MIGARO. TECHNICAL REPORT

ミガロ.テクニカルレポート

No.12
2019年秋

株式会社ミガロ.

Migaro.Technical Award 2019 お客様受賞論文／ミガロ.テクニカルアワード

Migaro. Special Report 2019 特別寄稿論文／ミガロ.テクニカルレポート

Migaro. Technical Award 2019 SE 論文／ミガロ.テクニカルレポート

ごあいさつ

いつもミガロ.製品をご愛用いただき誠にありがとうございます。

「ミガロ.テクニカルレポート」は、Delphi/400、Valence、SP4i などのミガロ.製品を使った開発に関する技術情報をお届けする論文集で、2008 年に創刊し、このたび第 12 号を発刊する運びとなりました。これもひとえに、技術論文をご寄稿いただいた多くのお客様、ならびに本レポートに対して貴重なご意見・ご要望をお寄せ下さった皆様のご支援の賜物と、心より感謝しております。

さて平成から令和となり、新時代の幕開けとともに、デジタル（IT）主導によるビジネス改革を意味するデジタルトランスフォーメーション（DX）という言葉をよく耳にするようになりました。令和元年版の総務省の情報通信白書では、今後、情報システム部門が新たな価値を創造するプロフィットセンターとなり企業の中核を担っていく必要がある、と指摘しています。弊社は、お客様の IBM i 基幹システムが周囲のシステムとも連携しつつ DX の中心となることを確信しており、IBM i の高度な活用と迅速な開発を可能とする開発ツールおよびサポートの提供により、お客様の情報システム変革のお役に立てるよう、さらなる努力をしてまいります。

テクニカルレポートの第 1 部は、「ミガロ.テクニカルアワード 2019」にご応募頂いたお客様論文の中から受賞作品を掲載しました。いずれの論文もお客様の実際の開発事例に基づく技術情報ですので、ぜひシステム開発の参考としてご活用ください。第 2 部は、弊社 SE による技術論文集です。弊社が行っているお客様システムの受託開発や日々のテクニカルサポートからヒントを得た生きた技術情報をお届けしています。さらに今回は、フランスの Metrixware 社による「スペシャルレポート」を掲載しました。Metrixware 社は Delphi/400、SP4i 開発部門（SystemObjects）を傘下に持ち、今後、同社オリジナル製品と融合した新しい IBM i ユーザー様向けソリューションを日本でも提供する予定です。

今回のお客様論文は、「電子帳簿保存法のスキャナ保存制度に適合するシステム開発」や「iPhone による宝飾品の販売管理システム開発」など、創意工夫にあふれる論文を多数ご寄稿いただきました。受賞作品は Delphi/400、SP4i、Valence が各 2 作品ずつとバランスのとれた構成となっております。また、弊社 SE 論文については、「FireMonkey フレームワークのカメラコンポーネントを使用したアプリ開発」や「ローコード開発ツール Valence AppBuilder で RPG を連携する方法」など、さまざまなテクニックを開発に活かしていただくための技術情報をご紹介しております。本レポートが少しでも皆様の開発・保守のお役に立てば幸いです。

最後に、ミガロ.テクニカルレポート第 12 号を発刊するにあたりまして、多くのお客様・パートナー様にご支援、ご協力をいただきましたことを、この場をお借りして、あらためて厚く御礼申し上げます。

2019 年秋

株式会社ミガロ.
代表取締役社長
上甲 將隆

Migaro. Technical Award 2019

お客様受賞論文／ミガロ. テクニカルアワード

最優秀賞

Delphi/400による電子帳簿保存法スキャナ保存制度への挑戦

石井 裕昭 様
豊鋼材工業株式会社
監査役

豊鋼材工業株式会社
http://www.yutaka-steel.co.jp/

鋼板加工のトータルコーディネーターとして、レベラー・スリット・溶断・開先・穴明け・プレス・ベンディング・溶接・マシニング・ショット・プライマー設備を有機的に活用し、幅広い産業分野のお客様に商品・サービスを提供している。九州・沖縄エリアでは業界トップシェア。伊藤忠丸紅鉄鋼・新日鐵住金系の会社である。

当社では国税関係書類（領収書等）のスキャナ保存に関する法的要件を満たすシステムを開発し、税務署への申請・承認を経て運用を開始した。この経緯およびシステム内容について以下に紹介する。

システムの概要

電子帳簿保存法に則したスキャナ保存制度の承認件数は、全法人の 0.07％（2017 年度末時点）と少なく、その中でも自社開発システムでの申請はほとんど例がない。今回、旅費交通費と交際費の立て替え精算を、従来の紙書類による運用から、複合機のスキャナ、ワークフロー、経理仕訳等を連携させたシステムに移行した。

システム化にあたってはペーパレス化、ワークフロー運用、経理処理連携、領収書のスキャン画像化など多くの開発要素がある。また取り扱う対象が国税関係の書類であるため、電子帳簿保存法の要件を満たす機能を実現する必要があった。

今回は領収書をスキャンデータとして保存する、いわゆるスキャナ保存制度の適用案件であり、【図 1】に示す各要件を考慮して構築した。本案件は、2018 年 12 月の税務署への申請後、2019 年 4 月よりみなし承認となり運用を開始している。

従来の課題と開発の概要

出張旅費、交際費（会議費を含む）等の申請･精算に関する一連の処理は従来、紙フォームで運用していた。紙の運用による非効率､内容のブラックボックス化、誤りの発生などもあり、また内部統制強化の観点からもシステム化が望まれていた。

今回、一連の作業をシステム化して情報の相互連携を可能にするとともに、以下の業務課題の解決を目的とした。【図 2】

・申請、承認に関わる業務の簡素化とスピードアップ
・自動仕訳連携（科目設定、負担部門配賦などを含む）による経理処理の容易化とミス防止
・自動単価設定および集計計算の自動化により誤りと手戻りを削減
・「駅すぱあと」の組み込みによる運賃確認の容易化とチェックの簡素化
・伺いと精算の情報連携による内部統制強化
・ワークフローによるペーパレス化
・領収書のスキャナ保存による電子帳簿保存法への対応
・各種実績情報のデータベース化による分析の可能化

今回の機器・システム構成は【図 3】のとおりである。機能開発後の新業務フローは【図 4】のような運用とした。また開発機能の一覧を【図 5】【図 6】に示す。

技術的なポイント

タイムスタンプ

タイムスタンプは、刻印された時刻以前にその電子文書が存在していたこと

図1　スキャナ保存要件

スキャナ保存要件一覧表（平成28年9月30日以後申請分）

スキャナ保存に係る要件の概要		規則第3条の該当箇所（注1）	帳簿・書類のうち決算関係書類	書類のうち決算関係書類以外のもの		
				重要書類（注2）		一般書類（注3）
				領収書・契約書及びこれらの写し	資金や物の流れに直結・連動する書類（請求書・納品書等）	左記以外の書類（見積書・注文書等）
スキャナ保存の可否		③	×	○	○	○
真実性の確保	入力期間の制限	⑤一イ ⑤一ロ ⑥	-	【早期入力方式】 国税関係書類に係る記録事項の入力をその受領等後、速やか（1週間以内）に行うこと 【業務処理サイクル方式】 国税関係書類に係る記録事項の入力をその業務の処理に係る通常の期間（1か月以内）を経過した後、速やか（1週間以内）に行うこと ※　国税関係書類の受領等から入力までの各事務の処理に関する規程を定めている場合に限る 【適時入力方式】適時に入力（注4）		
	一定水準以上の解像度及びカラー画像による読み取り	⑤二イ ⑥	-	(1)　解像度が200ｄｐｉ相当以上であること (2)　赤色、緑色及び青色の階調がそれぞれ256階調以上（24ビットカラー）であること (2)に関しては、白黒階調（いわゆるグレースケール）での読み取りも認められる。（注4）		
	タイムスタンプの付与	⑤二ロ ⑥	-	一般財団法人日本データ通信協会が認定する業務に係るタイムスタンプ（記録事項が変更されていないことについて、保存期間を通じて確認することができ、課税期間中の任意の期間を指定し、一括して検証することができるものに限る。）を、一の入力単位ごとの電磁的記録の記録事項に付すこと ※　国税関係書類の受領者等が読み取る場合は、受領等後、受領者等が署名の上、特に速やか（3日以内）にタイムスタンプを付すこと 受領者等が読み取る場合は、読み取る際に、又は受領等後、受領者等が署名の上、特に速やか（3日以内）にタイムスタンプを付すこと（注4）		
	読取情報の保存	⑤二ハ ⑥	-	読み取った際の解像度、階調及び当該国税関係書類の大きさに関する情報を保存すること ※　国税関係書類の受領者等が読み取る場合で、当該国税関係書類の大きさがA4以下であるときは、大きさに関する情報の保存は不要 大きさに関する情報の保存は不要（注4）		
	ヴァージョン管理	⑤二ニ	-	国税関係書類に係る電磁的記録の記録事項について訂正又は削除を行った場合には、これらの事実及び内容を確認することができる電子計算機処理システムを要すること		
	入力者等情報の確認	⑤三	-	国税関係書類に係る記録事項の入力を行う者又はその者を直接監督する者に関する情報を確認できるようにしておくこと		
	適正事務処理要件	⑤四 ⑥	-	国税関係書類の受領等から入力までの各事務について、次に掲げる事項に関する規程を定めるとともに、これに基づき当該各事務を処理すること (1)　相互に関連する各事務について、それぞれ別の者が行う体制（相互けんせい） (2)　当該各事務に係る処理の内容を確認するための定期的な検査を行う体制及び手続（定期的な検査） (3)　当該各事務に係る処理に不備があると認められた場合において、その報告、原因究明及び改善のための方策の検討を行う体制（再発防止） ※　小規模企業者の場合で、(2)を税務代理人が行うときは、(1)の要件は不要 不要（注4）		
可視性の確保	帳簿との相互関連性の確保	⑤五	-	国税関係書類に係る電磁的記録の記録事項と当該国税関係書類に関連する国税関係帳簿の記録事項との間において、相互にその関連性を確認することができるようにしておくこと		
	見読可能装置の備付け等	⑤六 ⑥	-	(1)　規則第3条第5項第6号に定める要件を満たしたカラーディスプレイ及びカラープリンタ並びに操作説明書を備え付けること (2)　電磁的記録について、整然とした形式や4ポイントの大きさの文字を認識できることなど、規則第3条第5項第6号に規定する状態で、速やかに出力することができるようにすること 白黒階調（いわゆるグレースケール）による保存の場合、ディスプレイ及びプリンタはカラー対応である必要はない。（注4）		
	電子計算機処理システムの開発関係書類等の備付け	①三 ⑤七	-	電子計算機処理システムの概要を記載した書類、そのシステムの開発に際して作成した書類、操作説明書、電子計算機処理並びに電磁的記録の備付け及び保存に関する事務手続を明らかにした書類を備え付けること		
	検索機能の確保	①五 ⑤七	-	電磁的記録の記録事項について、次の要件による検索ができるようにすること (1)　取引年月日その他の日付、取引金額その他主要な記録項目での検索 (2)　日付又は金額に係る記録項目について範囲を指定しての検索 (3)　2以上の任意の記録項目を組み合わせての検索		

注1　「電子計算機を使用して作成する国税関係帳簿書類の保存方法等の特例に関する法律施行規則」第3条の該当箇所を指し、丸囲いの数字は「項」を、漢数字は「号」を表します。
注2　「電子計算機を使用して作成する国税関係帳簿書類の保存方法等の特例に関する法律」第4条第3項に規定する国税関係書類のうち、一般書類以外の書類が該当します。
注3　「電子計算機を使用して作成する国税関係帳簿書類の保存方法等の特例に関する法律施行規則第3条第8項に規定する国税庁長官が定める書類を定める件（平成17年国税庁告示第4号）」に定める書類が該当します。
注4　本要件は、一般書類のスキャナ保存にのみ適用されます。また、当該電磁的記録の作成及び保存に関する事務の手続を明らかにした書類（当該事務の責任者が定められているもの。）の備付けを行う必要があります。

（存在証明）と、その時刻以降に当該文書が改ざんされていないこと（非改ざん証明）を証明するものである。

タイムスタンプは、時刻認証業務認定事業者（TSA、国内6社）の提供サービスを利用した（e-timing EVIDENCE 3161 PDF Lib-W ／アマノセキュアジャパンを使用）。インストールした実行ファイル（dll）に対して、コマンドライン引数を指定して実行することでタイムスタンプが発行される。

タイムスタンプは、コマンドプロンプトで指定のコマンドを実行すると、結果がコマンドプロンプトの画面に表示されるが、毎回手作業では引数を入力できないので、コマンド自体は入出力ファイル名をパラメータで渡す形で与えるとともに、実行結果をテキスト出力する方法で実現した。

これについては、Mr.XRAY の「552_パイプを使用したコマンドラインのリダイレクト」（http://mrxray.on.coocan.jp/Delphi/plSamples/552_PipeRedirect.htm）を参考に実装した。【図 7】

処理の成否は、得られた Text に含まれる文字列によって判断した。なお、今回利用したサービスでは PC の Bit 数に応じて実行ファイルを使い分ける必要があるため、あらかじめ PC の Bit 数を取得し、判定している。【図 8】

また申請者、スキャン画像チェック者、承認者以外の者による処理内容の妥当性検証が定期的に求められるため、規程を整備するとともに、システム的な検証機能（タイムスタンプによる改ざん履歴確認を含む）と検証履歴を残す機能も実装した。

QR コードの活用

領収書の電子画像のスキャナ保存が、今回の開発の目玉である。領収書の電子画像は、対応する支払データと連携させる必要がある。支払案件ごとに付与した ID を QR コード化して台紙に印字し、領収書を貼付したあと、スキャナ読み取り保存を行う。この方法により領収書画像と支払データの連携を実現した。QR コードは、スキャンした領収書画像 PDF のファイル名設定にも利用できる。

領収書貼付用の台紙は Excel でフォーム設計しており、QR コードはフリーのアドインツール（QRimgAdd）で作成した。

ただし、現在入れ替え途中の新しい PC に搭載される Excel（バージョン 19）では上記アドインツールが利用できないので、「Google Chart API の QR コード作成機能を呼び出す URL」でオブジェクトを作成する方法により対応した。【図 9】

全 PC の入れ替え完了までは、インストールされている Excel のバージョンに応じて、2 種類のフォーム用 Excel ファイルのいずれかを自動選択して呼び出すようにした。【図 10】【図 11】

アドインツールを使用した Excel テンプレートを利用する際には、PC ごとにアドインファイルの保存と Excel オプションの参照設定を事前に行う必要があり、若干汎用性が失われる。

XEROX 複合機との連携

XEROX 複合機のスキャナ機能で読み取った領収書画像を今回のシステムと連携するために、「ApeosFlowService」（最新版は ApeosWare Flow Management。以下、Apeos）というミドルウェアを活用した。

これは読み取った画像の加工、整理、分類、配信、OCR、QR コード読み取り、読み取り履歴のテキスト出力など、さまざまな処理を高い自由度でルール化し、複合機本体のスキャン画像の一時保存領域（BOX）と紐付けて処理する。【図 12】

使用予定の全複合機に領収書のスキャン画像取り込み用 BOX をあらかじめ作成し、以下の処理ルールを設定した。Apeos はサーバー上のサービスとして常駐し、BOX を取り込み元として監視（30 秒ごとにポーリング）する。読み取られた画像データがあれば、取り込んだうえで次の処理を実行する。

(1) QR コードの内容を読み取り、変数として格納（ハイフン等で分割されていれば複数属性として取得）
(2) 自動正立
(3) PDF 変換
(4) ファイル名設定（QR コード取得変数）
(5) サーバーの指定フォルダに保存
(6) 読み取り条件等をテキストファイルに出力

(1) の QR コード内容の読み取りについては、読み取った複数ページの文書中に QR コードが認識されると、次に QR コードが現れるページの前までを 1 文書として区切れる。ただし 1 ページ中に 2 つの QR コードがあると、エラーになる。

(6) は、電子帳簿保存のスキャナ保存の要件として規定されたスキャン条件（解像度、読み取りサイズ等）を担保する履歴として残すためのもので、後に IBM i のデータベースに取り込む。

カラーでのフォーム印刷必須化

電子帳簿のスキャナ保存の要件として、24Bit カラーでの読み取りが要求されているが、Apeos で出力するスキャン履歴のテキストファイルではこの情報を取得できない。このためスキャン用の台紙中にカラーサンプルを印刷し、スキャンされた画像がカラーで読み取られていることを目視のうえ、その確認履歴を残すこととした。【図 13】

Delphi アプリケーションでフォーム設計した Excel を起動し、Excel 側のマクロで台紙の印刷を自動実行する。Excel のマクロで常にカラー印刷となるように制御するのは難しいので、あらかじめデフォルト設定をカラー印刷にしたプリンタ情報をパラメータとして Excel 側に渡し、出力プリンタをマクロで指定するようにした。【図 14】

駅すぱあと連携

旅費精算業務では、運賃明細の申請後にネット等で各区間の料金を調査して記入することが多く、また運賃をチェックする側でも同様の調査が必要となる。今回、駅すぱあと SDK を Delphi のアプリケーションに組み込み、指定区間の料金を取得できるようにした。駅すぱあと SDK の組み込みについては、すでにミガロ．テクニカルアワード 2011 で事例紹介されており、同様の方法で開発を進めた。【図 15】

ここでは、駅すぱあとで取得したデータかどうかを判別できるように工夫している。これにより申請金額の妥当性判断を容易にし、チェック負荷の軽減を狙った【図 16】。また駅すぱあとで指定する出発地と到着地の履歴を保存し、これを ComboBox の候補として選択可能にす

図2　システム概念図

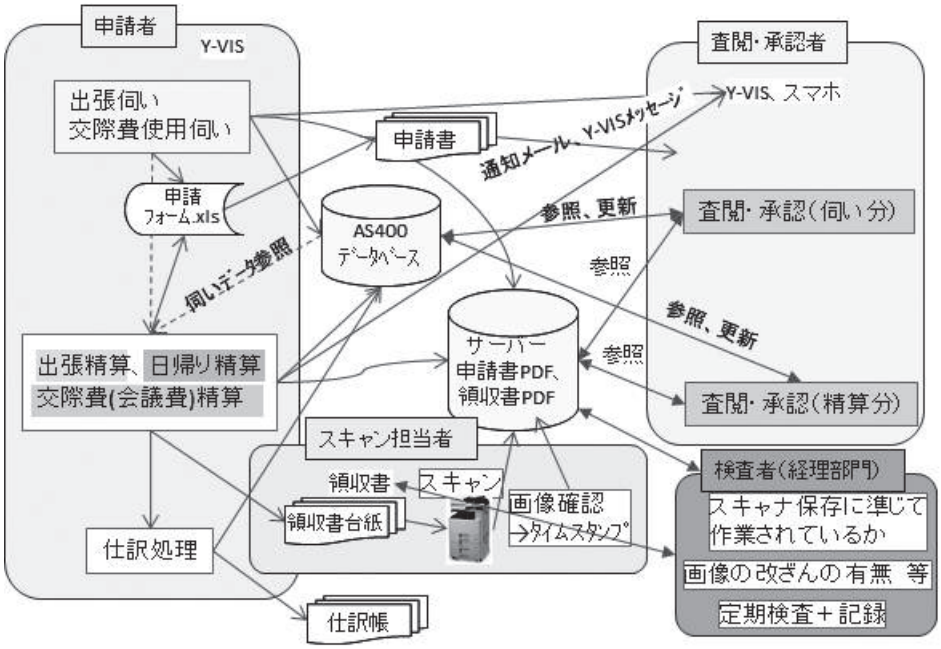


図3　機器･システム構成

分類	項目	内容(用途、目的等)	備考
ハードウェア	System I (IBM)	各種データベース登録、更新	既存利用
	PC（各利用者用）	伺い申請、精算申請、確認･承認	既存利用
	サーバー	領収書、申請書等PDF保存、 タイムスタンプ処理 スキャン画像のPDF変換、所定フォルダー保存	既存利用
	スキャナー	領収書スキャン	各拠点XEROX複合機 既存利用
	プリンター	領収書台紙(QR含)出力 申請書等出力(必要時)	各拠点プリンター 既存利用
ソフトウェア	Y-VIS	伺い申請、精算申請、確認･承認、履歴検索 タイムスタンプ処理、一括検証	既存に機能追加 社内開発 (各利用者PC)
	ApeosFlowService （XEROX)	スキャン画像のPDF変換、所定フォルダー保存	既存利用 取込ルール追加
	AMANO E-timing EVIDENCE 3161 PDF Lib-W	タイムスタンプ処理、および一括検証	Y-VISにてバッチ処理 用に組込み

ることで、入力を省略した【図 17】。さらに料金計算結果取得のオプションとして、IC カード使用の有無を選択できるようにした。

ワークフロー処理

内部統制の強化、処理スピードと効率アップのために、従来の紙ベース処理のフォームを極力踏襲しながら、PDF ファイルをベースとしたワークフロー機能を組み込んだ。処理 ID と紐付いたファイル名で PDF 化した申請書フォームおよび領収書は、TWebBrowser を利用して確認できるようにした。【図 18】

電子帳簿保存のスキャナ保存では、経費の申請者、領収書の確認者（スキャン実行者）、最終承認者が重複せず相互牽制を働かせる必要があり、承認ルート設定時のエラーチェック機能が必要となる。また、経理担当部門の支払い処理前の確認履歴も残せるようにし、チェック漏れがないように実装した。

ワークフローについては、社員コード、処理（申請）ID、処理実績日、進捗ステータス、摘要などの項目を備えたデータベースファイルを新たに作成し、申請・承認等の各役割の社員のアクションに応じてレコードが更新されるとともに、次の担当者に処理（判定）を促す仕組みとした。【図 19】

次に処理すべき担当者に早く気付かせるため、社内メールに加えて、通常使用している Delphi アプリケーションのフォーム上部に、ブリンク（点滅）するメッセージを割り込ませる仕組みを作った。【図 20】

連絡メッセージは、社員マスターとシステム機器一覧 DB からメールアドレス、PC の IP アドレスを送信先として取得し、パラメータ化した申請データよりメッセージ内容を作成する。社内メールには Web メールを利用しており、被通知者は任意のタイミングで確認できる。

一方、フォーム上部で点滅するメッセージは、送信側が Ping 機能で受信 PC の死活チェックを行い、繋がっていれば相手 PC の指定フォルダに指定テキストファイルを保存（すでにある場合はメッセージ追記）する。

受信者側のアプリケーションは上記フォルダを TShellChangeNotifier で監視しており、変化があれば指定テキストファイル内のメッセージを点滅状態（Timer でカラー変更）でフォーム上部に表示する。

死活チェックで NG の場合は、サーバーの指定フォルダに相手 PC 名をファイル名に付加して保存する。受信者側は、次にアプリケーションを起動する際にサーバーの指定フォルダを確認するようになっており、自分の PC 名のファイルがあれば同様にメッセージを表示させる。【図 21】【図 22】【図 23】

受信者は、ワークフロー上で自身の処理待ち状態となっているレコード分のみをリストで確認でき、Grid 上で選択した対象の PDF 表示で内容を確認できる。【図 24】

スマートフォン連携

電子帳簿保存のスキャナ保存制度では 2016 年より規制が緩和され、入力装置としてスマートフォン、デジタルカメラなどが認められている。ただしこれらの機器を使用する場合は、受領してから以降の処理の猶予期間が短いこと（受領者本人が処理する場合はタイムスタンプの付与までの期限が 3 日以内）もあり、今回の一連の開発では入力機器の対象外とした。

ただし、ワークフローの査閲・承認用の機能を Android アプリとして開発している。【図 25】

スマートフォンでも申請フォームの内容や領収書が PDF ファイルのイメージで参照できるように、TWebBrowser コンポーネントの利用を試みたが、PDF ファイルは表示できないことが判明した。これはミガロ .のサポートにより、intent という機能を使用した手法により実現できた。【図 26】

スマートフォンでの査閲・承認処理でのワークフローデータベースの更新は PC と同様であるが、メールおよびメッセージ通知方法が異なる。

通知先 PC の死活チェック等を直接スマートフォンから実行するのは難しいので、通知すべき内容を DataSnap サーバー経由でデータベースに書き込み、常駐起動する PC のアプリでそのデータベースを Timer 監視し、レコードがあればレコードの内容に応じて通知し、メールを送付するようにした。

その他の考慮事項

要件を満たしたスキャン画像の取得

複合機でのスキャンデータで、常に定められた要件を、すべて満足する条件で読み込むには、解像度、カラーモード、ファイル形式などを正しく設定する必要がある。しかし手作業の場合、設定の間違いなどで必ずしもルールどおりに運用できない危険性がある。このため、複合機の一連の各設定をまとめてメモリ登録する機能を利用して、条件設定のバラツキが生じないようにした。【図 27】

資格に応じた日当、手当設定

当社では出張などの日当、宿泊手当などは資格に応じて単価が定められている。従来、この単価はシステム上に反映されておらず、社内規程に準じて計算し記入していたが、精算頻度の少ない社員や昇格直後の社員などが間違って申請するケースも多く見られた。

今回、既存の社員マスターから参照する資格と新たに設けた資格ごとの単価マスターにより、正しい単価が自動設定されるようにした。

経理システム（仕訳データ）連動

経理部門のシステムも、同じ IBM i を利用した社内開発システムである。今回の精算システムでは、申請内容に基づく仕訳データが比較的容易に作成でき、IBM i で発番される仕訳データ ID の取得・更新も可能であるため、内容確認のうえで仕訳データ作成と仕訳帳発行を自動的に行えるようにした。

従来は申請者（立替本人）が作成した精算書をもとに、事務スタッフが経理仕訳を入力していたが、この処理は不要となった。また交際費等は 1 人当たりの負担金額により、勘定科目の妥当性が判定できるようになった。

ゴルフ利用税と部門間按分の考慮

仕訳データでは税計算の有無により明細行を分ける必要があるが、ゴルフの場合、領収書内に明記される利用税を二重に税計算しないように内数から差し引いて、それ以外の金額と 2 行分の仕訳データを作る必要がある。

今回、精算システム上で総額と利用税の金額をそれぞれ入力することにより、

図4　業務フロー図

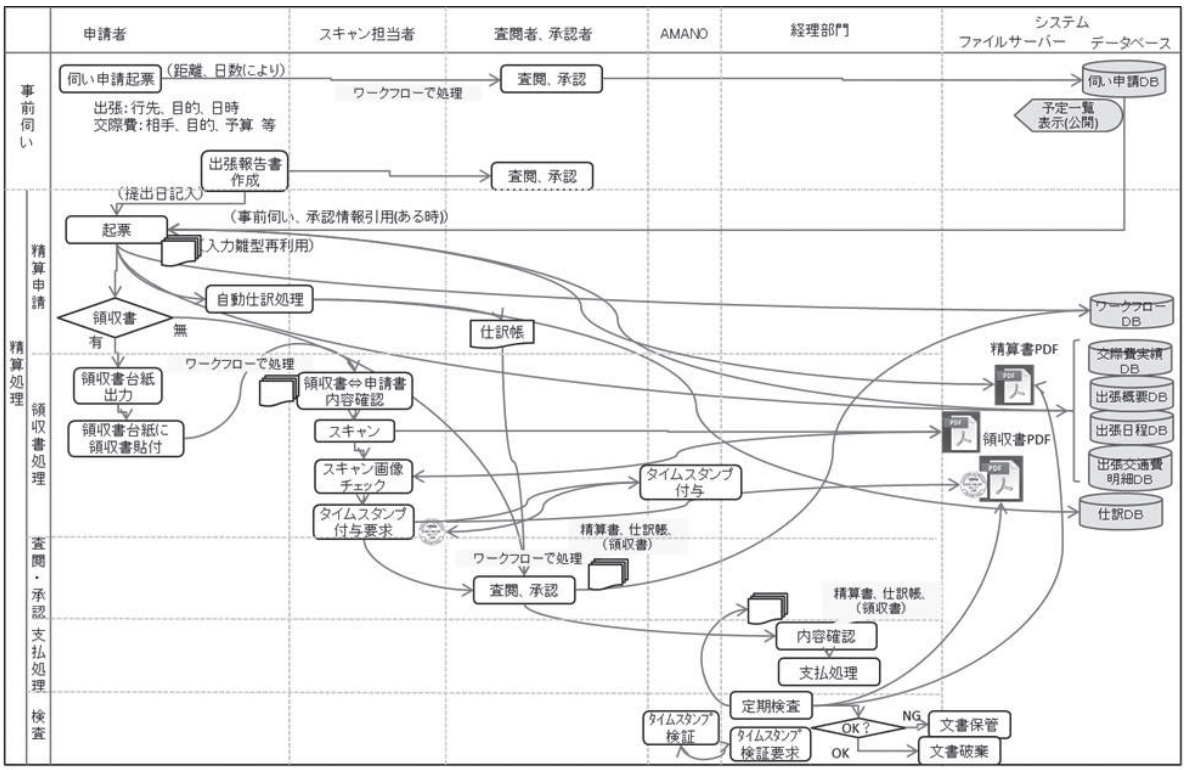


図5　組込み機能一覧(1)

	機能	内容(用途、目的、特徴等)	備考
事前伺い申請関係	承認待ち一覧表示	申請者分の承認待ち(ﾍﾝﾃﾞｨﾝｸﾞ)中の案件	ｵﾌﾟｼｮﾝで承認済分表示可能
	出張申請一覧表示	部門、全社単位の指定期間の出張申請状況	一覧ﾌｫｰﾑ出力可能
	出張伺い申請	出張予定共有、目的の確認、社有車利用申請	控えEXCELﾌｫｰﾑ自動保存
	交際費使用伺い申請	相手先、目的の確認、負担部門配分指定 社有車利用申請	控えEXCELﾌｫｰﾑ自動保存
	査閲、承認依頼通知	次の査閲者または承認者にメールおよび Y-VISの通知機能で確認依頼を通知	自動処理 精算申請時も同様
出張精算申請関係	事前伺い情報連携	事前伺いの登録済み内容を参照、入力補完 （主にヘッダー情報に再利用）	伺い申請のID表示 （申請ﾌｫｰﾑに明記）
	過去出張実績表示	過去の出張旅費精算の履歴一覧表示	精算申請ﾌｫｰﾑ入力用の雛型再利用可
	単価、総額自動設定	・マスター参照による申請者の資格に応じた 日当他の単価を自動ｾｯﾄ ・日当、交通費等の集計も自動	申請者の社員CD設定 （初回のみ）
	駅すぱあと連携	・駅すぱあとのｺﾝﾎﾟｰﾈﾝﾄを組込済み ・運賃の自動取込による信頼性向上、効率化 ・履歴あり駅はﾌﾟﾙﾀﾞｳﾝ可能(入力不要)	
	仕訳処理との連携	・仕訳入力画面へ連携 ・仕訳入力情報の自動作成および仕訳番号 との相互連携情報を保持	仕訳番号情報DB格納 （申請ﾌｫｰﾑにも明記）
	領収書関連機能	・領収書が要求される機関(ﾀｸｼｰ、駐車場等) の管理番号要求 ・スキャン用貼付台紙自動発行(識別用QR含)	領収書ｽｷｬﾝ担当者 指定が要求される QRｺｰﾄﾞはEXCELｱﾄﾞｲﾝ 設定要(初回のみ)
	日帰り、近距離出張対応	・任意の期間にわたって部分入力可能 （一回の処理でまとめて入力しなくて良い）	仮登録で情報ｷｰﾌﾟ （後日追加入力可能）
交際費精算申請関係	事前伺い情報連携	事前伺いの登録済み内容を参照、入力補完 （主にヘッダー情報に再利用）	伺い申請のID表示 （申請ﾌｫｰﾑに明記）
	仕訳処理との連携	・仕訳入力画面へ連携 ・仕訳入力情報の自動作成および仕訳番号 との相互連携情報を保持	仕訳番号情報DB格納 （申請ﾌｫｰﾑにも明記）
	負担部門別仕訳	指示された配分比率で支払金額を按分し、 該当部門数分の仕訳データが作成される	自動処理
	ゴルフ利用税仕訳	ゴルフの場合、自動的に利用税分用の別仕訳 明細を作成	金額入力欄表示
	支払方法別仕訳	立替払い、事後精算　の支払方法ごとの 仕訳の勘定科目自動設定	
申請共通	取下げ、修正機能	本人による取下げ、間違いの修正等は既存 データへのフラグ付加、管理番号修正で退避 の上、訂正情報は新たに発行	元のデータは削除されない 管理番号の関連性確認可能

自動的に2行の仕訳データを作成するようにした。

また、費用の目的によっては複数の部門で按分することがある。その場合は該当部門ごとに仕訳データを作成していたが、按分対象の部門と按分比率を指定することで、必要部門分の仕訳データを自動作成するようにした。

