済み通知ファイルを削除していることです。これは、STARFAX Server SDK が送信情報追加済み通知ファイルを作成しますが削除を行わないため、ユーザープログラムで管理を行う必要があるという事です。
なお、ソースファイル中に **SFCSSYS_** で始まる名称の関数があります。これらの関数は、サンプルプログラムで汎用的に利用される共通関数です。詳細は本書の ***付録*** に記述されていますのでご覧下さい。

MonDiff.cpp :

```
//**********************************************************************
// 関数名        : SendAddNotiProc
// 機能          : 送信情報追加済み通知処理
// 呼び出し      : SendAddNotiProc(void )
// 入力          : なし
// 出力          : 返値  なし
// 特記事項      :   送信情報追加済み通知ファイルがあれば、内容を表示して削除します。
// 作成者        :
// 作成日        : 2001.11.16
//**********************************************************************
void    SendAddNotiProc(void )
{
    WIN32_FIND_DATA FindData;
    HANDLE  hMission;

    char    szFolder[MAX_PATH+1];
    char    szWild[MAX_PATH+1];
    char    szMis[MAX_PATH+1];

    ////////////////////////////////////////////////////////////////////////////
    // (1)  送信情報追加済み通知フォルダパスを取得
    //

    if(!SFCSSYS_GetSubFolder(SFCS_SMODE_SENDADDNOTI, szFolder,  MAX_PATH + 1 ) ) {
        goto    EXIT;
    }

    ////////////////////////////////////////////////////////////////////////////
    // (2)  送信情報追加済み通知フォルダ調査
    //

    wsprintf(szWild, "%s¥¥%s", szFolder, SFCS_WID_SENDADDNOTI );
    FindData.dwFileAttributes = FILE_ATTRIBUTE_NORMAL;
    hMission = FindFirstFile((LPCTSTR) szWild, (LPWIN32_FIND_DATA) &FindData );
    if(hMission != INVALID_HANDLE_VALUE ) {

NEXT:

        if(!(FindData.dwFileAttributes & FILE_ATTRIBUTE_SYSTEM ) &&
            !(FindData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) ) {
```

```
            wsprintf(szMis, "%s¥¥%s", szFolder, FindData.cFileName );

            ////////////////////////////////////////////////////////////////////////////
            // (3)  送信情報追加済み通知ファイルを処理
            //

            fprintf(stdout,  "¥n_/_/_/_/_/  送信情報追加済み通知  _/_/_/_/_/¥n" );

            PrintAddFileInfo(szMis );  // 追加済み通知ファイル表示

            fprintf(stdout,  "¥n" );

            ////////////////////////////////////////////////////////////////////////////
            // (4)  送信情報追加済み通知ファイル削除
            //

            DeleteFile(szMis );
        }

        ////////////////////////////////////////////////////////////////////////////
        // (5)  次の送信情報追加済み通知フォルダ調査
        //

        if(FindNextFile(hMission, &FindData ) ) {
            goto    NEXT;
        }

        FindClose(hMission );
    }

EXIT:

    return;
}
```

2.3.2 受信情報が追加された事を知る

受信情報が追加された事を知るためのプログラミング例を、FAX 情報の追加、削除状況の取得プログラム 開発プロジェクトのソースファイルを元にご説明します。
FAX 情報の追加、削除状況の取得プログラム【MonDiff.exe】 の仕様、および操作方法については、*2.3 FAX 情報の追加、削除状況の取得* をご覧下さい。
開発プロジェクトは、本 CD-ROM の以下の位置に入っています。
¥サンプル¥VC6 SP6¥MonDiff¥ ... FAX 情報の追加、削除状況の取得プログラム 開発プロジェクト

■ 受信情報が追加された事を知る

受信情報が追加された事を知る方法は、受信情報追加済み通知ファイルを参照することにより行います。受信情報追加済み通知ファイルの詳細は**「STARFAX Server SDK ファイル de FAX」P14「追加されたログ情報だけを取得する」**に記述されていますので、以下と併せてご覧下さい。

ただし、 **STARFAX サービスマネージャ[環境設定]-「通知」-「通知ファイルを作成する」の「通信完了通知ファイル」で「受信」にチェックをいれていない場合**は、受信情報追加済み通知ファイルは作成されません。
受信情報追加済み通知ファイルの読み込みは、**MonDiff.c の RecvAddNotiProc()関数** で行っています。この関数は、以下のような処理を行っています。

① 受信情報追加済み通知フォルダ内の受信情報追加済み通知ファイルを探します。
② 受信情報追加済み通知ファイルがあれば、標準出力に出力し、削除します。

ここで重要な事は、受信情報追加済み通知ファイルを削除していることです。これは、STARFAX Server SDK が受信情報追加済み通知ファイルを作成しますが削除を行わないため、ユーザープログラムで管理を行う必要があるという事です。

なお、ソースファイル中に **SFCSSYS_** で始まる名称の関数があります。これらの関数は、サンプルプログラムで汎用的に利用される共通関数です。詳細は本書の ***付録*** に記述されていますのでご覧下さい。

MonDiff.cpp ：

```
//**********************************************************************
// 関数名　　：RecvAddNotiProc
// 機能　　：受信情報追加済み通知処理
// 呼び出し　：RecvAddNotiProc(void )
// 入力　　：なし
// 出力　　：返値　なし
// 特記事項　：　受信情報追加済み通知ファイルがあれば、内容を表示して削除します。
// 作成者　　：
// 作成日　　：2001.11.16
//**********************************************************************
void    RecvAddNotiProc(void )
{
    WIN32_FIND_DATA FindData;
    HANDLE  hMission;

    char    szFolder[MAX_PATH+1];
    char    szWild[MAX_PATH+1];
    char    szMis[MAX_PATH+1];

    ////////////////////////////////////////////////////////////////////////////
    // (1)　受信情報追加済み通知フォルダパスを取得
    //

    if(!SFCSSYS_GetSubFolder(SFCS_SMODE_RECVADDNOTI, szFolder,  MAX_PATH + 1 ) ) {
        goto    EXIT;
    }

    ////////////////////////////////////////////////////////////////////////////
    // (2)　受信情報追加済み通知フォルダ調査
    //

    wsprintf(szWild, "%s¥¥%s", szFolder, SFCS_WID_RECVADDNOTI );
    FindData.dwFileAttributes = FILE_ATTRIBUTE_NORMAL;
    hMission = FindFirstFile((LPCTSTR) szWild, (LPWIN32_FIND_DATA) &FindData );
    if(hMission != INVALID_HANDLE_VALUE ) {

NEXT:

        if(!(FindData.dwFileAttributes & FILE_ATTRIBUTE_SYSTEM ) &&
            !(FindData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) ) {
```

```
            wsprintf(szMis, "%s¥¥%s", szFolder, FindData.cFileName );

            ////////////////////////////////////////////////////////////////////////////
            // (3)  受信情報追加済み通知ファイルを処理
            //

            fprintf(stdout, "¥n_/_/_/_/_/  受信情報追加済み通知  _/_/_/_/_/¥n" );

            PrintAddFileInfo(szMis );  // 追加済み通知ファイル表示

            fprintf(stdout, "¥n" );

            ////////////////////////////////////////////////////////////////////////////
            // (4)  受信情報追加済み通知ファイル削除
            //

            DeleteFile(szMis );
        }

        ////////////////////////////////////////////////////////////////////////////
        // (5)  次の受信情報追加済み通知フォルダ調査
        //

        if(FindNextFile(hMission, &FindData ) ) {
            goto    NEXT;
        }

        FindClose(hMission );
    }

EXIT:

    return;
}
```

2.3.3 送信情報が削除された事を知る

送信情報が削除された事を知るためのプログラミング例を、FAX 情報の追加、削除状況の取得プログラム 開発プロジェクトのソースファイルを元にご説明します。

FAX 情報の追加、削除状況の取得プログラム【MonDiff.exe】 の仕様、および操作方法については、*2.3 FAX 情報の追加、削除状況の取得* をご覧下さい。

開発プロジェクトは、本 CD-ROM の以下の位置に入っています。

¥サンプル¥VC6 SP6¥MonDiff¥ ... FAX 情報の追加、削除状況の取得プログラム 開発プロジェクト

■ 送信情報が削除された事を知る

送信情報が削除された事を知る方法は、送信情報削除済み通知ファイルを参照することにより行います。
送信情報削除済み通知ファイルの詳細は**「STARFAX Server SDK ファイル de FAX」P13「ログを削除する」** に記述されていますので、以下と併せてご覧下さい。

ただし、 **STARFAX サービスマネージャ[環境設定]-「通知」-「通知ファイルを作成する」の「送受信ログの削除完了通知ファイル」で「送信」にチェックをいれていない場合は**、送信情報削除済み通知ファイルは作成されません。
送信情報削除済み通知ファイルの読み込みは、**MonDiff.c の SendDelNotiProc()関数** で行っています。この関数は、以下のような処理を行っています。

① 送信情報削除済み通知フォルダ内の、送信情報削除済み通知ファイルを探します。
② 送信情報削除済み通知ファイルがあれば、標準出力に出力し削除します。

ここで重要な事は、送信情報削除済み通知ファイルを削除していることです。これは、STARFAX Server SDK が、送信情報削除済み通知ファイルを作成しますが削除を行わないため、ユーザープログラムで管理を行う必要があるという事です。

なお、ソースファイル中に **SFCSSYS_** で始まる名称の関数があります。これらの関数は、サンプルプログラムで汎用的に利用される共通関数です。詳細は、本書の ***付録*** に記述されていますのでご覧下さい。

MonDiff.cpp ：

```
//**********************************************************************
// 関数名　　：SendDelNotiProc
// 機能　　：送信情報削除済み通知処理
// 呼び出し　：SendDelNotiProc(void )
// 入力　　：なし
// 出力　　：返値　なし
// 特記事項　：　送信情報削除済み通知ファイルがあれば、内容を表示して削除します。
// 作成者　　：
// 作成日　　：2001.11.16
//**********************************************************************
void    SendDelNotiProc(void )
{
    WIN32_FIND_DATA FindData;
    HANDLE  hMission;

    char    szFolder[MAX_PATH+1];
    char    szWild[MAX_PATH+1];
    char    szMis[MAX_PATH+1];

    ////////////////////////////////////////////////////////////////////////////
    // (1)　送信情報削除済み通知フォルダパスを取得
    //

    if(!SFCSSYS_GetSubFolder(SFCS_SMODE_SENDDELNOTI, szFolder,  MAX_PATH + 1 ) ) {
        goto    EXIT;
    }

    ////////////////////////////////////////////////////////////////////////////
    // (2)　送信情報削除済み通知フォルダ調査
    //

    wsprintf(szWild, "%s¥¥%s", szFolder, SFCS_WID_SENDDELNOTI );
    FindData.dwFileAttributes = FILE_ATTRIBUTE_NORMAL;
    hMission = FindFirstFile((LPCTSTR) szWild, (LPWIN32_FIND_DATA) &FindData );
    if(hMission != INVALID_HANDLE_VALUE ) {

NEXT:

        if(!(FindData.dwFileAttributes & FILE_ATTRIBUTE_SYSTEM ) &&
            !(FindData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) ) {
```

```
            wsprintf(szMis, "%s¥¥%s", szFolder, FindData.cFileName );

            ////////////////////////////////////////////////////////////////////////////
            // (3)  送信情報削除済み通知ファイルを処理
            //

            fprintf(stdout,  "¥n_/_/_/_/_/  送信情報削除済み通知  _/_/_/_/_/¥n" );

            PrintDelFileInfo(szMis );  // 削除済み通知ファイル表示

            fprintf(stdout,  "¥n" );

            ////////////////////////////////////////////////////////////////////////////
            // (4)  送信情報削除済み通知ファイル削除
            //

            DeleteFile(szMis );
        }

        ////////////////////////////////////////////////////////////////////////////
        // (5)  次の送信情報削除済み通知フォルダ調査
        //

        if(FindNextFile(hMission, &FindData ) ) {
            goto    NEXT;
        }

        FindClose(hMission );
    }

EXIT:

    return;
}
```

2.3.4 受信情報が削除された事を知る

受信情報が削除された事を知る為のプログラミング例を、FAX 情報の追加、削除状況の取得プログラム 開発プロジェクトのソースファイルを元にご説明します。
FAX 情報の追加、削除状況の取得プログラム【MonDiff.exe】 の仕様、及び、操作方法については、*2.3 FAX 情報の追加、削除状況の取得* をご覧下さい。
開発プロジェクトは、本 CD-ROM の以下の位置に入っています。
¥サンプル¥VC6 SP6¥MonDiff¥ ... FAX 情報の追加、削除状況の取得プログラム 開発プロジェクト

■ 受信情報が削除された事を知る

受信情報が削除された事を知る方法は、受信情報削除済み通知ファイルを参照することにより行います。
受信情報削除済み通知ファイルの詳細は**「STARFAX Server SDK ファイル de FAX」の P13「ログを削除する」** に記述されていますので、以下と合わせてご覧下さい。

ただし、**STARFAX サービスマネージャ[環境設定]-「通知」-「通知ファイルを作成する」の「送受信ログの削除完了通知ファイル」で「受信」にチェックをいれていない場合**は、受信情報削除済み通知ファイルは作成されません。
受信情報削除済み通知ファイルの読み込みは、**MonDiff.c の RecvDelNotiProc()関数** で行っています。この関数は、以下のような処理を行っています。

① 受信情報削除済み通知フォルダ内の、受信情報削除済み通知ファイルを探します。
② 受信情報削除済み通知ファイルがあれば、標準出力に出力し削除します。

ここで重要な事は、受信情報削除済み通知ファイルを削除していることです。これは、STARFAX Server SDK が受信情報削除済み通知ファイルを作成しますが削除を行わないため、ユーザープログラムで管理を行う必要があるという事です。
なお、ソースファイル中に **SFCSSYS_** で始まる名称の関数があります。これらの関数は、サンプルプログラムで汎用的に利用される共通関数です。詳細は本書の ***付録*** に記述されていますのでご覧下さい。

MonDiff.cpp ：

```
//**********************************************************************
// 関数名　　：RecvDelNotiProc
// 機能　　：受信情報削除済み通知処理
// 呼び出し　：RecvDelNotiProc(void )
// 入力　　：なし
// 出力　　：返値　なし
// 特記事項　：　受信情報削除済み通知ファイルがあれば、内容を表示して削除します。
// 作成者　　：
// 作成日　　：2001.11.16
//**********************************************************************
void    RecvDelNotiProc(void )
{
    WIN32_FIND_DATA FindData;
    HANDLE  hMission;

    char    szFolder[MAX_PATH+1];
    char    szWild[MAX_PATH+1];
    char    szMis[MAX_PATH+1];

    ////////////////////////////////////////////////////////////////////////////
    // (1)　受信情報削除済み通知フォルダパスを取得
    //

    if(!SFCSSYS_GetSubFolder(SFCS_SMODE_RECVDELNOTI, szFolder,  MAX_PATH + 1 ) ) {
        goto    EXIT;
    }

    ////////////////////////////////////////////////////////////////////////////
    // (2)　受信情報削除済み通知フォルダ調査
    //

    wsprintf(szWild, "%s¥¥%s", szFolder, SFCS_WID_RECVDELNOTI );
    FindData.dwFileAttributes = FILE_ATTRIBUTE_NORMAL;
    hMission = FindFirstFile((LPCTSTR) szWild, (LPWIN32_FIND_DATA) &FindData );
    if(hMission != INVALID_HANDLE_VALUE ) {

NEXT:

        if(!(FindData.dwFileAttributes & FILE_ATTRIBUTE_SYSTEM ) &&
            !(FindData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) ) {
```

```
            wsprintf(szMis, "%s¥¥%s", szFolder, FindData.cFileName );

            ////////////////////////////////////////////////////////////////////////////
            // (3)  受信情報削除済み通知ファイルを処理
            //

            fprintf(stdout, "¥n_/_/_/_/_/  受信情報削除済み通知  _/_/_/_/_/¥n" );

            PrintDelFileInfo(szMis );  // 削除済み通知ファイル表示

            fprintf(stdout, "¥n" );

            ////////////////////////////////////////////////////////////////////////////
            // (4)  受信情報削除済み通知ファイル削除
            //

            DeleteFile(szMis );
        }

        ////////////////////////////////////////////////////////////////////////////
        // (5)  次の受信情報削除済み通知フォルダ調査
        //

        if(FindNextFile(hMission, &FindData ) ) {
            goto    NEXT;
        }

        FindClose(hMission );
    }

EXIT:

    return;
}
```

2.4　ファイル印刷

ファイル印刷プログラム【PrtFile.exe】は、ファイルの印刷を行うサンプルプログラムです。本CD-ROMの以下の位置に入っています。

¥サンプル¥VC6 SP6¥PrtFile.exe　...　ファイル印刷プログラム
¥サンプル¥VC6 SP6¥PrtFile¥　　...　ファイル印刷プログラム 開発プロジェクト

主な、仕様、及び、操作方法は以下の通りです。

① STARFAX サービスマネージャ[環境設定]-「オプション」-「印刷設定」内「カレントプリンタ」 で印刷したいプリンタを設定します。

② STARFAX Server SDK を起動します。
(STARFAX Server SDK の起動は、「STARFAX Server SDK セットアップアニュアル」P21 参照)

③ ファイル印刷プログラム【PrtFile.exe】 を起動します。
(起動時の作業(カレント)フォルダの指定は特にありません)

④ 原稿を指定します。
- 最大４つのファイルを指定でき、印刷時に連結して印刷されます。
(これは、このサンプルプログラムの仕様です)
- 最低限、１つの原稿ファイルを指定する必要があります。
- 指定できるファイル形式は以下の通りです。
  - TIFF 形式　圧縮なし、修正 CCITT MH 圧縮、CCITT G3 MH 圧縮、CCITT G3 MR 圧縮
    PackBits 圧縮、Class F 圧縮、G4 圧縮、JPEG 圧縮
  - BMP ファイル
  - PCX ファイル
  - DCX ファイル
  - JPEG ファイル
  - テキストファイル
  - FAX ファイル

⑤ 印刷(P)ボタンを押して、ファイル印刷を行います。
この後、ファイル印刷が正常に動作していない場合は、 STARFAX ログ管理プログラム でイベントの内容を確認して下さい。

2.4　ファイル印刷

2.4.1　ファイルを印刷する

ファイルを印刷するためのプログラミング例を、ファイル印刷プログラム 開発プロジェクトのソースファイルを元にご説明します。
ファイル印刷プログラム【PrtFile.exe】 の仕様、および操作方法については、 *2.4　ファイル印刷* をご覧下さい。
開発プロジェクトは、本 CD-ROM の以下の位置に入っています。
¥サンプル¥VC6 SP6¥PrtFile¥　...　ファイル印刷プログラム 開発プロジェクト

■　ファイル印刷

STARFAX Server SDK へのファイル印刷命令は、印刷命令フォルダに印刷命令ファイルを置くことで行います。
印刷命令ファイルの詳細は**「STARFAX Server SDK ファイル de FAX」の P17「指定したファイルを印刷する」** に記述されていますので、以下と併せてご覧下さい。
印刷命令ファイルを作成する部分は、**ApiPrint.cpp の MakePrintMission()関数** に記述されています。
ここで重要なことは、直接印刷命令ファイルを作成せずに、一時ファイルを作成して、それをリネームしている点です。
これは、ファイルの書き出しに **WritePrivateProfileString()【Win32API】** を使用しているので、OS によって実際のファイルへの書き出しのタイミングが異なり、STARFAX Server SDK が中途半端な状態で読み出すことを防ぐためです。
なお、ソースファイル中に **SFCSSYS_** で始まる名称の関数があります。
これらの関数は、サンプルプログラムで汎用的に利用される共通関数です。詳細は本書の ***付録*** に記述されていますのでご覧下さい。

作成したユーザープログラムで、ファイル印刷が正常に動作していない場合は、**STARFAX ログ管理プログラム** でイベントの内容を確認して下さい。
そして、その内容を参考にしてプログラムを見直してみてください。