過去履歴コピー、事前申請連携機能による入力容易化

出張では固定の取引先や例会など、同じルート・交通機関を利用して定期的に訪問するケースも多い。したがって過去履歴の参照・コピー機能により、日付等の最低限の修正を行うだけで申請手続きが行えるようにした。

また､交際費の使用および宿泊（日当）を伴う出張では事前申請を義務づけているが、精算時は事前申請内容の再利用による補助入力を可能とし、精算分に対する事前申請情報が関連して確認できるようにした。

税務署への申請について

税務署へ電子帳簿保存法スキャナ保存制度の適用申請を行うには、適正な運用体制の整備とシステム構築が求められる。システム構築要件を大別すると､「真実性」と「可視性」を満足させる必要があり、それぞれについてさらに細かい要件が規定されている。【図1】にその概要を示すが、これらを担保するシステムの実装、運用ルール、マニュアル、規程の整備等が必要である。

所定フォーマットでの申請書のほかに、要求されている要件を満たしていると確認できる添付書類の提出および国税の担当者（技術官）への説明が求められる。当社で用意した申請書類の一覧を下記に記す。今回はさらに各要件に対する添付資料の説明箇所と対応方法をまとめた一覧表【図28】を準備し、提出した。

・スキャナ保存の承認申請書（控え）
・旅費・交際費精算システム設計仕様書
・旅費・交際費精算システム操作手順書
・業務フロー図
・タイムスタンプサービス契約手続き書類
・旅費・交際費精算にかかる事務処理規程
・スキャナによる電子化文書保存規定
・業務チェックシート
・スキャナ保存の事務処理にかかる検査報告書（兼対策書）
・職責簿

最後に

今回の開発に際しては、単に業務の利便性向上を目的とするだけではなく、電子帳簿保存法からの法的要件も満足させる必要があったが、当初は法律の解釈、運用ルールの決め方などがわからず、手探り状態であった。

同業者等でも参考にする事例がなかったため、富士ゼロックスに相談のうえ、税務署に申請するまでに数回のコンサルタントを依頼し､承認に繋げられた。

また、袖山喜久造氏の『改正電子帳簿保存法完全ガイド』と、ビジネス機械・情報システム産業協会 ドキュメントマネージメントシステム部会発行の解説書（※1）を参考にした。

ミガロ.からの技術サポートで実現した機能も多く、各関係者に謝意を表したい。

※1
『電子帳簿保存法スキャナ保存制度解説と検討の手引き～国税関係書類の電磁的記録の保存に向けて～』(別冊を含む)

M

図6　組込み機能一覧(2)

機能		内容(用途、目的、特徴等)	備考
確認、査閲・承認関係	承認・確認待ち一覧	現時点で照会者の処理待ち状態の案件を表示	オプションで承認済分表示可能
	申請書フォーム、領収書確認画面	照会者の処理待ち状態の案件の申請書フォームおよび領収書イメージ(いずれもPDF)を表示	拡大・縮小、印刷可能
	査閲・承認機能	上記で内容確認し、了承又はNG(保留)を指示	控えEXCELフォーム自動保存
	ワークフロー連動	次の査閲者または承認者にメールおよびY-VISの通知機能で確認依頼を通知 (処理が最終承認の場合は本人に通知発信)	自動処理
スキャン画像確認・タイムスタンプ処理	スキャン処理待ち一覧	現時点でスキャン担当者(該当分)の処理待ち状態の案件一覧を表示	オプションで処理済分表示可能
	スキャン済み画像表示	上記リスト選択行分のスキャン画像表示 (査閲・承認画面)	複合機でのスキャンし、数10秒後確認可能
	画像チェック結果登録	表示されたスキャン画像の内容確認し、確認した事項の結果を保存	全てOKであればタイムスタンプボタン押下可能となる
	タイムスタンプ処理	表示されたスキャン画像の妥当性を確認後に該当PDFにタイムスタンプを付与 (成功すると付与済みPDF画像に表示更新)	処理失敗時はエラーメッセージ表示 → 処理中止 成功で次の確認者へ通知発信
タイムスタンプ一括検証	領収書あり案件表示	・領収書含む精算案件一覧表示 ・期間、金額、部門、支払者等の複合条件検索 任意項目でのフィルター、並べ替え可能 ・修正、取下げ前の履歴も確認可能	領収書画像表示も連動
	タイムスタンプ一括検証	・一覧リスト表示分の該当PDFを一括検証 ・検証結果のリスト表示 ・検証結果のログ表示(CSV) ・検査履歴のDB登録 ・入力期間等の妥当性判定	タイムスタンプ処理後の内容訂正履歴確認可能

図7　タイムスタンプ発行および結果取得方法

```
procedure TForm1.ScanOKBtnClick(Sender: TObject);
var
 LCmdLine : TplCommandRedirect;
 FumPDF,OriPDF,TSPDF,cmdText,LCmdStr   : String;
 LStrText : TStrings;
 EChk:boolean;
begin
  TISTcmdMemo.lines.clear;
  TISTresMemo.Clear;
  TISTresMemo.Update;
  LCmdLine  := TplCommandRedirect.Create;
  LCmdStr   := 'cmd.exe  /k';
  Screen.Cursor := crHourGlass;

  try
    OriPDF:='¥¥10.10.10.17¥datas¥RSEISAN¥ST'+mainCDS[305].Fieldvalues['DWF001']+'.pdf';//入力ファイル
    TSPDF:='¥¥10.10.10.17¥datas¥RSEISAN¥TSFIN¥ST'+mainCDS[305].Fieldvalues['DWF001']+'.pdf';//出力ファイル

    if PCBit=64 then //PCが32Bit、64Bitで実行ファイルを使い分ける必要有り
    cmdText:='¥¥10.10.10.17¥C¥AMANO_3161pdf¥pdftimestamp  '+OriPDF+' '+TSPDF+'  /C /LI /L
¥¥10.10.10.17¥C¥AMANO_3161pdf¥A207470001000000010001TL00.atl /P SP6ZKN'
     else
     cmdText:='¥¥10.10.10.17¥C¥AMANO_3161pdf_32¥pdftimestamp  '+OriPDF+' '+TSPDF+'  /C /LI /L
¥¥10.10.10.17¥C¥AMANO_3161pdf¥A207470001000000010001TL00.atl /P SP6ZKN' ;

    TISTcmdMemo.Lines.Add(cmdText);
    TISTcmdMemo.Lines.Add('');

    LStrText    := TISTcmdMemo.Lines;
    TISTresMemo.Text  := LCmdLine.GrabStdOutText(LCmdStr,  LStrText);
  finally
    FreeAndNil(LCmdLine);
  end;
  Screen.Cursor := crDefault;
```

図8　パソコンのBit数を確認する方法

```
//------------------------------------------------------------------------------
// 32ビットのWindowsか64ビットのWindowsかを調べる関数
// 64ビット版のWindowsの場合はTrueを返す
//------------------------------------------------------------------------------
function Is64bitWindows: Boolean;
var
  Wow64Proc : function(hProcess: THandle; var Wow64: BOOL): BOOL stdcall;
  RetFlag   : LongBool;
begin
  @Wow64Proc := GetProcAddress(GetModuleHandle('Kernel32.dll'), 'IsWow64Process');
  if @Wow64Proc <> nil then begin
    Wow64Proc(GetCurrentProcess, RetFlag);

    if SizeOf(THandle) = 4 then begin
      Result := RetFlag;
    end else
    if SizeOf(THandle) = 8 then begin
      Result := True;
    end;
  end else begin
    //Windows XPの64ビット版か
    Result := CheckWin32Version(5, 2);
  end;
end;
```

図9　ExcelにQRコードをオブジェクトとして挿入（ExcelのVBA）

```
'GoogleAPIでQRコードを作成
        Set qr = ActiveSheet.Pictures _
              .Insert("http://chart.apis.google.com/chart?cht=qr&chs=140x140&chl=" _
                    + ActiveSheet.Range("K1").Value)

        'QRコードの表示位置を指定
        With qr
           .Top = ActiveSheet.Range("I1").Top  + 2
           .Left = ActiveSheet.Range("I1").Left  + 2
        End With
```

図10　Excelのバージョン判定方法

```
EXCELのバージョン判定
procedure TForm1.XLSVerChkBtnClick(Sender:  TObject);
var
  Excel : Variant;
  val   : Variant;
  ver :string;
begin
  Excel := CreateOleObject( 'Excel.Application' );
  ver:=Excel.Application.Version ;
  Excel.Application.quit;
end;
```

図11　Excelのバージョン値

Officeﾊﾞｰｼﾞｮﾝ	値
Office 95	7
Office 97	8
Office 2000	9
Office 2002	10
Office 2003	11
Office 2007	12
Office 2010	14
Office 2013	15
Office 2016	16
Office2019	16

図12　XEROXスキャン機能およびApeosFlowService機能

機能	内容(用途、目的、特徴等)	備考
XEROX 指定条件スキャン	解像度、階調　等を事前にパターン登録 （都度設定を不要化） 複合機の指定BOX(メモリ領域)に一時保存される	ジョブメモリー機能
ApeosFlowService スキャンデータ取込、PDF保存	各スキャナー(複合機)よりスキャン画像を 取込み、PDF変換、QRコードよりファイル名設定 のうえサーバーの指定フォルダーに保存	常駐サービスで一定 間隔でBox監視
スキャン条件、実績履歴保存	スキャン条件、日時等の履歴情報をCSV出力	Y-VISでCSV→DB登録

図13　24Bitカラーでの読み取り確認方法

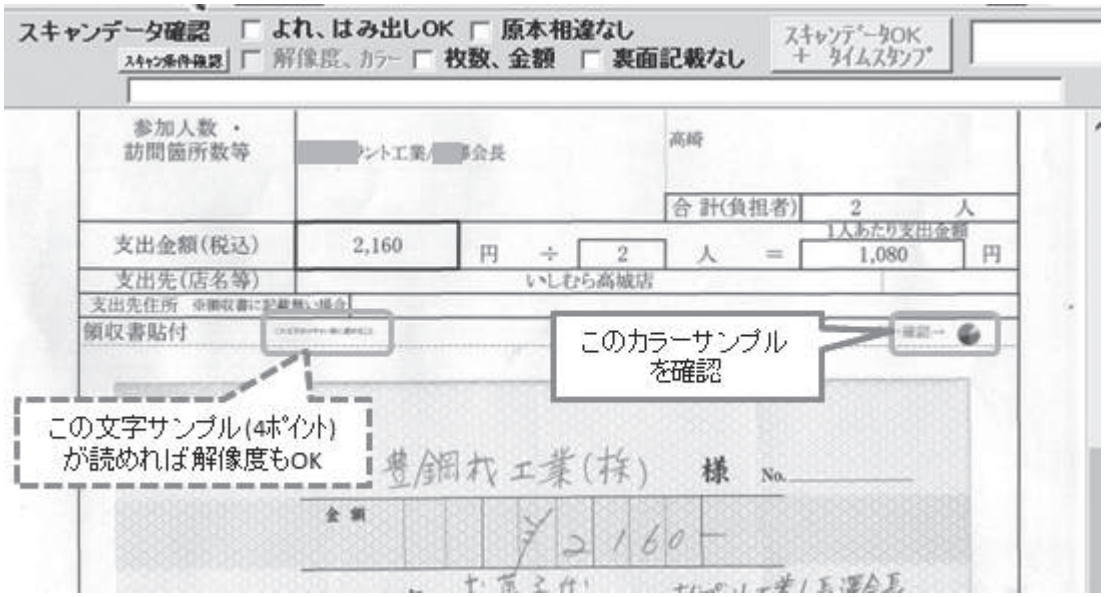


図14　カラープリンタからの強制印刷方法(Excel VBA)

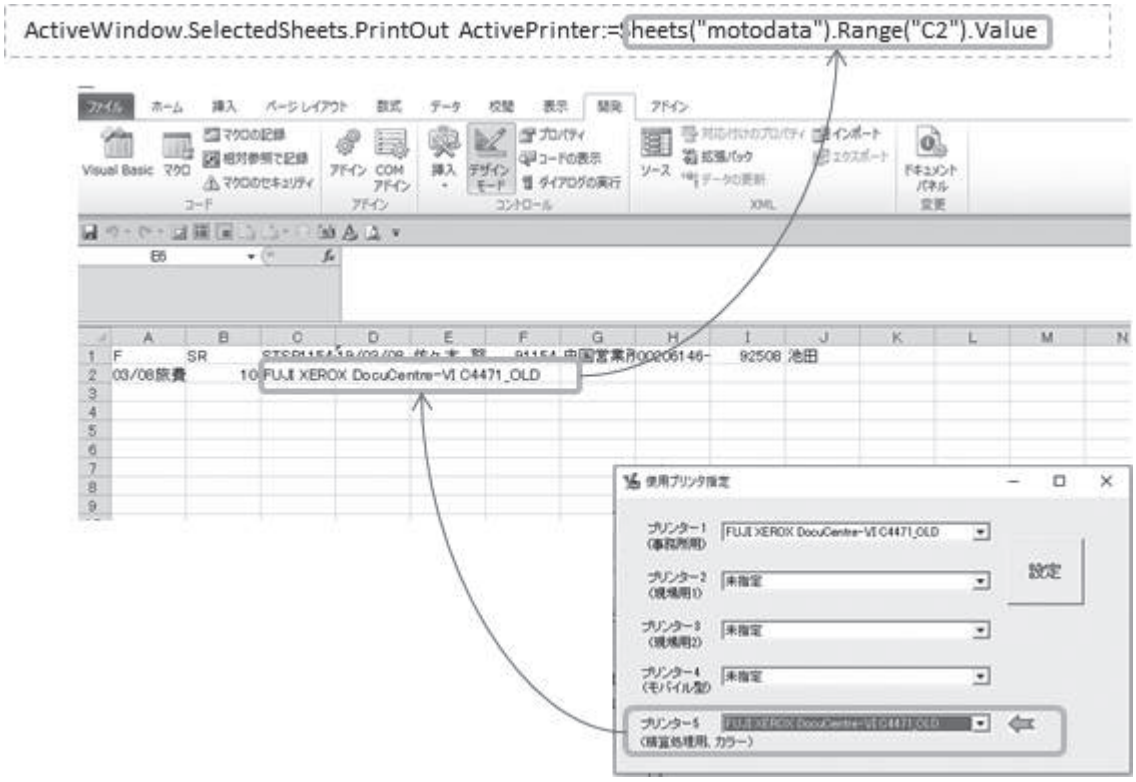


図15　駅すぱあと情報取込みフォーム

図16　駅すぱあと情報取込み明細(例)

取込み結果

一括 →

月	日	曜	機関	出発地	到着地	内容	距離km	金額	領収書	駅す
4	19	木	バス−鉄道	宇美	博多	TOTAL	0	640	0	使用

区間ごと →

月	日	曜	機関	出発地	到着地	内容	距離km	金額	領収書	駅す
4	19	木	バス	JR宇美	福岡空港		0	380	0	使用
4	19	木	鉄道	福岡空港	博多		0	260	0	使用

駅すぱあと
による情報である
証拠(修正不可)

図17　出発地、到着地の入力省略(履歴格納)

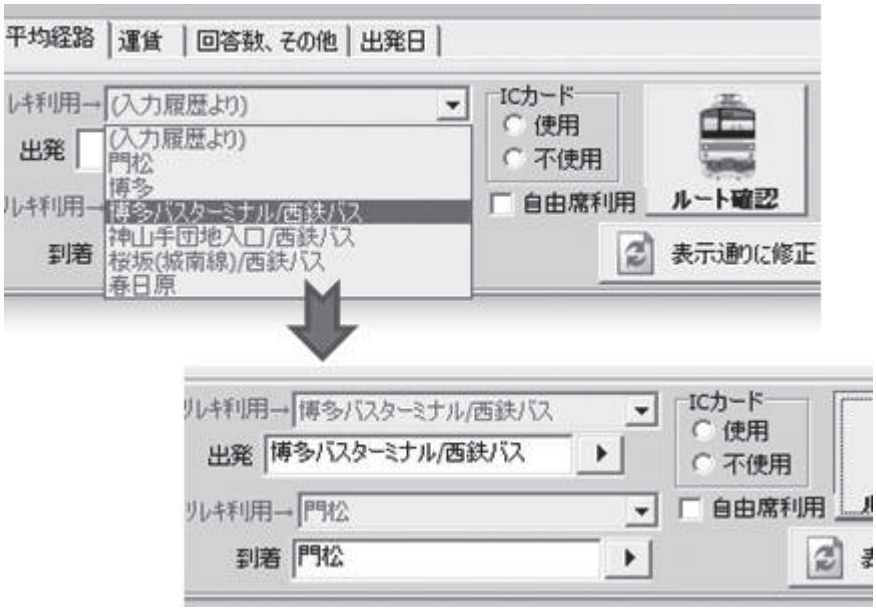


図18　ワークフロー査閲・承認画面

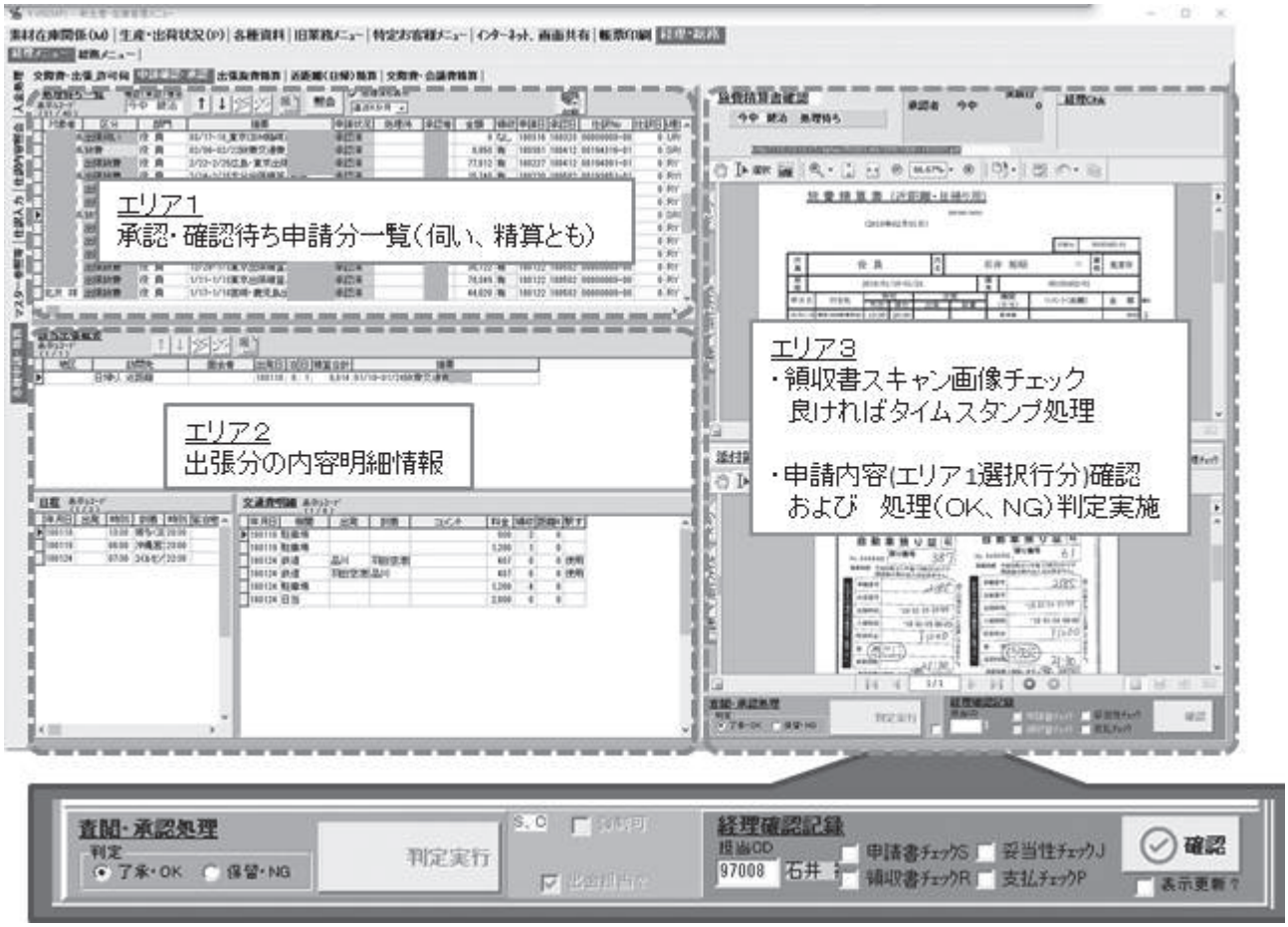


図19　新規作成データベース一覧

追加済みデータベース	主な項目	概要
役職・旅費規程マスター	職掌、級、資格名、日当、宿泊料	役職ごとの出張旅費に関する日当などの支給基準
出張・交際費許可申請DB	申請者ｺｰﾄﾞ、訪問先、日程、目的、予算、承認ルート、処理ID	出張旅費、交際費使用に関する内容の事前申請情報および承認予定・履歴
出張概要メインDB	処理ID、申請者ｺｰﾄﾞ、訪問先、期間、費用合計、仕訳No	1件の出張精算の全体概要
出張日程DB	処理ID、社員ｺｰﾄﾞ、出発地、到着地、移動年月日	1件の出張に関する行き、帰り、滞在日程明細
出張交通費明細	処理ID、社員ｺｰﾄﾞ、交通機関、支払日、料金、領収書No、乗車区間	1件の出張に関する使用交通機関と支払料金の明細
交際費DB	処理ID、社員ｺｰﾄﾞ、負担部門、接待相手、参加人数、相手との関係、支払先、目的、支払額	1件(支払先ごと)の交際費の使用状況
スキャン実績条件DB	読取日時、処理ID、解像度、用紙サイズ、スキャナ名、IPアドレス、階調	スキャン画像に関する読取条件履歴
ワークフローメインDB	処理ID、社員ｺｰﾄﾞ、承認ルート、承認実績日、検査実績日、事前申請ID、仕訳No	各種申請の承認ルートおよび実績日、進捗状況に関する全体経過確認用
スマホ用ワークフローDB	処理ID、社員ｺｰﾄﾞ、連絡先社員ｺｰﾄﾞ、送信社員ｺｰﾄﾞ、摘要、送付先PC、ｽﾃｰﾀｽ	スマホから通知のために中間ﾌｧｲﾙとして作成

図20　ワークフローメッセージ通知

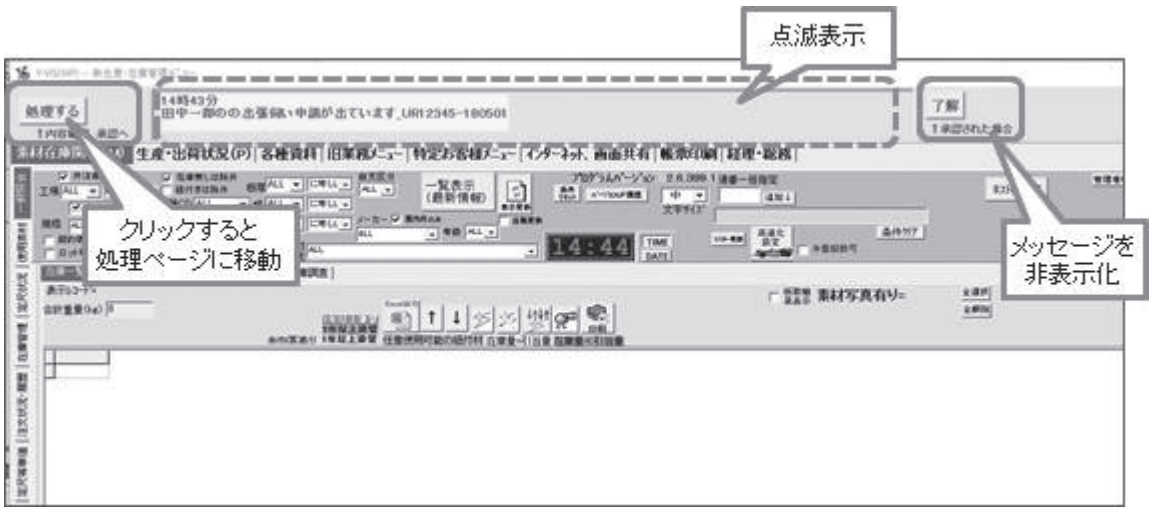
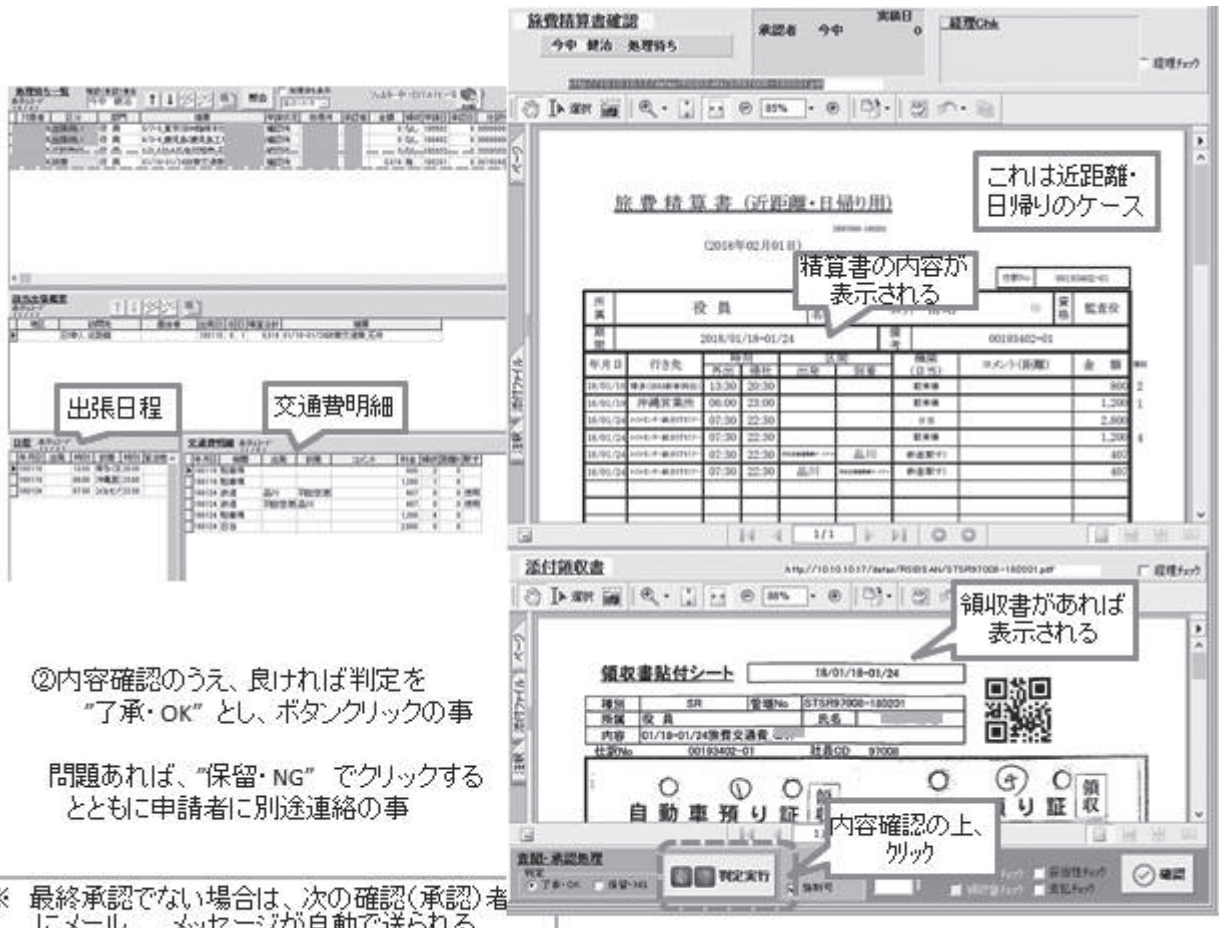


図21　ワークフロー査閲・承認画面（PDF表示例）



②内容確認のうえ、良ければ判定を
"了承・OK" とし、ボタンクリックの事

問題あれば、"保留・NG" でクリックする
とともに申請者に別途連絡の事

※　最終承認でない場合は、次の確認（承認）者にメール、メッセージが自動で送られる

最終承認の場合は申請者に承認済みのメールとメッセージが送られる

図22　メッセージ通知処理（死活チェック、ファイル送付）

```
ToPC:=Trim(multiSQLQ.fieldvalues['DJS006']);
ToIP:=multiSQLQ.fieldvalues['IP'];

if SL<>Nil then FreeAndNil(SL);
SL:=TStringList.create;
SL.add(MSG);

PingRes:=(-1);
try
PingRes:=SendPing(ToIP, pr) ;
except
PingRes:=(-1);
end;

msgFLS:=false;

 if (PingRes = 0) then //pingが通る場合
 begin  //無ければ作成
  if FileExists('¥¥'+ToIP+'¥c¥DELPHIアプリ¥FLUGWF¥WFnote.txt')=false then
  begin
   try
   SL.SaveToFile('¥¥'+ToIP+'¥c¥DELPHIアプリ¥FLUGWF¥WFnote.txt');
   except
    SL.SaveToFile('¥¥Yutext¥wkcom1¥DELPHI-HAIHU¥FLUGWF¥'+ToPC+'_WFnote.txt');
     if SL<>Nil then
     FreeAndNil(SL);
    end;
  end
  else
  begin //有れば追記
   try
   ToSL:=TStringList.create;
   ToSL.LoadFromFile('¥¥'+ToIP+'¥c¥DELPHIアプリ¥FLUGWF¥WFnote.txt');
   ToSL.add(MSG);
   ToSL.saveToFile('¥¥'+ToIP+'¥c¥DELPHIアプリ¥FLUGWF¥WFnote.txt');
   except
    if ToSL<>Nil then
    FreeAndNil(ToSL);
    msgFLS:=true;
   end;
  end;

  SmahoMSGBtn.hint:=ToPC;  //スマホ用メッセージ履歴用（指定端末でのみ利用）
 end
 else
 begin   //pingが通らない場合
  if FileExists('¥¥Yutext¥wkcom1¥DELPHI-HAIHU¥FLUGWF¥'+ToPC+'_WFnote.txt')=false  then
  SL.SaveToFile('¥¥Yutext¥wkcom1¥DELPHI-HAIHU¥FLUGWF¥'+ToPC+'_WFnote.txt')
   else
   begin
   ToSL:=TStringList.create;
   ToSL.LoadFromFile('¥¥Yutext¥wkcom1¥DELPHI-HAIHU¥FLUGWF¥'+ToPC+'_WFnote.txt');
   ToSL.add(MSG);
   ToSL.saveToFile('¥¥Yutext¥wkcom1¥DELPHI-HAIHU¥FLUGWF¥'+ToPC+'_WFnote.txt');
   end;
  SmahoMSGBtn.hint:='S';    //スマホ用メッセージ履歴用（指定端末でのみ利用）
 end;

 if SL<>Nil then
 FreeAndNil(SL);
 if ToSL<>Nil then
 FreeAndNil(ToSL);

 if msgFLS=true  then
 begin  //再チャレンジ
  ToSL:=TStringList.create;
  ToSL.LoadFromFile('¥¥Yutext¥wkcom1¥DELPHI-HAIHU¥FLUGWF¥'+ToPC+'_WFnote.txt');
  ToSL.add(MSG);
  ToSL.saveToFile('¥¥Yutext¥wkcom1¥DELPHI-HAIHU¥FLUGWF¥'+ToPC+'_WFnote.txt');
 end;

 if ToSL<>Nil then
 FreeAndNil(ToSL);
```

図23　Ping処理（死活チェック）

```
function SendPing(address: string; var pr: TPingResult): integer;
var
  Locator:  ISWbemLocator;
  Services: ISWbemServices;
  SObjSet:  ISWbemObjectSet;
  SObject:  ISWbemObject;
  Enum:     IEnumVariant;
  TempObj:  OleVariant;
  TempVal: Cardinal;
  Query: string;
begin
  Result := -1;

  if Failed(CoInitialize(nil)) then Exit;
  try
    Locator := CoSWbemLocator.Create;
    Services := Locator.ConnectServer('.', 'root¥cimv2', '', '', '', 0, nil);
    Query := 'SELECT  * FROM Win32_PingStatus WHERE address=' + QuotedStr(address);
    SObjSet := Services.ExecQuery(Query, 'WQL', wbemFlagReturnImmediately and wbemFlagForwardOnly, nil);
    TempVal := 0;
    Enum := (SObjSet._NewEnum)  as IEnumVariant;
    if (Succeeded(Enum.Next(1,  TempObj, TempVal)) and (TempVal > 0)) then
    begin
      try
        SObject := IUnknown(TempObj) as ISWBemObject;
        if (SObject <> nil) then
        begin
          if VarIsNull(SObject.Properties_.Item('StatusCode', 0).Get_Value) then exit;

          Result := (SObject.Properties_.Item('StatusCode', 0).Get_Value);

          if (Result = 0) then
          begin
            with SObject.Properties_ do
            begin
              pr.ProtocolAddress := VarToStr(Item('ProtocolAddress', 0).Get_Value);
              pr.BufferSize := StrToIntDef(VarToStr(Item('BufferSize',  0).Get_Value), 0);
              pr.ResponseTime := StrToIntDef(VarToStr(Item('ResponseTime',  0).Get_Value), 0);
              pr.ResponseTimeToLive := StrToIntDef(VarToStr(Item('ResponseTimeToLive',  0).Get_Value), 0);
            end;
          end;
        end;
      finally
        SObject := nil;
        VarClear(TempObj);
      end;
    end;
  finally
    Enum.Reset;
    SObjSet := nil;
    Services := nil;
    Locator := nil;
  end;
end;
```