ApiPrint.cpp ：

```
//*********************************************************************
// 関数名　　：MakePrintMission
// 機能　　：印刷命令ファイル作成
// 呼び出し　：MakePrintMission(PRINTFILE_MISSION *pInfo, int *piError )
// 入力　　：pInfo　= 印刷命令ファイル作成情報ポインタ
//　　：piError = エラー時、エラー取得用ポインタ
// 出力　　：返値　TRUE　：正常終了
//　　：　　FALSE：失敗
//　　：　　*piError = エラーコード
//　　：
//　　：　【エラーコード】
//　　：　　PRINTFILE_ERR_NoMisFolder　　... 印刷命令フォルダが存在しません
//　　：　　PRINTFILE_ERR_GetMisFolder　　... 印刷命令フォルダの取得に失敗しました
//　　：　　PRINTFILE_ERR_GetTempFolder　... 一時フォルダの取得に失敗しました
//　　：　　PRINTFILE_ERR_GetTempFile　　... 一時ファイルの取得に失敗しました
//　　：　　PRINTFILE_ERR_MakeMisName　　... 印刷命令ファイル名の作成に失敗しました
//　　：　　PRINTFILE_ERR_PARAM_INFO　　... 関数引数エラー：印刷命令ファイル作成情報が指定されて
//　　　　　　　　　　　　　　　　　　　　　いません。
//　　：　　PRINTFILE_ERR_PARAM_DOC　　... 関数引数エラー：印刷原稿ファイルが指定されていません。
//　　：　　PRINTFILE_ERR_PARAM_DOCNAME　... 関数引数エラー：印刷原稿ファイル名が指定されていません。
//　　：
// 特記事項　：印刷命令ファイル作成を作成します。
// 作成者　　：
// 作成日　　：2001.11.16
//*********************************************************************
BOOL APIENTRY　MakePrintMission(PRINTFILE_MISSION *pInfo, int *piError )
{
    BOOL　bRet = FALSE;　// 返値 ... 初期値は失敗

    int iErrorDummy;　// NULL 用ダミーエラーコード設定

    int i;

    BOOL bTemp = FALSE;　　// 一時ファイルが作成されたか否か

    char　szTempFolder[MAX_PATH+1];　// 一時ファイルフォルダ用
    char　szTempFile[MAX_PATH+1];　　// 一時ファイル用

    char　szPrintMisFolder[MAX_PATH+1];　// 印刷命令フォルダ用
    char　szPrintMis[MAX_PATH+1];　　// 印刷命令ファイル用
```

```
char    *pDocName;      // 印刷原稿用

char    szKey[32];      // キー設定用
char    szWork[512];    // 作業用

if(!piError ) {
    // エラーコード取得用ポインタにダミー設定
    piError = &iErrorDummy;
}
*piError = PRINTFILE_SUCCESS;       // エラーコードに初期値設定 ... 正常終了

////////////////////////////////////////////////////////////////////////////
// (1)  簡単な引数の内容チェック
//

    // 印刷命令ファイル作成情報チェック
if(!pInfo ) {
    *piError = PRINTFILE_ERR_PARAM_INFO;
    goto    EXIT;
}
    // 印刷原稿ファイルチェック
if(!pInfo->iDocNum ) {
    *piError = PRINTFILE_ERR_PARAM_DOC;
    goto    EXIT;
}
    // 印刷原稿ファイル名チェック
if(pInfo->iDocNum ) {
    for(i = 0 ;  i < pInfo->iDocNum ; i++ ) {
        pDocName = *(pInfo->ppDocName + i );
        if(!pDocName ) {
            *piError = PRINTFILE_ERR_PARAM_DOCNAME;
            goto    EXIT;
        }
    }
}

////////////////////////////////////////////////////////////////////////////
// (2)印刷命令フォルダパスのチェック
//

if(SFCSSYS_CheckSubFolder(SFCS_SMODE_PRTMIS ) != SFCS_SUCCESS ) {
```

```
    *piError = PRINTFILE_ERR_NoMisFolder;
    goto    EXIT;
}

////////////////////////////////////////////////////////////////////////////
// (3)印刷命令フォルダパスを取得
//

if(!SFCSSYS_GetSubFolder(SFCS_SMODE_PRTMIS, szPrintMisFolder, MAX_PATH + 1 ) ) {
    *piError = PRINTFILE_ERR_GetMisFolder;
    goto    EXIT;
}

////////////////////////////////////////////////////////////////////////////
// (4)  一時ファイル名パスを取得 【重要】
//
//    印刷命令ファイルの形式は、INI ファイル形式なので、ファイルの書き出しは、
//    WritePrivateProfileString() 【Win32API】 を使用します。
//    ただ、OS により、WritePrivateProfileString() 【Win32API】 を使用しての
//    実際のファイルへの書き出しのタイミングが異なる為、STARFAX Engine が
//    中途半端な状態で読み出すおそれがあります。それを防ぐ為に、一旦、一時ファイルを
//    作成し、それをリネームするようにします。
//

if(!GetTempPath(sizeof(szTempFolder), szTempFolder ) ) {
    *piError = PRINTFILE_ERR_GetTempFolder;
    goto    EXIT;
}
if(!GetTempFileName(szTempFolder, "SFCS", 0, szTempFile ) ) {
    *piError = PRINTFILE_ERR_GetTempFile;
    goto    EXIT;
}

////////////////////////////////////////////////////////////////////////////
// (5)  一時ファイルへ書き込み
//

bTemp = TRUE;

// セクション名: [Doc]        ... 印刷原稿
// ・ Num         ... 原稿数
if(pInfo->iDocNum ) {
```

```
    wsprintf(szWork, "%d", pInfo->iDocNum );
    WritePrivateProfileString("Doc", "Num", szWork, szTempFile );

    // ・ Name%d     ... 原稿ファイルパス (1～)
    for(i = 0 ;  i < pInfo->iDocNum ; i++ ) {
        pDocName = *(pInfo->ppDocName + i );
        wsprintf(szKey, "Name%d", i + 1 );
        wsprintf(szWork, "¥"%s¥"", pDocName );
        WritePrivateProfileString("Doc", szKey, szWork, szTempFile );
    }
}

////////////////////////////////////////////////////////////////////////////
// (6)  一時ファイルのフラッシュ 【重要】
//
//      OSにより、WritePrivateProfileString() 【Win32API】 を使用しての
//    実際のファイルへの書き出しのタイミングが異なる為、必ずこのフラッシュの
//    ステップが必要です。
//

WritePrivateProfileString(NULL, NULL, NULL, szTempFile );

////////////////////////////////////////////////////////////////////////////
// (7)印刷命令ファイル名作成
//

// 固有のファイル名作成
if(!SFCSSYS_GetUniqFileName(szPrintMisFolder, SFCS_FRM_PRTMIS, szPrintMis ) ) {
    *piError = PRINTFILE_ERR_MakeMisName;
    goto    EXIT;
}

////////////////////////////////////////////////////////////////////////////
// (8)  一時ファイルを印刷命令ファイル名にリネーム
//

DeleteFile(szPrintMis );   // 念のため削除
MoveFile(szTempFile, szPrintMis ); // リネーム

bRet = TRUE;    // 正常終了
```

```
EXIT:

    // エラー時は、一時ファイルを削除
    if(bTemp && !bRet ) {
        DeleteFile(szTempFile );
    }

    return bRet;
}
```

2.5　メール送信

メール送信プログラム【SendMail.exe】 は、メールの送信を行うサンプルプログラムです。本 CD-ROM の以下の位置に入っています。

¥サンプル¥VC6 SP6¥SendMail.exe　...　メール送信プログラム
¥サンプル¥VC6 SP6¥SendMail¥　　...　メール送信プログラム 開発プロジェクト

主な仕様、および操作方法は以下の通りです。

① STARFAX サービスマネージャ[メールサーバー]でメール送信で使用するメールサーバーを設定します。

② STARFAX Server SDK を起動します。
(STARFAX Server SDK の起動は、「STARFAX Server SDK セットアップアニュアル」P21 参照)

③ メール送信プログラム【SendMail.exe】 を起動します。
(起動時の作業(カレント)フォルダの指定は特にありません)

④ 宛先を指定します。
- 複数の宛先を指定する場合は、カンマ区切りで指定します。
- 最低限、１件の宛先を指定する必要があります。

⑤ 必要であれば、CC を指定します。
- 複数の CC を指定する場合は、カンマ区切りで指定します。

⑥ 必要であれば、件名を指定します。

⑦ 必要であれば、本文を指定します。

⑧ 必要であれば、添付ファイルを指定します。
- 最大２つの添付ファイルを指定できます。
(これは、このサンプルプログラムの仕様です)
- 指定できるファイル形式に特に制限はありません。

⑨ 送信(P)ボタンを押して、メール送信を行います。
この後、メール送信が正常に動作していない場合は、 STARFAX ログ管理プログラム でイベントの内容を確認して下さい。

2.5　メール送信

2.5.1　メールを送信する

メールを送信するためのプログラミング例を、メール送信プログラム 開発プロジェクトのソースファイルを元にご説明します。
メール送信プログラム【SnedMail.exe】 の仕様、および操作方法については、 *2.5　メール送信* をご覧下さい。
開発プロジェクトは、本 CD-ROM の以下の位置に入っています。
¥サンプル¥VC6 SP6¥SendMail¥　...　メール送信プログラム 開発プロジェクト

■　メール送信

STARFAX Server SDK へのメール送信命令は、メール送信命令フォルダにメール送信命令ファイルを置くことで行います。
メール送信命令ファイルの詳細は**「STARFAX Server SDK ファイル de FAX」の P19「指定したファイルをメールに添付して送信する」**に記述されていますので、以下と併せてご覧下さい。
メール送信命令ファイルを作成する部分は、**ApiEMail.cpp** の **MakeEMailMission()関数** に記述されています。

ここで重要なことは、直接メール送信命令ファイルを作成せずに、一時ファイルを作成して、それをリネームしている点です。これは、ファイルの書き出しに WritePrivateProfileString()【Win32API】を使用しているので、OS によって実際のファイルへの書き出しのタイミングが異なり、STARFAX Server SDK が中途半端な状態で読み出すことを防ぐためです。

なお、ソースファイル中に **SFCSSYS_** で始まる名称の関数があります。これらの関数は、サンプルプログラムで汎用的に利用される共通関数です。詳細は本書の ***付録*** に記述されていますのでご覧下さい。

作成したユーザープログラムで、メール送信が正常に動作していない場合は、 **STARFAX Server SDK ログ管理プログラム** でイベントの内容を確認して下さい。そして、その内容を参考にしてプログラムを見直してみてください。

ApiEMail.cpp ：

```
//*********************************************************************
// 関数名　　：MakeEMailMission
// 機能　　：メール送信命令ファイル作成
// 呼び出し　：MakeEMailMission(EMAIL_MISSION *pInfo, int *piError )
// 入力　　：pInfo　= メール送信命令ファイル作成情報ポインタ
//　　：piError = エラー時、エラー取得用ポインタ
// 出力　　：返値　TRUE　：正常終了
//　　：　　FALSE：失敗
//　　：　　*piError = エラーコード
//　　：
//　　：　【エラーコード】
//　　：　　SENDMAIL_ERR_NoMisFolder　　... メール送信命令フォルダが存在しません
//　　：　　SENDMAIL_ERR_GetMisFolder　　... メール送信命令フォルダの取得に失敗しました
//　　：　　SENDMAIL_ERR_GetTempFolder　... 一時フォルダの取得に失敗しました
//　　：　　SENDMAIL_ERR_GetTempFile　　... 一時ファイルの取得に失敗しました
//　　：　　SENDMAIL_ERR_MakeMisName　　... メール送信命令ファイル名の作成に失敗しました
//　　：　　SENDMAIL_ERR_PARAM_INFO　　... 関数引数エラー: メール送信命令ファイル作成情報が指定されて
//　　　　　　　　　　　　　　　　　　　　いません。
//　　：　　SENDMAIL_ERR_PARAM_TO　　... 関数引数エラー: 宛先が指定されていません。
//　　：　　SENDMAIL_ERR_PARAM_TOADR　　... 関数引数エラー: 宛先アドレスが指定されていません。
//　　：　　SENDMAIL_ERR_PARAM_CCADR　　... 関数引数エラー: CCアドレスが指定されていません。
//　　：　　SENDMAIL_ERR_PARAM_ATTACHNAME ... 関数引数エラー: 添付ファイルパスが指定されていません。
//　　：
// 特記事項　　：　メール送信命令ファイル作成を作成します。
// 作成者　　：
// 作成日　　：2002.05.14
//*********************************************************************
BOOL APIENTRY　　MakeEMailMission(EMAIL_MISSION *pInfo, int *piError )
{
    BOOL    bRet = FALSE;　　// 返値 ... 初期値は失敗

    int     iErrorDummy;　　// NULL用ダミーエラーコード設定

    int     i;

    BOOL    bTemp = FALSE;　　// 一時ファイルが作成されたか否か

    char    szTempFolder[MAX_PATH+1];　　// 一時ファイルフォルダ用
    char    szTempFile[MAX_PATH+1];　　// 一時ファイル用

    char    szEMailMisFolder[MAX_PATH+1];　　// メール送信命令フォルダ用
    char    szEMailMis[MAX_PATH+1];　　// メール送信命令ファイル用
```

```
char    *pToAddress;                    // 宛先用
char    *pCcAddress;                    // CC 用
char    *pAttachName;                   // 添付ファイル用

char    szKey[32];              // キー設定用
char    szWork[4096+1];         // 作業用

if(!piError ) {
    // エラーコード取得用ポインタにダミー設定
    piError = &iErrorDummy;
}
*piError = SENDMAIL_SUCCESS;                // エラーコードに初期値設定 ... 正常終了

////////////////////////////////////////////////////////////////////////////
// (1)  簡単な引数の内容チェック
//

    // メール送信命令ファイル作成情報チェック
if(!pInfo ) {
    *piError = SENDMAIL_ERR_PARAM_INFO;
    goto            EXIT;
}

    // 宛先チェック
if(!pInfo->iToNum ) {
    *piError = SENDMAIL_ERR_PARAM_TO;
    goto            EXIT;
}
    // 宛先アドレスチェック
if(pInfo->iToNum ) {
    for(i = 0 ;  i < pInfo->iToNum ; i++ ) {
        pToAddress = *(pInfo->ppToAddress + i );
        if(!pToAddress ) {
            *piError = SENDMAIL_ERR_PARAM_TOADR;
            goto    EXIT;
        }
    }
}
```

```
    // CC アドレスチェック
if(pInfo->iCcNum ) {
    for(i = 0 ;  i < pInfo->iCcNum ; i++ ) {
        pCcAddress = *(pInfo->ppCcAddress + i );
        if(!pCcAddress ) {
            *piError = SENDMAIL_ERR_PARAM_CCADR;
            goto   EXIT;
        }
    }
}
    // 添付ファイルパスチェック
if(pInfo->iAttachNum ) {
    for(i = 0 ;  i < pInfo->iAttachNum ; i++ ) {
        pAttachName = *(pInfo->ppAttachName + i );
        if(!pAttachName ) {
            *piError = SENDMAIL_ERR_PARAM_ATTACHNAME;
            goto   EXIT;
        }
    }
}

////////////////////////////////////////////////////////////////////////////
// (2)  メール送信命令フォルダパスのチェック
//

if(SFCSSYS_CheckSubFolder(SFCS_SMODE_EMSMIS ) != SFCS_SUCCESS ) {
    *piError = SENDMAIL_ERR_NoMisFolder;
    goto           EXIT;
}

////////////////////////////////////////////////////////////////////////////
// (3)  メール送信命令フォルダパスを取得
//

if(!SFCSSYS_GetSubFolder(SFCS_SMODE_EMSMIS, szEMailMisFolder, MAX_PATH + 1 ) ) {
    *piError = SENDMAIL_ERR_GetMisFolder;
    goto           EXIT;
}
```

```
////////////////////////////////////////////////////////////////////////////
// (4)  一時ファイル名パスを取得 【重要】
//
//      メール送信命令ファイルの形式は、INI ファイル形式なので、ファイルの書き出しは、
//    WritePrivateProfileString() 【Win32API】 を使用します。
//    ただ、OSにより、WritePrivateProfileString() 【Win32API】 を使用しての
//    実際のファイルへの書き出しのタイミングが異なる為、STARFAX Engine が
//    中途半端な状態で読み出すおそれがあります。それを防ぐ為に、一旦、一時ファイルを
//    作成し、それをリネームするようにします。
//

if(!GetTempPath(sizeof(szTempFolder), szTempFolder ) ) {
    *piError = SENDMAIL_ERR_GetTempFolder;
    goto           EXIT;
}
if(!GetTempFileName(szTempFolder, "SFCS", 0, szTempFile ) ) {
    *piError = SENDMAIL_ERR_GetTempFile;
    goto           EXIT;
}

////////////////////////////////////////////////////////////////////////////
// (5)  一時ファイルへ書き込み
//

bTemp = TRUE;

// セクション名: [To]  ... 宛先
// ・  Num       ... 宛先数
if(pInfo->iToNum ) {

    wsprintf(szWork, "%d", pInfo->iToNum );
    WritePrivateProfileString("To", "Num", szWork, szTempFile );

    // ・  Address%d ... 宛先  (1～)
    for(i = 0 ;  i < pInfo->iToNum ; i++ ) {
        pToAddress = *(pInfo->ppToAddress + i );
        wsprintf(szKey, "Address%d", i + 1 );
        wsprintf(szWork, "¥"%s¥"", pToAddress );
        WritePrivateProfileString("To", szKey, szWork, szTempFile );
    }
}
```

```
// セクション名: [Cc]  ... CC
// ・ Num       ... CC数
if(pInfo->iCcNum ) {

    wsprintf(szWork, "%d", pInfo->iCcNum );
    WritePrivateProfileString("Cc", "Num", szWork, szTempFile );

    // ・ Address%d ... CC (1～)
    for(i = 0 ;  i < pInfo->iCcNum ; i++ ) {
        pCcAddress = *(pInfo->ppCcAddress + i );
        wsprintf(szKey, "Address%d", i + 1 );
        wsprintf(szWork, "¥"%s¥"", pCcAddress );
        WritePrivateProfileString("Cc", szKey, szWork, szTempFile );
    }
}

// セクション名: [Text]  ... テキスト
// ・ Subject ... 件名
if(pInfo->pSubject ) {
    WritePrivateProfileString("Text", "Subject", pInfo->pSubject, szTempFile );
}

// セクション名: [Text]  ... テキスト
// ・ TheBody ... 本文 (改行: ¥n)
if(pInfo->pTheBody ) {

    int iLen;

    ZeroMemory(szWork, 4096 + 1 );
    iLen = lstrlen(pInfo->pTheBody );
    if(iLen ) {
        // CRLF("¥x0d¥x0a") を "¥n"に変換します。
        TheBodyEncode((unsigned char *)pInfo->pTheBody, (unsigned char *)szWork, 4096 );
        iLen = lstrlen(szWork );
    }
    if(iLen ) {
        WritePrivateProfileString("Text", "TheBody", szWork, szTempFile );
    }
}
```

```
// セクション名: [Attach]  ... 添付ファイル
// ・ Num        ... 添付数
if(pInfo->iAttachNum ) {

    wsprintf(szWork, "%d", pInfo->iAttachNum );
    WritePrivateProfileString("Attach", "Num", szWork, szTempFile );

    // ・ Name%d     ... 添付ファイルパス (1～)
    for(i = 0 ;  i < pInfo->iAttachNum ; i++ ) {
        pAttachName = *(pInfo->ppAttachName + i );
        wsprintf(szKey, "Name%d", i + 1 );
        wsprintf(szWork, "¥"%s¥"", pAttachName );
        WritePrivateProfileString("Attach", szKey, szWork, szTempFile );
    }
}

////////////////////////////////////////////////////////////////////////////
// (6)  一時ファイルのフラッシュ 【重要】
//
//      OSにより、WritePrivateProfileString() 【Win32API】 を使用しての
//    実際のファイルへの書き出しのタイミングが異なる為、必ずこのフラッシュの
//    ステップが必要です。
//

WritePrivateProfileString(NULL, NULL, NULL, szTempFile );

////////////////////////////////////////////////////////////////////////////
// (7)  メール送信命令ファイル名作成
//

// 固有のファイル名作成
if(!SFCSSYS_GetUniqFileName(szEMailMisFolder, SFCS_FRM_EMSMIS, szEMailMis ) ) {
    *piError = SENDMAIL_ERR_MakeMisName;
    goto           EXIT;
}

////////////////////////////////////////////////////////////////////////////
// (8)  一時ファイルをメール送信命令ファイル名にリネーム
//

DeleteFile(szEMailMis );        // 念のため削除
```

```
    MoveFile(szTempFile, szEMailMis );          // リネーム

    bRet = TRUE;        // 正常終了

EXIT:

    // エラー時は、一時ファイルを削除
    if(bTemp && !bRet ) {
        DeleteFile(szTempFile );
    }

    return bRet;
}
```

2.6　動作情報の参照

動作情報プログラム【MonEnv.exe】 は、 STARFAX Server SDK の 起動状況、および動作状況の表示を行うサンプルプログラムです。本 CD-ROM の以下の位置に入っています。

¥サンプル¥VC6 SP6¥MonEnv.exe　...　動作情報プログラム
¥サンプル¥VC6 SP6¥MonEnv¥　　...　動作情報プログラム 開発プロジェクト

主な仕様、および操作方法は以下の通りです。

① STARFAX Server SDK を起動します。
　(STARFAX Server SDK の起動は、「STARFAX Server SDK セットアップアニュアル」P21 参照)

② 動作情報プログラム【MonEnv.exe】 を起動します。
　(起動時の作業(カレント)フォルダの指定は特にありません)

③ ダイアログに各種動作状況が表示されます。

④ 更新(U)ボタンを押すと、表示内容が最新の状態に更新されます。

2.6.1　動作情報を参照する

STARFAX Server SDK の起動状況、および動作状況の参照するためのプログラミング例を、動作情報プログラム 開発プロジェクトのソースファイルを元にご説明します。
動作情報プログラム【MonEnv.exe】 の仕様、および操作方法については、 *2.6　動作情報の参照* をご覧下さい。
開発プロジェクトは、本 CD-ROM の以下の位置に入っています。
**　¥サンプル¥VC6 SP6¥MonEnv¥　...　動作情報プログラム 開発プロジェクト**

■STARFAX Server SDK の起動状況、および動作状況の参照

STARFAX Server SDK の起動状況、および動作状況の参照は、起動情報ファイルを参照することにより行います。
起動情報ファイルの詳細は**「STARFAX Server SDK ファイル de FAX」P22「STARFAX Server SDK の起動情報を取得する」**をご覧下さい。
起動情報ファイルの参照は、**MonEnvDlg.cpp の UpdateContorols()関数** で行っています。

なお、ソースファイル中に SFCSSYS_ で始まる名称の関数があります。これらの関数は、サンプルプログラムで汎用的に利用される共通関数です。
詳細は本書の ***付録*** に記述されていますのでご覧下さい。

MonEnvDlg.cpp ：

```
void CMonEnvDlg::UpdateContorols(void )
{
    char    szWork[512];
    char    szWarning[1024];
    int iRet;

    ////////////////////////////////////////////////////////////////////////////
    // (1)  STARFAX Engine サービス状態取得
    //

    m_bRun = (SFCSSYS_CheckServiceStatus() == SFCS_SERVICE_STOP ) ? FALSE : TRUE;
    m_ceRun.SetWindowText(m_bRun ? "動作中" : "停止中" );

    if(!m_bRun ) {
        goto    EXIT;
    }

    ////////////////////////////////////////////////////////////////////////////
    // (2)  その他情報を取得
    //
    //      ■ 起動情報ファイル
    //          [制御関連インターフェイスフォルダ¥SfCsRun.inf
    //        ● セクション名: [Version]  ... バージョン 情報
    //          ・ Product    ... 製品種類              ("SfCs" 固定)
    //          ・ Major      ... メジャーバージョン
    //          ・ Minor      ... マイナーバージョン
    //          ・ Revision   ... バージョン毎修正回数
    //        ● セクション名: [Line0]  ... 回線０情報
    //        ● セクション名: [Line1]  ... 回線１情報
    //        ● セクション名: [Line2]  ... 回線２情報
    //        ● セクション名: [Line3]  ... 回線３情報
    //          ・ Run        ... 回線起動状況 ("0":動作していない, "1":動作している)
    //          ・ Send       ... 送信状況 ("0":送信不可, "1":送信可)
    //          ・ Receive    ... 受信状況 ("0":受信不可, "1":受信可)
    //          ・ Modem      ... モデム名
    //        ● セクション名: [AutoDel]  ... 自動削除情報
    //          ・ AutoTx     ... 送信情報自動削除 (1:ON, 0:OFF)
    //          ・ AutoTxTime ... 送信情報自動削除対象 経過日数 (日)
    //          ・ AutoRx     ... 受信情報自動削除 (1:ON, 0:OFF)
    //          ・ AutoRxTime ... 受信情報自動削除対象 経過日数 (日)
    //        ● セクション名: [Warning]  ... 警告
```

```
//          ・ DiskFreeInstall   ... インストールフォルダのディスク空き容量
//                                  (1:残り 150M 以下, 0:正常)
//          ・ DiskFreeCtrl      ... 制御関連インターフェイスフォルダのディスク空き容量
//                                  (1:残り 150M 以下, 0:正常)
//          ・ DiskFreeData      ... データフォルダのディスク空き容量
//                                  (1:残り 150M 以下, 0:正常)
//          ・ SendIndexNum      ... 送信情報インデックスファイル件数
//                                  (1:8 万件以上, 0:正常)
//          ・ RecvIndexNum      ... 受信情報インデックスファイル件数
//                                  (1:8 万件以上, 0:正常)
//          ・ QueIndexNum       ... 未送信情報インデックスファイル件数
//                                  (1:8 万件以上, 0:正常)
//          ・ DustSendIndexNum  ... ごみ箱 送信情報インデックスファイル件数
//                                  (1:8 万件以上, 0:正常)
//          ・ DustRecvIndexNum  ... ごみ箱 受信情報インデックスファイル件数
//                                  (1:8 万件以上, 0:正常)
//

if(SFCSSYS_GetRunInfoString(SFCS_SEC_RUNVERSION, SFCS_KEY_RUNMAJOR, "", szWork, 512 ) ) {
    m_ceMajor.SetWindowText(szWork );
}
if(SFCSSYS_GetRunInfoString(SFCS_SEC_RUNVERSION, SFCS_KEY_RUNMINOR, "", szWork, 512 ) ) {
    m_ceMinor.SetWindowText(szWork );
}
if(SFCSSYS_GetRunInfoString(SFCS_SEC_RUNVERSION, SFCS_KEY_RUNREVISION, "", szWork, 512 ) ) {
    m_ceRevision.SetWindowText(szWork );
}

if(SFCSSYS_GetRunInfoInt(SFCS_SEC_AUTODEL, SFCS_KEY_AUTOTX, 0 ) ) {
    iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_AUTODEL, SFCS_KEY_AUTOTXTIME, 0 );
    wsprintf(szWork, "%d 日", iRet );
}
else {
    lstrcpy(szWork, "しない" );
}
m_ceAutoTx.SetWindowText(szWork );
if(SFCSSYS_GetRunInfoInt(SFCS_SEC_AUTODEL, SFCS_KEY_AUTORX, 0 ) ) {
    iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_AUTODEL, SFCS_KEY_AUTORXTIME, 0 );
wsprintf(szWork, "%d 日", iRet );
}
else {
```

```
    lstrcpy(szWork, "しない");
}
m_ceAutoRx.SetWindowText(szWork );

iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_LINE0, SFCS_KEY_LINERUN, 0 );
m_ceLine0_Run.SetWindowText(iRet ? "動作中" : "" );
if(iRet ) {
    iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_LINE0, SFCS_KEY_LINESEND, 0 );
    m_ceLine0_Send.SetWindowText(iRet ? "送信可" : "送信不可" );
    iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_LINE0, SFCS_KEY_LINERECV, 0 );
    m_ceLine0_Receive.SetWindowText(iRet ? "受信可" : "受信不可" );
    if(SFCSSYS_GetRunInfoString(SFCS_SEC_LINE0, SFCS_KEY_LINEMDM, "", szWork, 512 ) ) {
        m_ceLine0_Modem.SetWindowText(szWork );
    }
}
iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_LINE1, SFCS_KEY_LINERUN, 0 );
m_ceLine1_Run.SetWindowText(iRet ? "動作中" : "" );
if(iRet ) {
    iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_LINE1, SFCS_KEY_LINESEND, 0 );
    m_ceLine1_Send.SetWindowText(iRet ? "送信可" : "送信不可" );
    iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_LINE1, SFCS_KEY_LINERECV, 0 );
    m_ceLine1_Receive.SetWindowText(iRet ? "受信可" : "受信不可" );
    if(SFCSSYS_GetRunInfoString(SFCS_SEC_LINE1, SFCS_KEY_LINEMDM, "", szWork, 512 ) ) {
        m_ceLine1_Modem.SetWindowText(szWork );
    }
}
iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_LINE2, SFCS_KEY_LINERUN, 0 );
m_ceLine2_Run.SetWindowText(iRet ? "動作中" : "" );
if(iRet ) {
    iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_LINE2, SFCS_KEY_LINESEND, 0 );
    m_ceLine2_Send.SetWindowText(iRet ? "送信可" : "送信不可" );
    iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_LINE2, SFCS_KEY_LINERECV, 0 );
    m_ceLine2_Receive.SetWindowText(iRet ? "受信可" : "受信不可" );
    if(SFCSSYS_GetRunInfoString(SFCS_SEC_LINE2, SFCS_KEY_LINEMDM, "", szWork, 512 ) ) {
        m_ceLine2_Modem.SetWindowText(szWork );
    }
}
iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_LINE3, SFCS_KEY_LINERUN, 0 );
m_ceLine3_Run.SetWindowText(iRet ? "動作中" : "" );
if(iRet ) {
    iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_LINE3, SFCS_KEY_LINESEND, 0 );
    m_ceLine3_Send.SetWindowText(iRet ? "送信可" : "送信不可" );
```

```
    iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_LINE3, SFCS_KEY_LINERECV, 0 );
    m_ceLine3_Receive.SetWindowText(iRet ? "受信可" : "受信不可" );
    if(SFCSSYS_GetRunInfoString(SFCS_SEC_LINE3, SFCS_KEY_LINEMDM, "", szWork, 512 ) ) {
        m_ceLine3_Modem.SetWindowText(szWork );
    }
}

ZeroMemory(szWarning, 1024 );
iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_WARNING, SFCS_KEY_DISKFREEINST, 0 );
if(iRet ) {
lstrcat(szWarning,
  " STARFAX Engine がインストールされているフォルダのディスク空き容量が 150 メガバイト以下 になりました¥x0d¥x0a" );
}
iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_WARNING, SFCS_KEY_DISKFREECTRL, 0 );
if(iRet ) {
    lstrcat(szWarning,
      "制御関連情報のインターフェイスフォルダのディスク空き容量が 150 メガバイト以下 になりました¥x0d¥x0a" );
}
iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_WARNING, SFCS_KEY_DISKFREEDATA, 0 );
if(iRet ) {
    lstrcat(szWarning,
         "データ情報が設定されるフォルダのディスク空き容量が 150 メガバイト以下 になりました¥x0d¥x0a" );
}
iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_WARNING, SFCS_KEY_SENDIDXNUM, 0 );
if(iRet ) {
    lstrcat(szWarning, "送信情報インデックスファイルの件数が ８万件以上 になりました¥x0d¥x0a" );
}
iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_WARNING, SFCS_KEY_RECVIDXNUM, 0 );
if(iRet ) {
    lstrcat(szWarning, "受信情報インデックスファイルの件数が ８万件以上 になりました¥x0d¥x0a" );
}
iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_WARNING, SFCS_KEY_QUEIDXNUM, 0 );
if(iRet ) {
    lstrcat(szWarning, "未送信情報インデックスファイルの件数が ８万件以上 になりました¥x0d¥x0a" );
}
iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_WARNING, SFCS_KEY_DUSENDIDXNUM, 0 );
if(iRet ) {
    lstrcat(szWarning, "ごみ箱 送信情報インデックスファイルの件数が ８万件以上 になりました¥x0d¥x0a" );
}
iRet = SFCSSYS_GetRunInfoInt(SFCS_SEC_WARNING, SFCS_KEY_DURECVIDXNUM, 0 );
if(iRet ) {
    lstrcat(szWarning, "ごみ箱 受信情報インデックスファイルの件数が ８万件以上 になりました¥x0d¥x0a" );
```

```
    }
    m_ceWarning.SetWindowText(szWarning );

EXIT:

    return;
}
```

第Ⅲ章

クライアントプログラムの開発について

ユーザープログラムの開発の手順についてご説明しています。

3.1　開発の手順

3.1　開発の手順

STARFAX Server SDK をセットアップしていない別のコンピュータから FAX 送信を行う為には、１台のコンピュータに STARFAX Server SDK 本体のセットアップを行い、もう１台のコンピュータにクライアント環境のセットアップを行う必要がります。
そして、ユーザープログラム開発を行う前に、STARFAX Server SDK 本体の操作を簡単に理解しておく必要があります。
それらを考慮して、以下の手順でユーザープログラム開発を行うことをお奨めします。

① 「STARFAX Server SDK オペレーションマニュアル」をお読みください。

- ［ STARFAX Server SDK のセットアップ(本体) ］を行って下ください。
- STARFAX Server SDK の基本的な操作を理解してください。
- もう１台のコンピュータで以下を行ってください。

［ クライアント送信のサンプルとＯＣＸ のセットアップ ］
［ プリンタドライバ のセットアップ ］(※)
［ ビューア のセットアップ ］(※)
(※サンプルプログラムによって必要となります)

② ［ STARFAX Server SDK 本体をセットアップしているコンピュータ ］と［ クライアント送信を行うコンピュータ ］の連絡を、共有フォルダを利用して行う場合は、以下をご覧ください。
第Ⅳ章　FAX 送信命令フォルダを共有して FAX 送信

メール送信を利用して連絡を行う場合は、以下をご覧ください。
第Ⅴ章　［ メール de ＦＡＸ ］で FAX 送信

そして、クライアントコンピュータから FAX 送信する仕組み、及び、各種サンプルプログラムを学習してください。

③ ユーザープログラム を作成してください。

- ①～② を踏まえて、ユーザープログラムの作成・テストを行って下さい。

第Ⅳ章

FAX 送信命令フォルダを共有して FAX 送信

FAX 送信命令フォルダを共有して、STARFAX Server SDK 本体をセットアップしていないコンピュータから FAX 送信する方法についてご説明します。

4.1　FAX 送信命令フォルダを共有して FAX 送信

FAX 送信命令フォルダを共有して、STARFAX Server SDK 本体をセットアップしていないコンピュータから FAX 送信する方法は、以下の通りです。
（以降、STARFAX Server SDK 本体をセットアップしているコンピュータを [サーバー側]、STARFAX Server SDK 本体をセットアップしていないコンピュータを [クライアント側] とします）

① [サーバー側]で STARFAX Server SDK を起動します。
（STARFAX Server SDK の起動は、「STARFAX Server SDK セットアップアニュアル」P21 参照）

② [サーバー側] の 送信命令フォルダ[制御関連インターフェイスフォルダ¥SENDMIS] を共有設定して、他のコンピュータからアクセス(読み書き)できるようにします。
（※フォルダの共有設定方法は、ＯＳにより異なります）

③ [クライアント側]から以下の内容で、共有している送信命令フォルダに、送信命令ファイルを作成します。

- 送信命令フォルダに作業フォルダを作成します。
- 送信原稿、送付状は、作業フォルダに設定して、相対的に扱うモードを指定します。
- 送信命令処理後、作業フォルダを削除するように指定します。

（FAX 送信の詳細は「STARFAX Server SDK ファイル de FAX」P6「送信命令ファイルを作成し、FAX 送信を指示する」をご覧下さい）

4.2　サンプルプログラム

サンプルプログラムは、３種類のプログラムを用意しています。それぞれ、元となるサンプルプログラムがあり、FAX 送信に関する部分を共有フォルダに出力する仕組みに変更して、クライアント動作するようにしています。
共有フォルダ以外の処理については、元の各種マニュアルをご覧ください。

● FAX 送信する (SendFAX.exe)

元のプログラムは、「STARFAX Server SDK プログラミング(ファイル操作版)」の
2.1 FAX 送信 をご覧ください。

●印刷結果の FAX 送信 (PrtCli.exe)

元のプログラムは、「STARFAX Server SDK VC 開発向けプリンタドライバとビューア」の
2.1 印刷結果の FAX 送信 をご覧ください。

● TIFF ファイルの作成と FAX 送信 (PrtCli2.exe)

元のプログラムは、「STARFAX Server SDK VC 開発向けプリンタドライバとビューア」の
2.2 TIFF ファイルの作成と FAX 送信 をご覧ください。

4.2.1　FAX 送信

FAX 送信プログラム【SendFax.exe】 は、STARFAX Server SDK 本体をセットアップしていないコンピュータから FAX の送信を行うサンプルプログラムです。本 CD-ROM の以下の位置に入っています。

¥サンプル¥応用サンプル¥クライアント¥共有フォルダ¥VC6¥SendFax.exe ...　FAX 送信プログラム
¥サンプル¥応用サンプル¥クライアント¥共有フォルダ¥VC6¥SendFax¥
...　FAX 送信プログラム 開発プロジェクト

なお、このサンプルプログラムには、元となるサンプルプログラムがあり、FAX 送信に関する部分を共有フォルダに出力する仕組みに変更して、クライアント動作するようにしています。
共有フォルダ以外の処理については、「STARFAX Server SDK プログラミング(ファイル操作版)」の *2.1 FAX送信* をご覧ください。

主な仕様、および操作方法は以下の通りです。
(以降、STARFAX Server SDK 本体をセットアップしているコンピュータを [サーバー側]、STARFAX Server SDK 本体をセットアップしていないコンピュータを [クライアント側] とします)

① [サーバー側]で STARFAX Server SDK を起動します。
(STARFAX Server SDK の起動は、「STARFAX Server SDK セットアップアニュアル」P21 参照)

② [サーバー側] の 送信命令フォルダ[制御関連インターフェイスフォルダ¥SENDMIS] を共有設定して、他のコンピュータからアクセス(読み書き)できるようにします。
(※フォルダの共有設定方法は、ＯＳにより異なります)

③ [クライアント側]で FAX 送信プログラム【SendFax.exe】 を起動します。
(起動時の作業(カレント)フォルダの指定は特にありません)

④ [共有設定した FAX 送信フォルダ(X)]を指定します。

⑤ 各種送信内容を指定します。

- [相手先(S)]ボタンを押して、相手先を指定します。
- [原稿(D)]ボタンを押して、原稿を指定します。
- 必要であれば、[送付状(C)]ボタンを押して、送付状を指定します。
- 必要であれば、[発信元(U)]ボタンを押して、発信元情報を指定します。

⑥ [送信(G)]ボタンを押して、FAX 送信を行います。
この後、FAX 送信が正常に動作していない場合は、[サーバー側] の STARFAX ログ管理プログラム で [イベント] の内容を確認して下さい。

ApiSend.cpp ：

```
//**********************************************************************
// 関数名　　：MakeSendMissionToSharedFolder
// 機能　　：共有フォルダ形式の送信命令ファイル作成
// 呼び出し　：MakeSendMissionToSharedFolder(LPCTSTR pSharedFolder, SENDFAX_MISSION *pInfo, int *piError )
// 入力　　：pSharedFolder = 共有フォルダ
//　　：pInfo　　= 送信命令ファイル作成情報ポインタ
//　　：piError　　= エラー時、エラー取得用ポインタ
// 出力　　：返値　TRUE　：正常終了
//　　：　　FALSE：失敗
//　　：　　*piError = エラーコード
//　　：
//　　：　【エラーコード】
//　　：　　SENDFAX_ERR_GetTempFolder　　... 一時フォルダの取得に失敗しました
//　　：　　SENDFAX_ERR_GetTempFile　　... 一時ファイルの取得に失敗しました
//　　：　　SENDFAX_ERR_MakeMisName　　... 送信命令ファイル名の作成に失敗しました
//　　：　　SENDFAX_ERR_NoShareFolder　　... 共有フォルダが存在しません
//　　：　　SENDFAX_ERR_MakeShareDocFolder ... 共有フォルダに送信原稿フォルダを作成できませんでした
//　　：　　SENDFAX_ERR_MakeShareDocFile　... 共有フォルダに送信原稿をコピーできませんでした
//　　：　　SENDFAX_ERR_MakeShareCvrFile　... 共有フォルダに送付状をコピーできませんでした
//　　：　　SENDFAX_ERR_PARAM_INFO　　... 関数引数エラー: 送信命令ファイル作成情報が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_SENDNUM　　... 関数引数エラー: 相手先数の指定が 0 です。
//　　：　　SENDFAX_ERR_PARAM_SENDINFO　　... 関数引数エラー: 相手先情報が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_FAX　　... 関数引数エラー: FAX 番号が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_DOCorCOVER　... 関数引数エラー: 送信原稿ファイル、送付状ファイルが、
//　　：　　　　ともに指定されていません。
//　　：　　SENDFAX_ERR_PARAM_DOCNAME　　... 関数引数エラー: 送信原稿ファイル名が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_SHAREFOLDER　... 関数引数エラー: 共有フォルダが指定されていません。
//　　：
// 特記事項　　：　共有フォルダ形式の送信命令ファイルを作成します。
// 作成者　　：
// 作成日　　：2010.07.20
//**********************************************************************
BOOL APIENTRY MakeSendMissionToSharedFolder(LPCTSTR pSharedFolder, SENDFAX_MISSION *pInfo, int *piError )
{

　～
```

```
////////////////////////////////////////////////////////////////////////////
// (1)  簡単な引数の内容チェック
//

// 共有フォルダチェック
if(!pSharedFolder || !(*pSharedFolder) ) {
    *piError = SENDFAX_ERR_PARAM_SHAREFOLDER;
    Goto    EXIT;
}

～

////////////////////////////////////////////////////////////////////////////
// (2)  共有フォルダパスのチェック
//

if(GetFileAttributes((char *)pSharedFolder ) == -1 ) {
    *piError = SENDFAX_ERR_NoShareFolder;
    Goto    EXIT;
}

////////////////////////////////////////////////////////////////////////////
// (3)  共有フォルダに送信原稿フォルダを作成
//

if(!GetTempFileName(pSharedFolder, "SFCS", 0, szDocFolder ) ) {
    *piError = SENDFAX_ERR_MakeShareDocFolder;
    Goto    EXIT;
}

DeleteFile(szDocFolder );

if(!CreateDirectory(szDocFolder, NULL ) ) {
    *piError = SENDFAX_ERR_MakeShareDocFolder;
    Goto    EXIT;
}

SUB_GetFileName(szDocFolder, szDocFolderName );

bDocFolder = TRUE;
```

～

```
////////////////////////////////////////////////////////////////////////////
// (5)  一時ファイルへ書き込み
//

～

// セクション名: [Doc]        ... 送信原稿
// ・ Num         ... 原稿数
if(pInfo->iDocNum ) {

   ～

   // ・ RelPath    ... 原稿ファイルパスを送信命令フォルダの相対パスとする
   //                   ("0":絶対パス, "1":相対パス)
   //                   (指定がない場合は "0":絶対パス です)
   WritePrivateProfileString("Doc", "RelPath", "1", szTempFile );
}

// セクション名: [Cover]     ... 送付状
// ・ Name        ... 送付状ファイルパス(.txt)
if(pInfo->pCoverName ) {

   ～

   // ・ RelPath    ... 送付状ファイルパスを送信命令フォルダの相対パスとする
   //                   ("0":絶対パス, "1":相対パス)
   //                   (指定がない場合は "0":絶対パス です)
   WritePrivateProfileString("Cover", "RelPath", "1", szTempFile );
}

～
```

```
    // セクション名: [Delete]    ... 削除要求
    //    ・ Folder      ... 命令ファイル処理後、削除フォルダ

    // 共有フォルダの相対パス
    WritePrivateProfileString("Delete", "Folder", szDocFolderName, szTempFile );

    // ・ RelPath    ... 削除フォルダを送信命令フォルダの相対パスとする
    //                   ("0":絶対パス, "1":相対パス)
    //                   (指定がない場合は "0":絶対パス です)
    WritePrivateProfileString("Delete", "RelPath", "1", szTempFile );


    ～


    return bRet;
}
```

4.2.2 印刷結果のＦＡＸ送信

印刷結果のＦＡＸ送信プログラム【PrtCli.exe】 は、STARFAX Server SDK 本体をセットアップしていないコンピュータで 印刷結果の表示と FAX 送信を行うサンプルプログラムです。
本 CD-ROM の以下の位置に入っています。ワード・エクセル等のアプリケーションから手動で印刷後、プリンタドライバからユーザープログラムが起動されます。

¥サンプル¥応用サンプル¥クライアント¥共有フォルダ¥VC6¥PrtCli.exe
...印刷結果のＦＡＸ送信プログラム
¥サンプル¥応用サンプル¥クライアント¥共有フォルダ¥VC6¥PrtCli¥
...印刷結果のＦＡＸプログラム 開発プロジェクト