図24　ワークフローメッセージ通知処理内容

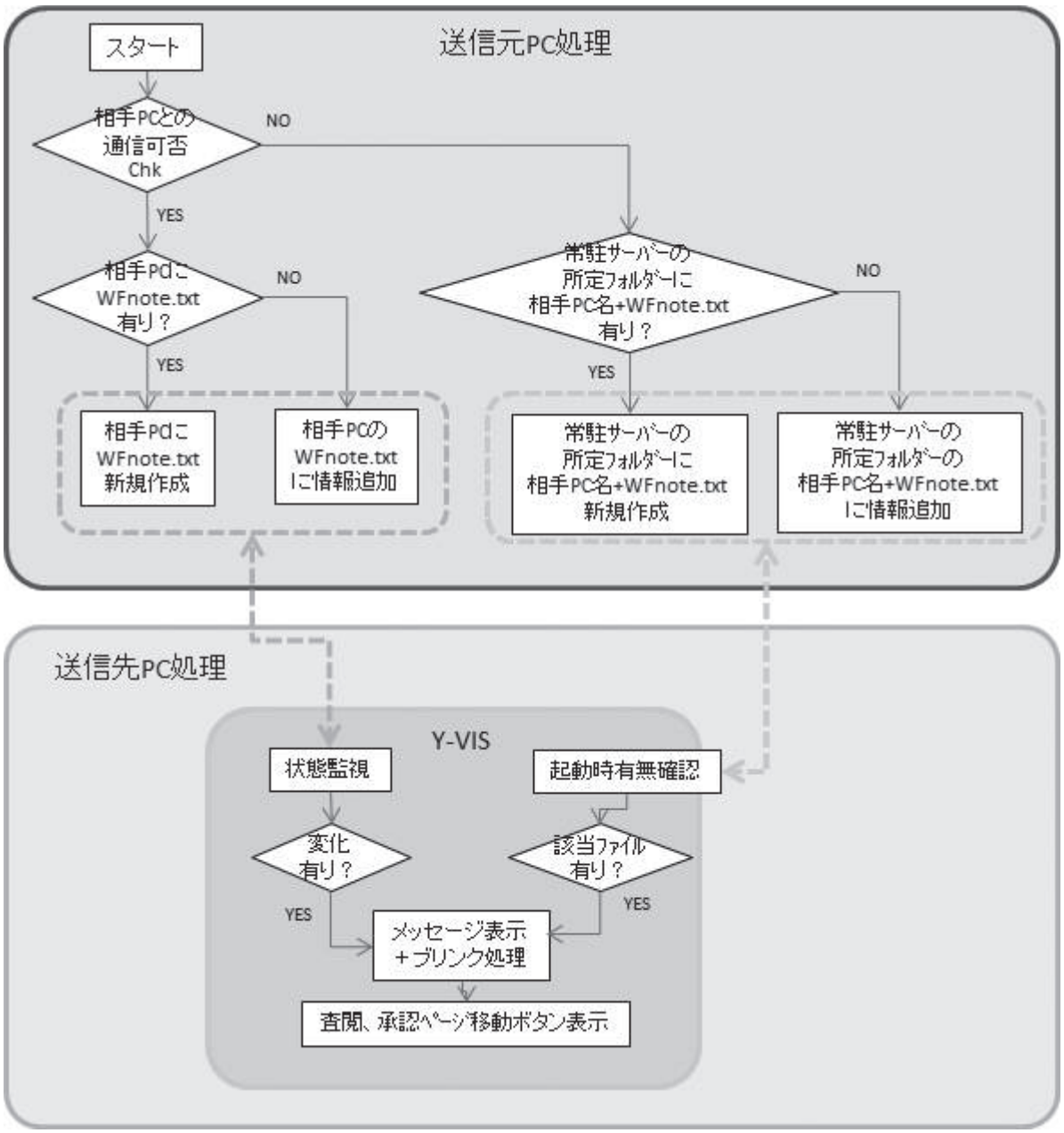


図25　Androidスマホでのワークフロー機能

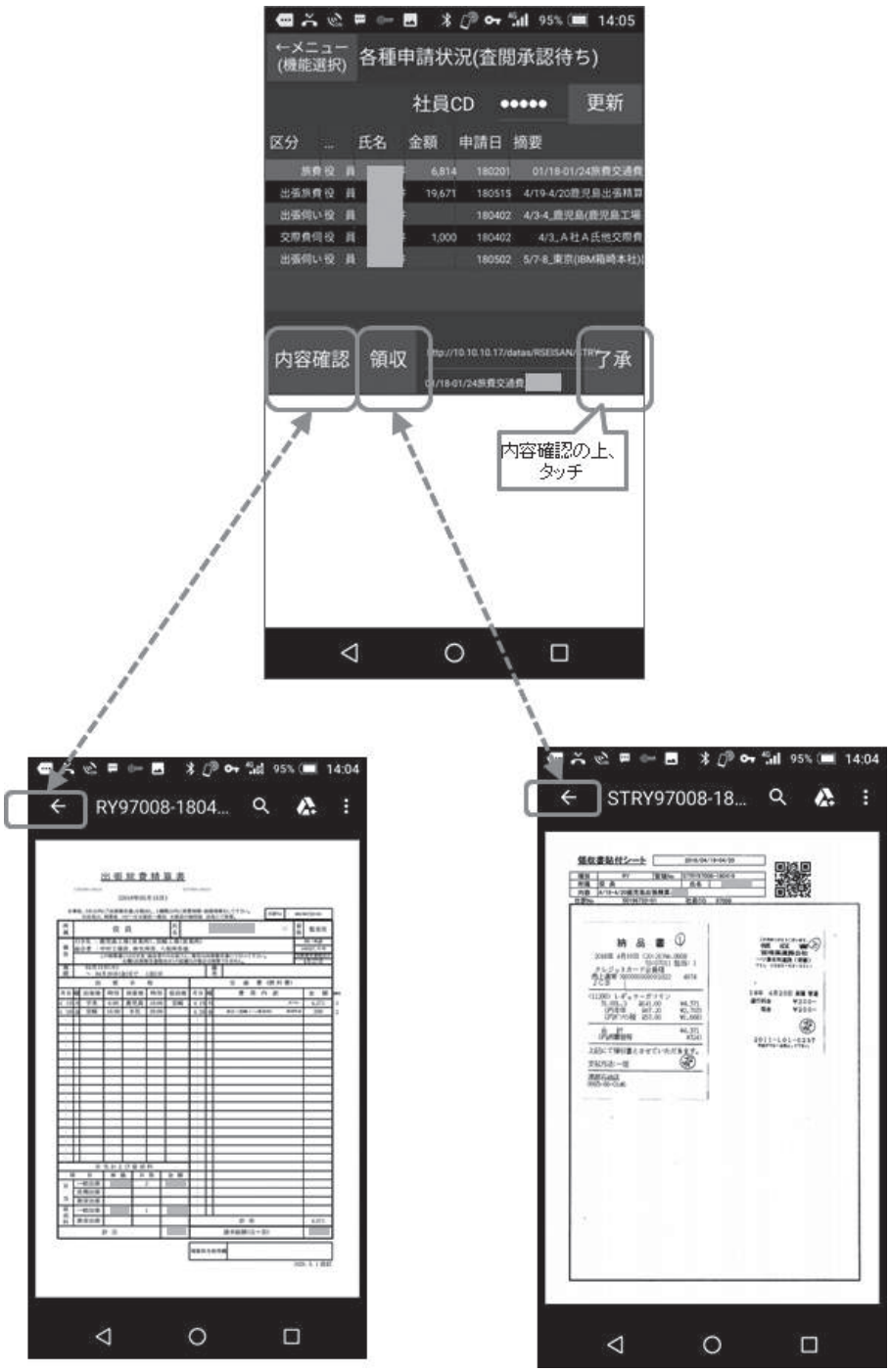


図26　AndroidでのPDF表示方法

```
procedure TForm2.WFRecChkBTNClick(Sender: TObject);
var
 URLStr:string;
 fName      : String;
 Intent     : JIntent;
 URI        : Jnet_Uri;
begin
 URLStr:='http://10.10.10.17/datas/RSEISAN/ST'+WorkFCDS.FieldValues['DWF001']+'.pdf';

   {$IFDEF ANDROID}
    URI := TJnet_Uri.JavaClass.parse(StringToJString(URLStr));
    intent := TJIntent.Create;
    intent.setAction(TJIntent.JavaClass.ACTION_VIEW);
    intent.setDataAndType(URI,StringToJString('application/pdf'));
    SharedActivity.startActivity(intent);
   {$ENDIF}
end;
```

図27　メモリ機能によるスキャン条件一定化

図28　スキャナ保存制度要件対応の要約説明資料(抜粋)

区分	#	スキャナ保存要件対応一覧	要件概要	システム対応説明	システム補足説明	該当ページ※
真実性の確保	1	入力方式(入力期間の制限)	速やか(1週間)、業務サイクル(Max1カ月と1週間)、連時(国税一般書類)	「スキャナによる電子化文書保存規定」にスキャン入力期間について記載するとともに、定期検査で入力日時の妥当性についての検証判定をおこなう。	・入力されたデータの保管先(本社サーバー室/データ保存サーバー) ・法定期間中、付属するHDDに定時自動バックアップし、サーバー室は室員不在時施錠 ・定期検査のタイミング:帳簿点検時に実施 ・定期検査実施者:財務経理審査室長が指名したもの	システム操作手順書C-4、F-2(4項)
	2	対象書類	国税庁長官が定める資金や物の流れに直結する重要書類 帳簿、決算書を除く国税関係書類、帳簿代用書類 国税関係書類の電磁的記録によるスキャナ保存の承認された開始日以降に作成又は受領した書類 契約書、納品書、送り状、請求書、領収書等及びこれらの写し スキャナ保存対象を「3万円未満の契約書・領収書及びこれらの写し」に限る規制は撤廃	対象書類は領収書であり重要度・高に分類される	今回は旅費交通費、交際費、会議費の領収書を対象とする 対象は本社地区(篠栗)にある全部署	
	3	一定水準以上の解像度及びカラー画像による読み取り	1)原稿台と一体となったスキャナを使用すること(H28年緩和により撤廃) 2)解像度が200dpi相当以上であること 3)赤色、緑色及び青色の階調がそれぞれ256階調以上(24ビットカラー)であること	規定の手順により要件を満たし、スキャンの実績条件をデータ保存する。	・スキャナ機の設定を固定 (モード:フルカラー、解像度300dpi、サイズ:A4) 上記設定はパスワードにて保護 (ゼロックス/サービス設定) ・法令要件を満たさないファイルの場合、警告メッセージを表示(一括検証機能) ・スキャン1枚の4ポイント文字、カラー見本での確認を手順化 ・領収書枚数の確認もチェック対象とする	システム操作手順書C-14、C-16、F2 システム設計書P5、P20
	4	タイムスタンプ	一の入力単位ごとに一般財団法人日本データ通信協会認定のタイムスタンプを付すこと	AMANOのタイムスタンプを使用、スキャン担当者がスキャン画像を原本照合し、問題ないことを確認したうえで付加処理を行う。 付加処理はシステムより対象PDF画像へのタイムスタンプ付加のコマンドを実行することで実現。 タイムスタンプ印影は画面確認できる。 タイムスタンプ付加処理はワークフローと連動し、次の査閲(承認)者に回る。	・e-timing EVIDENCE 3161 PDF Lib-W ・タイムスタンプの検証は一括でも単体でも可能 ・検証処理予想:年間最大2000枚程度、 ・検証の運用:最低1回/年で経理による部署毎の帳簿点検時に実施予定 ・一括検証時のソフトウェア負荷予想、1~2秒/枚程度	システム操作手順書C-15、C-16 システム設計書P21
	5	読取情報の保存	スキャナで読み取った際の解像度、階調及び当該国税関係書類の大きさに関する情報を保持すること	読取時の条件実績はゼロックスのApeosFlowServiceによりCSV形式で出力されるデータをデータベース格納する。	・「スキャンファイル一括チェック」機能 ・「個別確認・検証」機能 ・ApeosFlowService:スキャナーのスキャン画像を取込みPDF変換、ファイル名指定、指定フォルダーへの保管、読取時条件のCSV出力を行う ・法令要件を満たさないファイルの場合、警告メッセージを表示 (一括検証で解像度、階調、サイズ情報の妥当性、タイムスタンプ付与時期、改変の有無が確認される)	システム操作手順書F-2、F-3 システム設計書P5、P10、P20

図29　参考 事前申請および(申請者用)承認状況確認画面

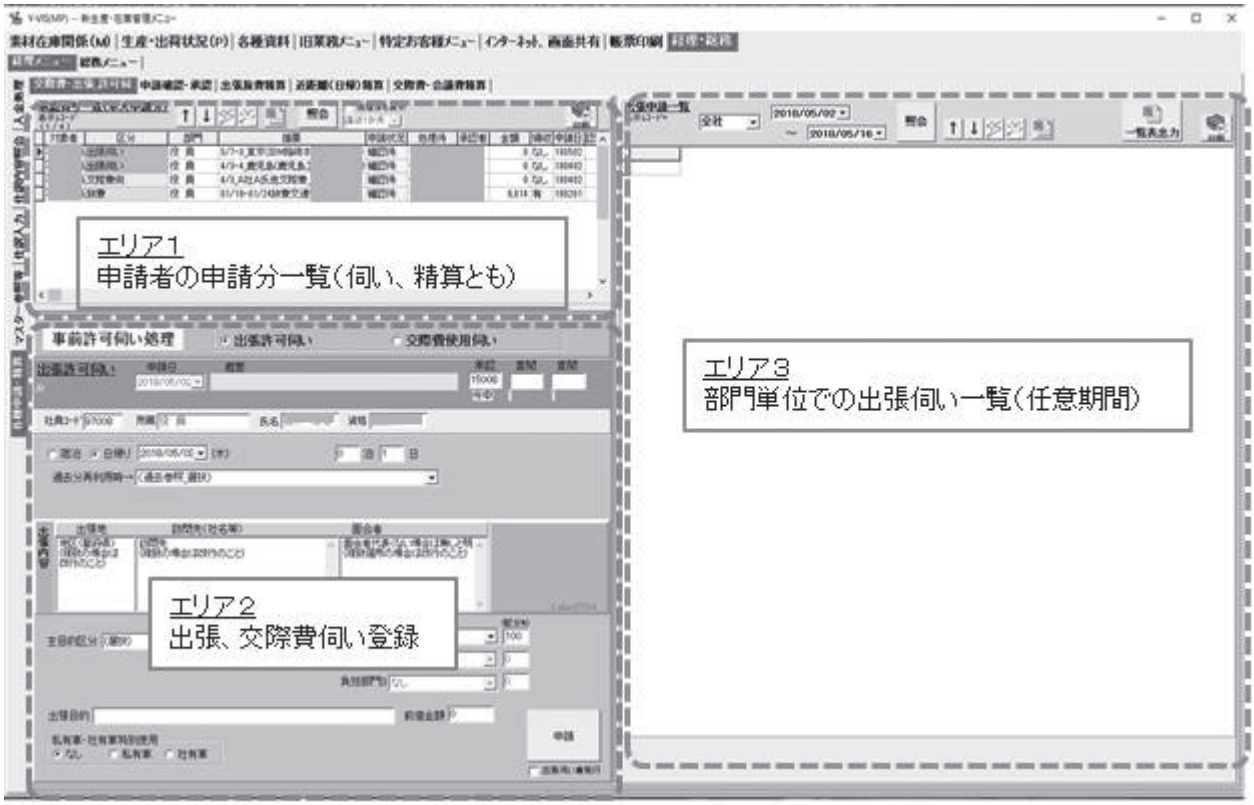


図30　参考 出張旅費精算申請画面

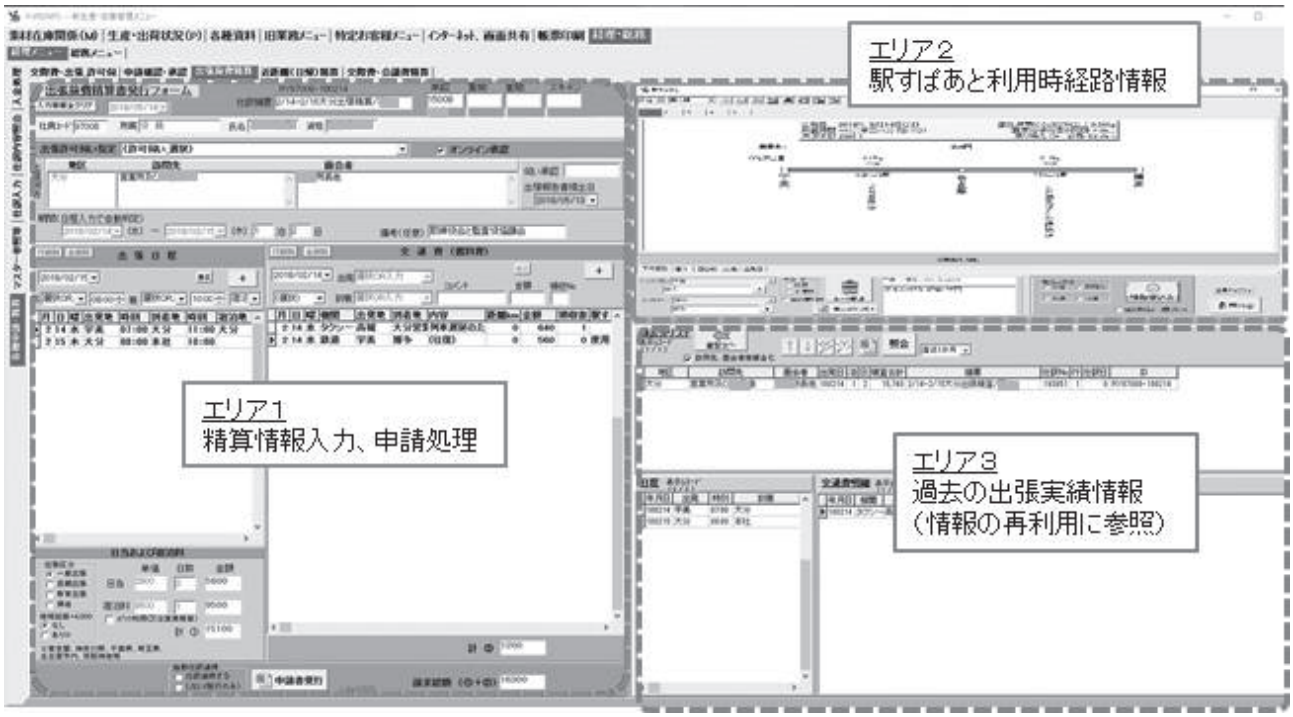


図31　参考 交際費精算申請画面

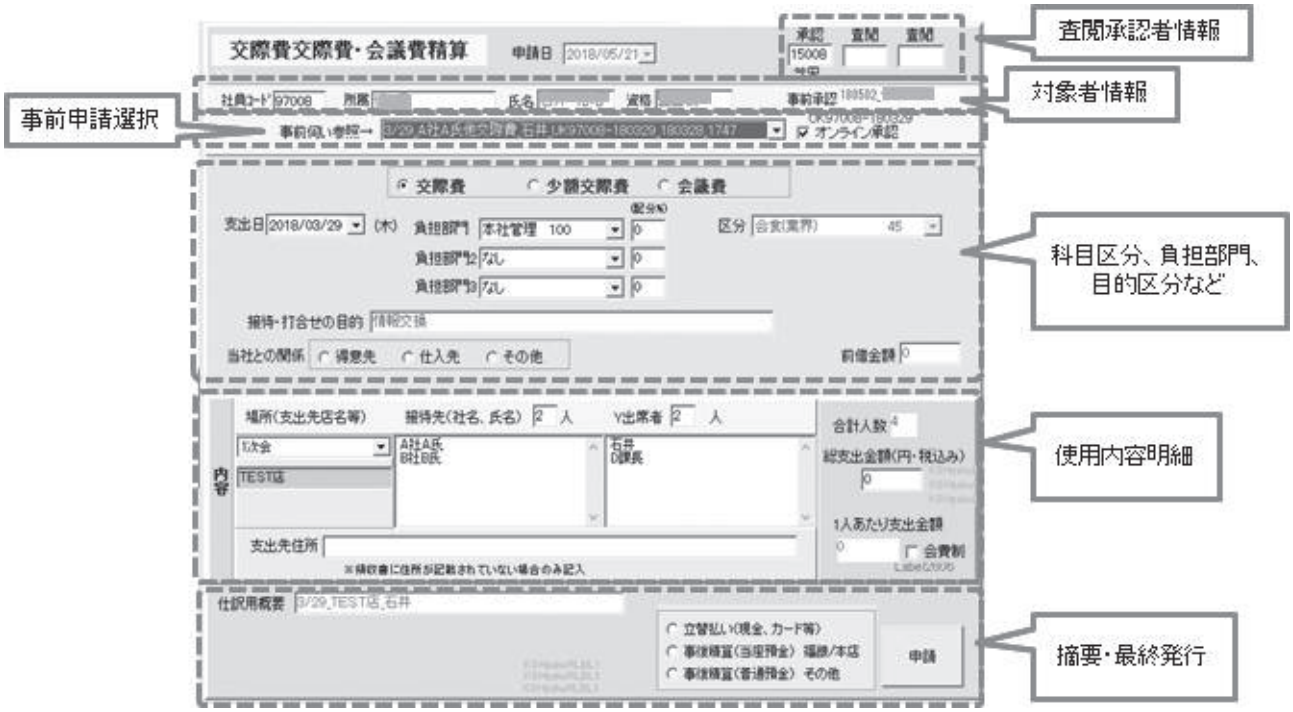
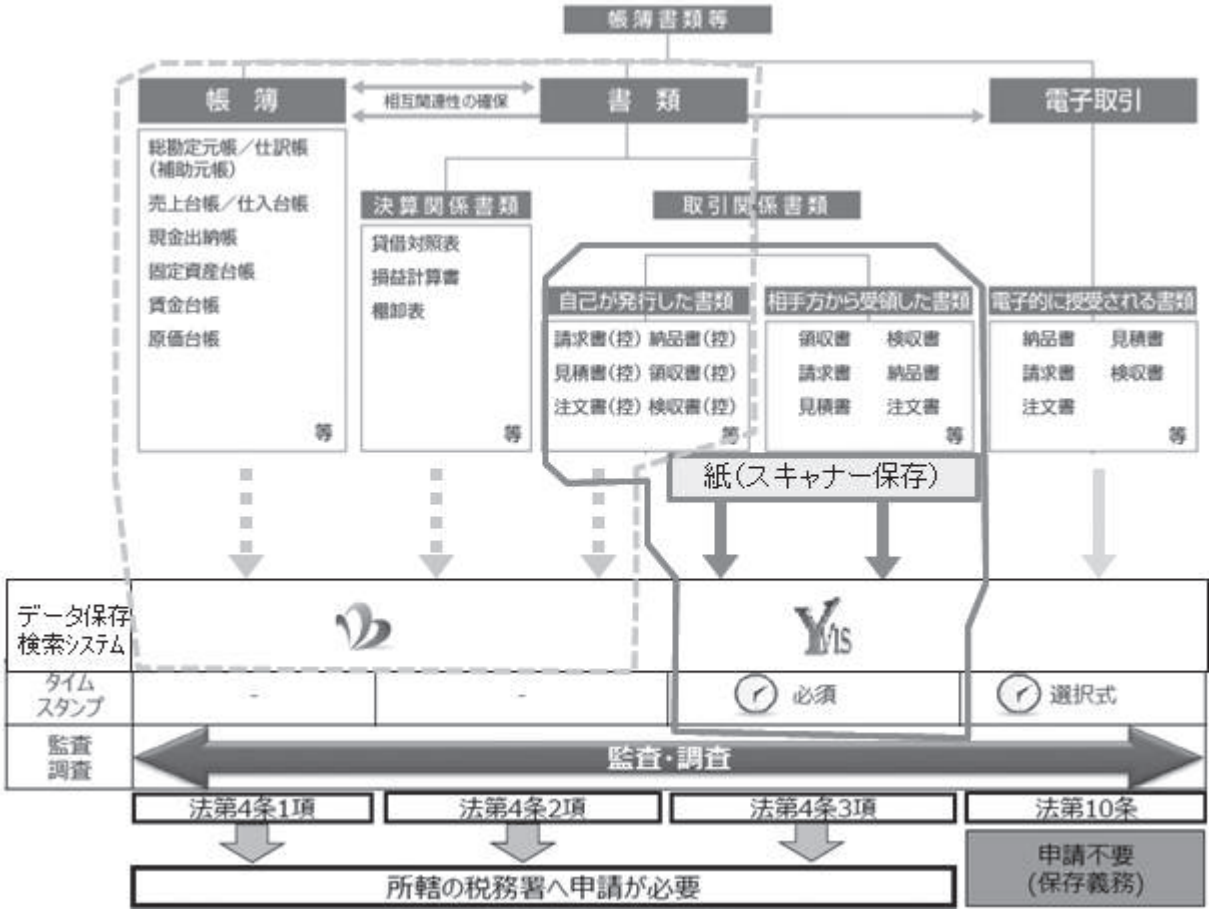


図32　参考 今回開発システムの運用対象書類



出荷業務従事者のモチベーションアップ大作戦
—Delphi/400で基幹データの見える化

池田 純子 様
錦城護謨株式会社
システム部
課長

錦城護謨株式会社
http://www.kinjogomu.jp/

工業用ゴム・樹脂製品の製造・販売を主力事業とし今年創業 83 年を迎える。「オンリーワン技術の開発」をモットーに、材料開発、金型設計、製造をワンストップで手掛ける技術力と、徹底管理された工場力を強みとする。土木事業では、独自工法を用いた軟弱地盤改良分野でトップシェアを獲得している。

業務の概要

当社は家電、ガス機器、OA 機器、医療用機器などの完成品メーカーに、工業用ゴム部品を納めている BtoB メーカーである。客先数は約 300 社で、約 3000 点の部品を扱っており、全国および海外に出荷している。また土木事業として、軟弱地盤改良工事の設計・施工や福祉事業を手掛けている。

基幹システムに関しては 1990 年から AS/400 を導入し、そのほとんどを内部開発してオリジナリティ ERP システムを構築している。データ収集は、独自開発の「Osaka-Ben Barcode System」を導入しており、リアルタイムで基幹データと連動したシステム環境となっている。【図 1】

開発の経緯

AS/400 の基幹データは昔ながらのグリーン画面で、一般のユーザーには受け入れられにくく、ビッグデータの利用がうまく進んでいなかった。そこで利用促進を図ろうと、2013 年に一部の管理用画面で Delphi/400 を採用し、Excel へアウトプットしたことで一気にデータ利用が促進された。

今回発表する開発内容の経緯は一昨年、筆者が業務改善のため、生産管理部門をフォローすることになったのがきっかけである。現状分析を実施し、各メンバーと面談した際、出荷業務の従事者が毎日の出荷作業に追われ、モチベーションも芳しくないことが判明。彼らのモチベーションを上げる手立てに頭を悩ましていたときに思いついたのが、「彼らの仕事の成果を他部門の社員に PR すればよいのではないか」ということであった。

そこで他部門の社員たちが目にする場所に大型モニターを設置し、Delphi/400 を利用してリアルタイム情報を掲示しようと思い立った。【図 2】

システムの概要

「Heart-Board プロジェクト」と名付けて取り組んだ内容は、各工程の状態をグラフや表にして、作業実績や状況を見やすく順次表示することで、作業者自身の確認はもとより、作業場の前を通る他部門やお客様に見てもらい、意識の向上を図っていくことであった。【図 3】に全体の構成を示す。

受入状況・出荷進捗状況・倉庫別進捗状況・各工程作業者の各データを Delphi/400 でコントロールして表示することで、各種データを同時に表示できるようになった。

受入状況の表示

当日の受入実績をバーコード転送と同時に累積表示し、受入物の行先別件数と過去 1 カ月の実績データを表示している。【図 4】

出荷進捗状況の表示

当日の出荷準備状況をリアルタイムに表示し、先 3 日間の事前準備状況と過去 1 カ月の出荷実績データを表示している。さらに画面下部に運送便ごとの集荷

図1　KINJO ERP SYSTEM

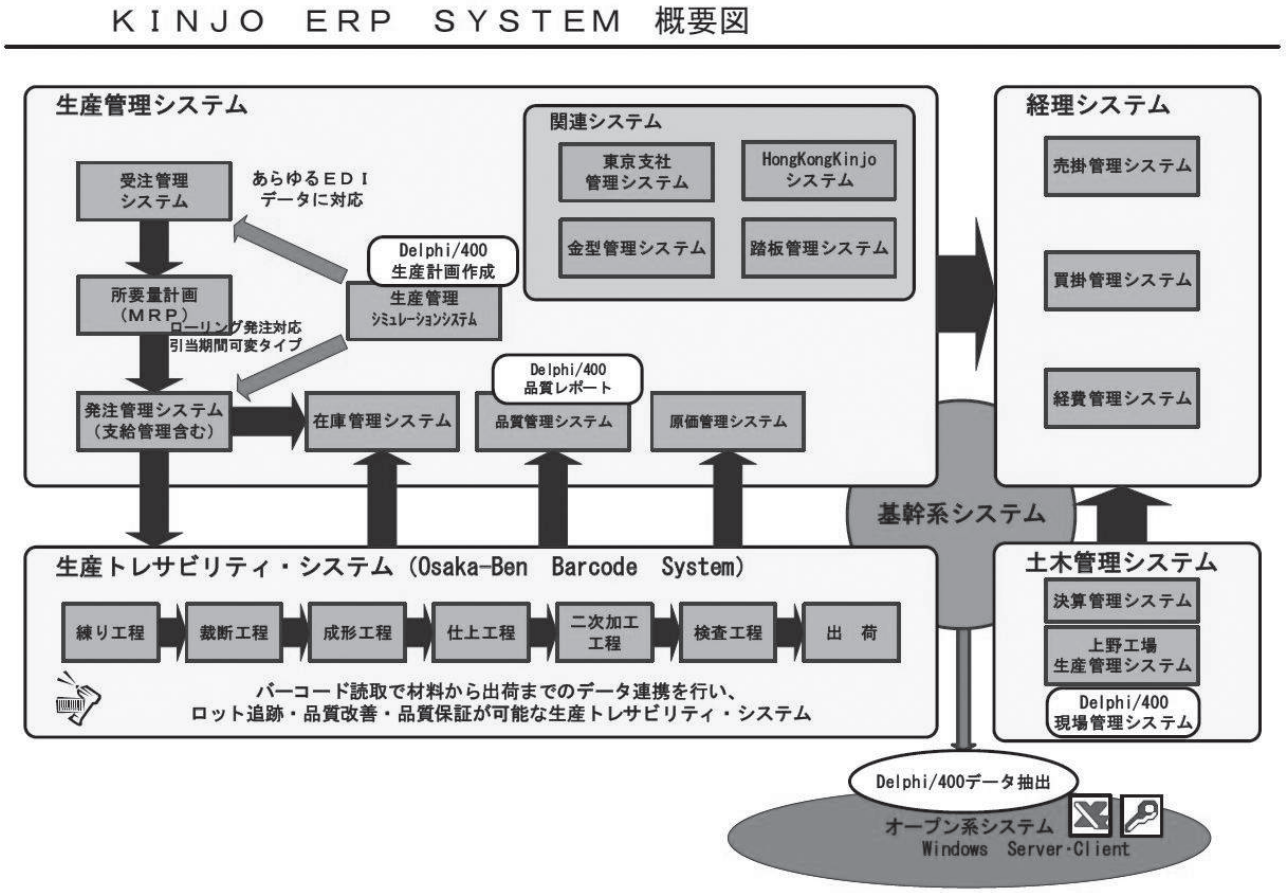


図2　完成した掲示板（65inchモニター）

状態を表示し、集配業者にもわかるようにした。【図 5】

倉庫別進捗状況の表示

出荷指図の品物から倉庫を特定し、倉庫場所別に出荷準備の進捗状況を表示する。これにより、各倉庫担当者の競争意識をもたせた。【図 6】

各工程作業者の表示

データ表示だけでは華やかさがないため、各工程作業者や関連スタッフの個人目標付き笑顔ポーズ写真を間に挟み込み、部署が一丸となって取り組んでいることをアピールするように工夫した。【図7】

画面の切り替え構成

データは AS/400 から、他の画像はファイルサーバーから送信し、設定 INI ファイルで画面切り替えや表示時間をコントロールしている。【図 8】

出荷場所にモニターを設置したあと、管理者のいる事務所にも同様の画面を設置し、内容を確認できるようにした。管理者用画面では、出荷場所のモニター情報に加えてトラブル情報をデータ入力と同時に表示させ、注意喚起を促している。【図 9】

工夫した点

今回もっとも工夫したのは、スライドメッセージを用いてトラブル情報を表示したことである。

スライドメッセージは、TLabel（メッセージ表示領域）の Left プロパティ（左位置）の値を時間の経過（TTimer でコントロール）とともに、一定量ずつマイナスしていくことで実現した。

連続してメッセージを表示させるための工夫として、メッセージ用の TLabel を 2 つ使用し、メッセージを連続表示できるように制御している。また、静止画面の中に横方向の動きや効果音をつけることで注目を惹きつけ、トラブル情報の共有や対策に活用している。【図 10】【図 11】

設置後の成果

部署内はもちろん他部署からも非常に高評価で、来客の方々からもよい取り組みだと評価されている。また事前準備も着実に成果を上げ、前日の出荷準備率は設置前 40％程度だったのが 80％まで改善し、出荷業務従事者のモチベーションもかなり向上した。【図 12】

今後の展開

Heart-Board プロジェクトは、ミガロ. の担当者からのアドバイスも取り入れ、Delphi/400 の仕組みを構築した。今後もミガロ. からの支援を受けながら、新しいアイデアを具現化していきたい。

今回の取り組みでは、Delphi/400 の新たな使い方を発見できた。今後は、今回の仕組みを他工程にも展開し、新たな Delphi/400 の活用を模索して、基幹データの有効活用を図りたいと考えている。

M

図3　システム全体図

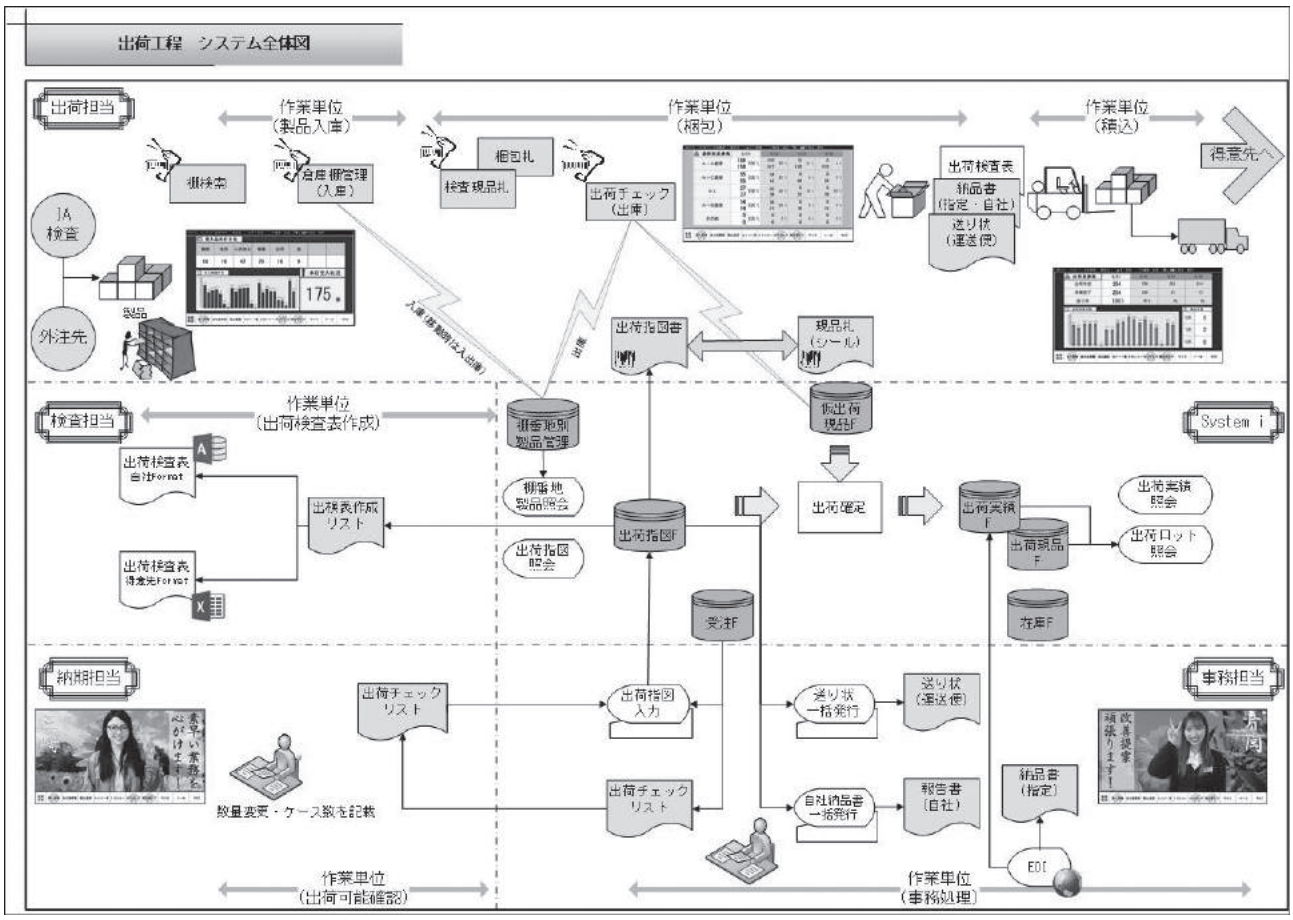


図4　受入状況の表示

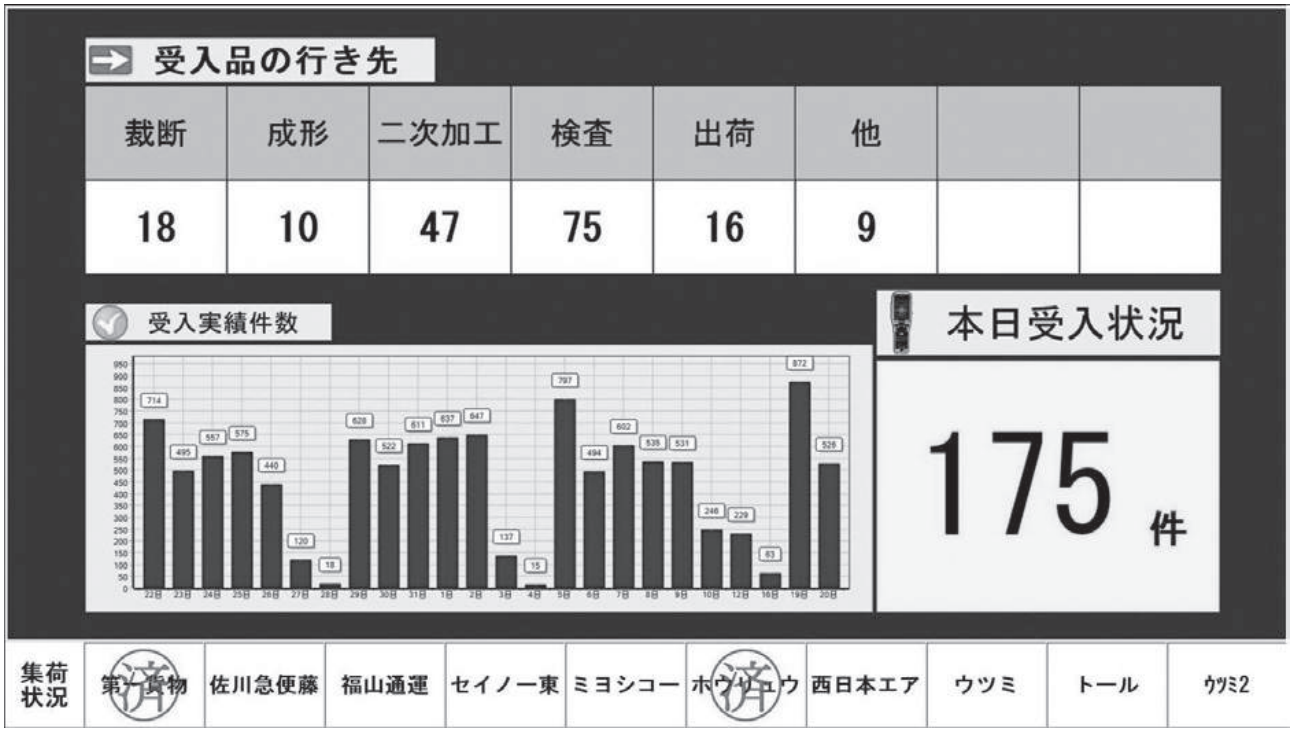
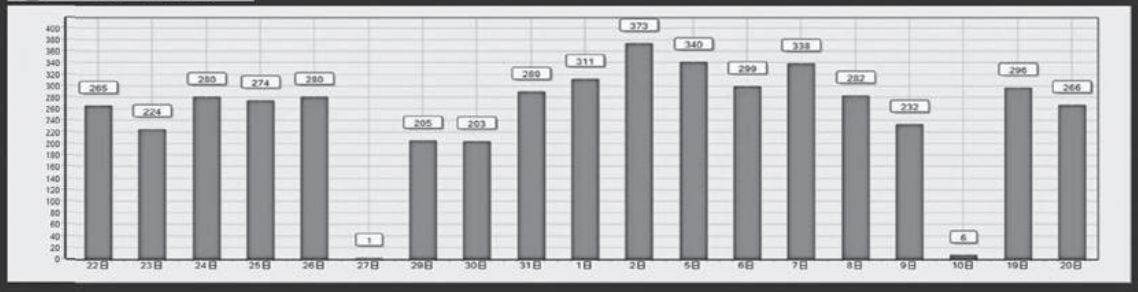


図5　出荷進捗状況の表示

出荷進捗表	8/21	8/22	8/23	8/26
出荷予定	254	258	255	214
準備完了	254	215	21	13
完了率	100%	83%	8%	6%

出荷実績件数



集荷状況	第一貨物	佐川急便藤	福山通運	セイノー東	ミヨシコー	ホウリュウ	西日本エア	ウツミ	トール	ｳﾂﾐ2
	済					済				

図6　倉庫別進捗状況の表示

倉庫別進捗表	8/21		8/22		8/23		8/26	
A－A倉庫	158 158	100 %	120 147	82 %	15 136	11 %	5 110	4 %
A－C倉庫	55 55	100 %	34 42	81 %	0 69	0 %	0 50	0 %
OA	27 27	100 %	35 38	92 %	6 32	18 %	8 39	20 %
A－B倉庫	14 14	100 %	26 31	84 %	0 18	0 %	0 15	0 %
その他	0 0	0 %	0 0	0 %	0 0	0 %	0 0	0 %

集荷状況	第一貨物	佐川急便藤	福山通運	セイノー東	ミヨシコー	ホウリュウ	西日本エア	ウツミ	トール	ｳﾂﾐ2
	済					済				

図7　各工程者の表示

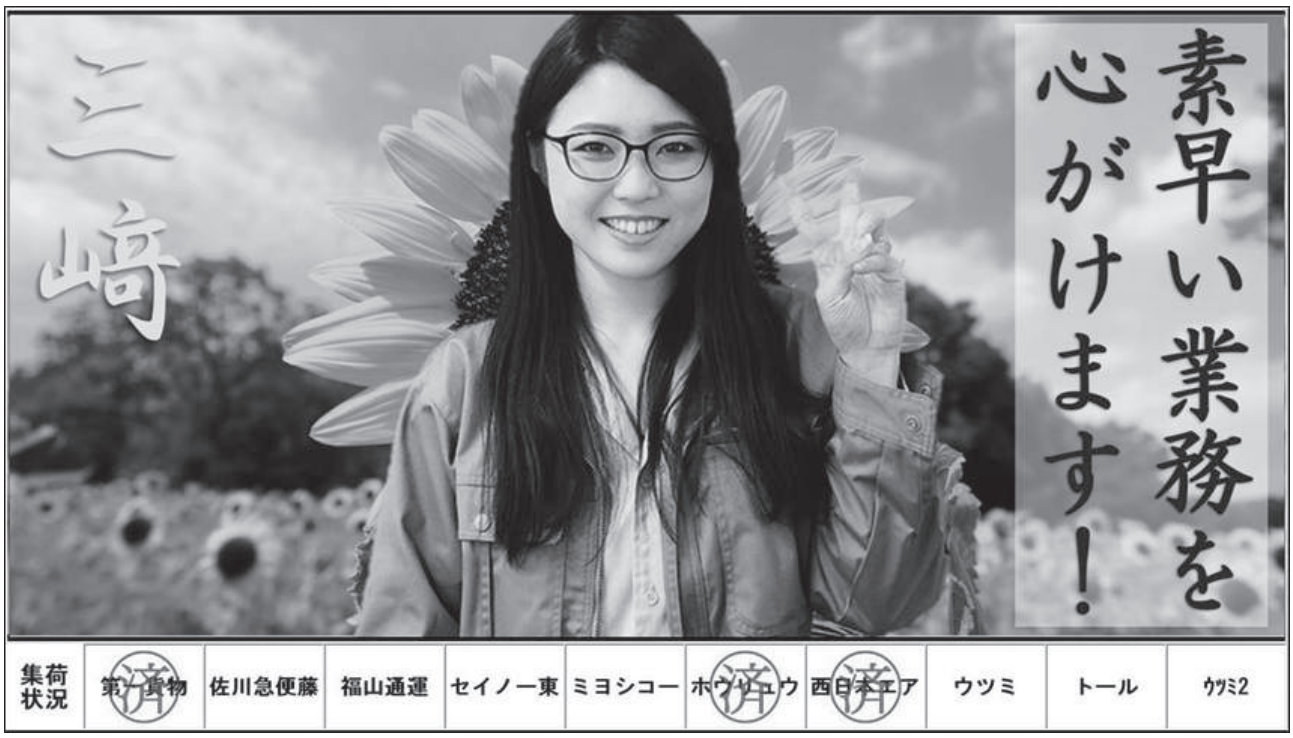


図8　画面の切り替え概要図

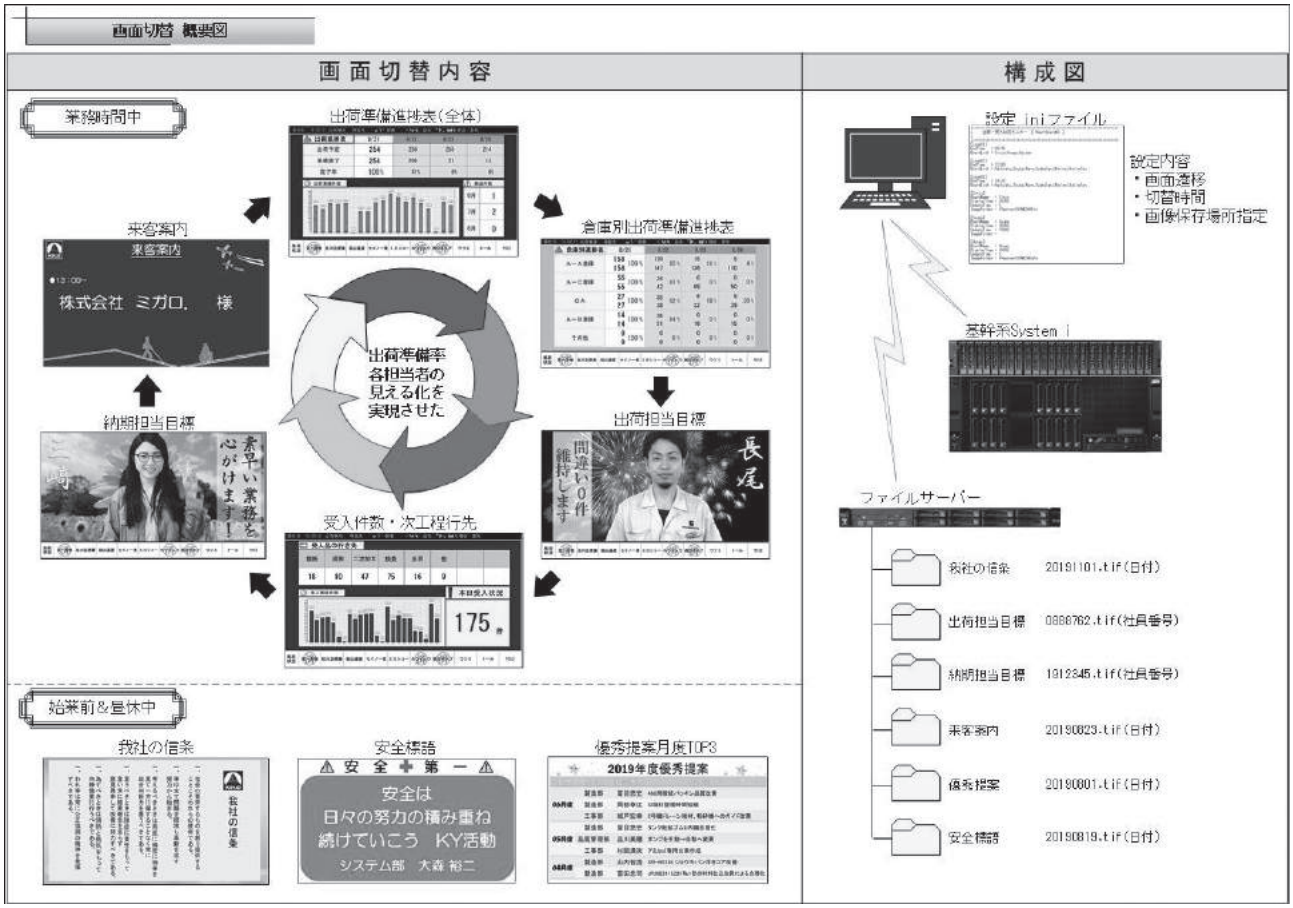


図9　スライドメッセージ(トラブル情報)

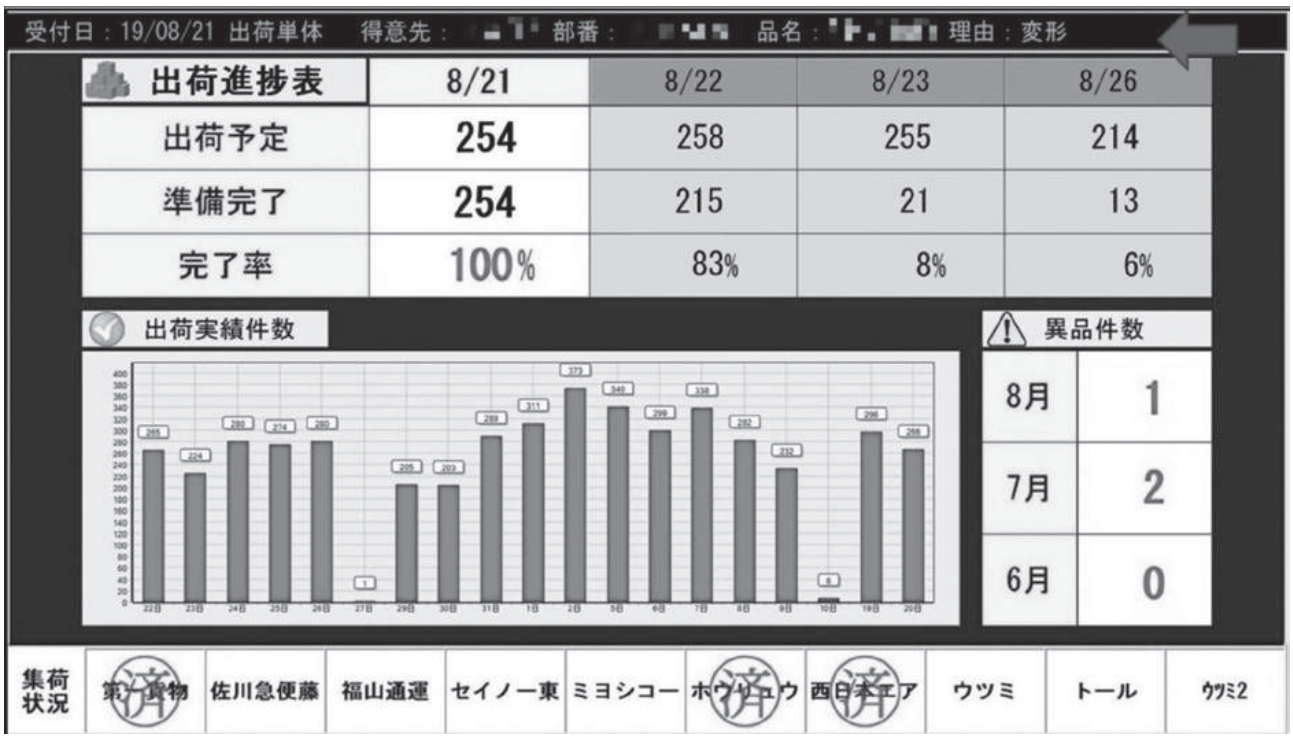


図10　スライドメッセージの仕組み①

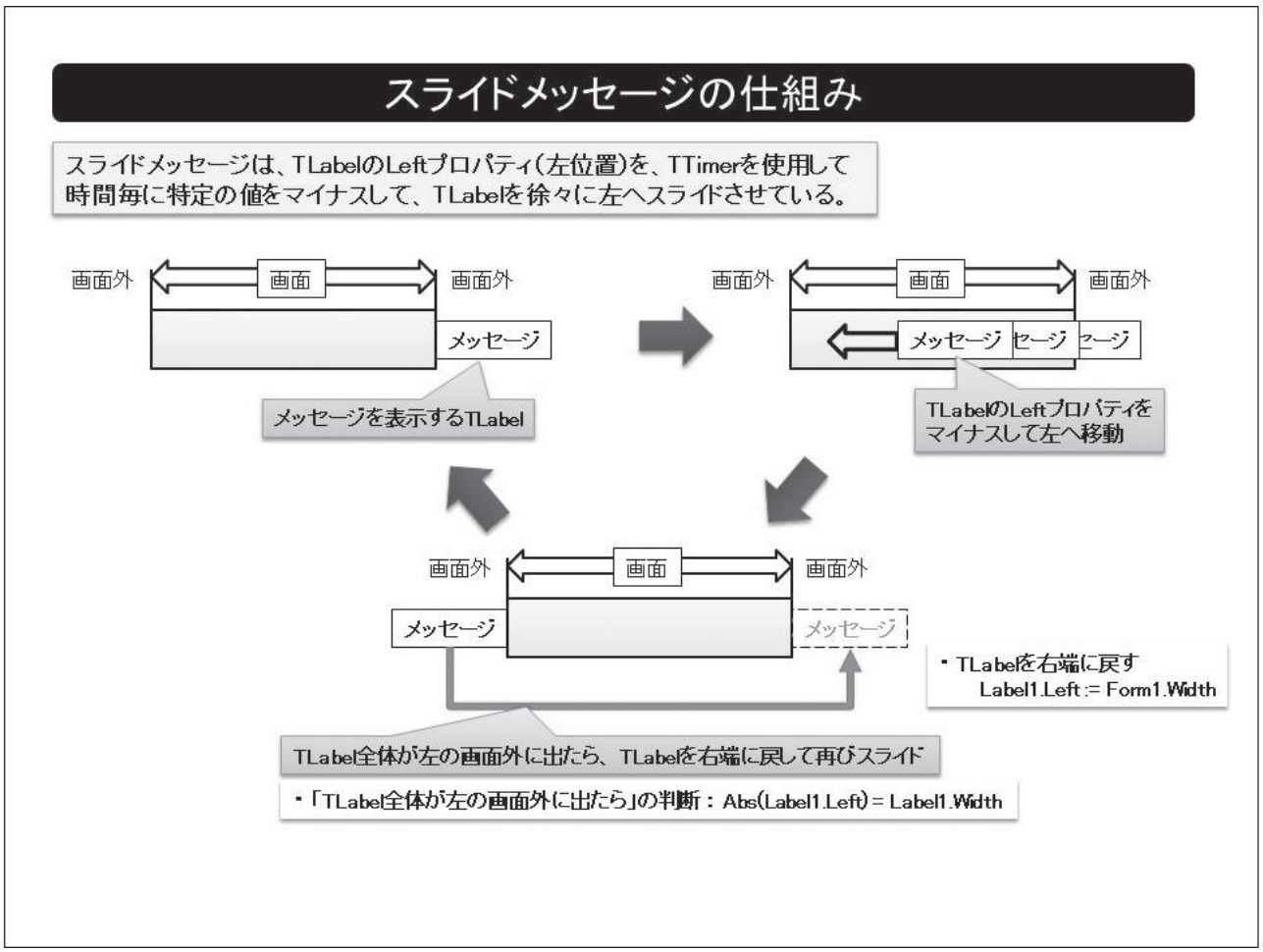


図11　スライドメッセージの仕組み②

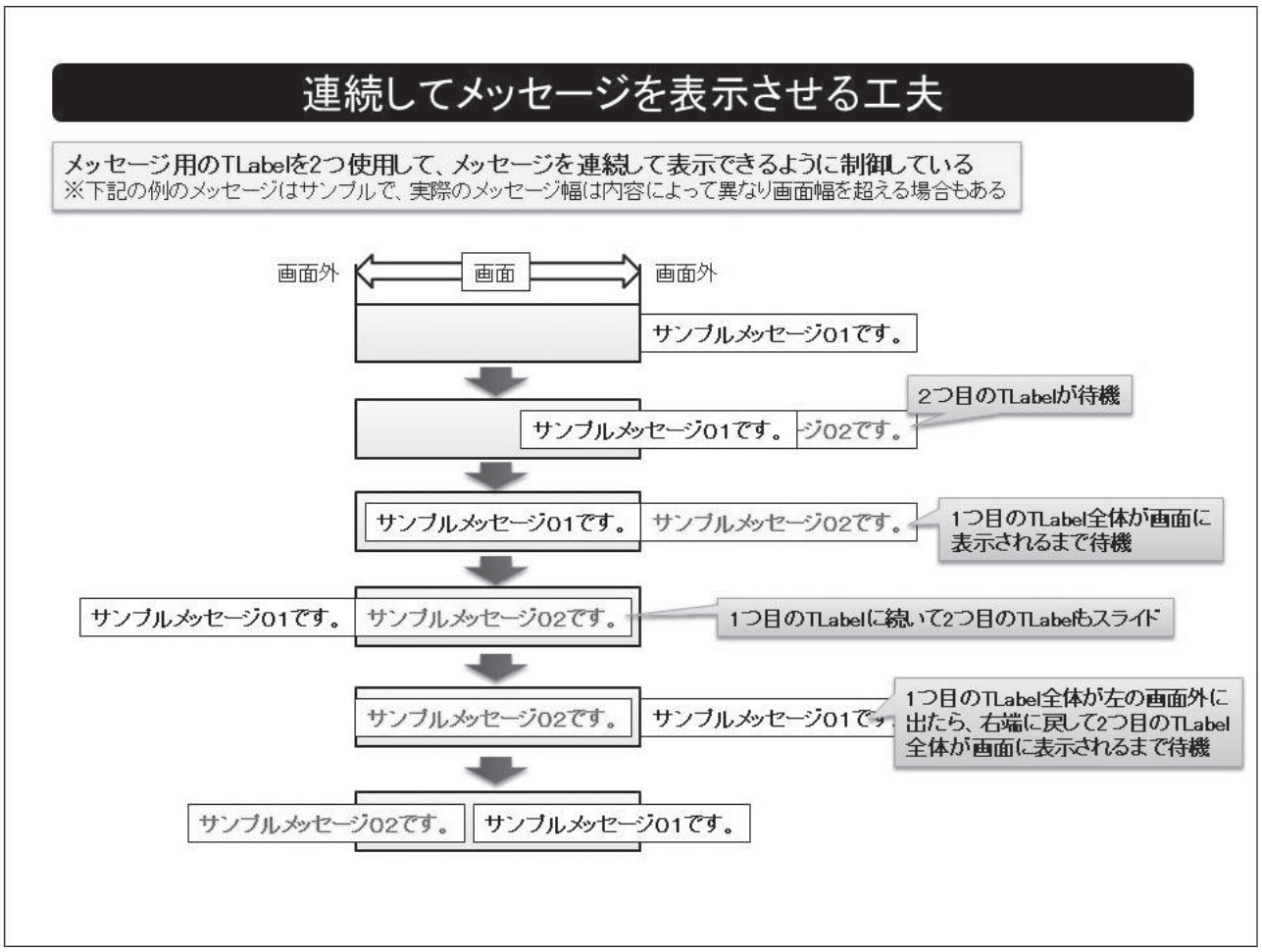
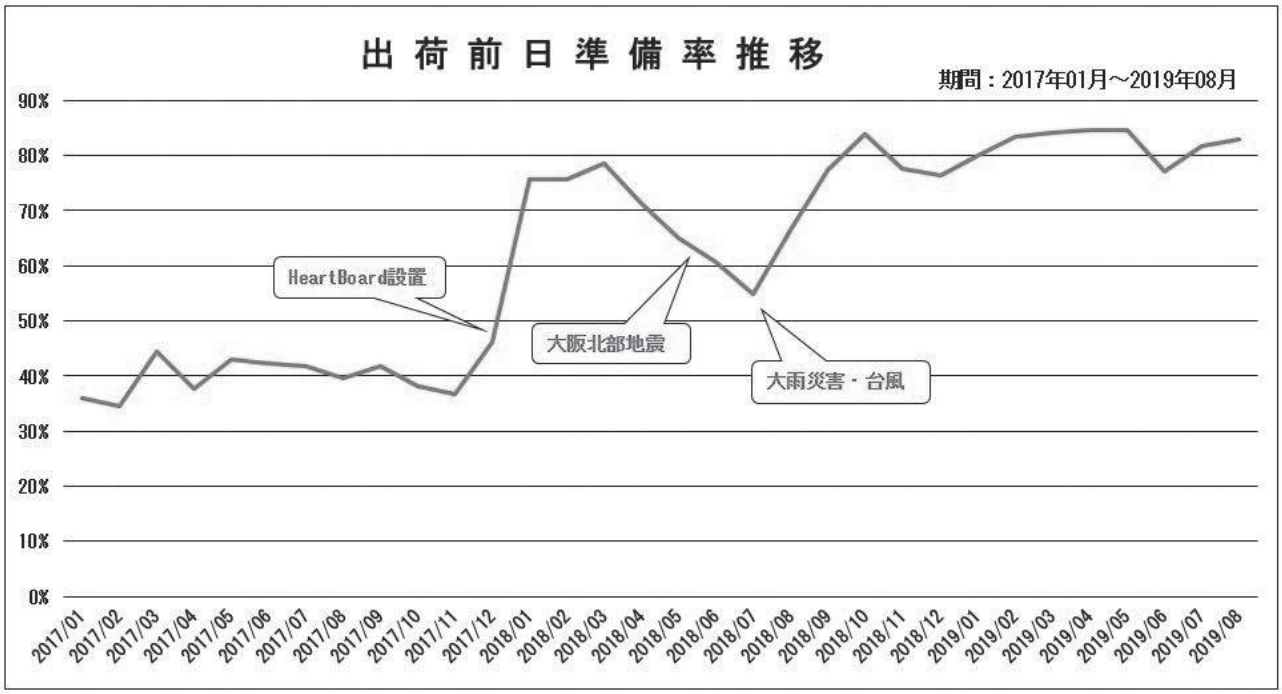


図12　出荷前日準備率の推移



iPhoneを利用した宝飾品の販売系システムをSP4iで構築

島本 佳昭 様
株式会社大月真珠
システム室
課長代理

株式会社大月真珠
https://www.otsuki-pearl.co.jp/

創業 1930 年。真珠の加工・販売・輸出および宝石、宝飾品の販売を行う。多様なニーズに応えて生産・加工・販売の一貫体制を取ることで、高い競争力と信頼を得ている。製品取扱量は非常に多く、アコヤ真珠の 1 級入札会シェアや真珠輸出額は業界第 1 位。アコヤ貝の養殖などの生産活動も積極的に行っている。

本論文では、iPhone で稼働する宝飾管理システムの構築について述べる。当社の基幹システムは 1986 年に AS/400 を導入後、一貫して AS/400 ～ IBM i で構築してきた。基幹システムは、本社で管理する加工、卸し販売、百貨店販売から経理までをカバーしている。販売系のシステムについては、製品画像を業務画面上で確認したいという要望が強いことから、PC で稼働する GUI システムを構築している。