なお、このサンプルプログラムには、元となるサンプルプログラムがあり、FAX 送信に関する部分を共有フォルダに出力する仕組みに変更して、クライアント動作するようにしています。共有フォルダ以外の処理については、「STARFAX Server SDK VC 開発向けプリンタドライバとビューア」の *2.1 印刷結果の FAX 送信* をご覧ください。

主な仕様、および操作方法は以下の通りです。
（以降、STARFAX Server SDK 本体をセットアップしているコンピュータを [サーバー側]、STARFAX Server SDK 本体をセットアップしていないコンピュータを [クライアント側] とします）

① [サーバー側]で STARFAX Server SDK を起動します。
（STARFAX Server SDK の起動は、「STARFAX Server SDK セットアップアニュアル」P21 参照）

② [サーバー側] の 送信命令フォルダ[制御関連インターフェイスフォルダ¥SENDMIS] を共有設定して、他のコンピュータからアクセス(読み書き)できるようにします。
（※フォルダの共有設定方法は、ＯＳにより異なります）

③ [クライアント側]で、以下の STARFAX Server SDK プリンタドライバの動作に関するレジストリを指定します。

- HKEY_LOCAL_MACHINE¥Software¥MEGASOFT¥SfCs¥OutFolder … ファイル出力フォルダ
  文字列項目で、任意の作業フォルダを作成して指定します。
  （例: "C:¥Program Files¥SfCs¥Temp" ）
- HKEY_LOCAL_MACHINE¥Software¥MEGASOFT¥SfCs¥DocName …ドキュメント名
  文字列項目で、 このサンプルプログラムの場合、任意の文字列を指定します。
  （例: "SFCSPRN" ）

- HKEY_LOCAL_MACHINE¥Software¥MEGASOFT¥SfCs¥ExecFlag …プログラム実行フラグ
  DWORD 項目で、 1 を指定します。
- HKEY_LOCAL_MACHINE¥Software¥MEGASOFT¥SfCs¥ExecPath …プログラムパス
  文字列項目で、 PrtCli.exe をフルパスで指定します。
  (例: "C:¥Program Files¥SfCs¥PrtCli.exe")
- HKEY_LOCAL_MACHINE¥Software¥MEGASOFT¥SfCs¥ExecParam …追加パラメータ
  文字列項目で、 何も指定していない状態("") を設定します。

④ 印刷可能な適当なアプリケーション(ワード等)から プリンタ名 "MEGASOFT STARFAX Server SDK" に対して印刷を行うと 印刷結果のＦＡＸ送信プログラム【PrtCli.exe】 が起動して、印刷結果リスト に 印刷結果が 登録された状態になります。

(または、印刷結果のＦＡＸ送信プログラム【PrtCli.exe】 を起動して、[テスト印刷(T)]ボタンでテスト印刷 を行っても 同じ状態になります)

⑤ [共有設定したFAX送信フォルダ(X)]を指定します。

⑥ [表示(V)]ボタンを押して、印刷結果の内容を確認します。

⑦ [ＦＡＸ送信(S)]ボタンを押して、FAX送信を行います。
この後、FAX 送信が正常に動作していない場合は、[サーバー側] の STARFAX ログ管理プログラム で [イベント] の内容を確認して下さい。

ApiSend.cpp ：

```
//**********************************************************************
// 関数名　　：MakeSendMissionToSharedFolder
// 機能　　　：共有フォルダ形式の送信命令ファイル作成
// 呼び出し　：MakeSendMissionToSharedFolder(LPCTSTR pSharedFolder, SENDFAX_MISSION *pInfo, int *piError )
// 入力　　　：pSharedFolder = 共有フォルダ
//　　　　　：pInfo　　= 送信命令ファイル作成情報ポインタ
//　　　　　：piError　　= エラー時、エラー取得用ポインタ
// 出力　　　：返値　TRUE　：正常終了
//　　　　　：　　FALSE：失敗
//　　　　　：　　　　*piError = エラーコード
//　　　　　：
//　　　　　：　【エラーコード】
//　　　　　：　　SENDFAX_ERR_GetTempFolder　　... 一時フォルダの取得に失敗しました
//　　　　　：　　SENDFAX_ERR_GetTempFile　　... 一時ファイルの取得に失敗しました
//　　　　　：　　SENDFAX_ERR_MakeMisName　　... 送信命令ファイル名の作成に失敗しました
//　　　　　：　　SENDFAX_ERR_NoShareFolder　　... 共有フォルダが存在しません
//　　　　　：　　SENDFAX_ERR_MakeShareDocFolder ... 共有フォルダに送信原稿フォルダを作成できませんでした
//　　　　　：　　SENDFAX_ERR_MakeShareDocFile　... 共有フォルダに送信原稿をコピーできませんでした
//　　　　　：　　SENDFAX_ERR_MakeShareCvrFile　... 共有フォルダに送付状をコピーできませんでした
//　　　　　：　　SENDFAX_ERR_PARAM_INFO　　... 関数引数エラー: 送信命令ファイル作成情報が指定されていません。
//　　　　　：　　SENDFAX_ERR_PARAM_SENDNUM　　... 関数引数エラー: 相手先数の指定が 0 です。
//　　　　　：　　SENDFAX_ERR_PARAM_SENDINFO　　... 関数引数エラー: 相手先情報が指定されていません。
//　　　　　：　　SENDFAX_ERR_PARAM_FAX　　... 関数引数エラー: FAX 番号が指定されていません。
//　　　　　：　　SENDFAX_ERR_PARAM_DOCorCOVER　... 関数引数エラー: 送信原稿ファイル、送付状ファイルが、
//　　　　　：　　　　ともに指定されていません。
//　　　　　：　　SENDFAX_ERR_PARAM_DOCNAME　　... 関数引数エラー: 送信原稿ファイル名が指定されていません。
//　　　　　：　　SENDFAX_ERR_PARAM_SHAREFOLDER　... 関数引数エラー: 共有フォルダが指定されていません。
//　　　　　：
// 特記事項　　：　共有フォルダ形式の送信命令ファイルを作成します。
// 作成者　　：
// 作成日　　：2010.07.20
//**********************************************************************
BOOL APIENTRY MakeSendMissionToSharedFolder(LPCTSTR pSharedFolder, SENDFAX_MISSION *pInfo, int *piError )
{

    ～
```

```
////////////////////////////////////////////////////////////////////////////
// (1)  簡単な引数の内容チェック
//

// 共有フォルダチェック
if(!pSharedFolder || !(*pSharedFolder) ) {
    *piError = SENDFAX_ERR_PARAM_SHAREFOLDER;
    Goto    EXIT;
}

〜

////////////////////////////////////////////////////////////////////////////
// (2)  共有フォルダパスのチェック
//

if(GetFileAttributes((char *)pSharedFolder ) == -1 ) {
    *piError = SENDFAX_ERR_NoShareFolder;
    Goto    EXIT;
}

////////////////////////////////////////////////////////////////////////////
// (3)  共有フォルダに送信原稿フォルダを作成
//

if(!GetTempFileName(pSharedFolder, "SFCS", 0, szDocFolder ) ) {
    *piError = SENDFAX_ERR_MakeShareDocFolder;
    Goto    EXIT;
}

DeleteFile(szDocFolder );

if(!CreateDirectory(szDocFolder, NULL ) ) {
    *piError = SENDFAX_ERR_MakeShareDocFolder;
    Goto    EXIT;
}

SUB_GetFileName(szDocFolder, szDocFolderName );

bDocFolder = TRUE;
```

～

```
////////////////////////////////////////////////////////////////////////////
// (5)  一時ファイルへ書き込み
//

～

// セクション名: [Doc]        ... 送信原稿
// ・ Num         ... 原稿数
if(pInfo->iDocNum ) {

   ～

   // ・ RelPath    ... 原稿ファイルパスを送信命令フォルダの相対パスとする
   //                   ("0":絶対パス, "1":相対パス)
   //                   (指定がない場合は "0":絶対パス です)
   WritePrivateProfileString("Doc", "RelPath", "1", szTempFile );
}

// セクション名: [Cover]     ... 送付状
// ・ Name        ... 送付状ファイルパス(.txt)
if(pInfo->pCoverName ) {

   ～

   // ・ RelPath    ... 送付状ファイルパスを送信命令フォルダの相対パスとする
   //                   ("0":絶対パス, "1":相対パス)
   //                   (指定がない場合は "0":絶対パス です)
   WritePrivateProfileString("Cover", "RelPath", "1", szTempFile );
}

～
```

```
// セクション名: [Delete]    ... 削除要求
   //    ・ Folder      ... 命令ファイル処理後、削除フォルダ

   // 共有フォルダの相対パス
   WritePrivateProfileString("Delete", "Folder", szDocFolderName, szTempFile );

   // ・ RelPath    ... 削除フォルダを送信命令フォルダの相対パスとする
   //                   ("0":絶対パス, "1":相対パス)
   //                   (指定がない場合は "0":絶対パス です)
   WritePrivateProfileString("Delete", "RelPath", "1", szTempFile );


   ～


   return bRet;
}
```

4.2.3　TIFF ファイルの作成とＦＡＸ送信

TIFF ファイルのＦＡＸ送信プログラム【PrtCli2.exe】 は、STARFAX Server SDK 本体をセットアップしていないコンピュータで TIFF ファイルの作成 と FAX 送信を行うサンプルプログラムです。本 CD-ROM の以下の位置に入っています。
ユーザープログラムがプリンタドライバを制御して印刷結果(TIFF ファイル)を取得します。

¥サンプル¥応用サンプル¥クライアント¥共有フォルダ¥VC6¥PrtCli2.exe
... TIFF ファイルのＦＡＸ送信プログラム
¥サンプル¥応用サンプル¥クライアント¥共有フォルダ¥VC6¥PrtCli2¥
... TIFF ファイルのＦＡＸ送信プログラム 開発プロジェクト

なお、このサンプルプログラムには、元となるサンプルプログラムがあり、FAX 送信に関する部分を共有フォルダに出力する仕組みに変更して、クライアント動作するようにしています。共有フォルダ以外の処理については、「STARFAX Server SDK VC 開発向けプリンタドライバとビューア」の *2.2 TIFF ファイルの作成と FAX 送信* をご覧ください。

主な仕様、および操作方法は以下の通りです。
(以降、STARFAX Server SDK 本体をセットアップしているコンピュータを [サーバー側]、STARFAX Server SDK 本体をセットアップしていないコンピュータを [クライアント側] とします)

① [サーバー側]で STARFAX Server SDK を起動します。
(STARFAX Server SDK の起動は、「STARFAX Server SDK セットアップアニュアル」P21 参照)

② [サーバー側] の 送信命令フォルダ[制御関連インターフェイスフォルダ¥SENDMIS] を共有設定して、他のコンピュータからアクセス(読み書き)できるようにします。
(※フォルダの共有設定方法は、ＯＳにより異なります)

③ [クライアント側]で PrtCli2.exe を起動します。

④ [共有設定した FAX 送信フォルダ(X)]を指定します。

⑤ 「操作１」の下欄に FAX 原稿に表示させる文字を入力します。

⑥ 「①FAX 原稿の作成」ボタンをクリックします。
(作成するファイル名を任意指定したい場合は、[指定]ラジオボタンをクリックして、ファイル名を入力してください)

⑦ 「②FAX 原稿の表示」ボタンをクリックすると、作成された FAX 原稿が STARFAX Server SDK ビューアで表示されます。

⑧ 「操作３」の下欄に送信先の FAX 番号を入力します。

⑨ 「FAX 送信」ボタンをクリックします。
この後、FAX 送信が正常に動作していない場合は、[サーバー側] の STARFAX Server SDK ログ管理プログラム で [イベント] の内容を確認して下さい。

ApiSend.cpp ：

```
//*********************************************************************
// 関数名　　：MakeSendMissionToSharedFolder
// 機能　　：共有フォルダ形式の送信命令ファイル作成
// 呼び出し　：MakeSendMissionToSharedFolder(LPCTSTR pSharedFolder, SENDFAX_MISSION *pInfo, int *piError )
// 入力　　：pSharedFolder = 共有フォルダ
//　　：pInfo　　= 送信命令ファイル作成情報ポインタ
//　　：piError　　= エラー時、エラー取得用ポインタ
// 出力　　：返値　TRUE　：正常終了
//　　：　　FALSE：失敗
//　　：　　*piError = エラーコード
//　　：
//　　：　【エラーコード】
//　　：　　SENDFAX_ERR_GetTempFolder　　... 一時フォルダの取得に失敗しました
//　　：　　SENDFAX_ERR_GetTempFile　　... 一時ファイルの取得に失敗しました
//　　：　　SENDFAX_ERR_MakeMisName　　... 送信命令ファイル名の作成に失敗しました
//　　：　　SENDFAX_ERR_NoShareFolder　　... 共有フォルダが存在しません
//　　：　　SENDFAX_ERR_MakeShareDocFolder ... 共有フォルダに送信原稿フォルダを作成できませんでした
//　　：　　SENDFAX_ERR_MakeShareDocFile　... 共有フォルダに送信原稿をコピーできませんでした
//　　：　　SENDFAX_ERR_MakeShareCvrFile　... 共有フォルダに送付状をコピーできませんでした
//　　：　　SENDFAX_ERR_PARAM_INFO　　... 関数引数エラー: 送信命令ファイル作成情報が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_SENDNUM　　... 関数引数エラー: 相手先数の指定が 0 です。
//　　：　　SENDFAX_ERR_PARAM_SENDINFO　　... 関数引数エラー: 相手先情報が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_FAX　　... 関数引数エラー: FAX 番号が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_DOCorCOVER　... 関数引数エラー: 送信原稿ファイル、送付状ファイルが、
//　　：　　　　ともに指定されていません。
//　　：　　SENDFAX_ERR_PARAM_DOCNAME　　... 関数引数エラー: 送信原稿ファイル名が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_SHAREFOLDER　... 関数引数エラー: 共有フォルダが指定されていません。
//　　：
// 特記事項　　：　共有フォルダ形式の送信命令ファイルを作成します。
// 作成者　　：
// 作成日　　：2010.07.20
//*********************************************************************
BOOL APIENTRY MakeSendMissionToSharedFolder(LPCTSTR pSharedFolder, SENDFAX_MISSION *pInfo, int *piError )
{

　～
```

```
////////////////////////////////////////////////////////////////////////////
// (1)  簡単な引数の内容チェック
//

// 共有フォルダチェック
if(!pSharedFolder || !(*pSharedFolder) ) {
    *piError = SENDFAX_ERR_PARAM_SHAREFOLDER;
    Goto    EXIT;
}

～

////////////////////////////////////////////////////////////////////////////
// (2)  共有フォルダパスのチェック
//

if(GetFileAttributes((char *)pSharedFolder ) == -1 ) {
    *piError = SENDFAX_ERR_NoShareFolder;
    Goto    EXIT;
}

////////////////////////////////////////////////////////////////////////////
// (3)  共有フォルダに送信原稿フォルダを作成
//

if(!GetTempFileName(pSharedFolder, "SFCS", 0, szDocFolder ) ) {
    *piError = SENDFAX_ERR_MakeShareDocFolder;
    Goto    EXIT;
}

DeleteFile(szDocFolder );

if(!CreateDirectory(szDocFolder, NULL ) ) {
    *piError = SENDFAX_ERR_MakeShareDocFolder;
    Goto    EXIT;
}

SUB_GetFileName(szDocFolder, szDocFolderName );

bDocFolder = TRUE;
```

～

```
////////////////////////////////////////////////////////////////////////////
// (5)  一時ファイルへ書き込み
//

～

// セクション名: [Doc]        ... 送信原稿
// ・ Num         ... 原稿数
if(pInfo->iDocNum ) {

   ～

   // ・ RelPath    ... 原稿ファイルパスを送信命令フォルダの相対パスとする
   //                   ("0":絶対パス, "1":相対パス)
   //                   (指定がない場合は "0":絶対パス です)
   WritePrivateProfileString("Doc", "RelPath", "1", szTempFile );
}

// セクション名: [Cover]     ... 送付状
// ・ Name        ... 送付状ファイルパス(.txt)
if(pInfo->pCoverName ) {

   ～

   // ・ RelPath    ... 送付状ファイルパスを送信命令フォルダの相対パスとする
   //                   ("0":絶対パス, "1":相対パス)
   //                   (指定がない場合は "0":絶対パス です)
   WritePrivateProfileString("Cover", "RelPath", "1", szTempFile );
}

～
```

```
// セクション名: [Delete]    ... 削除要求
   //    ・ Folder      ... 命令ファイル処理後、削除フォルダ

   // 共有フォルダの相対パス
   WritePrivateProfileString("Delete", "Folder", szDocFolderName, szTempFile );

   // ・ RelPath    ... 削除フォルダを送信命令フォルダの相対パスとする
   //                   ("0":絶対パス, "1":相対パス)
   //                   (指定がない場合は "0":絶対パス です)
   WritePrivateProfileString("Delete", "RelPath", "1", szTempFile );


   ～


   return bRet;
}
```

第Ⅴ章

[メール de ＦＡＸ]でFAX送信

[メール de ＦＡＸ]機能で、STARFAX Server SDK 本体をセットアップしていないコンピュータからFAX送信する方法についてご説明します。

5.1 ［ メール de ＦＡＸ ］でFAX送信

［ メール de ＦＡＸ ］機能で、STARFAX Server SDK本体をセットアップしていないコンピュータからFAX送信する方法は、以下の通りです。

以降、STARFAX Server SDK本体をセットアップしているコンピュータを［ サーバー側 ］、STARFAX Server SDK本体をセットアップしていないコンピュータを［ クライアント側 ］とします。

① ［ サーバー側 ］で［ メール de ＦＡＸ ］の設定を有効にします。

(1) タスクトレイの STARFAX Server SDK のアイコンを右クリックして表示されるメニューから［ メール de ＦＡＸ ］を実行します。

(2) 最低限、以下の項目を設定します。

- ［ メール de ＦＡＸ を有効にする(Y) ］をチェックします。
- ［ メールサーバーの設定(S) ］を行います。

(3) 必要に応じて、以下の項目を設定します。

- ［ 件名に含まれる文字で識別(I) ］
- ［ メール受信間隔(N) ］
- ［ 対象外のメールをEMLファイルに保存(P) ］、及び、関連項目
- ［ FAX送信結果をメール通知する(N) ］、及び、関連項目

② ［ サーバー側 ］で STARFAX Server SDK を起動します。
(STARFAX Server SDK の起動は、「STARFAX Server SDK セットアップアニュアル」P21 参照)

③ ［ クライアント側 ］から［ メール de ＦＡＸ ］の依頼メールを［ サーバー側 ］で設定した［ POP メールアドレス ］に送信します。

([メール de ＦＡＸ ］の依頼メールの詳細は *3.2 ［ メール de ＦＡＸ ］の依頼メールの仕様* をご覧下さい)

5.2 ［ メール de ＦＡＸ ］の依頼メールの仕様

STARFAX Server SDK の［ メール de ＦＡＸ ］機能が動作している場合、設定された［ POP メールアドレス ］に対しての依頼メールを送信すると、FAX 送信を行います。

この仕組みには、以下の特徴があります。

- **相手先を複数指定することができます。（同報送信）**
- **送信原稿ファイルを複数指定することができます。**
- **送付状を指定することができます。**
  （送付状ファイルは、テキストファイルで、差込内容を %? で指定できます）
- **発信元情報を指定することができます。**

依頼メールの仕様についてご説明します。

■ 依頼メールの構成

- **宛先**

  ［ メール de ＦＡＸ 設定 ］の［ メールサーバー設定(S) ］で設定されている［ POP メールアドレス ］を指定します。

- **件名**

  件名には、［ メール de ＦＡＸ 設定 ］の［ 件名に含まれる文字で識別(I) ］で設定されている文字列を含んでいる必要があります。含んでいないと依頼メールとして扱われません。

  例: ″ Mail to FAX″

- **本文**

  ［ メール de ＦＡＸ ］では扱われません。
  （設定されていても FAX 送信されることはありません）

● **添付ファイル**

添付ファイルは以下のファイルを指定します。

- **送信命令ファイル**
- **送信原稿ファイル**
- **送付状ファイル**

詳細については後述をご覧ください。

■ 送信命令ファイル

送信命令ファイルは、INI ファイル形式のファイルです。
ファイル名は、必ず、 **"Trans.txt"** である必要があります。
全ての項目を指定する必要はありません。
最低限、相手先の FAX 番号を指定すれば FAX 送信できます。(※)
(※: 添付ファイルで送信原稿ファイルが設定されている必要があります)

```
[SendInfo]
Num=1
[Send1]
Fax=0663868894
```

その他の情報は、送付状への差込、発信元情報への差込、ログに情報を残したい、等の必要に応じて設定します。

● **セクション名: [SendInfo] ... 送信のための相手先情報**

- Num ... 送信相手先数

● **セクション名: [Send%d]　... 送信のための相手先内容 (1～)**

- Fax　... FAX 番号 (最大 128 バイト)
- Company　... 会社名 (最大 128 バイト)
- Division　... 所属名 (最大 128 バイト)
- Position　... 役職名 (最大 128 バイト)
- Name　... 氏名 (最大 128 バイト)
- Title　... 敬称 (最大 128 バイト)
- Telephone　... 電話番号 (最大 128 バイト)
- ZipCode　... 郵便番号 (最大 128 バイト)
- Address1　... 住所 1 (最大 128 バイト)
- Address2　... 住所 2 (最大 128 バイト)
- Speed　... 通信速度 ("0":自動, "1":高速, "2":中速, "3":低速)
- Comp　... 圧縮方式 ("0":自動, "1":MH, "2":MR, "3":MMR)
- Ecm　... エラー訂正 ("0":自動, "1":利用する, "2":利用しない)
- FCode　... Ｆコード番号 (最大 20 バイト)
- FreeArea　... ユーザが自由に利用できるエリア (最大 128 バイト)
- Line　... 送信回線指定 (0～)
  (指定がない場合は、空いている回線から送信されます)
- Priority　... 優先順位 (0～15)
  (指定がない場合の優先順位は 8 です)
- Time　... 送信時刻 ("YYYYMMDDHHMMSS")