モバイル化の取り組みとSP4iの採用

販売系システムについては、以下の課題があった。

・販売系システムを外出先で利用したい
・販売した商品の画像登録を即時に行いたい
・場所を問わず棚卸しを行いたい
・報告系の入力を移動中に行いたい

これらの課題を解決するために、iPhone を使った販売システムの構築を検討した。

まず約 50 名の営業担当者に iPhone を支給したが、IBM i と連携する iPhone システムをどのように開発するか、という課題が残った。PC の販売システムで採用している IBM i の GUI 化ツールはモバイル対応ができなかったため、新たなツールを検討。AS/400 ～ IBM i の長年の開発経験により RPG プログラマーが多く在籍していることから、RPG で IBM i モバイルアプリを構築できる「SmartPad4i」（以下、SP4i）を採用するに至った。

iPhoneシステムの概要

今回開発した iPhone の販売システムは、部長、店長、営業担当者の約 50 名が利用している。使用端末は、会社支給の iPhone（iPhone8）のみに限定した。システム構築にあたり、PC の GUI システムの内容を基本的に踏襲しているが、モバイル環境で必要な範囲に絞り込んでいる。具体的にはダイヤ卸し系システムと百貨店系システムをモバイルで構築することとした。

本論文では、とくに百貨店系システムについて詳細に述べる。

モバイル用アプリ開発方法の概要

今回の iPhone 対応アプリは、SP4i のモバイル用アプリ「SP4iV2」アプリを使用して構築した。本アプリにより端末に合わせた画面のレイアウトの最適化、カメラ、バーコードリーダーが特別なシステム開発なしで利用できる。

個々のプログラム開発方法は通常の SP4i と同様で、作成済みの HTML をもとに SP4i デザイナーで RPG のひな形を作成し、生成された RPG プログラムをもとに業務ロジックを開発する。

ログイン機能については、iPhone の初期画面で SP4i システムアイコンをタップ後に、アプリケーションの実行を

図1　メインメニュー一般用

図2　メインメニュー店長用

図3　メインメニュー部長用

図4　持ち廻り1

図5　持ち廻り2

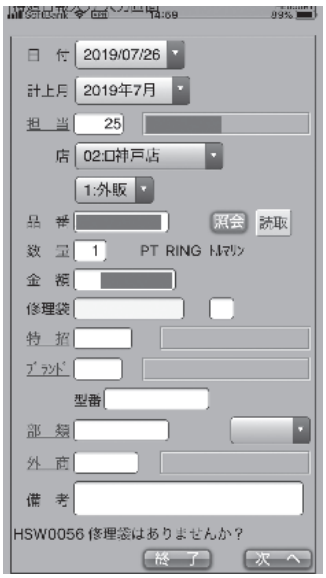
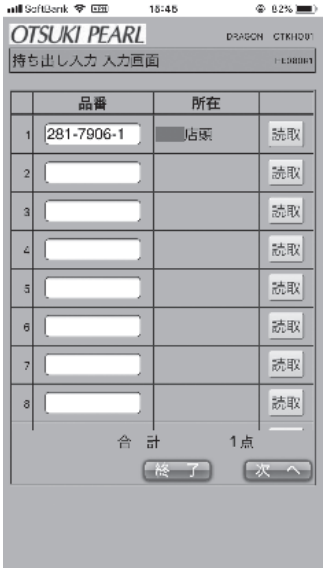


図6　持ち廻り3

図7　経費入力

図8　シフト

図9　持ち出し



タップすると、SP4iアプリの固有メニューをスキップし、初期業務メニューに直接遷移するようにした。VPNアプリを起動する際に、ユーザーごとに固有のIPアドレスを付与して個人を識別できる仕組みを入れたことで、ユーザーの職階に応じた個別の初期メニューをログイン操作することなく表示する。【図1】【図2】【図3】

業務系（入力／更新系）メニュー・機能の概要

各入力画面に共通の標準ルールを定めることにより、エンドユーザーがわかりやすくシステムを操作できるように工夫した。

入力（更新）系のルール

入力画面は、各項目への入力後に「次へ」ボタンをタップ→入力項目に保護がかかり確認画面に遷移→確認後に「更新」ボタンをタップすることで入力完了、という流れである。

エラー項目は赤で反転させ、フィールドにフォーカスをセットすることで、確認画面には進めなくする。

警告項目は黄色で反転させるが、黄色に反転したまま確認画面で更新可能とする。

入力・照会の共通ルール

画面サイズに制約があるため、コード検索を行いたい項目について、検索・照会画面を別画面でポップアップ表示できる仕組みとした。コード検索可能な項目は青字＋下線部、または検索ボタンを配置して識別できる。

日付や簡単なCD検索などは、プルダウンメニューを利用する。

また品番入力フィールドには読み取りボタンを用意し、バーコードリーダーからの読み込みを可能にした。複数明細のある画面では、連続読み込みも可能にしている。

営業担当者メニュー

持ち廻り入力（販売日報報告）

日々の販売日報報告を行う。読み取りボタンをタップすると、バーコードリーダーが起動する。【図4】【図5】【図6】

経費入力

日々使用した経費、同行した百貨店の外商員､商談回数などを入力する。【図7】

シフト入力

シフトが変更されたときに修正する。【図8】

持ち出し入力

商品を金庫から持ち出しするときに入力する。【図9】

持ち出し返却入力

持ち出しした商品を金庫に返却するために入力する。【図10】【図11】

画像登録入力

販売した商品の画像を登録する。ここでは次のような工夫を行った。

(1) SP4iのカメラ機能を利用して、指示画面からカメラを起動して撮影画像を登録する方法と、カメラロールに保存済みの画像を登録する方法の2つを可能にした。
(2) 適正な画像サイズとするために、20KB程度に圧縮をかけて保存した。
(3) 登録画像をIFS上のディレクトリに、品番に関連付いたファイル名で保管することにより、画像と品番の関連付けを容易にした。なお、撮影は横向きで行うことをルール化している。【図12】【図13】【図14】

棚卸入力

棚卸しを行うために入力する。読み取りボタンをタップするとバーコードリーダーが起動し、値札の品番バーコードを読み取った後に続けて、読み取りを行うかどうかの確認画面を表示する。YESをタップすると、続けてバーコードリーダーが起動する。【図15】【図16】

この画面では､次のような工夫を行った。

(1) SP4iのバーコード読み取り機能を利用して、連続読み取りとカーソル移動ができるようにする。
(2) 連続読み取りが速すぎて、同じバーコードを複数回読み取る事象が続いたため、読み取りを続行するかどうかの確認画面を間に入れることで､同一品番読み取りを防止できた。

店長向けメニュー

催事報告入力

催事報告を行うために入力する。催事報告のメイン画面から､「他社売上」「ブランド売上」「経費」「コメント」の各ボタンをタップすると､各入力画面が開き、詳細を入力できる。登録が完了した項目のボタンを緑色に変更することで、登録済みであることが一目でわかるように工夫した。【図17】【図18】【図19】【図20】【図21】

月末予測入力

毎月20日に、月末の売上予測の数字報告を行う｡入力ボタンをタップすると、入力用画面が開く。【図22】【図23】

月次報告入力

毎月末に、店長が月末時点の売上数字の速報値を報告するために入力する。メイン画面から各報告のボタンをタップし、それぞれの詳細入力画面に遷移。各項目の入力後はボタンが緑色に変わる（催事報告と同様の仕組み）。【図24】【図25】【図26】【図27】

照会系メニュー・機能の概要

営業担当・店長共通画面

シフト照会

日付指定でシフトを確認したい店のメンバーのシフトを照会する。【図28】【図29】

属性照会

商品属性を確認するために照会する。画面項目が多いため、中珠情報／脇石情報／デザイン情報／管理情報／コメントをセクション化し、セクションごとに明細を折りたたんで表示・非表示を切り替えられるように工夫した。【図30】【図31】【図32】

品番照会

品番の在庫の画像、在庫状況、在庫経歴を確認するための照会。画像／在庫状況／品番経歴をセクション化し、セクションごとに明細を折りたたんで表示・非表示を切り替えられるように工夫した（属性照会と同様）。この画面から「属」ボタンをタップして、属性照会を呼び出

図10　持ち出し戻り1

図11　持ち出し戻り2

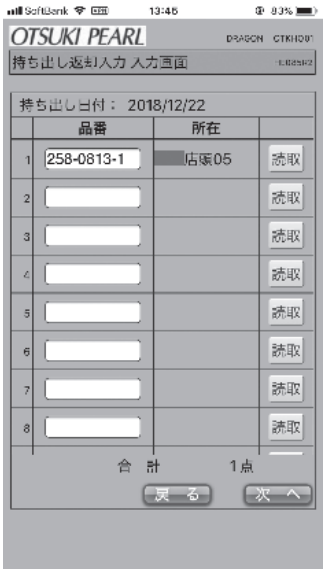


図12　画像登録

図13　画像登録1

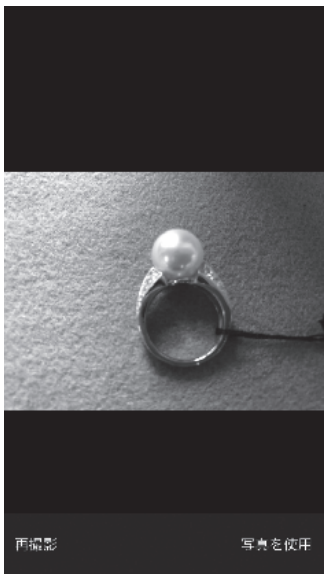


図14　画像登録2

図15　棚卸し1

図16　棚卸し2

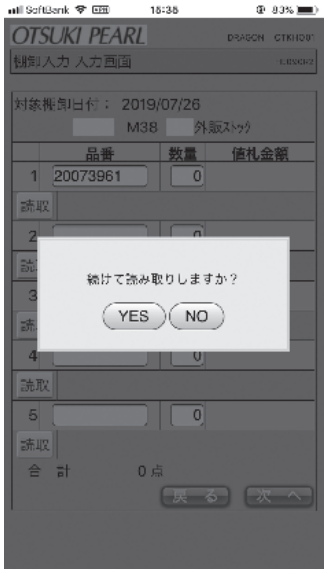


図17　催事報告

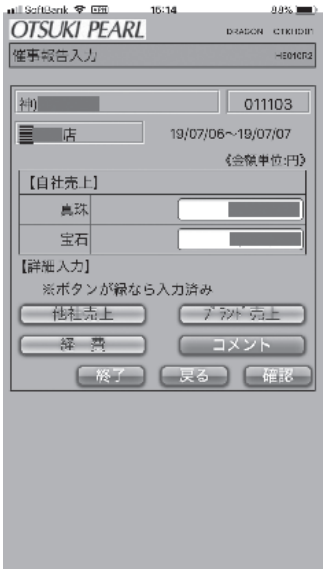


図18　催事報告コメント

せる。また、品番経歴部分は縦・横スクロールが可能である。【図33】【図34】

特招一覧照会

催事の開催情報を確認するために照会する。売上店をタップすると、催事報告の画面が呼び出せる（催事報告1）。また開始日をタップすると、催事計画の画面を呼び出せる（特招一覧1）。【図35】【図36】【図37】

出庫伝票一覧照会

本社から店舗に送った伝票の一覧を確認するために照会する。伝票番号をタップすると、伝票明細（品番単位の内訳）画面を呼び出せる。【図38】【図39】【図40】

在庫問い合わせ（宝石一般）

在庫を検索するために照会する。品番ごとに商品の画像や属性、在庫場所が確認できる。画像をタップすると画像照会に遷移する。【図41】【図42】【図43】

在庫問い合わせ（ダイヤ）

在庫を検索するために照会する。品番ごとに商品の画像や属性、在庫場所が確認できる。【図44】【図45】

在庫問い合わせ（ブランド）

在庫を検索するために照会する。品番ごとに商品の画像や属性、在庫場所が確認できる。【図46】【図47】

在庫問い合わせ（真珠）

在庫を検索するために照会する。品番ごとに商品の画像や属性、在庫場所が確認できる。宝石一般と同様に画像照会に遷移する。【図48】【図49】

未決明細照会

持ち廻り入力を行い、発生した売上を照会する。店CDと担当者CD、年月指定が必須となる。【図50】【図51】

店長用照会画面

シフト集計照会

店の所属メンバーのシフトを集計して照会する。店CDと年月指定が必須となる。年度を指定すると、累積有休数のみを照会できる（シフト集計4）。【図52】【図53】【図54】【図55】

未決速報

所属グループ店舗の売上実績を店舗一覧で照会する。店名をタップして、担当者別の速報照会を呼び出せる。縦・横のレスポンシブデザイン対応で、横スクロールも可能（未決速報3-1、3-2）。【図56】【図57】【図58】【図59】【図60】

部長用照会画面

月末予測一覧照会

店長が入力した月末予測を店舗別に一覧で照会する。店名をタップすることで、各店の月末予測を照会できる。未決速報と同様のデザイン設計。【図61】

月次報告一覧照会

店長が入力した月次報告を店舗別に一覧で照会する。店名をタップすることで、各店の月次報告を照会できる。未決速報と同様のデザイン設計。【図62】

開発上の苦労や工夫した点

本システムの開発で苦労した点や工夫した点は、以下のとおりである。

HTML作成

今回初めて取り組んだ。当初はホームページビルダーを使ってデザインしていたが､現在はツールなしで開発している。

Javaスクリプトの利用

スクロール時のヘッダ項目の固定化などの画面制御は、フリーのJavaスクリプトライブラリ（jQueryなど）を利用して実現した。

文字入力を極力削減

スマートフォンで使用するので、文字の入力をできるだけ少なくした。プルダウン選択、IBM iのマスターデータから候補リストを取得して選択、数字入力フィールドはフォーカスインした時点で入力値をすべて選択状態にして修正を容易にする、などの工夫を行った。また候補リストの選択は、項目に青下線を付けるルールで統一した。

使用文字の制限

データはIBM iに格納するため、入力はできても、保存はできないといった絵文字などによる問題が発生した。これは、HTML内にチェックロジックを埋め込むことで解決した。

レスポンシブデザインと縦横スクロール

多くの照会画面共通のレイアウトとして、画面を縦・横に向きを変更してもレイアウトが崩れないレスポンシブデザインを採用。また、ヘッダ項目を固定化して縦横スクロールできるように工夫した。

評価と今後の展望

開発面

SP4iでは、業務ロジックは基本的にRPGで開発するので、多くの画面に対応した多数のRPGプログラムをいかに効率的に開発するかが鍵となる。そのため、以下の例のように宣言部、自由記述部、構造体、テーブルなどにソースを分割し、コピー句も多用してソースコードの標準化を図ることで、効率的に開発できるように努めた。

例） 催事広告入力1　画面のプログラムコード構成

HE010R1B　RPGLE　SP4i
　催事報告入力1　各種宣言部
HE010R1C　RPGLE　SP4i
　催事報告入力1　自由記述部
HE010R1D　RPGLE　SP4i
　催事報告入力1　D,I（構造体）仕様書
HE010R1H　RPGLE　SP4i
　催事報告入力1　H,F,D（テーブル）仕様書

業務面

本システムの稼働後、それぞれの入力や照会の使用頻度は増えており、ユーザーに浸透しつつある。またユーザー部門からの要望に応えて、改善を実施している。毎月1日に利用店、担当者ごとに各プログラムの利用状況を分析するための統計表を作成し、この統計表をもとに使用頻度の低い入力・照会・ユーザーを洗い出しすることで、ユーザーにとってより役立つ使いやすいシステムとなるように改善を続けていく。【図63】

M

図19　催事報告ブランド

図20　催事報告経費

図21　催事報告他社情報

図22　月末予測

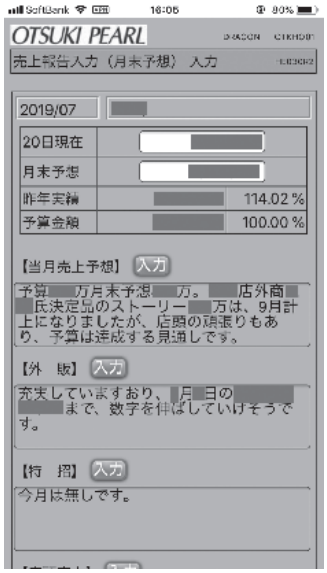


図23　月末予測1

図24　月次報告

図25　月次報告コメント

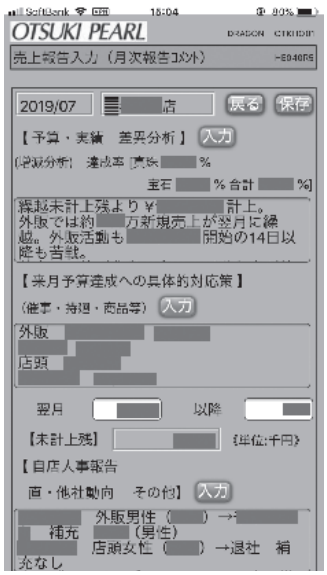


図26　月次報告他社

図27　月次報告特招

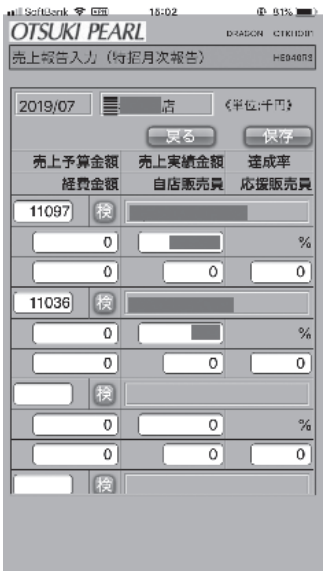


図28　シフト1

図29　シフト2

図30　属性照会1

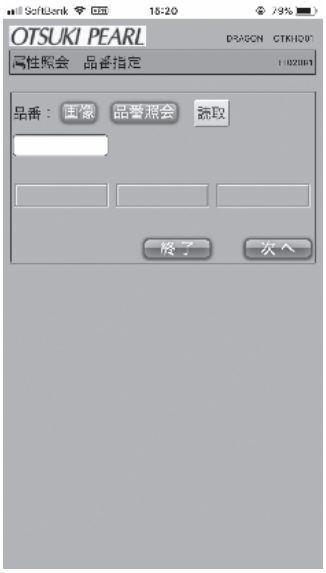


図31　属性照会2

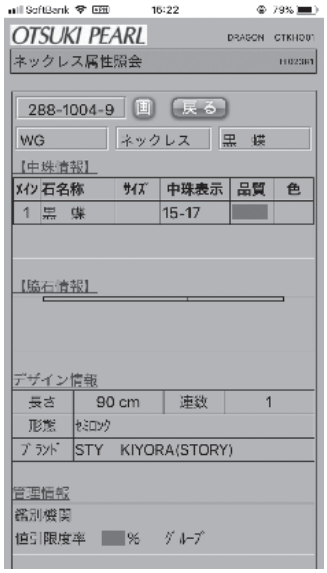


図32　属性照会3

図33　品番照会1

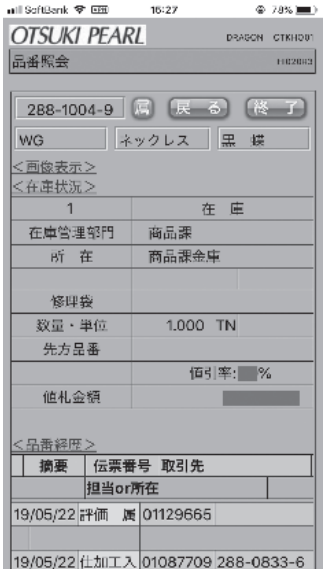


図34　品番照会2

図35　特招一覧

図36　特招一覧1

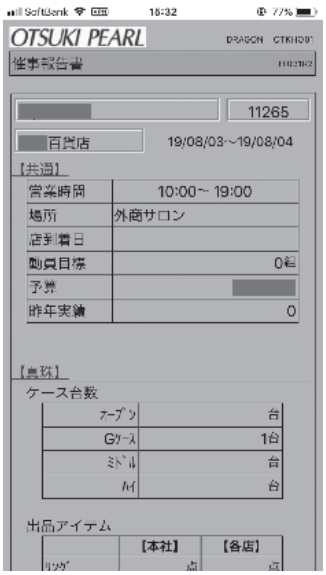


図37　催事報告1

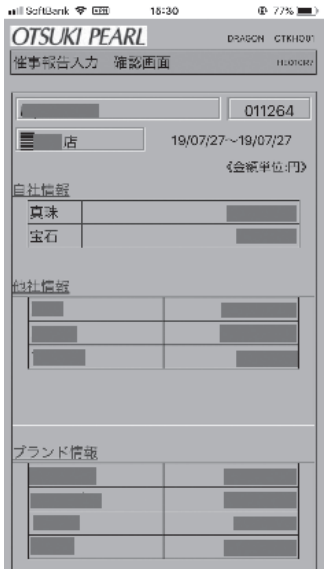


図38　出庫伝票一覧1

図39　出庫伝票一覧2

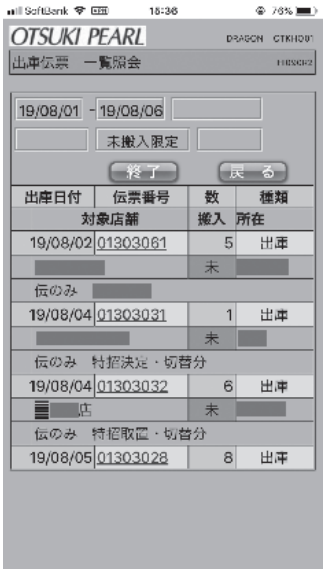


図40　出庫伝票一覧3

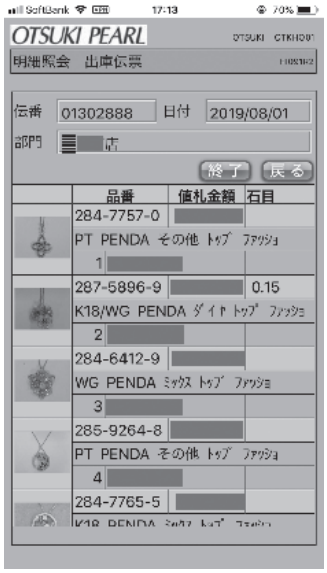


図41　在庫問い合わせ宝石1

図42　在庫問い合わせ宝石2

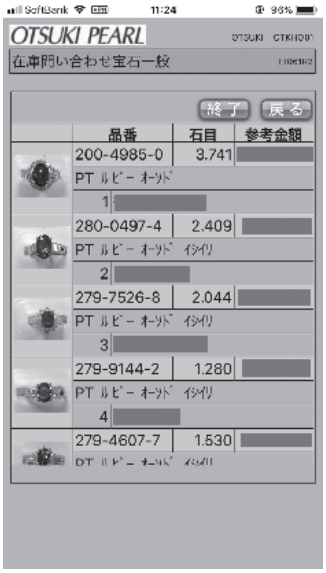


図43　在庫問い合わせ宝石3

図44　在庫問い合わせダイヤ1

図45　在庫問い合わせダイヤ2

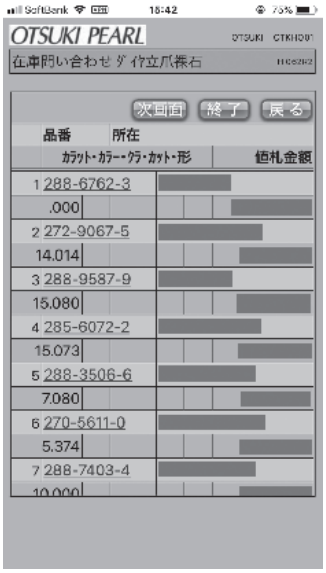


図46　在庫問い合わせブランド1

図47　在庫問い合わせブランド2

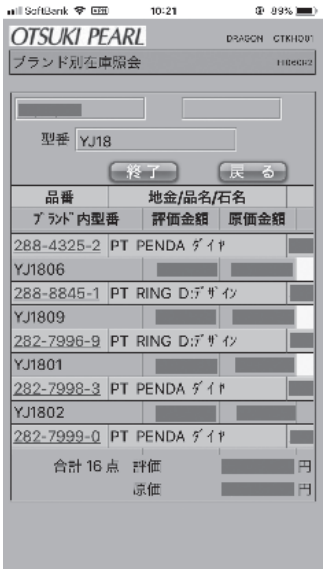


図48　在庫問い合わせ真珠1

図49　在庫問い合わせ真珠2

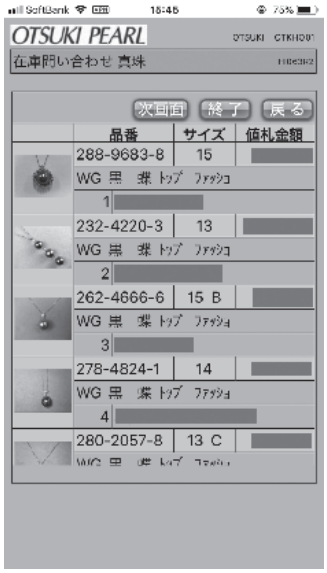


図50　未決明細1

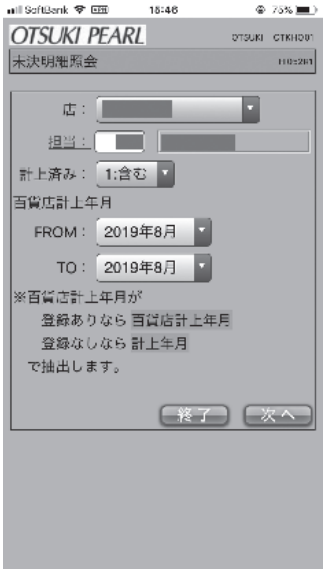


図51　未決明細2

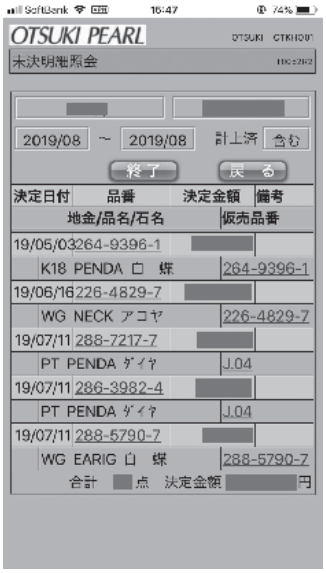


図52　シフト集計1

図53　シフト集計2

図54　シフト集計3

図55　シフト集計4

図56　未決速報1

図57　未決速報2

図58　未決速報3-1

図59　未決速報3-2

図60　未決速報4

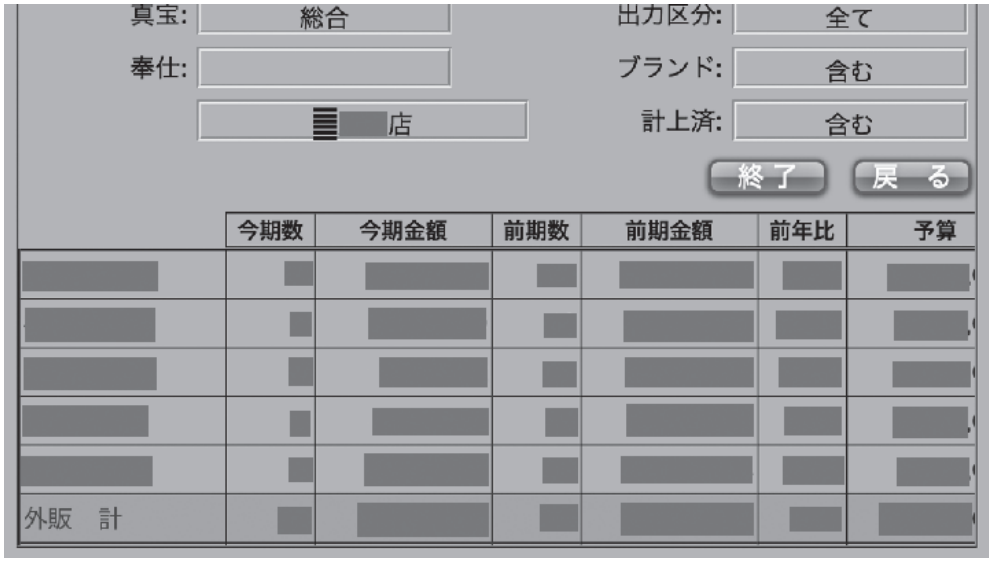


図61　月末予測

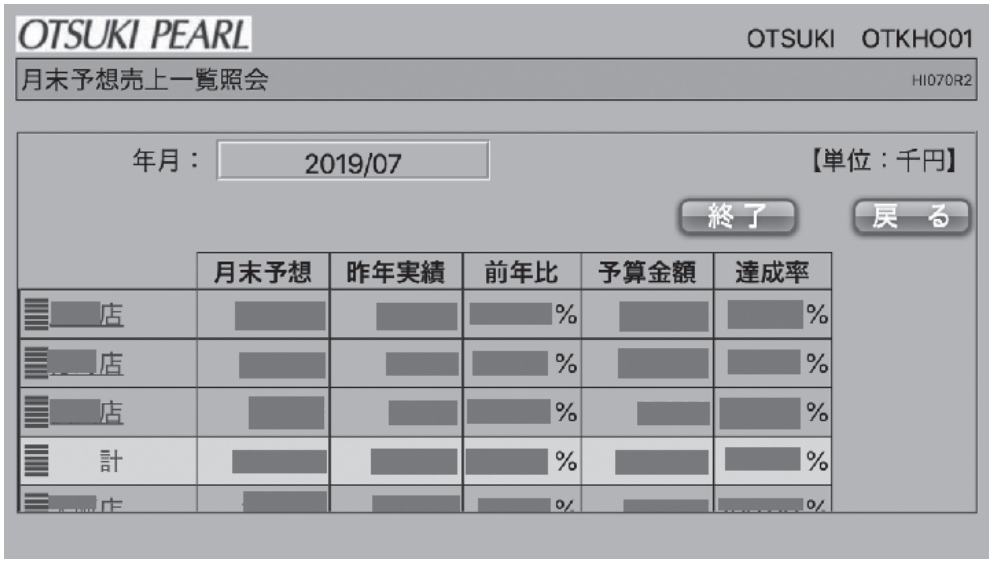


図62　月次報告

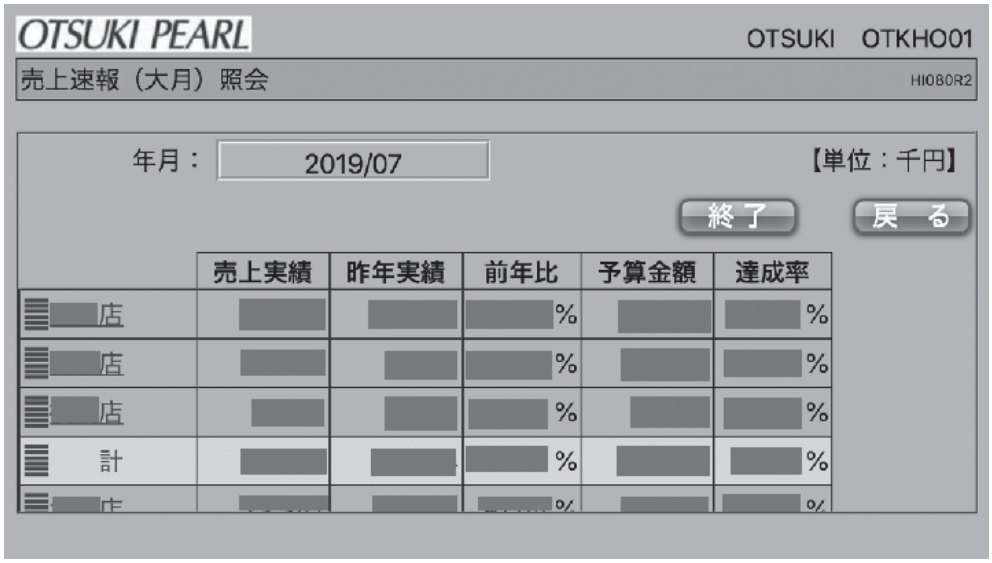


図63　統計表

[redacted]百貨店	64	[redacted]	40	HE050R1	持ち廻り入力	2
				HE070R1	シフト入力	8
				HE090R1	店舗別　棚卸入力	1
				HI010R1	シフト照会	26
				HI020R1	属性＆品番照会	3
		[redacted]	24	HE050R1	持ち廻り入力	1
				HE070R1	シフト入力	1
				HE090R1	店舗別　棚卸入力	3
				HI010R1	シフト照会	14
				HI011R1	シフト集計照会	1
				HI020R1	属性＆品番照会	1
				HI020R11	属性照会バーコード起動	1
				HI051R1	未決速報	1
				HI052R1	未決明細照会１	1
[redacted]	411	[redacted]	411	HE030R1	月末予測	9
				HE040R1	月次報告	13
				HE050R1	持ち廻り入力	1
				HE060R1	経費入力	2
				HE070R1	シフト入力	21
				HI010R1	シフト照会	49
				HI011R1	シフト集計照会	24
				HI020R11	属性照会バーコード起動	5
				HI020R12	経歴照会バーコード起動	10
				HI030R1	特招一覧照会	4
				HI051R1	未決速報	70
				HI052R1	未決明細照会１	162
				HI060R1	在庫明細（ブランド）	4
				HI063R1	在庫明細（真珠）	2
				HI090R1	出庫伝票　一覧照会１	35

優秀賞

SmartPad4iによる、導入後の改修を意識した設計

西村 直也 様

株式会社ジャストオートリーシング
営業企画部 システム課

株式会社ジャストオートリーシング
https://www.justauto.co.jp/

神奈川、東京都南西部を営業基盤とする独立系オートリース専門会社。神奈川県内で最大規模の自動車整備工場を運営し、自動車リースを中心に、車検や新車／中古車の販売、損害保険など、自動車に関する総合的なサービスの提供をコンセプトに日々、お客様に対応している。

業務課題

外回りの社員より、スマートフォンを使用して外出先で各種情報の確認や登録ができるWeb化画面を要望されていた。お客様サイト用に導入した「SmartPad4i」（以下、SP4i）で開発可能であるものの、以下の課題により導入できずにいた。

(1) 導入後も高頻度で改善・機能追加を求められることが予想されるが、社内はRPGのみの技術者が多く、定形化されたソースをもとに開発しており、HTMLなどRPG以外の仕組みに不慣れである。
(2) SP4iの標準機能のみでは個々の画面に合わせた多様なHTML設計が必要となるため、開発できても導入後の運用が困難と推測される（開発を担当した本人にしか改修できない）。

改修を意識した設計

(1) HTMLの構成は極力シンプルなものとし、入力欄やボタン等の大きさは共通CSSで定義することで、導入後の修正担当者がHTMLの構成を極力意識せずに済むよう設計する。【図1】
(2) 既存の定形化されたRPGソースを元に、SP4iの自動生成ソースに埋め込んで使用する定形RPGソースを設計。新規画面開発時には、定形ソースの利用により開発の標準化と効率アップを図る。【図2】【図3】

業務課題の解決

まずは、社員より要望の多い情報の照会・登録画面を開発し、運用を開始。その後、感想や要望を踏まえ、機能追加を繰り返しているが、(2)の設計により、新規画面開発時の負荷は少なく済んでいる。また、各画面の見栄えや操作性が統一されている。総じて、システム部門の管理負荷を軽減し、社員からの要望に迅速に応えられている。

M

図1　共通CSSの説明

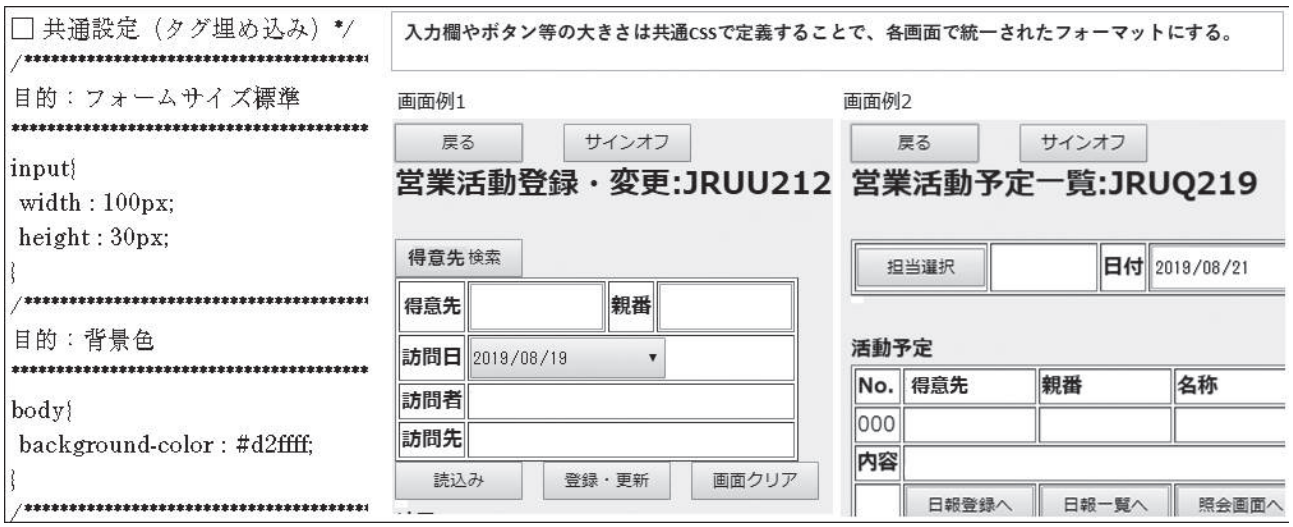
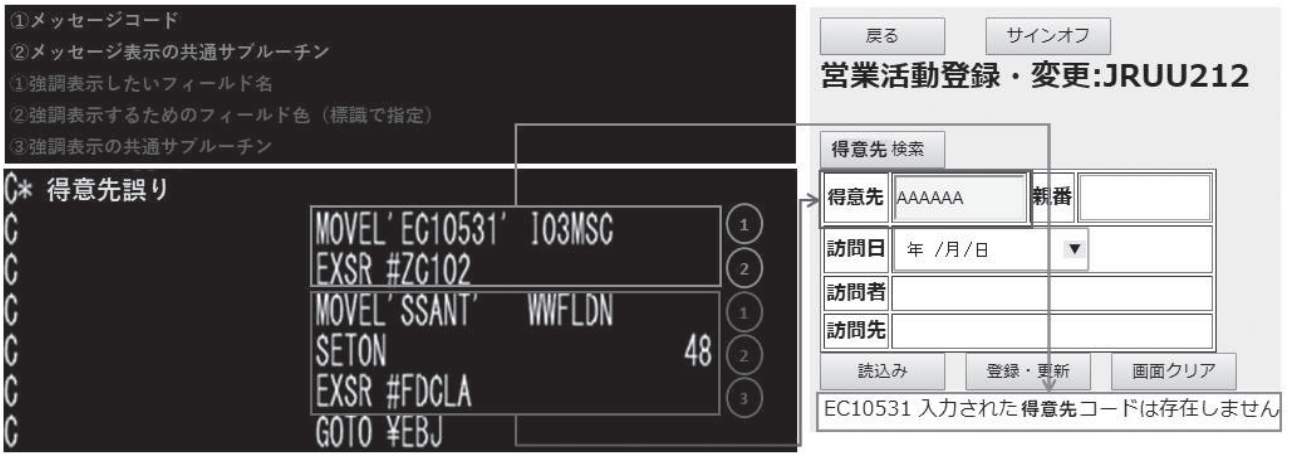


図2　ボタン処理の定型ソース

図3　エラー時の定型ソース



優秀賞

Valenceを使用した 温度･湿度ログ表示
―サーバー室内温度･湿度変化の見える化

中谷 佳史 様
株式会社保健科学西日本
情報管理課
顧問

株式会社保健科学西日本
http://www.hk-wj.co.jp/

医科学の研究と応用を通して人々の命を守り、医療業務に携わる人達を支援することを目標として、臨床検査事業を中心とした業務を行っている。最新機器やトレーサビリティ確保のためのバーコードシステムの導入により、医療機関、企業、教育機関などに向けて正確かつ迅速に検査情報を提供している。

業務課題

当社は臨床検査事業、検診事業などを主な事業として、医療の発展を通した社会貢献を目指して活動している。このような当社の業務の性格上、データの収集と分析を常に行っており、各種データの見える化を進めている。

本論文では、情報システム部門で管理するサーバールームの温度、湿度データを見える化した事例を紹介する。この取り組みは､以下の 2 点を狙いとしている。

・サーバールームの温度、湿度の異常値を迅速に発見できる態勢を確立し、システムを安定的に運用する。
・今回開発した見える化の仕組みを今後、医療検査機器の管理に応用する。

画面機能の詳細説明

温度・湿度データは、センサーを使って約 15 分ごとに IBM i に取り込んでいる。【図 1】IBM i データの見える化は、Valence App Builder を使用して実装した。【図 2】の画面には下記のデータが表示されている。

画面上部

サーバールームの温度と湿度を時系列の折れ線グラフで表示している。左軸の単位は温度（℃）と湿度（%）を表している。

画面下部

各回の測定値のログを一覧リスト形式で表示している。ここではページ切り替え機能とデータダウンロード機能を実装した。

Valence App Builder では、「ウィジェット」と呼ばれる部品の中から表示内容に合わせてグラフ等を選択する。今回、1 画面内に折れ線グラフと一覧リストの 2 つのウィジェットを配置したが、基本的にコーディングは行わずに、設定のみで画面を構築できた。DB 構造上、日付データと時間データの SQL による結合のみ行ったが、これを含めても開発に要した時間はわずか 30 分であった。

画面による業務課題の解決

今回の開発によりサーバールームの温度・湿度が視覚的、直感的に把握可能になった。また測定値ログのダウンロード機能により、統計資料の作成などにも役立てられる。さらに今後、Valence App Builder を使った IBM i データの見える化にも展望を開くことができた。次の開発として、医療検査機器の精度管理を考えている。各検査機器は精度の基準値が定められており、この基準を満たしているか定期的に測定し、各検査機器から IBM i にデータを取り込んでいる。

このデータのモニタリングに今回の手法を応用したい。さらに最新データを定期的に自動取得しながら、グラフ画面の切り替え表示を行う Valence の「キオスクモード」機能の実装も今後検討していきたい。 M

図1　温度・湿度センサーからのデータ取得

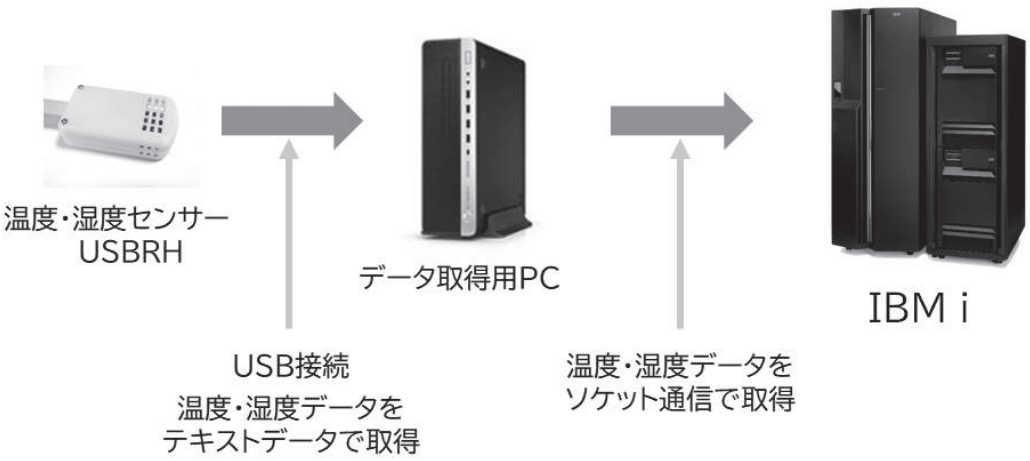


図2　温度・湿度ログ表示画面

優秀賞

RPGソースコードを Valence File Editorで改修

福島 利昭 様

株式会社ランドコンピュータ
代表取締役社長

株式会社ランドコンピュータ
https://www.rand.co.jp/

高校、大学等のコンピュータ教室で使われる「授業支援システム」の設計、開発、販売を行っており、学校等の教育関連施設に対して 5000 件以上の納入実績がある。業務ソリューションに必要なソフトウェア開発と、画像・音声処理などの機器製造をトータルに展開している。

業務課題

元号の変更に伴い、IBM i から出力している伝票の「平成」表記を西暦に変更する。そのために、プログラム内の固定値で管理している元号表記を編集する必要があった。

技術課題

RPG や CL を編集する際、5250 画面ではなく操作しやすい GUI 画面で編集したい。GUI でソースコードを編集するためのツールは存在するが、当社の IBM i のバージョンに対応するものは販売終了となり、入手できない。

技術課題の解決策

Valence ユーティリティの機能の 1 つである「File Editor」は、IBM i の DB やテーブルを参照・更新できるツールで、Web ブラウザで利用する Valence メニュー画面から簡単に呼び出せる。

Valence メニューは、大きなアイコンを配置しており、使い勝手がよい。【図 1】

また File Editor 画面は、最近使ったファイルがショートカット化されるため、頻繁に使用するファイル／ライブラリー名の都度入力は不要となる。【図 2】

この File Editor を使ってプログラムソースコードの編集を行うところが、今回工夫した活用方法である。編集手順は以下のとおり。

まず、ソースライブラリーから編集・確認したいメンバーを選択する。【図 3】

選択したメンバーの編集画面【図 4】で、編集したいソースの行をダブルクリックすると、編集用のダイアログが表示される。【図 5】

従来の方法【図 6】と比べると、File Editor を使ってソースコードの内容を簡単かつ確実に編集でき、とくに本事例の日付表示形式変更のような既存プログラムの特定箇所の編集作業には非常に有効だった。ちなみにコンパイルは 5250 画面で実行している。【図 7】

業務課題解決と効果

ブラウザベースなので、メインで使用している Mac からでもソースコードの確認、編集ができるようになった。ソースコードを Excel 形式で簡単にダウンロードできるので、バックアップ目的やソースコードの確認が楽になった。

今後も消費税変更対応など、IBM i のテーブル／ソースコードを変更する際に、Valence File Editor を利用する予定である。

M

図1　Valenceメニューの画面

図2　File Editorの画面

図3　ソースメンバーの選択

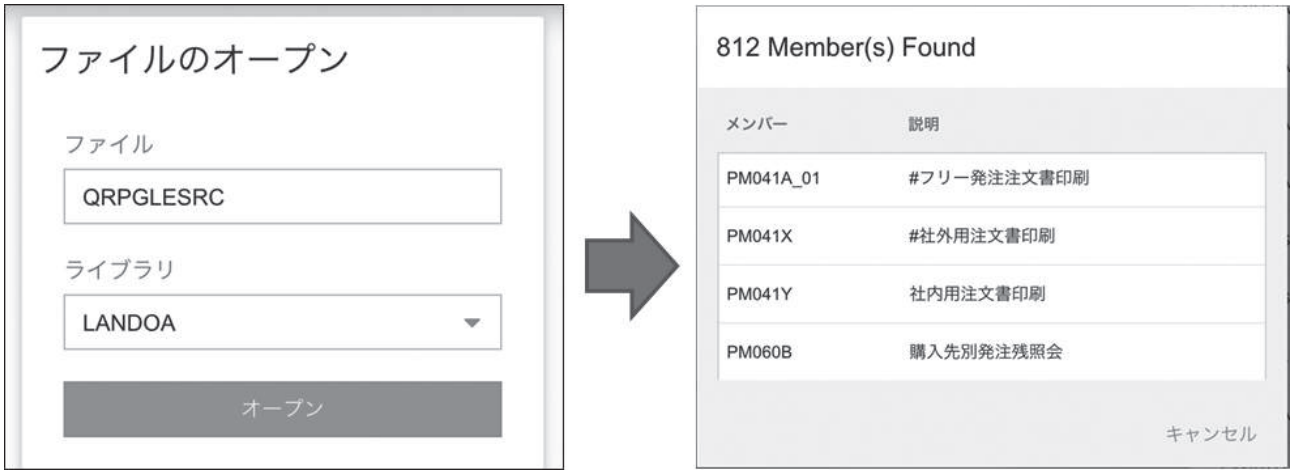


図4　File Editorのメンバー編集画面

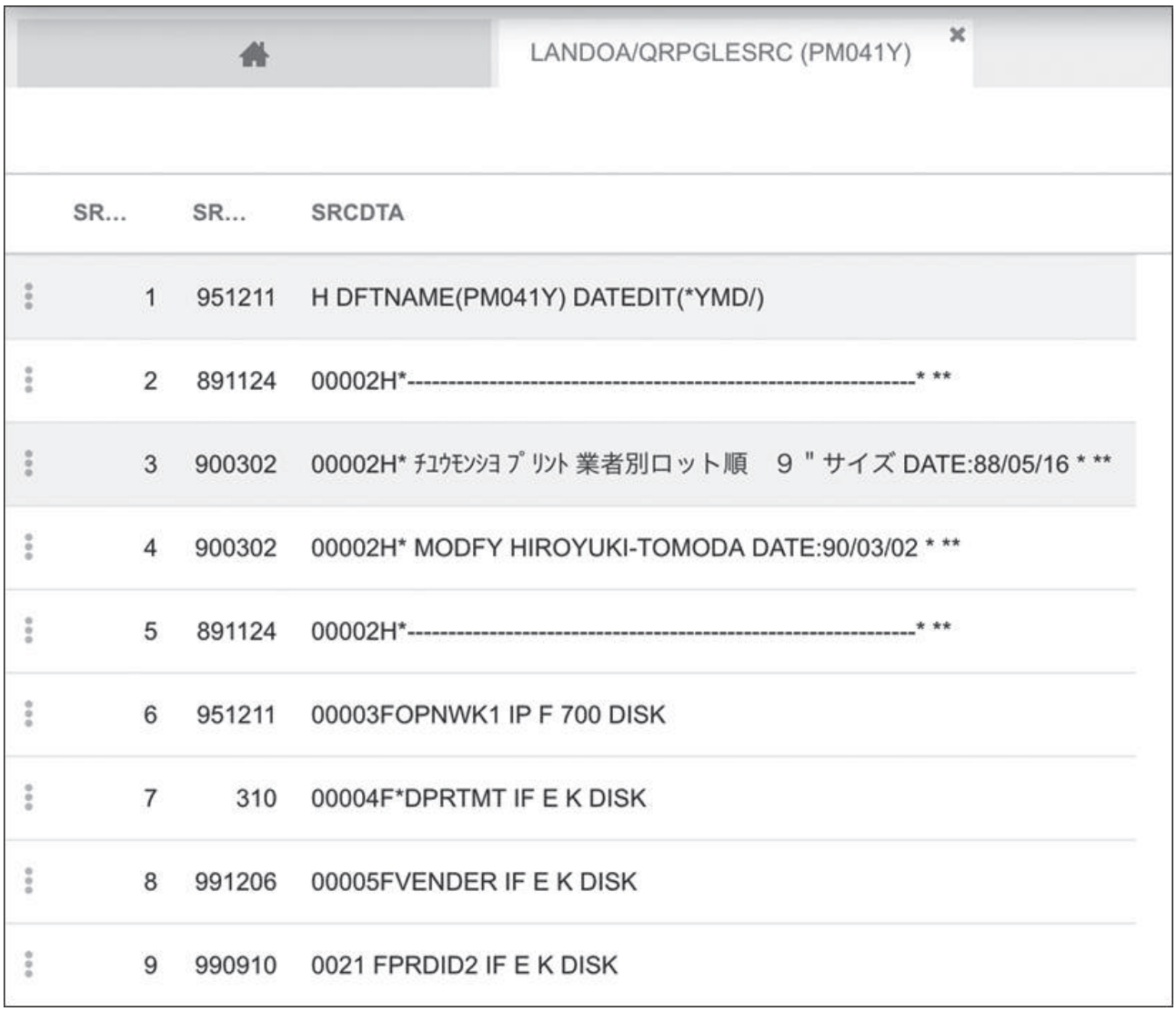


図5　編集用ダイアログの表示

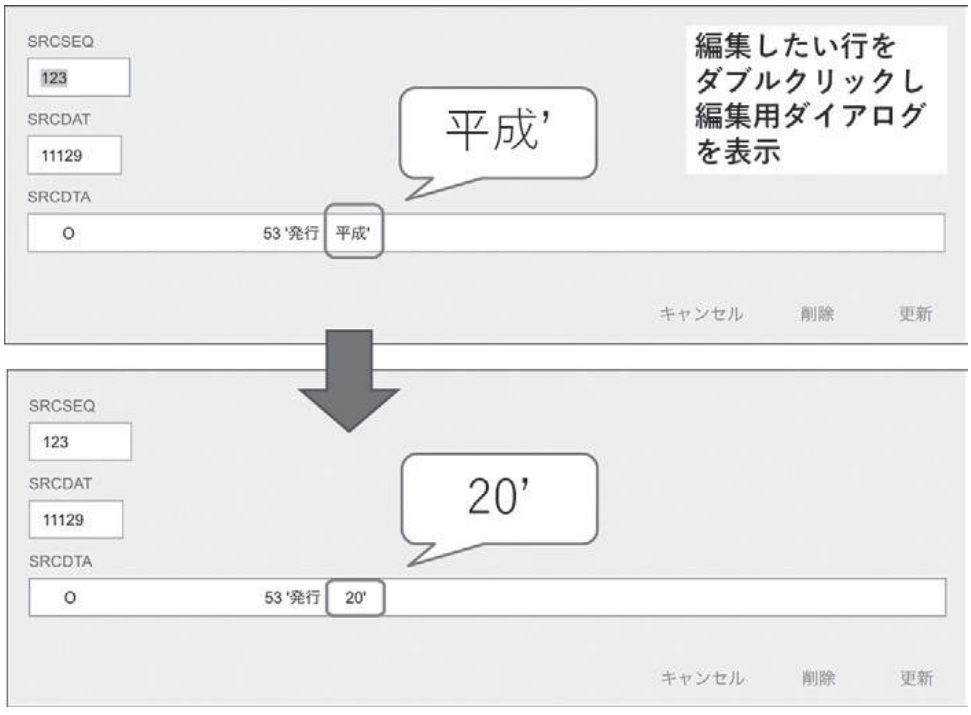


図6　5250画面で編集

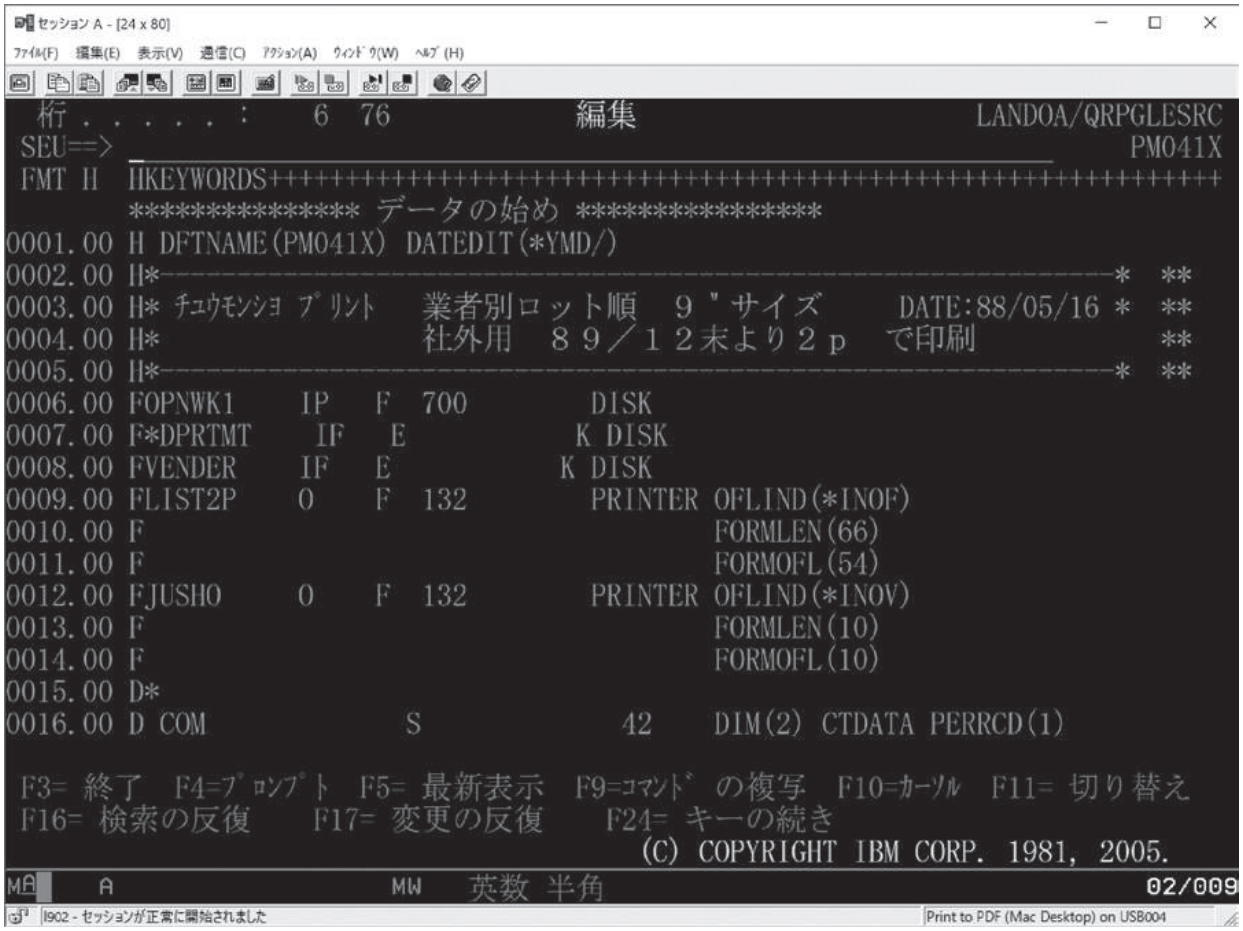
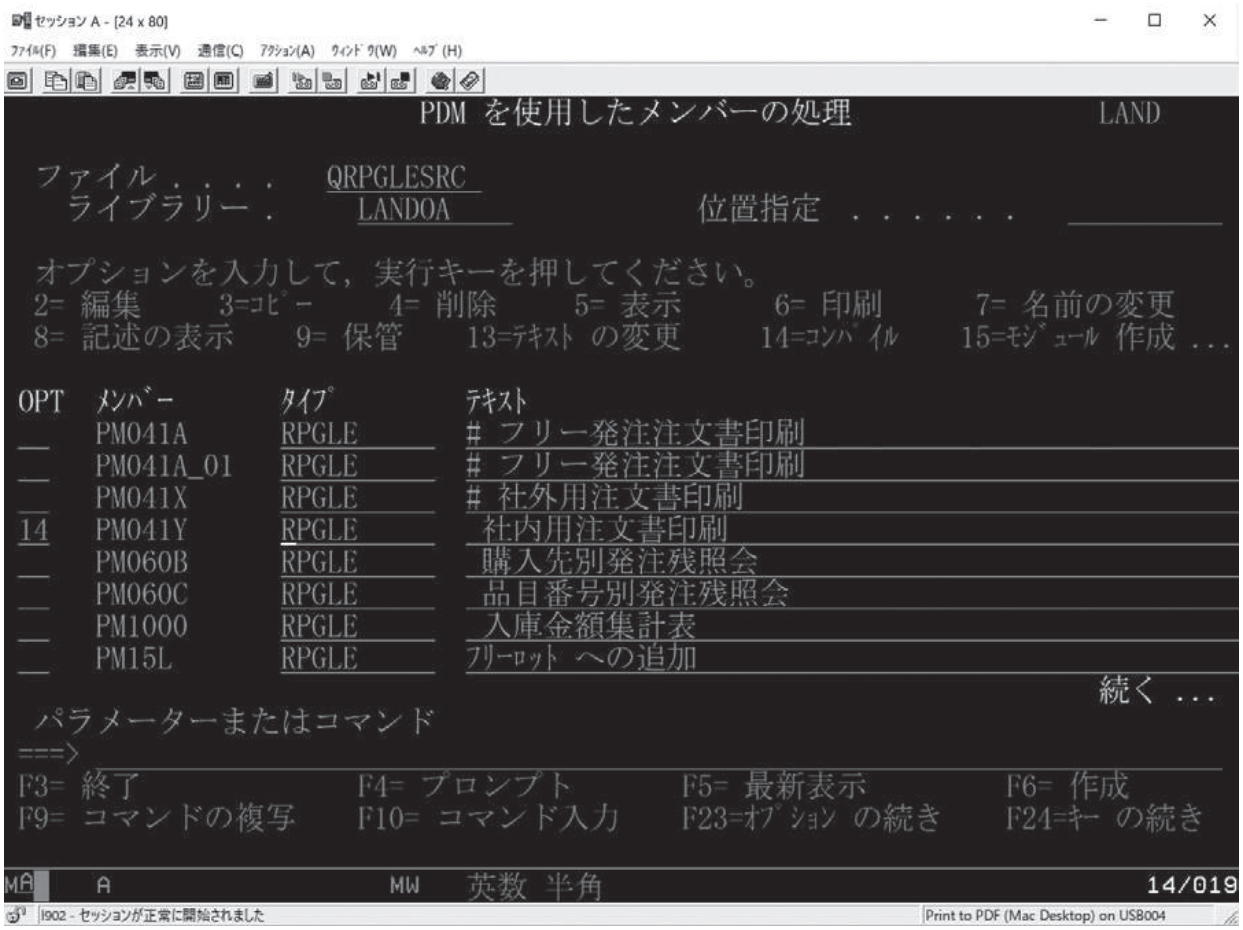


図7　コンパイルは5250で実行



Migaro. Special Report 2019

特別寄稿論文／ミガロ. スペシャルレポート

統合開発環境Cobosのご紹介
—RPG･SP4iの効率的な開発に

Julie Dumortier 様
Metrixware group
President

Metrixware group
https://www.metrixware.com/

Metrixware は 25 年以上にわたり、企業の基幹システムのモダナイゼーションおよび開発効率向上に役立つソフトウェアを提供してきた。IBM 汎用機および IBM i のソリューションを得意としている。本社はフランスで、欧州の銀行などをはじめ大企業から中堅企業まで顧客は世界中に広がっている。

2018 年より、Delphi/400、SmartPad4i（以下、SP4i）、Business4mobile の開発元 SystemObjects 社は、当社 Metrixware（メトリクスウェア）グループに加わることとなった。Metrixware は、IBM i およびメインフレームの開発を効率的に行うための統合開発環境、エミュレータなどのツールで、特に欧州の金融機関向けなどで多くの導入実績がある。

本レポートでは、当社の基幹ツールである Cobos 製品の仕組みと特徴を、IBM i、SP4i ユーザー向け機能を含めてご紹介する。

Cobosの概要

Cobos は、COBOL、RPG、PL/Ⅰ、汎用機のアセンブラといった汎用機系の言語を標準的なワークステーション上で開発するための SDE（スマートな開発環境）を提供する。【図 1】また、Cobos にプラグインを追加することにより、日々の開発・運用作業やメインフレームへのアクセスを自動化できる。

Cobos を利用することのメリットは、次のとおりである。【図 2】

- 他の言語（Java 等）の開発者が使用している標準的な開発環境が利用可能
- 開発を補助するツールによる生産性の向上
- カスタマイズや拡張が可能なオープンな環境

Cobosのシステム構成

Cobos 環境は、次の要素で構成されている。

- PC：Neon4.6 以降のバージョンの Eclipse および Java SE8
- 対象サーバーとして以下が利用可能
  - IBM z/OS：必要なコンポーネントを区分データセットに保管したメインフレームサーバー
  - IBM i：アプリケーションの保管・ビルド・配布を行うサーバー。Cobos は、Web・モバイルアプリ開発ツール SP4i との完全な統合を実現している
  - Linux または Windows サーバー：アプリケーションの保管・ビルド・配布を行う
- 構成管理ツール（オプション）：Subversion、Git/Gerrit、その他
- 継続的インテグレーションツール（オプション）：Jenkins、その他

プラグイン機能の紹介

Cobos は、Eclipse のプラグインとして開発された。このため、顧客のさまざまなニーズに適合するプラグイン機能を追加して、よりリッチな開発環境を継続的に構築することが可能となっている。