● **セクション名: [Cover]　... 送付状**

- Name　... 送付状ファイルパス(.txt)
  (さらに、添付ファイルで設定する必要があります)
- FontName　... 送付状フォント名
  (指定がない場合は "ＭＳ ゴシック" です)
- FontSize　... 送付状フォントサイズ (8 ～ 72 (ポイント))
  (指定がない場合は 10 ポイント です)

● **セクション名: [User]　... 発信元情報**

- UserInfo　... 発信元情報記録 ("記録位置,記録情報")
  ("0":記録しない, "1":原稿の内側に記録, "2":原稿の外側に記録)
- UserID　... 自局電話番号 (FAXID として、相手ファクシミリに通知されます)
  (最大 20バイトで、半角数字、スペース、+を設定してください)

【 発信元情報の仕様 】

発信元情報とは、FAX 送信時に、各ページ毎、送信原稿の先頭に付加する情報です。
発信元情報は、[User]-UserInfo に、"記録位置,記録情報" の形式で指定します。

● 記録位置

0: 記録しません。
1: 原稿の内側に記録します。原稿の内容によっては、原稿の先頭が少し消えてしまう可能性があります。
2: 原稿の外側に記録します。

1、または、2 を指定すると、記録情報を指定していなくも日付とページ数は記録されます。

```
[User]
UserInfo=1,
```

↓

```
                              2001 12/ 1 10:12      Page 01
```

● 記録情報

自由に文字を指定できます。
以下の差込も使用できます。

- %S ... 会社名 ([Send%d]セクション-Company が差し込まれます)
- %N ... 氏名 ([Send%d]セクション-Name と [Send%d]セクション-Title が差し込まれます)
- %T ... FAX 番号 ([Send%d]セクション-Fax が差し込まれます)

```
[Send1]
Company=山田サークル
Name=山田
[User]
UserInfo=1,メガ太郎 → %S %N
```

↓

```
     メガ太郎 → 山田サークル 山田様          2001 12/ 1 10:12      Page 01
```

■ 送信原稿ファイル

送信原稿ファイルは、以下の形式のファイルを複数指定可能です。

ファイル形式:
- TIFF 形式　圧縮なし
  - 修正 CCITT MH 圧縮
  - CCITT G3 MH 圧縮
  - CCITT G3 MR 圧縮
  - PackBits 圧縮
  - Class F 圧縮
  - G4 圧縮
  - JPEG 圧縮
- BMP ファイル
- PCX ファイル
- DCX ファイル
- JPEG ファイル
- テキストファイル
- FAX ファイル
- LNK ファイル

■ 送付状ファイルの仕様

テキストファイル(*.txt)で、自由に作成したファイルを指定できます。
文字フォントは、"ＭＳ ゴシック"、大きさは１０ポイントとして扱われ、FAX 送信されます。

以下の差込が有効です。

- **%S ... 会社名 ([Send%d]セクション-Company が差し込まれます)**
- **%D ... 所属名 ([Send%d]セクション-Division が差し込まれます)**
- **%Y ... 役職名 ([Send%d]セクション-Position が差し込まれます)**
- **%N ... 氏名 ([Send%d]セクション-Name が差し込まれます)**
- **%Z ... 郵便番号 ([Send%d]セクション-ZipCode が差し込まれます)**
- **%A ... 住所１ ([Send%d]セクション-Address1 が差し込まれます)**
- **%B ... 住所２ ([Send%d]セクション-Address2 が差し込まれます)**
- **%H ... 電話番号 ([Send%d]セクション-Telephone が差し込まれます)**
- **%T ... FAX 番号 ([Send%d]セクション-Fax が差し込まれます)**
- **%K ... 敬称 ([Send%d]セクション-Title が差し込まれます)**
- **%P ... ページ数 (自動的に計算され、差し込まれます)**
- **%x ... xxxx ([Send%d]セクション- が差し込まれます)**

ファクシミリ送付のご案内

%S
%D
%Y
%N　%K

メガソフト株式会社
メガ太郎

毎度格別のお引き立てにあずかり、まことにありがとうございます。
下記の書類を拝送しますので、よろしくご査収下さいますようお願い申し上げます。

記

以上

5.3　サンプルプログラム

サンプルプログラムは、３種類のプログラムを用意しています。それぞれ、元となるサンプルプログラムがあり、FAX 送信に関する部分を[メール de ＦＡＸ]の仕組みに変更して、クライアント動作するようにしています。
[メール de ＦＡＸ]以外の処理については、元の各種マニュアルをご覧ください。

● FAX 送信する (SendFAX.exe)

元のプログラムは、「STARFAX Server SDK プログラミングマニュアル」の
2.1 FAX 送信 をご覧ください。

●印刷結果の FAX 送信 (PrtCli.exe)

元のプログラムは、「STARFAX Server SDK VC 開発向けプリンタドライバとビューア」の
2.1 印刷結果の FAX 送信 をご覧ください。

● TIFF ファイルの作成と FAX 送信 (PrtCli2.exe)

元のプログラムは、「STARFAX Server SDK VC 開発向けプリンタドライバとビューア」の
2.2 TIFF ファイルの作成と FAX 送信 をご覧ください。

5.3.1　FAX 送信

FAX 送信プログラム【SendFax.exe】 は、STARFAX Server SDK 本体をセットアップしていないコンピュータから FAX の送信を行うサンプルプログラムです。本 CD-ROM の以下の位置に入っています。

¥サンプル¥応用サンプル¥クライアント¥メール de ＦＡＸ¥VC6¥SendFax.exe
... FAX 送信プログラム
¥サンプル¥応用サンプル¥クライアント¥メール de ＦＡＸ¥VC6¥SendFax¥
... FAX 送信プログラム 開発プロジェクト

なお、このサンプルプログラムには、元となるサンプルプログラムがあり、FAX 送信に関する部分を[メール de ＦＡＸ]の仕組みに変更して、クライアント動作するようにしています。
[メール de ＦＡＸ]以外の処理については、「STARFAX Server SDK プログラミング(ファイル操作版)」の *2.1 FAX送信* をご覧ください。

主な仕様、および操作方法は以下の通りです。
(以降、STARFAX Server SDK 本体をセットアップしているコンピュータを [サーバー側]、STARFAX Server SDK 本体をセットアップしていないコンピュータを [クライアント側] とします)

① [サーバー側]で [メール de ＦＡＸ] の設定を有効にします。

(1) タスクトレイの STARFAX Server SDK のアイコンを右クリックして表示されるメニューから [メール de ＦＡＸ] を実行します。

(2) 最低限、以下の項目を設定します。

- [メール de ＦＡＸ を有効にする(Y)] をチェックします。
- [メールサーバーの設定(S)] を行います。

(3) 必要に応じて、以下の項目を設定します。

- [件名に含まれる文字で識別(I)]
- [メール受信間隔(N)]
- [対象外のメールを EML ファイルに保存(P)]、及び、関連項目
- [FAX 送信結果をメール通知する(N)] 、及び、関連項目

② [サーバー側]で STARFAX Server SDK を起動します。
(STARFAX Server SDK の起動は、「STARFAX Server SDK セットアップアニュアル」P21 参照)

③ [クライアント側]で FAX 送信プログラム【SendFax.exe】 を起動します。
(起動時の作業(カレント)フォルダの指定は特にありません)

④ 各種送信内容を指定します。

- [相手先(S)]ボタンを押して、相手先を指定します。
- [原稿(D)]ボタンを押して、原稿を指定します。
- 必要であれば、[送付状(C)]ボタンを押して、送付状を指定します。
- 必要であれば、[発信元(U)]ボタンを押して、発信元情報を指定します。

⑤ [メール送信(G)]ボタンを押して、メール送信を行います。
この後､FAX 送信が正常に動作していない場合は､[サーバー側] の以下の表示をご確認下さい。

- STARFAX ログ管理プログラム の [イベント]
  (主に､[コントロール] と [メール] をご確認下さい)
- [メール de ＦＡＸ 設定] の システムメニュー の [イベントログ]

ApiSend.cpp ：

```
//*********************************************************************
// 関数名　　：MakeTransFileForMailToFax
// 機能　　：[ メール de ＦＡＸ ]の送信命令ファイル作成
// 呼び出し　：MakeTransFileForMailToFax(LPCTSTR pTransFile, SENDFAX_MISSION *pInfo, int *piError )
// 入力　　：pTransFile = 送信命令ファイル
//　　：pInfo　= 送信命令ファイル作成情報ポインタ
//　　：piError　= エラー時、エラー取得用ポインタ
// 出力　　：返値　TRUE　：正常終了
//　　：　FALSE：失敗
//　　：　　*piError = エラーコード
//　　：
//　　：　【エラーコード】
//　　：　　SENDFAX_ERR_GetTempFolder　... 一時フォルダの取得に失敗しました
//　　：　　SENDFAX_ERR_GetTempFile　... 一時ファイルの取得に失敗しました
//　　：　　SENDFAX_ERR_CreateTransFile　... 送信命令ファイルの作成に失敗しました
//　　：　　SENDFAX_ERR_PARAM_INFO　... 関数引数エラー: 送信命令ファイル作成情報が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_SENDNUM　... 関数引数エラー: 相手先数の指定が 0 です。
//　　：　　SENDFAX_ERR_PARAM_SENDINFO　... 関数引数エラー: 相手先情報が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_FAX　... 関数引数エラー: FAX 番号が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_TRANSFILE　... 関数引数エラー: 送信命令ファイルが指定されていません。
//　　：
// 特記事項　　：　[ メール de ＦＡＸ ]の送信命令ファイルを作成します。
// 作成者　　：
// 作成日　　：2010.07.20
//*********************************************************************
BOOL APIENTRY　MakeTransFileForMailToFax(LPCTSTR pTransFile, MAIL2FAX_MISSION *pInfo, int *piError )
{
    BOOL　bRet = FALSE;　// 返値 ... 初期値は失敗

    int iErrorDummy;　// NULL 用ダミーエラーコード設定

    int i;

    char　szGetName[MAX_PATH+1];　　// ファイル名取得用

    SENDFAX_SENDINFO　*pSendInfo; // 相手先情報用

    char　szSection[32];　// セクション設定用
    char　szWork[512];　// 作業用

    if(!piError ) {
        // エラーコード取得用ポインタにダミー設定
        piError = &iErrorDummy;
```

```
}
*piError = SENDFAX_SUCCESS;      // エラーコードに初期値設定 ... 正常終了

////////////////////////////////////////////////////////////////////////////
// (1)  簡単な引数の内容チェック
//

    // 送信命令ファイルチェック
if(!pTransFile || !(*pTransFile) ) {
    *piError = SENDFAX_ERR_PARAM_TRANSFILE;
    goto    EXIT;
}

    // 送信命令ファイル作成情報チェック
if(!pInfo ) {
    *piError = SENDFAX_ERR_PARAM_INFO;
    goto    EXIT;
}
    // 相手先数チェック
if(!pInfo->iSendNum ) {
    *piError = SENDFAX_ERR_PARAM_SENDNUM;
    goto    EXIT;
}
    // 相手先情報チェック
for(i = 0 ;  i < pInfo->iSendNum ; i++ ) {
    pSendInfo = *(pInfo->ppSend + i );
    if(!pSendInfo ) {
        *piError = SENDFAX_ERR_PARAM_SENDINFO;
        goto    EXIT;
    }
        // FAX 番号チェック
    if(!pSendInfo->szFax[0] ) {
        *piError = SENDFAX_ERR_PARAM_FAX;
        goto    EXIT;
    }
}

////////////////////////////////////////////////////////////////////////////
// (2)  送信命令ファイルへ書き込み
//

DeleteFile(pTransFile );   // 念のため削除

// セクション名: [SendInfo]  ... 送信のための相手先情報
```

```
// ・ Num        ... 送信相手先数
wsprintf(szWork, "%d", pInfo->iSendNum );
WritePrivateProfileString("SendInfo", "Num", szWork, pTransFile );

// セクション名: [Send%d]    ... 送信のための相手先内容 (1～)
for(i = 0 ;  i < pInfo->iSendNum ; i++ ) {
    pSendInfo = *(pInfo->ppSend + i );

    wsprintf(szSection, "Send%d", i + 1 );

    // ・ Fax        ... FAX番号 (最大128バイト)
    WritePrivateProfileString(szSection, "Fax", pSendInfo->szFax, pTransFile );
    // ・ Company    ... 会社名 (最大128バイト)
    if(pSendInfo->szCompany[0] ) {
        wsprintf(szWork, "¥"%s¥"", pSendInfo->szCompany );
        WritePrivateProfileString(szSection, "Company", szWork, pTransFile );
    }
    // ・ Division   ... 所属名 (最大128バイト)
    if(pSendInfo->szDivision[0] ) {
        wsprintf(szWork, "¥"%s¥"", pSendInfo->szDivision );
        WritePrivateProfileString(szSection, "Division", szWork, pTransFile );
    }
    // ・ Position   ... 役職名 (最大128バイト)
    if(pSendInfo->szPosition[0] ) {
        wsprintf(szWork, "¥"%s¥"", pSendInfo->szPosition );
        WritePrivateProfileString(szSection, "Position", szWork, pTransFile );
    }
    // ・ Name       ... 氏名 (最大128バイト)
    if(pSendInfo->szName[0] ) {
        wsprintf(szWork, "¥"%s¥"", pSendInfo->szName );
        WritePrivateProfileString(szSection, "Name", szWork, pTransFile );
    }
    // ・ Title      ... 敬称 (最大128バイト)
    if(pSendInfo->szTitle[0] ) {
        wsprintf(szWork, "¥"%s¥"", pSendInfo->szTitle );
        WritePrivateProfileString(szSection, "Title", szWork, pTransFile );
    }
    // ・ Telephone  ... 電話番号 (最大128バイト)
    if(pSendInfo->szTelephone[0] ) {
        WritePrivateProfileString(szSection, "Telephone", pSendInfo->szTelephone, pTransFile );
    }
    // ・ ZipCode    ... 郵便番号 (最大128バイト)
    if(pSendInfo->szZipCode[0] ) {
        WritePrivateProfileString(szSection, "ZipCode", pSendInfo->szZipCode, pTransFile );
    }
```

```
// ・ Address1   ... 住所1（最大128バイト）
if(pSendInfo->szAddress1[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szAddress1 );
    WritePrivateProfileString(szSection, "Address1", szWork, pTransFile );
}
// ・ Address2   ... 住所2（最大128バイト）
if(pSendInfo->szAddress2[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szAddress2 );
    WritePrivateProfileString(szSection, "Address2", szWork, pTransFile );
}
// ・ FCode      ... Ｆコード番号（最大20バイト）
if(pSendInfo->szFCode[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szFCode );
    WritePrivateProfileString(szSection, "FCode", szWork, pTransFile );
}
// ・ FreeArea   ... ユーザが自由に利用できるエリア（最大128バイト）
if(pSendInfo->szFreeArea[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szFreeArea );
    WritePrivateProfileString(szSection, "FreeArea", szWork, pTransFile );
}
// ・ Speed      ... 通信速度（"0":自動, "1":高速, "2":中速, "3":低速）
if(pSendInfo->iSpeed ) {
    wsprintf(szWork, "%d", pSendInfo->iSpeed );
    WritePrivateProfileString(szSection, "Speed", szWork, pTransFile );
}
// ・ Comp       ... 圧縮方式（"0":自動, "1":MH, "2":MR, "3":MMR）
if(pSendInfo->iComp ) {
    wsprintf(szWork, "%d", pSendInfo->iComp );
    WritePrivateProfileString(szSection, "Comp", szWork, pTransFile );
}
// ・ Ecm        ... エラー訂正（"0":自動, "1":利用する, "2":利用しない）
if(pSendInfo->iEcm ) {
    wsprintf(szWork, "%d", pSendInfo->iEcm );
    WritePrivateProfileString(szSection, "Ecm", szWork, pTransFile );
}
// ・ Line       ... 送信回線指定（0～）
if(pSendInfo->bLine ) {
    wsprintf(szWork, "%d", pSendInfo->iLine );
    WritePrivateProfileString(szSection, "Line", szWork, pTransFile );
}
// ・ Priority   ... 優先順位（0～15）
if(pSendInfo->bPriority ) {
    wsprintf(szWork, "%d", pSendInfo->iPriority );
    WritePrivateProfileString(szSection, "Priority", szWork, pTransFile );
}
```

```
    // ・ Time        ... 送信時刻 ("YYYYMMDDHHMMSS")
    if(pSendInfo->bTime ) {
        wsprintf(szWork, "%s", pSendInfo->szTime );
        WritePrivateProfileString(szSection, "Time", szWork, pTransFile );
    }
}

// セクション名: [Cover]     ... 送付状
// ・ Name        ... 送付状ファイルパス(.txt)
if(pInfo->pCoverName ) {

    SUB_GetFileName(pInfo->pCoverName, szGetName );

    wsprintf(szWork, "¥"%s¥"", szGetName );
    WritePrivateProfileString("Cover", "Name", szWork, pTransFile );
}
// ・ FontName   ... 送付状フォント名
if(pInfo->pCoverFontName ) {
    wsprintf(szWork, "¥"%s¥"", pInfo->pCoverFontName );
    WritePrivateProfileString("Cover", "FontName", szWork, pTransFile );
}
// ・ FontSize   ... 送付状フォントサイズ (8 ～ 72 (ポイント ) )
if(pInfo->iCoverFontSize ) {
    wsprintf(szWork, "%d", pInfo->iCoverFontSize );
    WritePrivateProfileString("Cover", "FontSize", szWork, pTransFile );
}

// セクション名: [User]       ... 発信元情報
// ・ UserInfo   ... 発信元情報記録 ("記録位置,記録情報") (最大 140 バイト)
//                    ("0":記録しない, "1":原稿の内側に記録, "2":原稿の外側に記録)
if(pInfo->szUserInfo[0] ) {
    wsprintf(szWork, "¥"%s¥"", pInfo->szUserInfo );
    WritePrivateProfileString("User", "UserInfo", szWork, pTransFile );
}
// ・ UserID     ... 自局電話番号 (FAXID として、相手ファクシミリに通知されます) (最大 20 バイト)
if(pInfo->szUserID[0] ) {
    wsprintf(szWork, "¥"%s¥"", pInfo->szUserID );
    WritePrivateProfileString("User", "UserID", szWork, pTransFile );
}

////////////////////////////////////////////////////////////////////////
// (3) 送信命令ファイルのフラッシュ 【重要】
//
//      OS により、WritePrivateProfileString() 【Win32API】 を使用しての
//    実際のファイルへの書き出しのタイミングが異なる為、必ずこのフラッシュの
```

```
    //    ステップが必要です。
    //

    WritePrivateProfileString(NULL, NULL, NULL, pTransFile );

    bRet = TRUE;    // 正常終了

EXIT:

    // エラー時は、送信命令ファイルを削除
    if(!bRet ) {
        DeleteFile(pTransFile );
    }

    return bRet;
}
```

SendFaxDlg.cpp ：

```
void CSendFaxDlg::OnOK()
{

    ～

    ////////////////////////////////////////////////////////////////////////////
    // (2) [ メール de ＦＡＸ ]の送信命令ファイル作成関数 呼び出し

    if(!MakeTransFileForMailToFax(szTrans, &MailToFaxInfo, &iError ) ) {

        switch(iError ) {
        case    SENDFAX_ERR_GetTempFolder:
            MessageBox("一時フォルダの取得に失敗しました", "FAX 送信", MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_GetTempFile:
            MessageBox("一時ファイルの取得に失敗しました", "FAX 送信", MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_CreateTransFile:
            MessageBox("送信命令ファイルの作成に失敗しました", "FAX 送信", MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_PARAM_INFO:
            MessageBox("関数引数エラー: 送信命令ファイル作成情報が指定されていません。", "FAX 送信",
                        MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_PARAM_SENDNUM:
            MessageBox(" 関数引数エラー: 相手先数の指定が 0 です。", "FAX 送信", MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_PARAM_SENDINFO:
            MessageBox("関数引数エラー: 相手先情報が指定されていません。", "FAX 送信", MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_PARAM_FAX:
            MessageBox("関数引数エラー: FAX 番号が指定されていません。", "FAX 送信", MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_PARAM_TRANSFILE:
            MessageBox("関数引数エラー: 送信命令ファイルが指定されていません。","FAX 送信",MB_ICONEXCLAMATION|MB_OK);
            break;
        default:
            break;
        }

    }
    else {
```

MessageBox(″このサンプルプログラムは MAPI を利用してメール送信を行います。¥n 宛先 に STARFAX Engine の［ メール de ＦＡＸ ］で設定した¥nPOP メールアドレス を指定して送信してください。¥n¥n なお、送信後、OUTLOOK 等メーラーを起動して送信を行わないと¥n 実際に送信されないことがあります。″, "FAX 送信", MB_ICONINFORMATION | MB_OK);

```
////////////////////////////////////////////////////////////////////////////
//
// メール送信
//
////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
// MAPI 初期化

HINSTANCE hInstMail = ::LoadLibraryA("MAPI32.DLL" );

if(hInstMail == NULL ) {
    AfxMessageBox(AFX_IDP_FAILED_MAPI_LOAD );
    goto    EXIT;
}

ULONG (PASCAL *lpfnSendMail )(ULONG, ULONG, MapiMessage*, FLAGS, ULONG );
(FARPROC&)lpfnSendMail = GetProcAddress(hInstMail, "MAPISendMail" );
if(lpfnSendMail == NULL ) {
    AfxMessageBox(AFX_IDP_INVALID_MAPI_DLL );
    ::FreeLibrary(hInstMail );
    goto    EXIT;
}

////////////////////////////////////////////////////////////////////////////
// 添付ファイル設定

// prepare the file description (for the attachment)
MapiFileDesc    fileDesc[6];

int iAttach = 0;

// 送信命令ファイル
ZeroMemory(&fileDesc[iAttach], sizeof(MapiFileDesc) );
fileDesc[iAttach].nPosition = iAttach;
fileDesc[iAttach].lpszPathName = szTrans;
fileDesc[iAttach].lpszFileName = GetFileNamePtr(szTrans );
iAttach++;
```

```
// 送信原稿
if(m_szDocName1[0] ) {
    ZeroMemory(&fileDesc[iAttach], sizeof(MapiFileDesc) );
    fileDesc[iAttach].nPosition = iAttach;
    fileDesc[iAttach].lpszPathName = m_szDocName1;
    fileDesc[iAttach].lpszFileName = GetFileNamePtr(m_szDocName1 );
    iAttach++;
}
if(m_szDocName2[0] ) {
    ZeroMemory(&fileDesc[iAttach], sizeof(MapiFileDesc) );
    fileDesc[iAttach].nPosition = iAttach;
    fileDesc[iAttach].lpszPathName = m_szDocName2;
    fileDesc[iAttach].lpszFileName = GetFileNamePtr(m_szDocName2 );
    iAttach++;
}
if(m_szDocName3[0] ) {
    ZeroMemory(&fileDesc[iAttach], sizeof(MapiFileDesc) );
    fileDesc[iAttach].nPosition = iAttach;
    fileDesc[iAttach].lpszPathName = m_szDocName3;
    fileDesc[iAttach].lpszFileName = GetFileNamePtr(m_szDocName3 );
    iAttach++;
}
if(m_szDocName4[0] ) {
    ZeroMemory(&fileDesc[iAttach], sizeof(MapiFileDesc) );
    fileDesc[iAttach].nPosition = iAttach;
    fileDesc[iAttach].lpszPathName = m_szDocName4;
    fileDesc[iAttach].lpszFileName = GetFileNamePtr(m_szDocName4 );
    iAttach++;
}

// 送付状
if(MailToFaxInfo.pCoverName ) {
    ZeroMemory(&fileDesc[iAttach], sizeof(MapiFileDesc) );
    fileDesc[iAttach].nPosition = iAttach;
    fileDesc[iAttach].lpszPathName = MailToFaxInfo.pCoverName;
    fileDesc[iAttach].lpszFileName = GetFileNamePtr(MailToFaxInfo.pCoverName );
    iAttach++;
}

////////////////////////////////////////////////////////////////////////////
// その他メール設定

MapiMessage message;

ZeroMemory(&message, sizeof(message) );
```

```
message.nFileCount = iAttach;
message.lpFiles = fileDesc;
message.lpszNoteText = "Mail to FAX";
message.lpszSubject = "Mail to FAX";

//////////////////////////////////////////////////////////////////////////
// メール送信

AfxGetApp()->EnableModeless(FALSE );
HWND hWndTop;
CWnd* pParentWnd = CWnd::GetSafeOwner(NULL, &hWndTop );

pParentWnd->SetCapture();
::SetFocus(NULL );

pParentWnd->m_nFlags |= WF_STAYDISABLED;

int nError = lpfnSendMail(
    0,
    (ULONG)pParentWnd->GetSafeHwnd(),
    &message,
    MAPI_LOGON_UI | MAPI_DIALOG,
    0 );

Sleep(500 );

::ReleaseCapture();
pParentWnd->m_nFlags &= ~WF_STAYDISABLED;

pParentWnd->EnableWindow(TRUE );
::SetActiveWindow(NULL );
pParentWnd->SetActiveWindow();
pParentWnd->SetFocus();

if(hWndTop != NULL ) {
    ::EnableWindow(hWndTop, TRUE );
}
AfxGetApp()->EnableModeless(TRUE );

if(nError != SUCCESS_SUCCESS &&
    nError != MAPI_USER_ABORT &&
    nError != MAPI_E_LOGIN_FAILURE ) {

    AfxMessageBox(AFX_IDP_FAILED_MAPI_SEND );
}
```

```
        ////////////////////////////////////////////////////////////////////////////
        // MAPI 後始末

        ::FreeLibrary(hInstMail );
    }

EXIT:

    ～

}
```

5.3.2 印刷結果のＦＡＸ送信

印刷結果のＦＡＸ送信プログラム【PrtCli.exe】 は、STARFAX Server SDK 本体をセットアップしていないコンピュータで 印刷結果の表示と FAX 送信を行うサンプルプログラムです。
本 CD-ROM の以下の位置に入っています。ワード・エクセル等のアプリケーションから手動で印刷後、プリンタドライバからユーザープログラムが起動されます。

¥サンプル¥応用サンプル¥クライアント¥メール de ＦＡＸ¥VC6¥PrtCli.exe
...印刷結果のＦＡＸ送信プログラム
¥サンプル¥応用サンプル¥クライアント¥メール de ＦＡＸ¥VC6¥PrtCli¥
...印刷結果のＦＡＸプログラム 開発プロジェクト

なお、このサンプルプログラムには、元となるサンプルプログラムがあり、FAX 送信に関する部分を[メール de ＦＡＸ]の仕組みに変更して、クライアント動作するようにしています。[メール de ＦＡＸ] 以外の処理については、「STARFAX Server SDK VC 開発向けプリンタドライバとビューア」の *2.1 印刷結果の FAX 送信* をご覧ください。

主な仕様、および操作方法は以下の通りです。
（以降、STARFAX Server SDK 本体をセットアップしているコンピュータを [サーバー側]、STARFAX Server SDK 本体をセットアップしていないコンピュータを [クライアント側] とします）

① [サーバー側]で [メール de ＦＡＸ] の設定を有効にします。

(1) タスクトレイの STARFAX Server SDK のアイコンを右クリックして表示されるメニューから [メール de ＦＡＸ] を実行します。

(2) 最低限、以下の項目を設定します。

- [メール de ＦＡＸ を有効にする(Y)] をチェックします。
- [メールサーバーの設定(S)] を行います。

(3) 必要に応じて、以下の項目を設定します。

- [件名に含まれる文字で識別(I)]
- [メール受信間隔(N)]
- [対象外のメールを EML ファイルに保存(P)]、及び、関連項目
- [FAX 送信結果をメール通知する(N)] 、及び、関連項目

② [サーバー側]で STARFAX Server SDK を起動します。
(STARFAX Server SDK の起動は、「STARFAX Server SDK セットアップアニュアル」P21 参照)

③ [クライアント側]で、以下の STARFAX Server SDK プリンタドライバの動作に関するレジストリを指定します。

- HKEY_LOCAL_MACHINE¥Software¥MEGASOFT¥SfCs¥OutFolder … ファイル出力フォルダ
  文字列項目で、任意の作業フォルダを作成して指定します。
  (例: "C:¥Program Files¥SfCs¥Temp")
- HKEY_LOCAL_MACHINE¥Software¥MEGASOFT¥SfCs¥DocName …ドキュメント名
  文字列項目で、 このサンプルプログラムの場合、任意の文字列を指定します。
  (例: "SFCSPRN")
- HKEY_LOCAL_MACHINE¥Software¥MEGASOFT¥SfCs¥ExecFlag …プログラム実行フラグ
  DWORD 項目で、 1 を指定します。
- HKEY_LOCAL_MACHINE¥Software¥MEGASOFT¥SfCs¥ExecPath …プログラムパス
  文字列項目で、 PrtCli.exe をフルパスで指定します。
  (例: "C:¥Program Files¥SfCs¥PrtCli.exe")
- HKEY_LOCAL_MACHINE¥Software¥MEGASOFT¥SfCs¥ExecParam …追加パラメータ
  文字列項目で、 何も指定していない状態("") を設定します。

④印刷可能な適当なアプリケーション(ワード等)から プリンタ名 "MEGASOFT STARFAX Server SDK" に対して印刷を行うと 印刷結果のＦＡＸ送信プログラム【PrtCli.exe】 が起動して、印刷結果リスト に 印刷結果が 登録された状態になります。

(または、印刷結果のＦＡＸ送信プログラム【PrtCli.exe】 を起動して、[テスト印刷(T)]ボタンでテスト印刷 を行っても 同じ状態になります)

⑤ [表示(V)]ボタンを押して、印刷結果の内容を確認します。

⑥ [メール送信(S)]ボタンを押して、メール送信を行います。
この後、FAX 送信が正常に動作していない場合は、[サーバー側] の以下の表示をご確認下さい。

- STARFAX ログ管理プログラム の [イベント]
  (主に、[コントロール] と [メール] をご確認下さい)
- [メール de ＦＡＸ 設定] の システムメニュー の [イベントログ]

ApiSend.cpp ：

```
//*********************************************************************
// 関数名　　：MakeTransFileForMailToFax
// 機能　　：[ メール de ＦＡＸ ]の送信命令ファイル作成
// 呼び出し　：MakeTransFileForMailToFax(LPCTSTR pTransFile, SENDFAX_MISSION *pInfo, int *piError )
// 入力　　：pTransFile = 送信命令ファイル
//　　：pInfo　　= 送信命令ファイル作成情報ポインタ
//　　：piError　= エラー時、エラー取得用ポインタ
// 出力　　：返値　TRUE　：正常終了
//　　：　　FALSE：失敗
//　　：　　　　*piError = エラーコード
//　　：
//　　：　【エラーコード】
//　　：　　SENDFAX_ERR_GetTempFolder　... 一時フォルダの取得に失敗しました
//　　：　　SENDFAX_ERR_GetTempFile　　... 一時ファイルの取得に失敗しました
//　　：　　SENDFAX_ERR_CreateTransFile ... 送信命令ファイルの作成に失敗しました
//　　：　　SENDFAX_ERR_PARAM_INFO　　... 関数引数エラー: 送信命令ファイル作成情報が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_SENDNUM　... 関数引数エラー: 相手先数の指定が 0 です。
//　　：　　SENDFAX_ERR_PARAM_SENDINFO　... 関数引数エラー: 相手先情報が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_FAX　　... 関数引数エラー: FAX 番号が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_TRANSFILE ... 関数引数エラー: 送信命令ファイルが指定されていません。
//　　：
// 特記事項　　：　[ メール de ＦＡＸ ]の送信命令ファイルを作成します。
// 作成者　　：
// 作成日　　：2010.07.20
//*********************************************************************
BOOL APIENTRY　MakeTransFileForMailToFax(LPCTSTR pTransFile, MAIL2FAX_MISSION *pInfo, int *piError )
{
    BOOL　bRet = FALSE;　// 返値 ... 初期値は失敗

    int iErrorDummy;　// NULL 用ダミーエラーコード設定

    int i;

    char　szGetName[MAX_PATH+1];　　// ファイル名取得用

    SENDFAX_SENDINFO　*pSendInfo; // 相手先情報用

    char　szSection[32];　// セクション設定用
    char　szWork[512];　// 作業用

    if(!piError ) {
        // エラーコード取得用ポインタにダミー設定
        piError = &iErrorDummy;
```

```
}
*piError = SENDFAX_SUCCESS;      // エラーコードに初期値設定 ... 正常終了

////////////////////////////////////////////////////////////////////////////
// (1)  簡単な引数の内容チェック
//

    // 送信命令ファイルチェック
if(!pTransFile || !(*pTransFile) ) {
    *piError = SENDFAX_ERR_PARAM_TRANSFILE;
    goto    EXIT;
}

    // 送信命令ファイル作成情報チェック
if(!pInfo ) {
    *piError = SENDFAX_ERR_PARAM_INFO;
    goto    EXIT;
}
    // 相手先数チェック
if(!pInfo->iSendNum ) {
    *piError = SENDFAX_ERR_PARAM_SENDNUM;
    goto    EXIT;
}
    // 相手先情報チェック
for(i = 0 ;  i < pInfo->iSendNum ; i++ ) {
    pSendInfo = *(pInfo->ppSend + i );
    if(!pSendInfo ) {
        *piError = SENDFAX_ERR_PARAM_SENDINFO;
        goto    EXIT;
    }
        // FAX 番号チェック
    if(!pSendInfo->szFax[0] ) {
        *piError = SENDFAX_ERR_PARAM_FAX;
        goto    EXIT;
    }
}

////////////////////////////////////////////////////////////////////////////
// (2)  送信命令ファイルへ書き込み
//

DeleteFile(pTransFile );   // 念のため削除

// セクション名: [SendInfo]  ... 送信のための相手先情報
```

```
// ・ Num        ... 送信相手先数
wsprintf(szWork, "%d", pInfo->iSendNum );
WritePrivateProfileString("SendInfo", "Num", szWork, pTransFile );

// セクション名: [Send%d]    ... 送信のための相手先内容 (1～)
for(i = 0 ;  i < pInfo->iSendNum ; i++ ) {
    pSendInfo = *(pInfo->ppSend + i );

    wsprintf(szSection, "Send%d", i + 1 );

    // ・ Fax        ... FAX番号 (最大128バイト)
    WritePrivateProfileString(szSection, "Fax", pSendInfo->szFax, pTransFile );
    // ・ Company    ... 会社名 (最大128バイト)
    if(pSendInfo->szCompany[0] ) {
        wsprintf(szWork, "¥"%s¥"", pSendInfo->szCompany );
        WritePrivateProfileString(szSection, "Company", szWork, pTransFile );
    }
    // ・ Division   ... 所属名 (最大128バイト)
    if(pSendInfo->szDivision[0] ) {
        wsprintf(szWork, "¥"%s¥"", pSendInfo->szDivision );
        WritePrivateProfileString(szSection, "Division", szWork, pTransFile );
    }
    // ・ Position   ... 役職名 (最大128バイト)
    if(pSendInfo->szPosition[0] ) {
        wsprintf(szWork, "¥"%s¥"", pSendInfo->szPosition );
        WritePrivateProfileString(szSection, "Position", szWork, pTransFile );
    }
    // ・ Name       ... 氏名 (最大128バイト)
    if(pSendInfo->szName[0] ) {
        wsprintf(szWork, "¥"%s¥"", pSendInfo->szName );
        WritePrivateProfileString(szSection, "Name", szWork, pTransFile );
    }
    // ・ Title      ... 敬称 (最大128バイト)
    if(pSendInfo->szTitle[0] ) {
        wsprintf(szWork, "¥"%s¥"", pSendInfo->szTitle );
        WritePrivateProfileString(szSection, "Title", szWork, pTransFile );
    }
    // ・ Telephone  ... 電話番号 (最大128バイト)
    if(pSendInfo->szTelephone[0] ) {
        WritePrivateProfileString(szSection, "Telephone", pSendInfo->szTelephone, pTransFile );
    }
    // ・ ZipCode    ... 郵便番号 (最大128バイト)
    if(pSendInfo->szZipCode[0] ) {
        WritePrivateProfileString(szSection, "ZipCode", pSendInfo->szZipCode, pTransFile );
    }
```

```
// ・ Address1   ... 住所1 (最大128バイト)
if(pSendInfo->szAddress1[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szAddress1 );
    WritePrivateProfileString(szSection, "Address1", szWork, pTransFile );
}
// ・ Address2   ... 住所2 (最大128バイト)
if(pSendInfo->szAddress2[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szAddress2 );
    WritePrivateProfileString(szSection, "Address2", szWork, pTransFile );
}
// ・ FCode      ... Ｆコード番号 (最大20バイト)
if(pSendInfo->szFCode[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szFCode );
    WritePrivateProfileString(szSection, "FCode", szWork, pTransFile );
}
// ・ FreeArea   ... ユーザが自由に利用できるエリア (最大128バイト)
if(pSendInfo->szFreeArea[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szFreeArea );
    WritePrivateProfileString(szSection, "FreeArea", szWork, pTransFile );
}
// ・ Speed      ... 通信速度 ("0":自動, "1":高速, "2":中速, "3":低速)
if(pSendInfo->iSpeed ) {
    wsprintf(szWork, "%d", pSendInfo->iSpeed );
    WritePrivateProfileString(szSection, "Speed", szWork, pTransFile );
}
// ・ Comp       ... 圧縮方式 ("0":自動, "1":MH, "2":MR, "3":MMR)
if(pSendInfo->iComp ) {
    wsprintf(szWork, "%d", pSendInfo->iComp );
    WritePrivateProfileString(szSection, "Comp", szWork, pTransFile );
}
// ・ Ecm        ... エラー訂正 ("0":自動, "1":利用する, "2":利用しない)
if(pSendInfo->iEcm ) {
    wsprintf(szWork, "%d", pSendInfo->iEcm );
    WritePrivateProfileString(szSection, "Ecm", szWork, pTransFile );
}
// ・ Line       ... 送信回線指定 (0～)
if(pSendInfo->bLine ) {
    wsprintf(szWork, "%d", pSendInfo->iLine );
    WritePrivateProfileString(szSection, "Line", szWork, pTransFile );
}
// ・ Priority   ... 優先順位 (0～15)
if(pSendInfo->bPriority ) {
    wsprintf(szWork, "%d", pSendInfo->iPriority );
    WritePrivateProfileString(szSection, "Priority", szWork, pTransFile );
}
```

```
    // ・ Time        ... 送信時刻 ("YYYYMMDDHHMMSS")
    if(pSendInfo->bTime ) {
        wsprintf(szWork, "%s", pSendInfo->szTime );
        WritePrivateProfileString(szSection, "Time", szWork, pTransFile );
    }
}

// セクション名: [Cover]     ... 送付状
// ・ Name        ... 送付状ファイルパス(.txt)
if(pInfo->pCoverName ) {

    SUB_GetFileName(pInfo->pCoverName, szGetName );

    wsprintf(szWork, "¥"%s¥"", szGetName );
    WritePrivateProfileString("Cover", "Name", szWork, pTransFile );
}
// ・ FontName   ... 送付状フォント名
if(pInfo->pCoverFontName ) {
    wsprintf(szWork, "¥"%s¥"", pInfo->pCoverFontName );
    WritePrivateProfileString("Cover", "FontName", szWork, pTransFile );
}
// ・ FontSize   ... 送付状フォントサイズ (8 ～ 72 (ポイント ) )
if(pInfo->iCoverFontSize ) {
    wsprintf(szWork, "%d", pInfo->iCoverFontSize );
    WritePrivateProfileString("Cover", "FontSize", szWork, pTransFile );
}

// セクション名: [User]       ... 発信元情報
// ・ UserInfo   ... 発信元情報記録 ("記録位置,記録情報") (最大 140 バイト)
//                    ("0":記録しない, "1":原稿の内側に記録, "2":原稿の外側に記録)
if(pInfo->szUserInfo[0] ) {
    wsprintf(szWork, "¥"%s¥"", pInfo->szUserInfo );
    WritePrivateProfileString("User", "UserInfo", szWork, pTransFile );
}
// ・ UserID     ... 自局電話番号 (FAXID として、相手ファクシミリに通知されます) (最大 20 バイト)
if(pInfo->szUserID[0] ) {
    wsprintf(szWork, "¥"%s¥"", pInfo->szUserID );
    WritePrivateProfileString("User", "UserID", szWork, pTransFile );
}

////////////////////////////////////////////////////////////////////////
// (3) 送信命令ファイルのフラッシュ 【重要】
//
//      OS により、WritePrivateProfileString() 【Win32API】 を使用しての
//    実際のファイルへの書き出しのタイミングが異なる為、必ずこのフラッシュの
```

```
    //    ステップが必要です。
    //

    WritePrivateProfileString(NULL, NULL, NULL, pTransFile );

    bRet = TRUE;    // 正常終了

EXIT:

    // エラー時は、送信命令ファイルを削除
    if(!bRet ) {
        DeleteFile(pTransFile );
    }

    return bRet;
}
```

PrtCliDlg.cpp ：

```
////////////////////////////////////////////////////////////////////////////
// メール送信ボタン

void CPrtCliDlg::OnOK()
{

    ～

    //――――――――――――――――――――――――――――――――――――――――
    // (2) [ メール de ＦＡＸ ]の送信命令ファイル作成関数 呼び出し

    if(!(bRet = MakeTransFileForMailToFax(szTrans, &MailToFaxInfo, &iError ) ) ) {
        switch(iError ) {
        case    SENDFAX_ERR_GetTempFolder:
            MessageBox("一時フォルダの取得に失敗しました", CAP_APR_PRTCLI, MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_GetTempFile:
            MessageBox("一時ファイルの取得に失敗しました", CAP_APR_PRTCLI, MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_CreateTransFile:
            MessageBox("送信命令ファイルの作成に失敗しました", CAP_APR_PRTCLI, MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_PARAM_INFO:
            MessageBox("関数引数エラー: 送信命令ファイル作成情報が指定されていません。",
                        CAP_APR_PRTCLI, MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_PARAM_SENDNUM:
            MessageBox(" 関数引数エラー: 相手先数の指定が 0 です。", CAP_APR_PRTCLI, MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_PARAM_SENDINFO:
            MessageBox("関数引数エラー: 相手先情報が指定されていません。", CAP_APR_PRTCLI, MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_PARAM_FAX:
            MessageBox("関数引数エラー: FAX 番号が指定されていません。", CAP_APR_PRTCLI, MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_PARAM_TRANSFILE:
            MessageBox("関数引数エラー: 送信命令ファイルが指定されていません。", CAP_APR_PRTCLI,
                        MB_ICONEXCLAMATION | MB_OK );
            break;
        default:
            break;
        }
```

```
}
else {
```