Cobos 環境は、2009 年のリリース以降も成長を続けており、以下のような拡張機能を提供して幅広いニーズに応えている。

図1　Cobos開発環境（RPG編集画面）

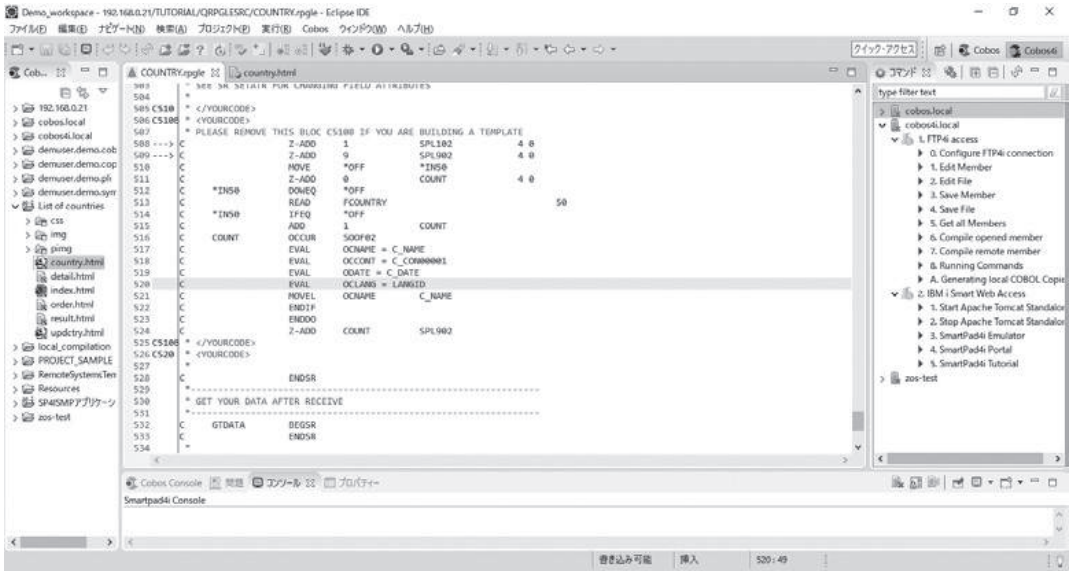


図2　Cobosの特徴

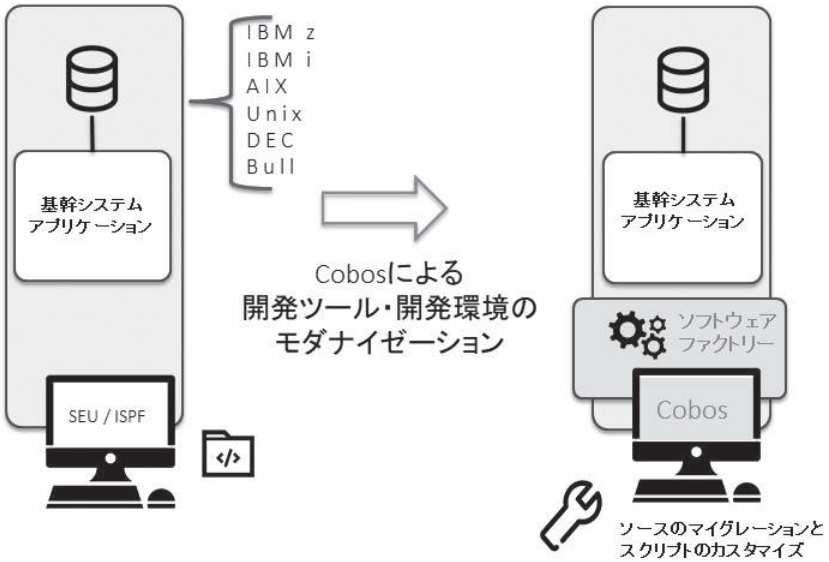
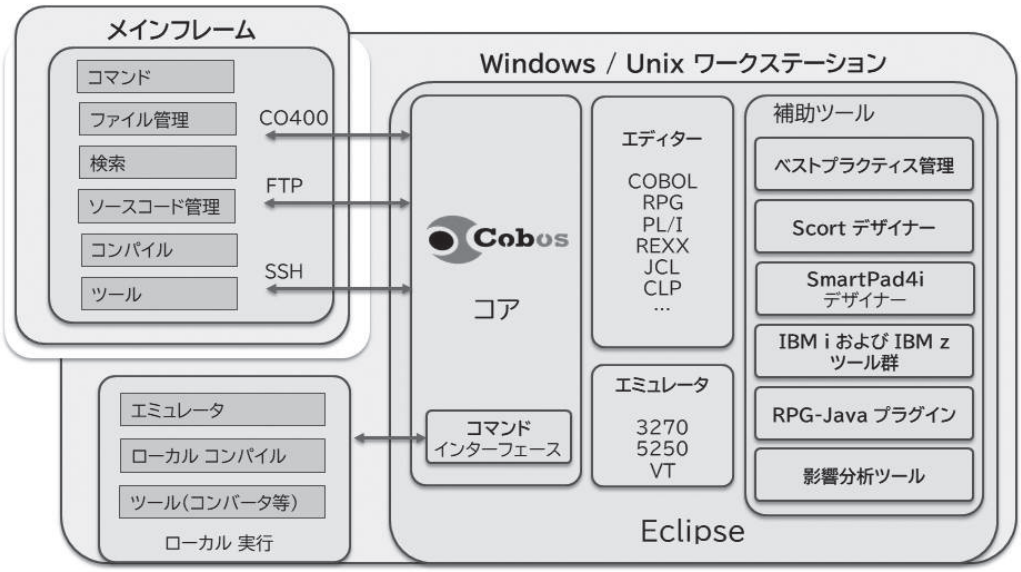


図3　Cobosの仕組み



・COBOL エディター：COBOL 開発環境を Eclipse の統合開発環境に統合
・RPG エディター：固定フォームおよびフリーフォーム RPG に対応して、RPG の構文着色（構文強調）、コンテンツアシスト（コード補完）、アウトラインビューの機能を利用できる。
・GnuCOBOL（旧 OpenCOBOL）コンパイラ：フリーの COBOL コンパイラ機能。これにより、ローカルワークステーション上でソースコードの構文チェックが可能となる。
・PL/1 エディター：構文着色（構文強調）、コンテンツアシスト（コード補完）、アウトラインビュー、ダークテーマ、メインフレームのリモートコンパイルの各機能を利用できる。
・JCL エディター：構文着色（構文強調）、コンテンツアシスト（コード補完）、アウトラインビュー、およびメインフレームのジョブ実行（サブミット）ができる。
・REXX エディター：オープンソースプロジェクトより移植
・Scort デザイナー：Scort をプラグインで組み込むことにより、COBOL、RPG プログラムを Web アプリケーションとして利用できる。
注：Scort は当社製ツール（https://www.metrixware.com/products/mis/ − lang = en）
・Cobos4i の各機能：IBM i 向けに以下のプラグインを用意している。

− SP4i プラグイン：Web およびモバイル向けアプリケーションの開発が可能となる。
− 5250 エミュレータプラグイン：5250 エミュレーションから IBM i サーバーにアクセスできる。本機能は SP4i 機能を利用している。
− FTP4i プラグイン：IBM i のプログラムへのアクセスと自動コンパイルを行う。

・Cobos4z の各機能：IBM z/OS メインフレーム向けに以下のプラグインを用意している。

− Z/Navigator プラグイン：メインフレーム上のデータセットに直接アクセスできる。
− Z/Jobs プラグイン：z/OS のジョブ管理を行う。
− Z/Search プラグイン：メインフレーム上で、特定の文字列を含むソースを検索する。
− je3270 プラグイン：3270 エミュレータによりメインフレームにアクセスする。

以上のほかにも Cobos は、アプリケーションマッピング、影響分析、DevOps 環境へのインターフェース、利用企業のニーズに応じたソフトウェアファクトリーなど、開発を支援するさまざまな拡張機能を提供する。

Cobosの仕組み

Cobos は Eclipse の標準的な機能拡張の仕組みを利用している。構造図は【図 3】のとおり。

DevOps との統合

Cobos は、専用のウィザードやスクリプトのカスタマイズにより、開発者の端末にオープンソースのソフトウェアファクトリー環境（DevOps）を組み入れることができる。【図 4】は、組み入れ可能な DevOps の一例である。

Cobosの新機能

最新版のバージョン 4.2.0 では、以下のような新機能を追加した。

・FTP4i 接続について、全角文字に対応（日本語対応）
・Scort デザイナーに Web サービス関連機能を追加【図 5】

− WSDL のインポート
− Web サービスに対して情報のリクエストと受け取りを行う COBOL コピーブックを生成
−ランタイム環境に配布するための Java クラスを生成
−アーティファクトをメインフレームおよび Web サーバーに配布
−生成したコードをテストタブによりテスト

・SP4i デザイナー機能を完全に組み込み【図 6】

−既存の SP4i プロジェクトを Eclipse プロジェクトとして取り込み
−同時に複数のプロジェクト、テンプレートを開ける
−すべてのデスクトップ機能を実装
− Eclipse のオリジナル機能を利用可能
− HTML エディター、Web ページエディター、デザイナーのすべてのツールを 1 か所に集約
−接続管理、配布ビューの使いやすさの改善

・コマンドビューの改善

− URL コマンドのパラメータに対応
−フォームの変数を使ってホストのユーザー ID を使用
−各リソースで使用している文字コードをフォーム上に表示

・Z/Navigator で、メインフレームのメンバーをコピー／ペーストするためのショートカットに対応

最後に

RPG、COBOL の開発者不足が叫ばれているが、若い開発者に人気のある Java 等の開発言語と比べて、言語仕様だけでなく、RPG や COBOL の開発環境が大きく異なっていることも一因と思われる。

Cobos はオープン系の開発手法に慣れた技術者を IBM i や IBM Z の世界に招き入れるのに最適なツールと考えている。昨年の SystemObjects 社との統合以来、IBM i/RPG 関連機能強化、SP4i との統合、全角文字対応（日本語対応）などの Cobos の改善を実施してきた。当社では、Cobos がミガロ. 製品ユーザー様の日々の開発業務の効率アップに貢献できることを願い、さらに機能改善を重ねていく予定である。

M

図4　DevOpsとの統合

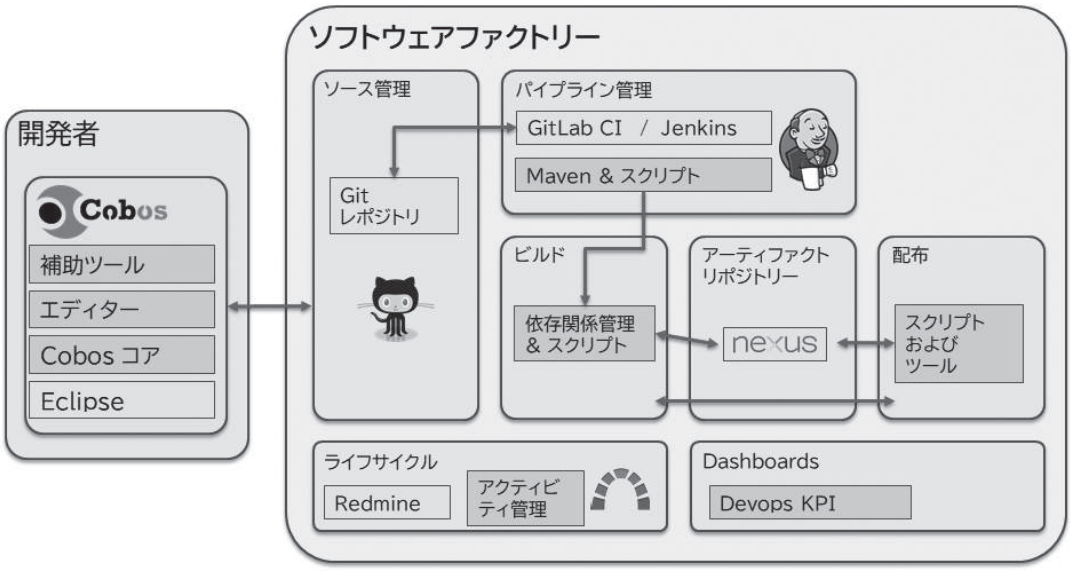


図5　Scort Webサービス連携

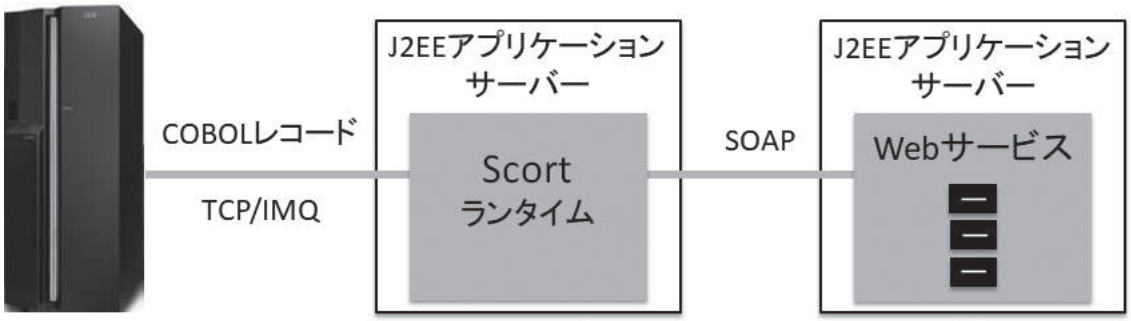
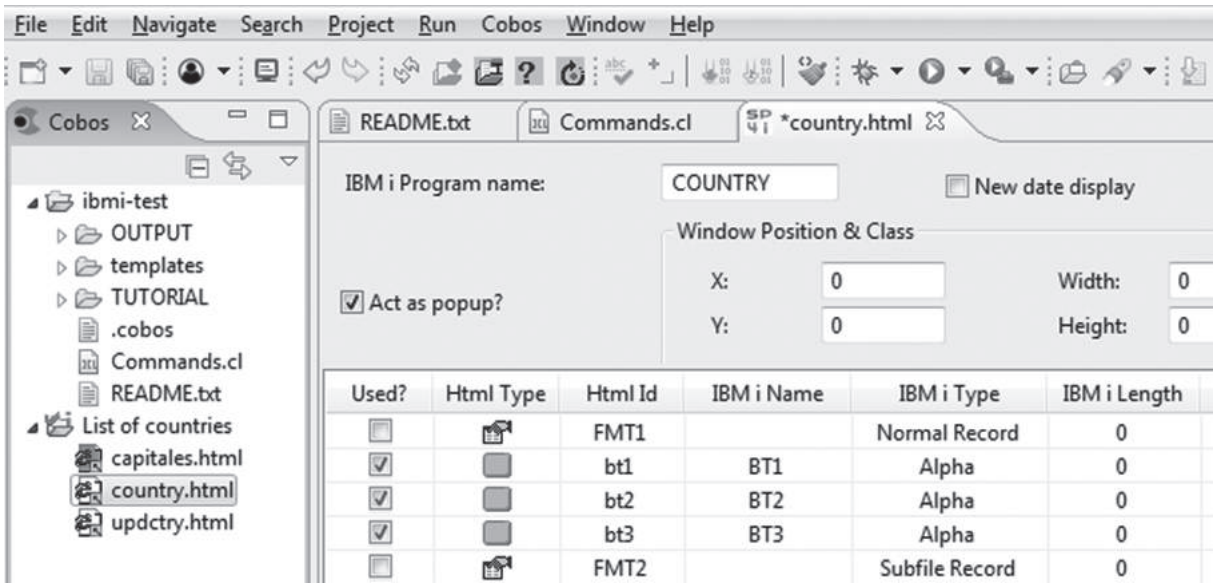


図6　SP4iの組み込み



Migaro. Technical Report 2019

SE 論文／ミガロ. テクニカルレポート

田村 洋一郎 / 宮坂 優大 / 都地 奈津美

株式会社ミガロ.

システム事業部 システム1課

[Delphi/400]

IBM iデータベースへのFTP送信手法の紹介

1. はじめに

外部システムのデータやユーザーが作成したデータを CSV ファイルに保存し、IBM i データベースに更新する際、SQL や RPG で更新処理を実装しようとすると、更新先のデータベースごとにプログラムを開発する必要がある。【図 1】

また iSeries Access for Windows や PCOMM のデータ転送機能を利用して更新する際も、データベースごとに設定ファイルを作成する必要がある。

本稿では、データベースごとにプログラムを開発することなく汎用的にデータを更新する手法として、FTP 送信処理を紹介する。FTP 処理の手法については、『ミガロ . テクニカルレポート 2014』の「大量データ処理テクニック ～ FTP を利用したデータ転送～」にも紹介されているので、ぜひこちらも参考にしていただきたい。

本稿では、まず第 2 章で IBM i データベースへダイレクトに FTP 送信する方法を紹介する。次に第 3 章では、第 2 章で作成した送信処理をもとに送信先のファイルレイアウトを考慮し、汎用的に送信する手法について紹介する。【図 2】

2. IBM iデータベースへのFTP送受信処理

2-1. FTP送受信処理について

ここでは「Indy」を使用し、TIdFTP コンポーネントを用いて FTP サーバーとの通信を行う方法、並びに FTP サーバーに IBM i データベースを設定し、CSV ファイルを送信する手法を紹介する。

「Indy」とは、Delphi/400 で使用できるオープンソースのネットワーク関連コンポーネント群のことであり、Delphi/400 に標準で付属している。また FTP（File Transfer Protocol、ファイル転送プロトコル）とはサーバー間、またはサーバー・クライアント間でファイル送受信を行う際に利用される手法の 1 つである。

略歴 田村 洋一郎
1983 年 9 月 27 日生まれ
2006 年 3 月 近畿大学 理工学部卒業
2006 年 4 月 株式会社ミガロ. 入社
2006 年 4 月 システム事業部配属

現在の仕事内容
RPG や Delphi/400 などの開発経験を経て、現在は要件定義から安定稼働フォローまで、システム開発全般に携わっている。

略歴 宮坂 優大
1982 年 11 月 19 日生まれ
2006 年 3 月 近畿大学 理工学部卒業
2006 年 4 月 株式会社ミガロ. 入社
2006 年 4 月 システム事業部配属

現在の仕事内容
主に Delphi/400 を利用したシステムの受託開発をメインに担当。Delphi および Delphi/400 のスペシャリストを目指して精進する日々である。

略歴 都地 奈津美
1989 年 8 月 19 日生まれ
2012 年 3 月 関西学院大学 理工学部卒業
2012 年 4 月 株式会社ミガロ. 入社
2012 年 4 月 システム事業部配属

現在の仕事内容
主に Delphi/400 を使用したシステム受託開発とシステム保守を担当。開発スキルの向上を目指し、日々精進している。

図1　IBM iデータベースへの取込例1(データベース毎にプログラム開発)

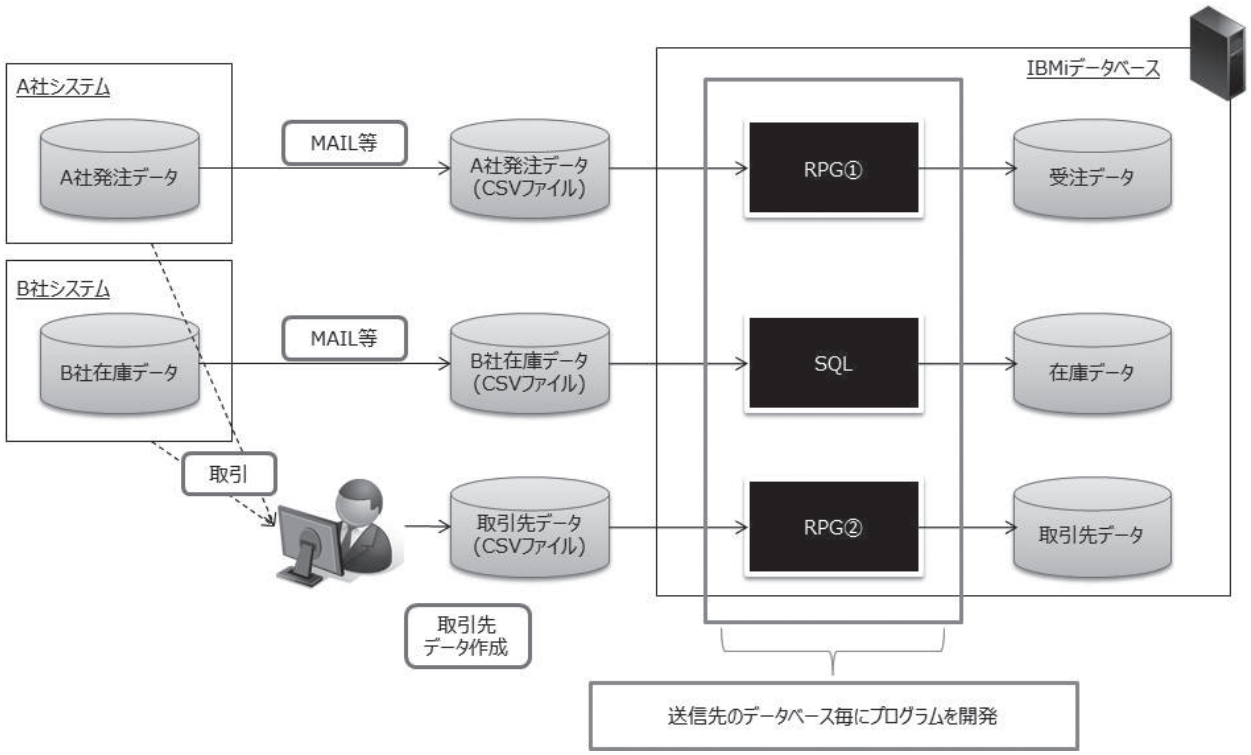
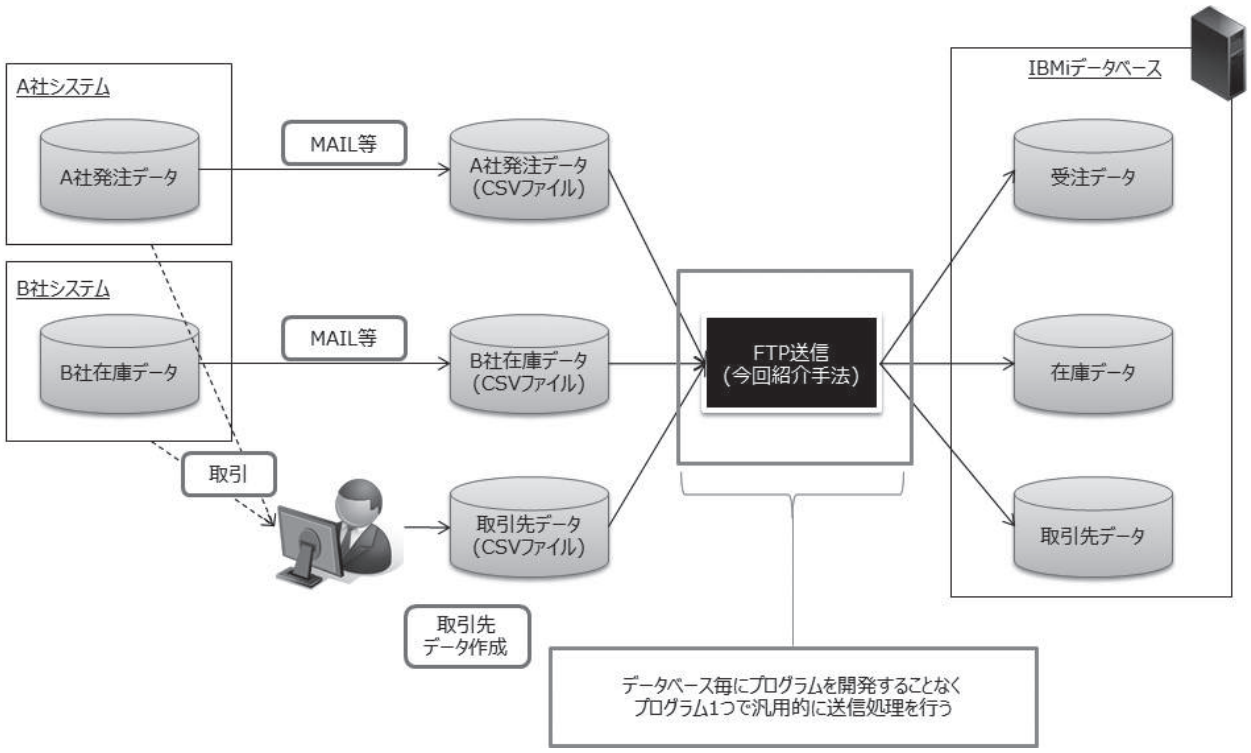


図2　IBM iデータベースへの取込例2(汎用的に送信処理を開発)



Indy のコンポーネントの 1 つである TIdFTP を利用することで、IBM i データベースやファイルサーバー等とクライアントの間で、FTP 通信を利用したファイルの送受信を行える。

なお、本章で作成しているプログラムは、Delphi/400 10.2Tokyo を使用している。

2-2. FTP送受信処理の前提条件

まず、IBM i データベースに対して FTP を利用したファイルの送受信を行うための前提条件として、IBM i データベースが FTP 送受信を許可しているかを確認する必要がある。

5250 画面で「NETSTAT *CNN」コマンドを実行すると、IPV4 接続状況の一覧が表示される。

接続状況一覧の中で、「ftp-con」または 21 番のローカルポートが「接続待機」になっていれば、FTP 接続が許可されている。【図 3】

存在しない場合は、5250 画面で「STRTCPSVR SERVER(*FTP)」コマンドを実行すると、FTP 接続を許可できる。また、IBM i で接続待機になっていても FTP 通信ができない場合、クライアント側のファイアウォールの設定で IBM i との FTP 通信が許可されているかを確認する。

次に、送信する CSV ファイルの書式について解説する。

CSV ファイルの項目の並び順および桁数は、送信先となるデータベースファイルのフィールド順と揃える必要がある。

今回使用する送信先のファイルのレイアウトを確認する。送信先のファイルは、次のように定義している。【図 4】

・半角文字フィールド（A タイプ 10 桁）
・全角文字フィールド（O タイプ 20 桁）
・数値フィールド（S タイプ 10 桁）

送信する CSV ファイルは、【図 5】のように作成した。

2-3. IBM i データベースへのアップロード処理

2-2. の準備事項が確認できたら、基本的な FTP 送信を行うプログラムを実際に作成していく。

(1) コンポーネントの配置

接続先情報と送信元ファイルを設定するための TEdit、FTP 通信を行うための TIdFTP、並びに TLabel や TButton をそれぞれ画面に配置する。【図 6】

送信元となる CSV ファイルは、画面上の「転送元 CSV ファイル」で設定する。また、送信先は「転送先ライブラリ名」「転送先ファイル名」に設定する。

(2) FTP 送信処理の実装

配置した「FTP 送信開始」ボタンの OnClick 処理を作成したら、実際に FTP 送信するソースコードを記述する。

次に、TIdFTP コンポーネントが持つプロパティや、今回の処理で行っているメソッドについて解説する。

①接続設定と接続処理

Host、Username、Password の各プロパティに、FTP 通信を行うための値を設定する。【ソース 1-①】

Host には接続先 IBM i データベースの IP アドレスを、Username と Password には IBM i データベースサインオン時のユーザー名とパスワードを設定する。

接続設定が完了したら、Connect メソッドで接続する。接続後は、try ～ finally で処理を囲み、処理終了後は Disconnect メソッドで接続を終了する。

② Passive プロパティ

パッシブモードで送信するため、Passive プロパティは True を設定する。【ソース 1-②】

③ TransferType プロパティ

ftASCII、ftBinary の 2 種類が存在し、ファイルの送受信をテキストファイル形式、バイナリ形式のどちらで行うかを設定できる。IBM i データベースは EBCDIC のため、ftBinary を設定する。なお、ftBinary は IdFTPCommon.pas で定義されているため、uses に「IdFTPCommon」を追加する。【ソース 1-③】

④送信元 CSV ファイルの文字列成型（CreateString メソッド）

FTP 送信する CSV ファイルは、カンマ区切りで構成されている。そのまま FTP 送信するとカンマ自体もデータとして送信されるので、文字列成型を行う。文字列の成型方法については以下、④-1. ～④-3. に記載する。

④-1. 送信元 CSV ファイルの読み込み

TStringList（slReadCSV）を生成し、LoadFromFile メソッドにて IBM i データベースへ FTP 送信する CSV ファイルを設定し、ファイルを読み込む。読み込んだ CSV ファイルをさらに 1 行ごとに読み込み、以下④-2. の処理で文字列成型を行う。【ソース 2-④-1】

④-2. 読み込んだファイルの文字列成型

TStringList（slRowCSV）の Comma Text に対して、④-1. で読み込んだデータ（slReadCSV）を代入する。slRowCSV の各項目を文字列として接続し、String 型変数（sCVSText）へ代入し、カンマを除去する。カンマを除去した sCSVText を、ファイル保存用 TStringList（slSaveCSV）に追加する。【ソース 2-④-2】

④-3. 文字列成型後の CSV ファイルを保存

カンマを除去した文字列を保管した TStringList（slSaveCSV） を、SaveToFile メソッドにて保存する。【ソース 2-④-3】

⑤ Put メソッド

各プロパティの設定および送信元 CSV ファイルの成型が完了したら、Put メソッドを呼び出し、FTP 送信処理を行う。引数で設定した条件で、ファイルを FTP 送信先にアップロードする。第 2 以降の引数は省略可能である。

第 1 引数：

アップロード元ファイルのフルパスを設定する。(1) で配置した「転送元 CSV ファイル」を設定する。

第 2 引数：

アップロード先のファイル名を設定する。(1) で配置した「転送先ライブラリ名」「転送先ファイル名」を設定する。ブランク設定時または省略時は、第 1 引

図3　IBM iのFTP送受信許可の確認

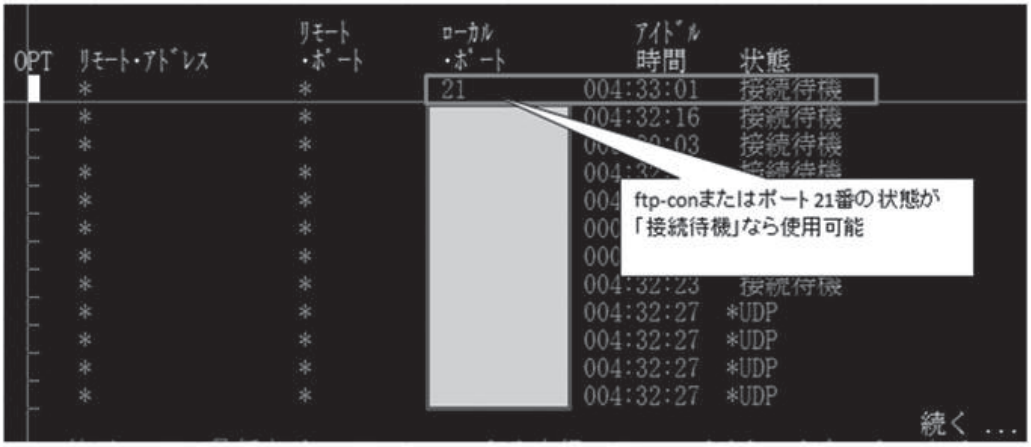


図4　送信先ファイルレイアウト

```
A*. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
********** データの始め **********************************************
A*-------------------------------------------------------------------*
A*       ファイル ID: FTPTEST
A*       ファイル名 :  サンプルファイル
A*-------------------------------------------------------------------*
A          R TESTR                        TEXT(' テスト ')
A            DATA1          10A          COLHDG(' 半角フィールド ')
A            DATA2          200          COLHDG(' 全角フィールド ')
A            DATA3          10S 0        COLHDG(' 数値フィールド ')
A*
********** データの終わり ********************************************
```

図5　送信元CSVファイル

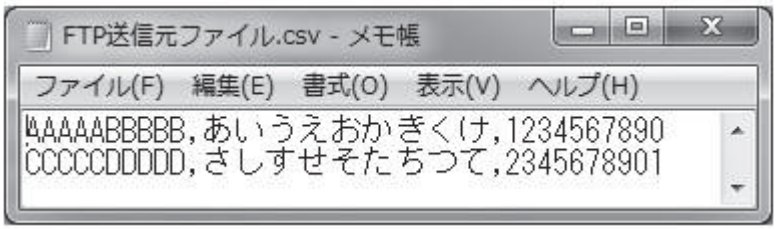
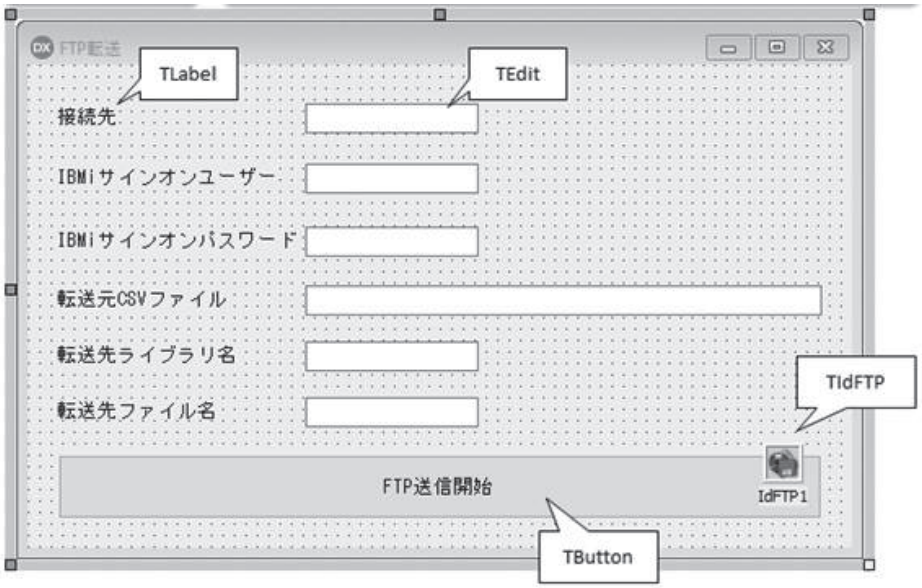