MessageBox("このサンプルプログラムは MAPI を利用してメール送信を行います。¥n 宛先 に STARFAX Engine の [メール de ＦＡＸ] で設定した¥nPOP メールアドレス を指定して送信してください。¥n¥n なお、送信後、OUTLOOK 等メーラーを起動して送信を行わないと¥n 実際に送信されないことがあります。", CAP_APR_PRTCLI, MB_ICONINFORMATION | MB_OK);

```
////////////////////////////////////////////////////////////////////////////
//
// メール送信
//
////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
// MAPI 初期化

HINSTANCE hInstMail = ::LoadLibraryA("MAPI32.DLL" );

if(hInstMail == NULL ) {
    AfxMessageBox(AFX_IDP_FAILED_MAPI_LOAD );
    goto    EXIT;
}

ULONG (PASCAL *lpfnSendMail )(ULONG, ULONG, MapiMessage*, FLAGS, ULONG );
(FARPROC&)lpfnSendMail = GetProcAddress(hInstMail, "MAPISendMail" );
if(lpfnSendMail == NULL ) {
    AfxMessageBox(AFX_IDP_INVALID_MAPI_DLL );
    ::FreeLibrary(hInstMail );
    goto    EXIT;
}

////////////////////////////////////////////////////////////////////////////
// 添付ファイル設定

MapiFileDesc    *pFileDesc;

pFileDesc = new MapiFileDesc[m_clPRT.GetSelectedCount() + 1];

int iAttach = 0;

// 送信命令ファイル
ZeroMemory(&pFileDesc[iAttach], sizeof(MapiFileDesc) );
pFileDesc[iAttach].nPosition = iAttach;
pFileDesc[iAttach].lpszPathName = szTrans;
```

```
pFileDesc[iAttach].lpszFileName = pSUB_GetFileNamePtr(szTrans );
iAttach++;

POSITION pos = m_clPRT.GetFirstSelectedItemPosition();
while(pos ) {
    int nItem = m_clPRT.GetNextSelectedItem(pos );
    PRES_DATINFO    *pItem = (PRES_DATINFO *)(m_clPRT.GetItemData(nItem )) ;
    if(pItem ) {
        ZeroMemory(&pFileDesc[iAttach], sizeof(MapiFileDesc) );
        pFileDesc[iAttach].nPosition = iAttach;
        pFileDesc[iAttach].lpszPathName = pItem->pTiffFile;
        pFileDesc[iAttach].lpszFileName = pSUB_GetFileNamePtr(pItem->pTiffFile );

        iAttach++;
    }
}

////////////////////////////////////////////////////////////////////////////
// その他メール設定

MapiMessage message;

ZeroMemory(&message, sizeof(message) );
message.nFileCount = iAttach;
message.lpFiles = pFileDesc;
message.lpszNoteText = "Mail to FAX";
message.lpszSubject = "Mail to FAX";

////////////////////////////////////////////////////////////////////////////
// メール送信

AfxGetApp()->EnableModeless(FALSE );
HWND hWndTop;
CWnd* pParentWnd = CWnd::GetSafeOwner(NULL, &hWndTop );

pParentWnd->SetCapture();
::SetFocus(NULL );

pParentWnd->m_nFlags |= WF_STAYDISABLED;

int nError = lpfnSendMail(
    0,
    (ULONG)pParentWnd->GetSafeHwnd(),
    &message,
    MAPI_LOGON_UI | MAPI_DIALOG,
```

```
            0 );

        Sleep(500 );

        ::ReleaseCapture();
        pParentWnd->m_nFlags &= ~WF_STAYDISABLED;

        pParentWnd->EnableWindow(TRUE );
        ::SetActiveWindow(NULL );
        pParentWnd->SetActiveWindow();
        pParentWnd->SetFocus();

        if(hWndTop != NULL ) {
            ::EnableWindow(hWndTop, TRUE );
        }
        AfxGetApp()->EnableModeless(TRUE );

        if(nError != SUCCESS_SUCCESS &&
            nError != MAPI_USER_ABORT &&
            nError != MAPI_E_LOGIN_FAILURE ) {

            AfxMessageBox(AFX_IDP_FAILED_MAPI_SEND );
        }

        delete pFileDesc;

        ////////////////////////////////////////////////////////////////////////////
        // MAPI後始末

        ::FreeLibrary(hInstMail );
    }

EXIT:

    ～

}
```

5.3.3 TIFF ファイルの作成とＦＡＸ送信

TIFF ファイルのＦＡＸ送信プログラム【PrtCli2.exe】 は、STARFAX Server SDK 本体をセットアップしていないコンピュータで TIFF ファイルの作成 と FAX 送信を行うサンプルプログラムです。本 CD-ROM の以下の位置に入っています。ユーザープログラムがプリンタドライバを制御して印刷結果(TIFF ファイル)を取得します。

¥サンプル¥応用サンプル¥クライアント¥メール de ＦＡＸ¥VC6¥PrtCli2.exe
... TIFF ファイルのＦＡＸ送信プログラム
¥サンプル¥応用サンプル¥クライアント¥メール de ＦＡＸ¥VC6¥PrtCli2¥
... TIFF ファイルのＦＡＸ送信プログラム 開発プロジェクト

なお、このサンプルプログラムには、元となるサンプルプログラムがあり、FAX 送信に関する部分を[メール de ＦＡＸ]の仕組みに変更して、クライアント動作するようにしています。[メール to ＦＸ] 以外の処理については、「STARFAX Server SDK VC 開発向けプリンタドライバとビューア」の *2.2 TIFF ファイルの作成と FAX 送信* をご覧ください。

主な仕様、および操作方法は以下の通りです。

(以降、STARFAX Server SDK 本体をセットアップしているコンピュータを [サーバー側]、STARFAX Server SDK 本体をセットアップしていないコンピュータを [クライアント側] とします)

① [サーバー側]で [メール de ＦＡＸ] の設定を有効にします。

(1) タスクトレイの STARFAX Server SDK のアイコンを右クリックして表示されるメニューから [メール de ＦＡＸ] を実行します。

(2) 最低限、以下の項目を設定します。

- [メール de ＦＡＸ を有効にする(Y)] をチェックします。
- [メールサーバーの設定(S)] を行います。

(3) 必要に応じて、以下の項目を設定します。

- [件名に含まれる文字で識別(I)]
- [メール受信間隔(N)]
- [対象外のメールを EML ファイルに保存(P)]、及び、関連項目
- [FAX 送信結果をメール通知する(N)] 、及び、関連項目

② [サーバー側]で STARFAX Server SDK を起動します。
(STARFAX Server SDK の起動は、「STARFAX Server SDK セットアップアニュアル」P21 参照)

③ [クライアント側]で PrtCli2.exe を起動します。

④ 「操作１」の下欄に FAX 原稿に表示させる文字を入力します。

⑤ 「①FAX 原稿の作成」ボタンをクリックします。
(作成するファイル名を任意指定したい場合は、[指定]ラジオボタンをクリックして、ファイル名を入力してください)

⑥ 「②FAX 原稿の表示」ボタンをクリックすると、作成された FAX 原稿が STARFAX Server SDK ビューアで表示されます。

⑦ 「操作３」の下欄に送信先の FAX 番号を入力します。

⑧ 「メール送信」ボタンをクリックします。
この後､FAX 送信が正常に動作していない場合は､[サーバー側] の以下の表示をご確認下さい。

- STARFAX ログ管理プログラム の [イベント]
  (主に、[コントロール] と [メール] をご確認下さい)
- [メール de ＦＡＸ 設定] の システムメニュー の [イベントログ]

ApiSend.cpp ：

```
//*********************************************************************
// 関数名　　：MakeTransFileForMailToFax
// 機能　　：[ メール de ＦＡＸ ]の送信命令ファイル作成
// 呼び出し　：MakeTransFileForMailToFax(LPCTSTR pTransFile, SENDFAX_MISSION *pInfo, int *piError )
// 入力　　：pTransFile = 送信命令ファイル
//　　：pInfo　= 送信命令ファイル作成情報ポインタ
//　　：piError　= エラー時、エラー取得用ポインタ
// 出力　　：返値　TRUE　：正常終了
//　　：　FALSE：失敗
//　　：　　*piError = エラーコード
//　　：
//　　：　【エラーコード】
//　　：　　SENDFAX_ERR_GetTempFolder　... 一時フォルダの取得に失敗しました
//　　：　　SENDFAX_ERR_GetTempFile　... 一時ファイルの取得に失敗しました
//　　：　　SENDFAX_ERR_CreateTransFile　... 送信命令ファイルの作成に失敗しました
//　　：　　SENDFAX_ERR_PARAM_INFO　... 関数引数エラー: 送信命令ファイル作成情報が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_SENDNUM　... 関数引数エラー: 相手先数の指定が 0 です。
//　　：　　SENDFAX_ERR_PARAM_SENDINFO　... 関数引数エラー: 相手先情報が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_FAX　... 関数引数エラー: FAX 番号が指定されていません。
//　　：　　SENDFAX_ERR_PARAM_TRANSFILE　... 関数引数エラー: 送信命令ファイルが指定されていません。
//　　：
// 特記事項　　：　[ メール de ＦＡＸ ]の送信命令ファイルを作成します。
// 作成者　　：
// 作成日　　：2010.07.20
//*********************************************************************
BOOL APIENTRY　MakeTransFileForMailToFax(LPCTSTR pTransFile, MAIL2FAX_MISSION *pInfo, int *piError )
{
    BOOL　bRet = FALSE;　// 返値 ... 初期値は失敗

    int iErrorDummy;　// NULL 用ダミーエラーコード設定

    int i;

    char　szGetName[MAX_PATH+1];　　// ファイル名取得用

    SENDFAX_SENDINFO　*pSendInfo; // 相手先情報用

    char　szSection[32];　// セクション設定用
    char　szWork[512];　// 作業用

    if(!piError ) {
        // エラーコード取得用ポインタにダミー設定
        piError = &iErrorDummy;
```

```
}
*piError = SENDFAX_SUCCESS;      // エラーコードに初期値設定 ... 正常終了

////////////////////////////////////////////////////////////////////////////
// (1)  簡単な引数の内容チェック
//

    // 送信命令ファイルチェック
if(!pTransFile || !(*pTransFile) ) {
    *piError = SENDFAX_ERR_PARAM_TRANSFILE;
    goto    EXIT;
}

    // 送信命令ファイル作成情報チェック
if(!pInfo ) {
    *piError = SENDFAX_ERR_PARAM_INFO;
    goto    EXIT;
}
    // 相手先数チェック
if(!pInfo->iSendNum ) {
    *piError = SENDFAX_ERR_PARAM_SENDNUM;
    goto    EXIT;
}
    // 相手先情報チェック
for(i = 0 ;  i < pInfo->iSendNum ; i++ ) {
    pSendInfo = *(pInfo->ppSend + i );
    if(!pSendInfo ) {
        *piError = SENDFAX_ERR_PARAM_SENDINFO;
        goto    EXIT;
    }
        // FAX 番号チェック
    if(!pSendInfo->szFax[0] ) {
        *piError = SENDFAX_ERR_PARAM_FAX;
        goto    EXIT;
    }
}

////////////////////////////////////////////////////////////////////////////
// (2)  送信命令ファイルへ書き込み
//

DeleteFile(pTransFile );   // 念のため削除

// セクション名: [SendInfo]  ... 送信のための相手先情報
```

```
// ・ Num        ... 送信相手先数
wsprintf(szWork, "%d", pInfo->iSendNum );
WritePrivateProfileString("SendInfo", "Num", szWork, pTransFile );

// セクション名: [Send%d]    ... 送信のための相手先内容 (1～)
for(i = 0 ;  i < pInfo->iSendNum ; i++ ) {
    pSendInfo = *(pInfo->ppSend + i );

    wsprintf(szSection, "Send%d", i + 1 );

    // ・ Fax        ... FAX番号 (最大128バイト)
    WritePrivateProfileString(szSection, "Fax", pSendInfo->szFax, pTransFile );
    // ・ Company    ... 会社名 (最大128バイト)
    if(pSendInfo->szCompany[0] ) {
        wsprintf(szWork, "¥"%s¥"", pSendInfo->szCompany );
        WritePrivateProfileString(szSection, "Company", szWork, pTransFile );
    }
    // ・ Division   ... 所属名 (最大128バイト)
    if(pSendInfo->szDivision[0] ) {
        wsprintf(szWork, "¥"%s¥"", pSendInfo->szDivision );
        WritePrivateProfileString(szSection, "Division", szWork, pTransFile );
    }
    // ・ Position   ... 役職名 (最大128バイト)
    if(pSendInfo->szPosition[0] ) {
        wsprintf(szWork, "¥"%s¥"", pSendInfo->szPosition );
        WritePrivateProfileString(szSection, "Position", szWork, pTransFile );
    }
    // ・ Name       ... 氏名 (最大128バイト)
    if(pSendInfo->szName[0] ) {
        wsprintf(szWork, "¥"%s¥"", pSendInfo->szName );
        WritePrivateProfileString(szSection, "Name", szWork, pTransFile );
    }
    // ・ Title      ... 敬称 (最大128バイト)
    if(pSendInfo->szTitle[0] ) {
        wsprintf(szWork, "¥"%s¥"", pSendInfo->szTitle );
        WritePrivateProfileString(szSection, "Title", szWork, pTransFile );
    }
    // ・ Telephone  ... 電話番号 (最大128バイト)
    if(pSendInfo->szTelephone[0] ) {
        WritePrivateProfileString(szSection, "Telephone", pSendInfo->szTelephone, pTransFile );
    }
    // ・ ZipCode    ... 郵便番号 (最大128バイト)
    if(pSendInfo->szZipCode[0] ) {
        WritePrivateProfileString(szSection, "ZipCode", pSendInfo->szZipCode, pTransFile );
    }
```

```
// ・ Address1   ... 住所1 (最大128バイト)
if(pSendInfo->szAddress1[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szAddress1 );
    WritePrivateProfileString(szSection, "Address1", szWork, pTransFile );
}
// ・ Address2   ... 住所2 (最大128バイト)
if(pSendInfo->szAddress2[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szAddress2 );
    WritePrivateProfileString(szSection, "Address2", szWork, pTransFile );
}
// ・ FCode      ... Fコード番号 (最大20バイト)
if(pSendInfo->szFCode[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szFCode );
    WritePrivateProfileString(szSection, "FCode", szWork, pTransFile );
}
// ・ FreeArea   ... ユーザが自由に利用できるエリア (最大128バイト)
if(pSendInfo->szFreeArea[0] ) {
    wsprintf(szWork, "¥"%s¥"", pSendInfo->szFreeArea );
    WritePrivateProfileString(szSection, "FreeArea", szWork, pTransFile );
}
// ・ Speed      ... 通信速度 ("0":自動, "1":高速, "2":中速, "3":低速)
if(pSendInfo->iSpeed ) {
    wsprintf(szWork, "%d", pSendInfo->iSpeed );
    WritePrivateProfileString(szSection, "Speed", szWork, pTransFile );
}
// ・ Comp       ... 圧縮方式 ("0":自動, "1":MH, "2":MR, "3":MMR)
if(pSendInfo->iComp ) {
    wsprintf(szWork, "%d", pSendInfo->iComp );
    WritePrivateProfileString(szSection, "Comp", szWork, pTransFile );
}
// ・ Ecm        ... エラー訂正 ("0":自動, "1":利用する, "2":利用しない)
if(pSendInfo->iEcm ) {
    wsprintf(szWork, "%d", pSendInfo->iEcm );
    WritePrivateProfileString(szSection, "Ecm", szWork, pTransFile );
}
// ・ Line       ... 送信回線指定 (0～)
if(pSendInfo->bLine ) {
    wsprintf(szWork, "%d", pSendInfo->iLine );
    WritePrivateProfileString(szSection, "Line", szWork, pTransFile );
}
// ・ Priority   ... 優先順位 (0～15)
if(pSendInfo->bPriority ) {
    wsprintf(szWork, "%d", pSendInfo->iPriority );
    WritePrivateProfileString(szSection, "Priority", szWork, pTransFile );
}
```

```
    // ・ Time        ... 送信時刻 ("YYYYMMDDHHMMSS")
    if(pSendInfo->bTime ) {
        wsprintf(szWork, "%s", pSendInfo->szTime );
        WritePrivateProfileString(szSection, "Time", szWork, pTransFile );
    }
}

// セクション名: [Cover]     ... 送付状
// ・ Name        ... 送付状ファイルパス(.txt)
if(pInfo->pCoverName ) {

    SUB_GetFileName(pInfo->pCoverName, szGetName );

    wsprintf(szWork, "¥"%s¥"", szGetName );
    WritePrivateProfileString("Cover", "Name", szWork, pTransFile );
}
// ・ FontName   ... 送付状フォント名
if(pInfo->pCoverFontName ) {
    wsprintf(szWork, "¥"%s¥"", pInfo->pCoverFontName );
    WritePrivateProfileString("Cover", "FontName", szWork, pTransFile );
}
// ・ FontSize   ... 送付状フォントサイズ (8 ～ 72 (ポイント ) )
if(pInfo->iCoverFontSize ) {
    wsprintf(szWork, "%d", pInfo->iCoverFontSize );
    WritePrivateProfileString("Cover", "FontSize", szWork, pTransFile );
}

// セクション名: [User]       ... 発信元情報
// ・ UserInfo   ... 発信元情報記録 ("記録位置,記録情報") (最大 140 バイト)
//                    ("0":記録しない, "1":原稿の内側に記録, "2":原稿の外側に記録)
if(pInfo->szUserInfo[0] ) {
    wsprintf(szWork, "¥"%s¥"", pInfo->szUserInfo );
    WritePrivateProfileString("User", "UserInfo", szWork, pTransFile );
}
// ・ UserID     ... 自局電話番号 (FAXID として、相手ファクシミリに通知されます) (最大 20 バイト)
if(pInfo->szUserID[0] ) {
    wsprintf(szWork, "¥"%s¥"", pInfo->szUserID );
    WritePrivateProfileString("User", "UserID", szWork, pTransFile );
}

////////////////////////////////////////////////////////////////////////
// (3) 送信命令ファイルのフラッシュ 【重要】
//
//      OS により、WritePrivateProfileString() 【Win32API】 を使用しての
//    実際のファイルへの書き出しのタイミングが異なる為、必ずこのフラッシュの
```

```
    //    ステップが必要です。
    //

    WritePrivateProfileString(NULL, NULL, NULL, pTransFile );

    bRet = TRUE;    // 正常終了

EXIT:

    // エラー時は、送信命令ファイルを削除
    if(!bRet ) {
        DeleteFile(pTransFile );
    }

    return bRet;
}
```

PrtCli2Dlg.cpp ：

```
////////////////////////////////////////////////////////////////////////////
// [メール送信]ボタン

void CPrtCli2Dlg::OnButtonFax()
{
    ～

    //―――――――――――――――――――――――――――――――――――――――――
    // (2) [ メール de ＦＡＸ ]の送信命令ファイル作成関数 呼び出し

    if(!(bRet = MakeTransFileForMailToFax(szTrans,  &MailToFaxInfo, &iError ) ) ) {
        switch(iError ) {
        case    SENDFAX_ERR_GetTempFolder:
            MessageBox("一時フォルダの取得に失敗しました", CAP_APR_PRTCLI2, MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_GetTempFile:
            MessageBox("一時ファイルの取得に失敗しました", CAP_APR_PRTCLI2, MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_CreateTransFile:
            MessageBox("送信命令ファイルの作成に失敗しました", CAP_APR_PRTCLI2, MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_PARAM_INFO:
            MessageBox("関数引数エラー: 送信命令ファイル作成情報が指定されていません。",
                        CAP_APR_PRTCLI2, MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_PARAM_SENDNUM:
            MessageBox(" 関数引数エラー: 相手先数の指定が 0 です。", CAP_APR_PRTCLI2, MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_PARAM_SENDINFO:
            MessageBox("関数引数エラー: 相手先情報が指定されていません。", CAP_APR_PRTCLI2, MB_ICONEXCLAMATION | MB_OK);
            break;
        case    SENDFAX_ERR_PARAM_FAX:
            MessageBox("関数引数エラー: FAX 番号が指定されていません。", CAP_APR_PRTCLI2, MB_ICONEXCLAMATION | MB_OK );
            break;
        case    SENDFAX_ERR_PARAM_TRANSFILE:
            MessageBox("関数引数エラー: 送信命令ファイルが指定されていません。", CAP_APR_PRTCLI2,
                        MB_ICONEXCLAMATION | MB_OK );
            break;
        default:
            break;
        }
    }
```

```
else {

    MessageBox("このサンプルプログラムは MAPI を利用してメール送信を行います。¥n 宛先 に
STARFAX Engine の [ メール de ＦＡＸ ] で設定した¥nPOP メールアドレス を指定して送信してくだ
さい。¥n¥n なお、送信後、OUTLOOK 等メーラーを起動して送信を行わないと¥n 実際に送信されないこと
があります。", CAP_APR_PRTCLI2, MB_ICONINFORMATION | MB_OK );

    ////////////////////////////////////////////////////////////////////////////
    //
    // メール送信
    //
    ////////////////////////////////////////////////////////////////////////////

    ////////////////////////////////////////////////////////////////////////////
    // MAPI 初期化

    HINSTANCE hInstMail = ::LoadLibraryA("MAPI32.DLL" );

    if(hInstMail == NULL ) {
        AfxMessageBox(AFX_IDP_FAILED_MAPI_LOAD );
        goto    EXIT;
    }

    ULONG (PASCAL *lpfnSendMail )(ULONG, ULONG, MapiMessage*, FLAGS, ULONG );
    (FARPROC&)lpfnSendMail = GetProcAddress(hInstMail, "MAPISendMail" );
    if(lpfnSendMail == NULL ) {
        AfxMessageBox(AFX_IDP_INVALID_MAPI_DLL );
        ::FreeLibrary(hInstMail );
        goto    EXIT;
    }

    ////////////////////////////////////////////////////////////////////////////
    // 添付ファイル設定

    MapiFileDesc    fileDesc[2];

    int iAttach = 0;

    // 送信命令ファイル
    ZeroMemory(&fileDesc[iAttach], sizeof(MapiFileDesc) );
    fileDesc[iAttach].nPosition = iAttach;
    fileDesc[iAttach].lpszPathName = szTrans;
    fileDesc[iAttach].lpszFileName = pSUB_GetFileNamePtr(szTrans );
    iAttach++;
```

```
ZeroMemory(&fileDesc[iAttach], sizeof(MapiFileDesc) );
fileDesc[iAttach].nPosition = iAttach;
fileDesc[iAttach].lpszPathName = m_szTIFFFILE;
fileDesc[iAttach].lpszFileName = pSUB_GetFileNamePtr(m_szTIFFFILE );
iAttach++;

////////////////////////////////////////////////////////////////////////////
// その他メール設定

MapiMessage message;

ZeroMemory(&message, sizeof(message) );
message.nFileCount = iAttach;
message.lpFiles = fileDesc;
message.lpszNoteText = "Mail to FAX";
message.lpszSubject = "Mail to FAX";

////////////////////////////////////////////////////////////////////////////
// メール送信

AfxGetApp()->EnableModeless(FALSE );
HWND hWndTop;
CWnd* pParentWnd = CWnd::GetSafeOwner(NULL, &hWndTop );

pParentWnd->SetCapture();
::SetFocus(NULL );

pParentWnd->m_nFlags |= WF_STAYDISABLED;

int nError = lpfnSendMail(
    0,
    (ULONG)pParentWnd->GetSafeHwnd(),
    &message,
    MAPI_LOGON_UI | MAPI_DIALOG,
    0 );

Sleep(500 );

::ReleaseCapture();
pParentWnd->m_nFlags &= ~WF_STAYDISABLED;

pParentWnd->EnableWindow(TRUE );
::SetActiveWindow(NULL );
pParentWnd->SetActiveWindow();
pParentWnd->SetFocus();
```

```
        if(hWndTop != NULL ) {
            ::EnableWindow(hWndTop, TRUE );
        }
        AfxGetApp()->EnableModeless(TRUE );

        if(nError != SUCCESS_SUCCESS &&
            nError != MAPI_USER_ABORT &&
            nError != MAPI_E_LOGIN_FAILURE ) {

            AfxMessageBox(AFX_IDP_FAILED_MAPI_SEND );
        }

        ////////////////////////////////////////////////////////////////////////
        // MAPI 後始末

        ::FreeLibrary(hInstMail );
    }

EXIT:

    ～

}
```

付録

サンプルプログラムの共通関数

サンプルプログラムの共通関数の説明です。

A　概要

ユーザープログラム において、インストールの確認や動作状況のチェックを行う必要があります。
それらの頻繁に利用されると思われる動作の関数を、サンプルプログラムの以下のソースファイルにまとめてあります。

- SfCsSys.h

  STARFAX Server SDK システム確認関数ヘッダーファイル

- SfCsSys.c または、SfCsSys.cpp　※

  STARFAX Server SDK システム確認関数ファイル

関数の内容は、フォルダの確認をしたりレジストリを参照したりと、技術的には簡単なものです。
そしてヘッダーファイルに、フォルダやレジストリの各種定数を定義しています。

※ SfCsSys.c と SfCsSys.cpp の違いは、.cpp 版 において 標準ヘッダの#include が、MFC を考慮しているだけで、各関数の内容は同じです。

B　関数、定数一覧

■　関数一覧

【システムチェック関数】

《 SFCSSYS_CheckInstall 》　　STARFAX Server SDK インストール確認
《 SFCSSYS_CheckSubFolder 》　　STARFAX Server SDK サブフォルダ確認
《 SFCSSYS_CheckServiceStatus 》STARFAX Server SDK サービス状態取得

【フォルダ取得関数】

《 SFCSSYS_GetSfcsFolder 》　　インストールフォルダパス取得
《 SFCSSYS_GetCtrlFolder 》　　制御関連インターフェイスフォルダパス取得
《 SFCSSYS_GetDataFolder 》　　データフォルダパス取得
《 SFCSSYS_GetSubFolder 》　　各種サブフォルダパス取得

【起動情報ファイル参照関数】

《 SFCSSYS_GetRunInfoString 》　起動情報文字列取得
《 SFCSSYS_GetRunInfoInt 》　　起動情報 DWORD 取得

【その他補助関数】

《 SFCSSYS_GetUniqFileName 》　日付、時刻による固有のファイル名作成

■ 定数一覧

● レジストリ、フォルダ、ファイル関連

プログラム定義	値	内容
SFCS_LMKEY_SFCS	Software¥¥MEGASOFT¥¥SfCs	STARFAX Server SDK レジストリサブキー
SFCS_KEY_INSTALL	SFCS Folder	インストールフォルダ
SFCS_KEY_CTRL	CTRL Folder	制御関連インターフェイスフォルダ
SFCS_KEY_DATA	DATA Folder	データフォルダ
SFCS_FILE_RUN	SfCsRun.inf	起動情報ファイル
SFCS_FILE_SFCSRCV	SfCsRcv.Idx	受信情報インデックスファイル
SFCS_FILE_SFCSSND	SfCsSnd.Idx	送信情報インデックスファイル
SFCS_FILE_SFCSQUE	SfCsQue.Idx	未送信情報インデックスファイル
SFCS_MAX_LINE	32	回線情報 MAX
SFCS_SEC_RUNVERSION	Version	バージョン 情報
SFCS_KEY_RUNPRODUCT	Product	製品種類
SFCS_KEY_RUNMAJOR	Major	メジャーバージョン
SFCS_KEY_RUNMINOR	Minor	マイナーバージョン
SFCS_KEY_RUNREVISION	Revision	バージョン毎修正回数
SFCS_VAL_RUNPRODUCT	SfCs	製品種類 値
SFCS_SEC_RUNSFCS	StarFax	STARFAX Server SDK 情報
SFCS_KEY_RUNRUN	Run	STARFAX Server SDK 起動状況 ("0":起動していない, "1":起動している)
SFCS_KEY_RUNACTIVE	Acount	アカウント数 (0～)
SFCS_SEC_LINEW	Line%d	回線情報 WILD
SFCS_SEC_LINE0	Line0	回線 0 情報
SFCS_SEC_LINE1	Line1	回線 1 情報
SFCS_SEC_LINE2	Line2	回線 2 情報
SFCS_SEC_LINE3	Line3	回線 3 情報
SFCS_KEY_LINERUN	Run	回線起動状況 ("0":動作していない, "1":動作している)
SFCS_KEY_LINESEND	Send	送信状況 ("0":送信不可, "1":送信可)
SFCS_KEY_LINERECV	Receive	受信状況 ("0":受信不可, "1":受信可)
SFCS_KEY_LINEMDM	Modem	モデム名
SFCS_SEC_AUTODEL	AutoDel	自動削除情報
SFCS_KEY_AUTOTX	AutoTx	送信情報自動削除 (1:ON, 0:OFF)
SFCS_KEY_AUTOTXTIME	AutoTxTime	送信情報自動削除対象 経過日数 (日)
SFCS_KEY_AUTORX	AutoRx	受信情報自動削除 (1:ON, 0:OFF)
SFCS_KEY_AUTORXTIME	AutoRxTime	受信情報自動削除対象 経過日数 (日)
SFCS_SEC_WARNING	Warning	警告
SFCS_KEY_DISKFREEINST	DiskFreeInstall	STARFAX Server SDK がインストールされているフォルダのディスク空き容量 (1:残り 150M 以下, 0:正常)

プログラム定義	値	内容
SFCS_KEY_DISKFREECTRL	DiskFreeCtrl	制御関連インターフェイスフォルダのディスク空き容量（1:残り 150M 以下, 0:正常）
SFCS_KEY_DISKFREEDATA	DiskFreeData	データフォルダのディスク空き容量（1:残り 150M 以下, 0:正常）
SFCS_KEY_SENDIDXNUM	SendIndexNum	送信情報インデックスファイル件数 (1:8 万件以上, 0:正常)
SFCS_KEY_RECVIDXNUM	RecvIndexNum	受信情報インデックスファイル件数 (1:8 万件以上, 0:正常)
SFCS_KEY_QUEIDXNUM	QueIndexNum	未送信情報インデックスファイル件数 (1:8 万件以上, 0:正常)
SFCS_KEY_DUSENDIDXNUM	DustSendIndexNum	ごみ箱 送信情報インデックスファイル件数（1:8 万件以上, 0:正常）
SFCS_KEY_DURECVIDXNUM	DustRecvIndexNum	ごみ箱 受信情報インデックスファイル件数（1:8 万件以上, 0:正常）
SFCS_FLD_SENDMIS	SENDMIS	送信命令フォルダ
SFCS_FRM_SENDMIS	S%04d%02d%02d%02d%02d%02d%03d.WSM	送信命令ファイル 固有ファイル名作成用フォーム
SFCS_FLD_RECVINFO	INFORECV	受信情報作成フォルダ
SFCS_FLD_RECVDELMIS	DELMIS	受信情報削除命令フォルダ
SFCS_FRM_RECVDELMIS	DR%04d%02d%02d%02d%02d%02d%03d.WSM″	受信情報削除命令ファイル 固有ファイル名作成用フォーム
SFCS_FLD_SENDINFO	INFOSEND	送信情報作成フォルダ
SFCS_FLD_SENDDELMIS	DELMIS	送信情報削除命令フォルダ
SFCS_FRM_SENDDELMIS	DS%04d%02d%02d%02d%02d%02d%03d.WSM	送信情報削除命令ファイル 固有ファイル名作成用フォーム
SFCS_FLD_QUEINFO	INFOQUE	未送信情報作成フォルダ
SFCS_FLD_QUEDELMIS	DELMIS	未送信情報削除命令フォルダ
SFCS_FRM_QUEDELMIS	DQ%04d%02d%02d%02d%02d%02d%03d.WSM	未送信情報削除命令ファイル 固有ファイル名作成用フォーム
SFCS_FLD_PRTMIS	PRTMIS	印刷命令フォルダ
SFCS_FRM_PRTMIS	PD%04d%02d%02d%02d%02d%02d%03d.WSM	印刷命令ファイル 固有ファイル名作成用フォーム
SFCS_FLD_EMSMIS	EMSMIS	メール送信命令フォルダ
SFCS_FRM_EMSMIS	SD%04d%02d%02d%02d%02d%02d%03d.WSM	メール送信命令ファイル 固有ファイル名作成用フォーム
SFCS_FLD_NOTIINFO	NOTICE	通知情報フォルダ
SFCS_FLD_RECVADDNOTI	ADDRECV	受信情報追加済み通知フォルダ
SFCS_WID_RECVADDNOTI	AR*.WSM	受信情報追加済み通知ファイル ワイルドカード
SFCS_FLD_SENDADDNOTI	ADDSEND	送信情報追加済み通知フォルダ
SFCS_WID_SENDADDNOTI	AS*.WSM	送信情報追加済み通知ファイル ワイルドカード

プログラム定義	値	内容
SFCS_FLD_RECVDELNOTI	DELRECV	受信情報削除済み通知フォルダ
SFCS_WID_RECVDELNOTI	DR*.WSM	受信情報削除済み通知ファイル ワイルドカード
SFCS_FLD_SENDDELNOTI	DELSEND	送信情報削除済み通知フォルダ
SFCS_WID_SENDDELNOTI	DS*.WSM	送信情報削除済み通知ファイル ワイルドカード

● **エラーコード**

プログラム定義	値	内容
SFCS_SUCCESS	0	正常終了
SFCS_ERR_NoRegInstall	1	インストールフォルダがレジストリに設定されていません。
SFCS_ERR_NoInstall	2	インストールフォルダが存在しません。
SFCS_ERR_NoRegCtrl	3	制御関連インターフェイスフォルダがレジストリに設定されていません。
SFCS_ERR_NoCtrl	4	制御関連インターフェイスフォルダが存在しません。
SFCS_ERR_NoRegData	5	データフォルダがレジストリに設定されていません。
SFCS_ERR_NoData	6	データフォルダが存在しません。
SFCS_ERR_NoFolder	7	フォルダが存在しません。
SFCS_ERR_PARAM_MODE	100	関数引数エラー: サブフォルダモードの指定に誤りがあります。

● **サブフォルダモード**

プログラム定義	値	内容
SFCS_SMODE_SENDMIS	0	送信命令フォルダ
SFCS_SMODE_RECVINFO	1	受信情報フォルダ
SFCS_SMODE_SENDINFO	2	送信情報フォルダ
SFCS_SMODE_QUEINFO	3	未送信情報フォルダ
SFCS_SMODE_RECVDELMI	4	受信情報削除命令フォルダ
SFCS_SMODE_SENDDELMIS	5	送信情報削除命令フォルダ
SFCS_SMODE_QUEDELMIS	6	未送信情報削除命令フォルダ
SFCS_SMODE_PRTMIS	7	印刷命令フォルダ
SFCS_SMODE_EMSMIS	8	メール送信命令フォルダ
SFCS_SMODE_RECVADDNOTI	9	受信情報追加済み通知フォルダ
SFCS_SMODE_SENDADDNOTI	10	送信情報追加済み通知フォルダ
SFCS_SMODE_RECVDELNOTI	11	受信情報削除済み通知フォルダ
SFCS_SMODE_SENDDELNOTI	12	送信情報削除済み通知フォルダ

● **サービス状態**

プログラム定義	値	内容
SFCS_SERVICE_STOP	0	停止中
SFCS_SERVICE_SERVICE	1	サービスモード 動作中
SFCS_SERVICE_TRAY	2	非サービスモード 動作中

C　関数仕様

ヘッダー："SfCsSys.h"内で宣言。

◇　SFCSSYS_CheckInstall

書式

int SFCSSYS_CheckInstall(void)

入力

なし

返値

エラーコード

SFCS_SUCCESS	(0)	:	正常終了
SFCS_ERR_NoRegInstall	(1)	:	インストールフォルダがレジストリに設定されていません。
SFCS_ERR_NoInstall	(2)	:	インストールフォルダが存在しません。
SFCS_ERR_NoRegCtrl	(3)	:	制御関連インターフェイスフォルダがレジストリに設定されていません。
SFCS_ERR_NoCtrl	(4)	:	制御関連インターフェイスフォルダが存在しません。
SFCS_ERR_NoRegData	(5)	:	データフォルダがレジストリに設定されていません。
SFCS_ERR_NoData	(6)	:	データフォルダが存在しません。

機能

STARFAX Server SDK のインストール状況の確認をします。

◇ SFCSSYS_CheckSubFolder

書式

int SFCSSYS_CheckSubFolder(int iMode)

入力

iMode : サブフォルダモード

SFCS_SMODE_SENDMIS	(0)	: 送信命令フォルダ
SFCS_SMODE_RECVINFO	(1)	: 受信情報フォルダ
SFCS_SMODE_SENDINFO	(2)	: 送信情報フォルダ
SFCS_SMODE_QUEINFO	(3)	: 未送信情報フォルダ
SFCS_SMODE_RECVDELMIS	(4)	: 受信情報削除命令フォルダ
SFCS_SMODE_SENDDELMIS	(5)	: 送信情報削除命令フォルダ
SFCS_SMODE_QUEDELMIS	(6)	: 未送信情報削除命令フォルダ
SFCS_SMODE_PRTMIS	(7)	: 印刷命令フォルダ
SFCS_SMODE_EMSMIS	(8)	: メール送信命令フォルダ
SFCS_SMODE_RECVADDNOTI	(9)	: 受信情報追加済み通知フォルダ
SFCS_SMODE_SENDADDNOTI	(10)	: 送信情報追加済み通知フォルダ
SFCS_SMODE_RECVDELNOTI	(11)	: 受信情報削除済み通知フォルダ
SFCS_SMODE_SENDDELNOTI	(12)	: 送信情報削除済み通知フォルダ

返値

エラーコード

SFCS_SUCCESS	(0)	: 正常終了
SFCS_ERR_NoRegCtrl	(3)	: 制御関連インターフェイスフォルダがレジストリに設定されていません。
SFCS_ERR_NoCtrl	(4)	: 制御関連インターフェイスフォルダが存在しません。
SFCS_ERR_NoFolder	(7)	: フォルダが存在しません。
SFCS_ERR_PARAM_MODE	(100)	: 関数引数エラー: サブフォルダモードの指定に誤りがあります。

機能

STARFAX Server SDK の各種サブフォルダの存在確認をします。

◇ SFCSSYS_CheckServiceStatus

書式

int SFCSSYS_CheckServiceStatus(void)

入力

なし

返値

サービス状態	SFCS_SERVICE_STOP	(0)	：停止中
	SFCS_SERVICE_SERVICE	(1)	：サービスモード動作中
	SFCS_SERVICE_TRAY	(2)	：非サービスモード 動作中

機能

STARFAX Server SDK のサービスの状態を取得します。

◇ SFCSSYS_GetSfcsFolder

書式

DWORD SFCSSYS_GetSfcsFolder(char *pGet, DWORD dwBufLen)

入力

pGet : 取得バッファ
dwBufLen : 取得バッファバイト数

返値

インストールフォルダパスのバイト数

機能

STARFAX Server SDK のインストールフォルダパスを取得します。
なお、取得されたフォルダ名の終端に '¥' は付加されていません。

◇ SFCSSYS_GetCtrlFolder

書式

DWORD SFCSSYS_GetCtrlFolder(char *pGet, DWORD dwBufLen)

入力

pGet : 取得バッファ
dwBufLen : 取得バッファバイト数

返値

制御関連インターフェイスフォルダパスのバイト数

機能

STARFAX Server SDK の制御関連インターフェイスフォルダパスを取得します。
なお、取得されたフォルダ名の終端に '¥' は付加されていません。

◇ SFCSSYS_GetDataFolder

書式

DWORD SFCSSYS_GetDataFolder(char *pGet, DWORD dwBufLen)

入力

pGet : 取得バッファ
dwBufLen : 取得バッファバイト数

返値

データフォルダパスのバイト数

機能

STARFAX Server SDK のデータフォルダパスを取得します。
なお、取得されたフォルダ名の終端に '¥' は付加されていません。

◇ SFCSSYS_GetSubFolder

書式

DWORD SFCSSYS_GetSubFolder(int iMode, char *pGet, DWORD dwBufLen)

入力

iMode : サブフォルダモード

SFCS_SMODE_SENDMIS	(0)	:	送信命令フォルダ
SFCS_SMODE_RECVINFO	(1)	:	受信情報フォルダ
SFCS_SMODE_SENDINFO	(2)	:	送信情報フォルダ
SFCS_SMODE_QUEINFO	(3)	:	未送信情報フォルダ
SFCS_SMODE_RECVDELMIS	(4)	:	受信情報削除命令フォルダ
SFCS_SMODE_SENDDELMIS	(5)	:	送信情報削除命令フォルダ
SFCS_SMODE_QUEDELMIS	(6)	:	未送信情報削除命令フォルダ
SFCS_SMODE_PRTMIS	(7)	:	印刷命令フォルダ
SFCS_SMODE_EMSMIS	(8)	:	メール送信命令フォルダ
SFCS_SMODE_RECVADDNOTI	(9)	:	受信情報追加済み通知フォルダ
SFCS_SMODE_SENDADDNOTI	(10)	:	送信情報追加済み通知フォルダ
SFCS_SMODE_RECVDELNOTI	(11)	:	受信情報削除済み通知フォルダ
SFCS_SMODE_SENDDELNOTI	(12)	:	送信情報削除済み通知フォルダ

pGet : 取得バッファ

dwBufLen : 取得バッファバイト数

返値

サブフォルダパスのバイト数

機能

STARFAX Server SDK の各種サブフォルダパスを取得します。

なお、取得されたフォルダ名の終端に '¥' は付加されていません。

補足: [CTRL FOLDER] ... 制御関連インターフェイスフォルダ

◇ SFCSSYS_GetRunInfoString

書式

DWORD SFCSSYS_GetRunInfoString(char *pSection, char *pKey, char *pDefault, char *pGet, DWORD dwBufLen)

入力

pSection : セクション名
pKey : キー名
pDefault : デフォルト文字列
pGet : 取得バッファ
dwBufLen : 取得バッファバイト数

返値

取得した文字列のバイト数

機能

起動情報ファイル([CTRL FOLDER]¥SfCsRun.inf)より、任意の文字列項目を取得します。

補足: [CTRL FOLDER] ... 制御関連インターフェイスフォルダ

◇ SFCSSYS_GetRunInfoInt

書式

DWORD SFCSSYS_GetRunInfoInt(char *pSection, char *pKey, DWORD dwDefault)

入力

pSection : セクション名
pKey : キー名
dwDefault : デフォルト数値

返値

入力引数で指定した項目の設定数値

機能

起動情報ファイル([CTRL FOLDER]¥SfCsRun.inf)より、任意の数値項目を取得します。

◇ SFCSSYS_GetUniqFileName

書式

BOOL SFCSSYS_GetUniqFileName(char *pFolder, char *pWild, char *pMakePath)

入力

pFolder : フォルダパス
pWild : 作成用フォーム
SFCS_FRM_SENDMIS : 送信命令ファイル
SFCS_FRM_RECVDELMIS : 受信情報削除命令ファイル
SFCS_FRM_SENDDELMIS : 送信情報削除命令ファイル
SFCS_FRM_QUEDELMIS : 未送信情報削除命令ファイル
SFCS_FRM_PRTMIS : 印刷命令ファイル
SFCS_FRM_EMSMIS : メール送信命令ファイル
pMakePath : 日付、時刻によるユニークなファイル名取得バッファ

返値

TRUE : 正常終了
FALSE : エラー

機能

日付、時刻によるユニークなファイル名を作成します。
これは、各種命令ファイル名の作成を支援する為の関数です