図6　コンポーネントの配置



数と同じファイル名になる。
第3引数：
ファイルダウンロード時、上書き保存するかどうかを設定する。Trueを設定すると、上書き保存する。省略時はFalse。

上記プログラムを実行し、「FTP送信開始」ボタンを押した結果、送信されたデータを5250画面の「RUN QUERY」コマンドで確認する。【図7】
これを見ると、FTP送信処理は正しく完了していたが、すべてのフィールドで文字化けが発生している。文字化けの原因はFTP送信時に、Windowsの文字コードとIBM iデータベースの文字コードのマッピングが正しく行われていないためである。
この文字化けを回避するには、マッピングテーブルを合わせる必要がある。FTP送信時に使用するマッピングテーブルは、「Quote」メソッドを使って設定できる。文字化けを発生させずに正しく送信するため、「TYPE C 943」を設定する。【ソース3】
修正したプログラムを実行し、再度「FTP送信開始」ボタンを押した結果、送信されたデータを5250画面の「RUN QUERY」コマンドで確認する。【図8】

このようにQuoteメソッドを使用することで、文字コードのマッピングが正しく行われ、送信元CSVファイルの内容がそのままIBM iデータベース上のファイルにFTP送信される。

3. ファイルレイアウトを考慮したFTP送信処理

3-1. ファイルレイアウトを考慮する理由

前述の第2章で、データベースへダイレクトにFTP送信する方法を紹介した。ただ前述の送信処理では、CSVファイルのフィールド長と送信先のフィールド長が一致していない場合、桁ずれを起こすことになる。【図9】のように、CSVファイルのフィールド長がデータベースのフィールド長よりも長い場合、FTP送信すると、この事象が発生する。
このようなケースに対応させるため、ここでは送信先のファイルレイアウトのフィールド情報を取得し、桁ずれを起こさないよう考慮した送信手法について紹介する。
この手法を採用することで、送信先データベースのファイルレイアウトが異なる場合でも、プログラムを個別に開発する必要がなく、汎用的な送信処理を実現できる。

3-2. ファイルレイアウトを考慮した送信処理

第2章で作成したサンプルプログラムを利用し、ファイルレイアウトを考慮したFTP送信を行うプログラムを作成していく。

(1) コンポーネントの配置

まず、第2章で作成したサンプルプログラムに対し、ファイルレイアウトを取得するための情報を入力するTEditならびにTLabelをそれぞれ画面に配置する。【図10】

(2) ファイルレイアウトの取得

FTP受信処理にて、ファイルレイアウトを取得する。第2章で作成したサンプルプログラムの、「FTP送信開始」ボタンのOnClick処理の最初に、ファイルレイアウト取得を行うよう処理を組み込む。

以下に、ファイルレイアウトを取得するのに必要なTIdFTPコンポーネントが持つプロパティや、今回の処理で行っているメソッドについて解説する。なお、第2章で解説しているプロパティ、メソッドについての解説は本稿では割愛する。また、DDSライブラリ・DDSオブジェクト・DDSファイルについて【図11】に補足する。

①接続設定と接続処理
第2章2-3の（2）FTP送信処理の実装を参考に、接続処理を行う。【ソース4-①】

②Getメソッド
各プロパティの設定が完了したら、Getメソッドを呼び出し、ファイルレイアウトの取得（FTP受信処理）を行う。引数で指定した条件で、ファイルをFTP受信先にダウンロードする。第3引数は省略可能である。【ソース4-②】

第1引数：
ダウンロード元のファイル名を設定する。(1) で配置した「DDSライブラリ名」「DDSオブジェクト名」「DDSファイル名」を設定する。
第2引数：
ダウンロード先ファイルのフルパスを設定する。(1) で配置した「DDS保管テキストファイル」を設定する。
第3引数：
ファイルダウンロード時、上書き保存するかどうかを設定する。Trueを設定すると上書き保存する。省略時はFalse。

以上のサンプルプログラムを実行した結果、FTP受信したファイルレイアウトのテキストファイルを確認する。今回は、第2章で使用したファイルのファイルレイアウトを取得する。【図12】

(3) 送信元CSVファイルの文字列成型

次に、(2) で受信したファイルレイアウトをもとに、送信元CSVファイルの文字列成型を行う。送信元CSVファイルは【図13】のように作成した。
送信元CSVファイルについて、第2章では桁数やシフト・コード文字（シフトイン文字、シフトアウト文字）を考慮した文字列を作成していたが、本章ではそれらを考慮せずに文字列を作成する。
まず、作成した送信元CSVファイルを、文字列成型せずにFTP送信する。IBM iデータベースへのFTP送信処理は、第2章で作成したサンプルプログラムで実行し、送信されたデータを5250画面の「RUN QUERY」コマンドで確認する。【図14】
送信結果を確認すると、ファイルレイアウトを考慮せずに作成したCSVファイルの場合、FTP送信結果に文字化けや桁ずれが発生していることがわかる。そこでFTP送信前に、送信元CSVファイルをファイルレイアウトに合わせて文字列成型を行い、FTP送信結果に文字化けや桁ずれが発生しないようにする。
文字列の成型方法については以下、①

ソース1　FTP送信プログラムソース

```
{*************************************************************************
 目的 : FTP送信開始ボタン押下
 引数 :
 戻値 :
*************************************************************************}
procedure TfrmSample.btnTransferFTPClick(Sender: TObject);
begin
  IdFTP1.Host     := edtDataBaseName.Text;  // 接続先
  IdFTP1.Username := edtSignon.Text;        // IBMiサインオンユーザー
  IdFTP1.Password := edtPassword.Text;      // IBMiサインオンパスワード
  IdFTP1.Passive  := True;                  // パッシブモード

  IdFTP1.Connect;                           // FTP接続
  try
    if IdFTP1.Connected then
    begin
      IdFTP1.TransferType := ftBinary;      // FTP転送タイプ

      // 文字列成型処理
      CreateString;

      try
        // FTP送信処理
        IdFTP1.Put(edtText.Text, edtLibrary.Text + '/' + edtFileName.Text);

        // 完了メッセージの表示
        ShowMessage('転送が終了しました');
      except
        // エラーメッセージの表示
        ShowMessage('転送中にエラーが発生しました');
      end;
    end;
  finally
    IdFTP1.Disconnect;  // FTP切断
  end;
end;
```

①接続設定
②Passiveプロパティ
①接続処理
③TransferTypeプロパティ
④送信元CSVファイルの文字列成型
⑤Putメソッド

ソース2　文字列成型メソッド

```
{*************************************************************************
 目的 : 文字列成型メソッド
 引数 :
 戻値 :
*************************************************************************}
procedure TfrmSample.CreateString;
var
  slReadCSV: TStringList;
  slSaveCSV: TStringList;
  slRowCSV : TStringList;
  sCSVText: String;
  iReadCSV, iRowCSV: Integer;
begin
  // 転送元ファイル成型用StringListの生成
  slReadCSV := TStringList.Create;
  slSaveCSV := TStringList.Create;
  slRowCSV  := TStringList.Create;
  try

    // CSVファイルの読み込み
    slReadCSV.LoadFromFile(edtText.Text);

    // 読み込んだデータをslRowCSVに保管
    for iReadCSV := 0 to slReadCSV.Count - 1 do
    begin
      // n行目のデータをCommaTextで取得
      slRowCSV.CommaText := slReadCSV[iReadCSV];

      // 成型後文字列保管用変数の初期化
      sCSVText := '';

      // カンマを取り除いた文字列に成型
      for iRowCSV := 0 to slRowCSV.Count - 1 do
      begin
        sCSVText := sCSVText + slRowCSV[iRowCSV];
      end;

      // カンマを取り除いた文字列を保存用StringListに保存
      slSaveCSV.Add(sCSVText);
    end;

    // 成型した転送元ファイルの保存
    slSaveCSV.SaveToFile(edtText.Text);

  finally
    // 内部生成したStringListの解放
    FreeAndNil(slRowCSV);
    FreeAndNil(slReadCSV);
    FreeAndNil(slSaveCSV);
  end;
end;
```

④-1.送信元CSVファイルの読み込み
④-2.読み込んだファイルの文字列成型
④-3.文字列成型後のCSVファイルを保存

～③に記載する。

① IBM i データベースの項目属性の取得

TStringList を生成し、LoadFromFile メソッドにて（2）で保存したテキストファイルを指定し、ファイルを読み込む。読み込んだテキストファイルを 1 行ごとに読み込み、各フィールドの項目属性を取得する。

ファイルレイアウトの構文規則として、30 ～ 34 桁目に桁数、35 桁目に文字タイプが保管されている。また、2 桁目に「*」が指定されている場合、コメント行となっている。これらの構文規則をもとに、フィールドの桁数と文字タイプを取得する。【ソース 5】

取得した桁数と文字タイプをもとに、以下②～③にて送信元 CSV ファイルの文字列成型を行う。【ソース 6 ～ 11】

文字列成型処理は、第 2 章サンプルプログラムの「CreateString」手続き内で、送信元ファイルのカンマ（,）を排除する処理（【ソース 2-④ -2】）の代わりに実装する。

② O タイプフィールド以外の文字列成型

O タイプフィールド以外の場合、ファイルレイアウトの桁数に対して桁数超過している場合、送信元文字列をファイルレイアウトの桁数に合わせてカットする。【ソース 7 ～ 8】

またファイルレイアウトの桁数に満たない場合、不足桁数分、S タイプフィールドの場合は 0 を送信元文字列の前方に付与【ソース 7】、A タイプフィールドの場合は半角スペースを送信元文字列の後方に付与する。【ソース 8】

③ O タイプフィールドの文字列成型

O タイプフィールドの場合、シフト・コード文字を考慮した文字列成型を行う。【ソース 9 ～ 11】

シフトアウト文字は 2 バイト文字の始まり、シフトイン文字は 2 バイト文字の終わりを表すので、1 バイト文字の直後に 2 バイト文字が発生した場合、または 2 バイト文字の直後に 1 バイト文字が発生した場合に、一時的に半角スペースを埋め込むことで、シフト・コード文字を考慮した文字列を成型し【ソース 10】、ファイルレイアウトの桁数に合わせて文字列をカットする。

FTP 送信する際は、一時的に埋め込んだシフト・コード文字用の半角スペースは不要なので削除する。【ソース 11】

このとき、送信元の文字列はバイト単位で処理する必要があるので、送信元文字列を AnsiString として扱うことに注意する。

④文字列成型した送信元ファイルの保存

①～③で文字列成型を行った TString List を、SaveToFile メソッドにより保存することで、ファイルレイアウトを考慮した FTP 送信用の送信元 CSV ファイルが作成できる。【図 15】

（4）FTP 送信処理

第 2 章 2-3（2）の⑤ Put メソッドを参考に、(3) で保存した送信元 CSV ファイルを IBM i データベースへ FTP 送信する。

上記プログラムを実行し、「FTP 送信開始」ボタンを押した結果、送信されたデータを 5250 画面の「RUN QUERY」コマンドで確認する。【図 16】

文字列成型前の送信では文字化けや桁ずれが発生していたが、シフト・コード文字を考慮した文字列成型を行うことにより、正常に FTP 送信できていることが確認できる。

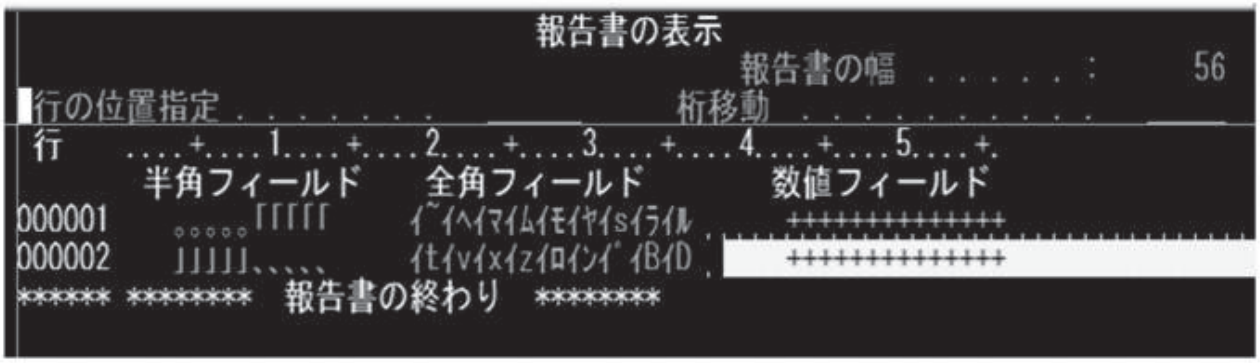
4. さいごに

以上本稿では、CSV ファイルのデータを IBM i データベースに送信する手法を紹介した。FTP 送信は処理自体が高速なので、速いレスポンスが要求されるアプリケーション開発でも効果が期待できる手法である。

また今回は IBM i データベースへの送信処理を紹介したが、IBM i データベースからの受信処理も行えるので、IBM i データベースとの転送手段の 1 つとして今回の手法を役立てていただければ幸いである。

M

図7　送信データの確認



ソース3　Quoteメソッドの追加

```
{*************************************************************************************
 目的 : FTP送信開始ボタン押下
 引数 :
 戻値 :
*************************************************************************************}
procedure TfrmSample.btnTransferFTPClick(Sender: TObject);
begin
  IdFTP1.Host     := edtDataBaseName.Text;  // 接続先
  IdFTP1.Username := edtSignon.Text;        // IBMiサインオンユーザー
  IdFTP1.Password := edtPassword.Text;      // IBMiサインオンパスワード
  IdFTP1.Passive  := True;                  // パッシブモード

  IdFTP1.Connect;                           // FTP接続
  try
    if IdFTP1.Connected then
    begin
      IdFTP1.TransferType := ftBinary;      // FTP転送タイプ
      IdFTP1.Quote('TYPE C 943');           // Quoteタイプの指定

      // 文字列成型処理
      CreateString;

      try
        // FTP送信処理
        IdFTP1.Put(edtText.Text, edtLibrary.Text + '/' + edtFileName.Text);

        // 完了メッセージの表示
        ShowMessage('転送が終了しました');
      except
        // エラーメッセージの表示
        ShowMessage('転送中にエラーが発生しました');
      end;
    end;
  finally
    IdFTP1.Disconnect;  // FTP切断
  end;
end;
```

Quoteメソッドの追加

図8　Quoteメソッド追加後の送信データの確認

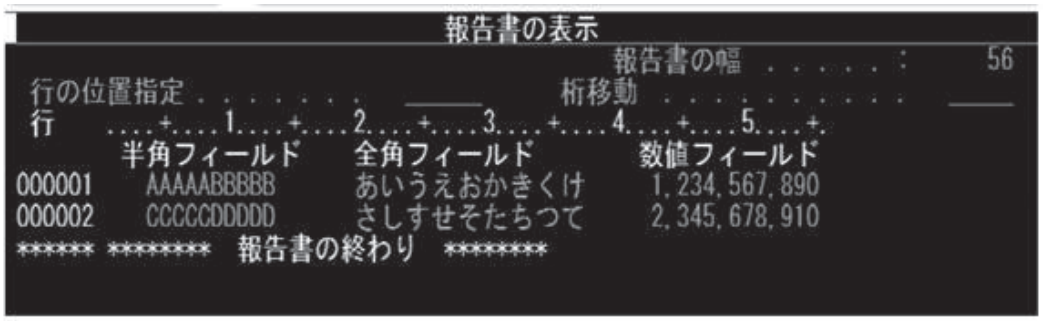


図9　フィールド長の違いによる桁ずれのイメージ

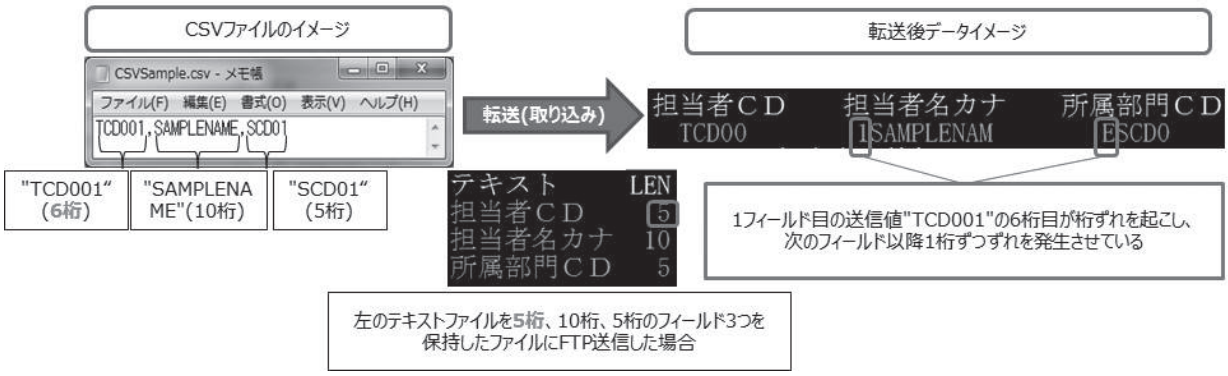


図10　コンポーネントの配置

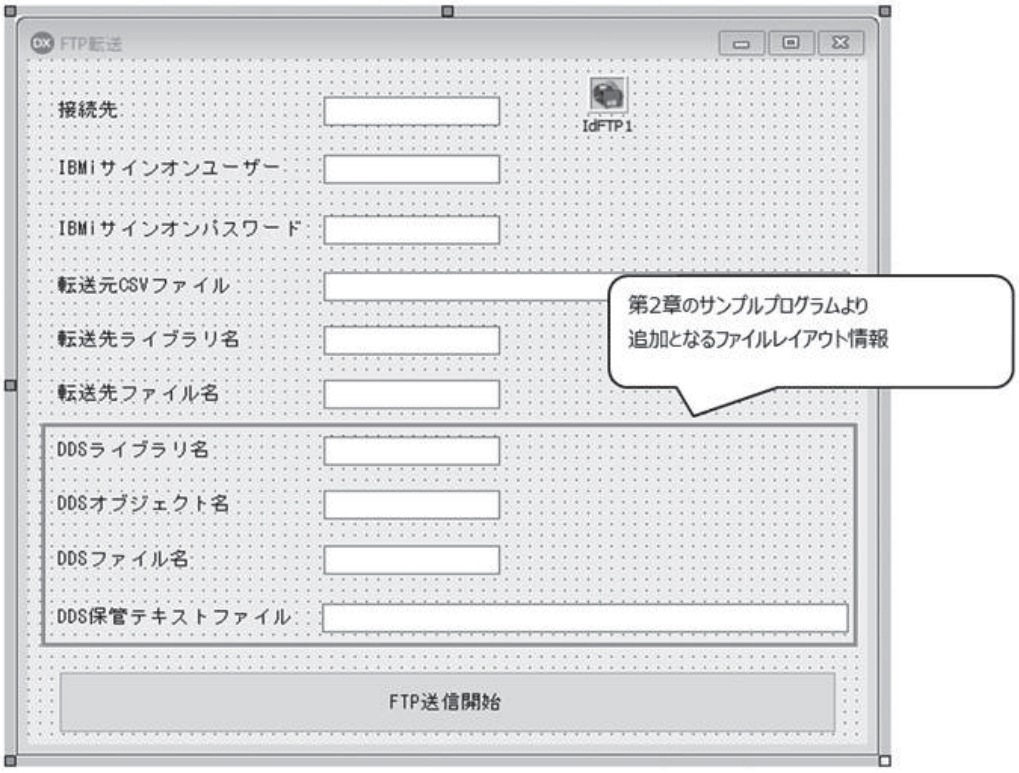
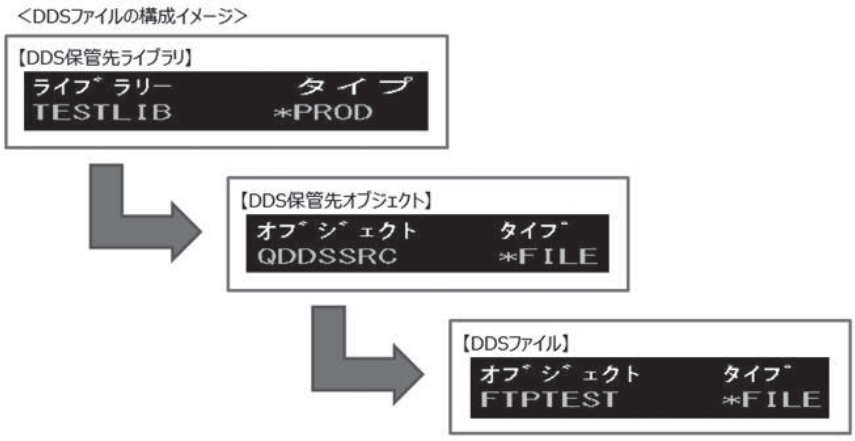


図11　DDSファイルの構成イメージ



<上記イメージのファイルレイアウトを取得するサンプル画面のTEdit指定例>

ソース4　FTP受信によるファイルレイアウトの取得

```
// DDSの取得
IdFTP1.Host     := edtDataBaseName.Text;  // 接続先
IdFTP1.Username := edtSignon.Text;        // IBMiサインオンユーザー
IdFTP1.Password := edtPassword.Text;      // IBMiサインオンパスワード
IdFTP1.Passive  := True;                  // パッシブモード

IdFTP1.Connect;                           // FTP接続
try
  if IdFTP1.Connected then
  begin
    IdFTP1.TransferType := ftBinary;      // FTP転送タイプ
    IdFTP1.Quote('TYPE C 943');           // Quoteタイプの指定

    try
      // FTP受信処理
      IdFTP1.Get(edtDDSLibrary.Text + '/' +  // 転送元：ライブラリ名
                 edtDDSObject.Text  + '.' +  //         オブジェクト名
                 edtDDSFileName.Text,        //         ファイル名
                 edtDDSText.Text,            // 転送先：テキストファイル名
                 True);                      // 上書き保存
    except
      // エラーメッセージの表示
      ShowMessage('DDS取得中にエラーが発生しました');
    end;
  end;
finally
  IdFTP1.Disconnect;  // FTP切断
end;
```

①接続設定と接続処理

②Getメソッド

図12　ファイルレイアウトの取得結果

図13　IBM iへFTP送信するテキストファイル

(Ctrl)

【送信元CSVの作成内容】
1行目：全項目、ファイルレイアウトに準じた桁数の1バイト文字を入力
2行目：半角フィールド（1項目目）のみ、桁数を超過した1バイト文字を入力
3行目：全角フィールド（2項目目）のみ、桁数を超過した2バイト文字を入力
4行目：全角フィールド（2項目目）のみ、桁数を超過した1バイト文字と2バイト文字を混合入力

図14　文字列成型前のFTP送信結果

	半角フィールド	全角フィールド	数値フィールド
000001	1234567890	12345678901234567890	1,234,567,890
000002	1234567890	AAA12345678901234567	8,901,234,567
000003	ABCDEFGHIJ	++++++++++++++++++++	0
000004	1234567890	++++++++++++++++++++	0

【FTP送信結果】
1行目：正常に転送完了
2行目：半角フィールドで超過桁数分、全角フィールド・数値フィールドにて桁ずれが発生
3行目：全角フィールドにて文字化けが発生、数値フィールドにて桁ずれが発生
4行目：全角フィールドにて文字化けが発生、数値フィールドにて桁ずれが発生

ソース5　ファイルレイアウトから項目属性を取得

```
// ファイルレイアウトの読み込み用StringListの生成
slDDS := TStringList.Create;

try
  // ファイルレイアウトの読み込み
  slDDS.LoadFromFile(edtDDSText.Text);

  // フィールドの開始桁
  iStrLen := 1;

  // ファイルレイアウトの行数分処理を繰り返す
  for iDDS := 0 to slDDS.Count - 1 do
  begin
    // コメント行の場合、次の処理へ
    if (Copy(slDDS[iDDS], 7, 1) = '*') then
    begin
      Continue;
    end;

    // 文字タイプ
    sType := Copy(slDDS[iDDS], 35, 1);

    // 桁数
    iKeta := StrToIntDef(Trim(Copy(slDDS[iDDS], 30, 5)),0);

    // 桁数が取得できていない場合、次の処理へ
    if (iKeta = 0) then
    begin
      Continue;
    end;

    // ※※ 以下、(3)-③～④(文字列成型)の処理を記述 ※※



    // フィールドの開始桁
    iStrLen := iStrLen + iKeta;
  end;

  // 成型した転送元ファイルの保存
  slSaveCSV.SaveToFile(edtText.Text);
finally
  FreeAndNil(slDDS);
end;
```

ソース6　文字列成型①処理の流れ

```
// 成型後の文字列の初期化
sCSVText_After := '';

// ファイルレイアウトを基に転送元ファイルを成型
for iReadCSV := 0 to slReadCSV.Count - 1 do
begin
  slRowCSV.CommaText := slReadCSV[iReadCSV];

  // 初期化
  iRowCSV        := 0;    // CSV行情報の読み込みフィールド番号
  sCSVText_After := '';   // 成型後の文字列

  // ファイルレイアウトの行数分処理を繰り返す
  for iDDS := 0 to slDDS.Count - 1 do
  begin
    // ※※ 以下、(3)-①(IBMiデータベースの項目属性の取得)の処理を記述 ※※


    // 処理対象のフィールドの文字列を取得
    sCSVText := slRowCSV[iRowCSV];

    // ※※ 以下、(3)-②～③の処理を記述 ※※



    // CSV行情報の読み込みフィールド番号
    iRowCSV := iRowCSV + 1;
  end;

  // 成型した文字列を保存用StringListに保存
  slSaveCSV.Add(sCSVText_After);
end;
```

sCSVTextは、バイト単位での処理を行う為、
AnsiString型の変数として定義する

ソース7　文字列成型②Sタイプフィールド

```
// Sタイプフィールドの場合
if (sType = 'S') then
begin
  // 桁数に満たない場合、不足桁分の0を文字列前方に付与
  if (Length(sCSVText) < iKeta) then
  begin
    for i := Length(sCSVText) + 1 to iKeta do
    begin
      sCSVText := '0' + sCSVText;
    end;
  end
  else
  // 桁数を超える場合、超過分の文字を削除
  if (Length(sCSVText) > iKeta) then
  begin
    sCSVText := Copy(sCSVText, 1, iKeta);
  end;

  // 成型した文字列を保管
  sCSVText_After := sCSVText_After + sCSVText;
end;
```

桁数不足

桁数超過

ソース8　文字列成型③Aタイプフィールド

```
// Aタイプフィールドの場合
if (sType = 'A') then
begin
  // 桁数に満たない場合、不足桁分の半角スペースを文字列後方に付与
  if (Length(sCSVText) < iKeta) then
  begin
    for i := Length(sCSVText) + 1 to iKeta do
    begin
      sCSVText := sCSVText + ' ';
    end;
  end
  else
  // 桁数を超える場合、超過分の文字を削除
  if (Length(sCSVText) > iKeta) then
  begin
    sCSVText := Copy(sCSVText, 1, iKeta);
  end;

  // 成型した文字列を保管
  sCSVText_After := sCSVText_After + sCSVText;
end
```

桁数不足

桁数超過

ソース9　文字列成型④Oタイプフィールド

```
// Oタイプフィールドの場合、シフト・コード文字を考慮した文字列の成型
if (sType = 'O') then
begin
  // 桁数に満たない場合、不足桁分の半角スペースを文字列後方に付与
  if (Length(sCSVText) < iKeta) then
  begin
    for i := Length(sCSVText) + 1 to iKeta do
    begin
      sCSVText := sCSVText + ' ';
    end;
  end;

  // シフト・コード文字を考慮した文字列に変換
  sCSVText := AddSISO(sCSVText);

  // 文字列成型結果文字列を桁数分でカット
  sCSVText := Copy(sCSVText, 1, iKeta);

  // 最終桁が全角の1バイト目の場合、半角スペースに置き換える
  if (ByteType(sCSVText, iKeta) = mbLeadByte) then
  begin
    sCSVText := Copy(sCSVText, 1, iKeta - 1) + ' ';
  end
  // 最終桁が全角2バイト目の場合、シフト・コード文字部分があふれるため
  // 半角スペースに置き換える
  else
  if (ByteType(sCSVText, iKeta) = mbTrailByte) then
  begin
    sCSVText := Copy(sCSVText, 1, iKeta - 2) + '  ';
  end;

  // シフト・コード文字を考慮した文字列に変換
  sCSVText := RmvSISO(sCSVText);

  // 成型した文字列を保管
  sCSVText_After := sCSVText_After + sCSVText;
end
```

ソース10

ソース11

ソース10　文字列成型⑤Oタイプフィールド(シフト・コード文字の付与)

```
{*******************************************************************************
 目的：文字列に対して、シフト・コード文字の位置にダミーの半角スペースをセット
 引数：AStr - 元の文字列
 戻値：成型された文字列
*******************************************************************************}
function TfrmSample.AddSISO(AStr: AnsiString): String;
var
  i: Integer;
  S: AnsiString;
begin
  // 初期化
  // 1バイト目が全角の場合
  if (ByteType(AStr, 1) = mbLeadByte) then
  begin
    S := ' ';
  end
  // 1バイト目が半角の場合
  else
  begin
    S := '';
  end;

  // 文字列を確認し、全角と半角の切替ポイントにダミーの半角スペースをセット
  for i := 1 to Length(AStr) do
  begin
    // 1文字ずつ保管
    S := S + AStr[i];

    // 現在の文字が半角 かつ 1文字先が全角の1バイト目の場合
    // シフト・コード文字の代わりに半角スペースを付与
    if (ByteType(AStr, i) = mbSingleByte) then
    begin
      if (ByteType(AStr, i + 1) = mbLeadByte) then
      begin
        S := S + ' ';
      end;
    end;

    // 現在の全角2バイト目 かつ 1文字先が半角の場合
    // シフト・コード文字の代わりに半角スペースを付与
    if (ByteType(AStr, i) = mbTrailByte) then
    begin
      if (ByteType(AStr, i + 1) = mbSingleByte) then
      begin
        S := S + ' ';
      end;
    end;
  end;

  // 結果をセット
  Result := String(S);
end;
```

ソース11　文字列成型⑥Oタイプフィールド(シフト・コード文字の削除)

```
{*******************************************************************************
 目的：文字列に対して、シフト・コード文字の位置の半角スペースをカット
 引数：AStr - 元の文字列(※シフト・コード文字考慮済みの文字列)
 戻値：成型された文字列
*******************************************************************************}
function TfrmSample.RmvSISO(AStr: WideString): String;
var
  i: Integer;
  bSI: Boolean;
begin
  // 初期化
  Result := '';
  bSI    := False;

  // 文字単位で後ろからカウント
  for i := Length(AStr) downto 2 do
  begin
    // 半角スペース(シフト・コード文字)の場合、セットしない
    if (bSI) then
    begin
      bSI := False;
    end
    else
    // 半角スペース(シフト・コード文字) かつ 1文字前が全角の場合、セットしない
    if (AStr[i] = ' ') and (Length(AnsiString(AStr[i - 1])) = 2) then
    begin
      // 何もしない
    end
    // 上記以外の場合、文字をセット
    else
    begin
      Result := AStr[i] + Result;

      // 全角かつ1文字前が半角スペース(シフト・コード文字)のとき、フラグセット
      if (Length(AnsiString(AStr[i])) = 2) and (AStr[i - 1] = ' ') then
      begin
        bSI := True;
      end;
    end;
  end;

  // 全角始まりでない場合は1文字目(シフト・コード文字でない)を最後に足す
  if (not bSI) then
  begin
    Result := AStr[1] + Result;
  end;
end;
```

図15　文字列成型後の送信元ファイル

＜文字列成型前の送信元CSVファイル＞

＜文字列成型後の送信元CSVファイル＞

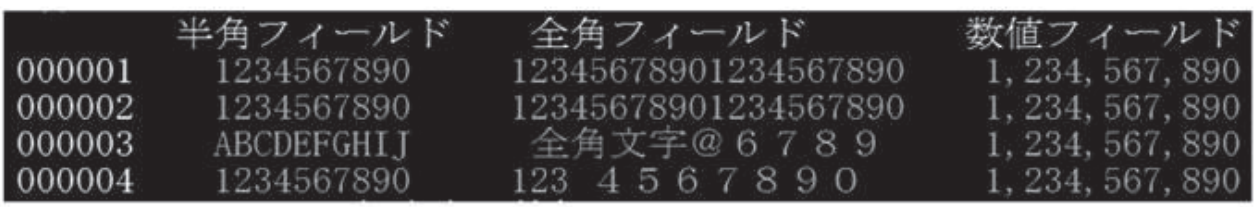
【送信元CSVファイルの文字列成型内容】
1行目：文字列成型前からファイルレイアウトに準じているため、文字列成型前と同じ文字列
2行目：半角フィールド（1項目目）のみ、桁数を超過した文字列を削除
3行目：全角フィールド（2項目目）のみ、シフト・コード文字を考慮＆桁数を超過した文字列を削除
4行目：全角フィールド（2項目目）のみ、シフト・コード文字を考慮＆桁数を超過した文字列を削除

図16　文字列成型後のFTP送信結果

＜文字列成型前のFTP送信結果＞

	半角フィールド	全角フィールド	数値フィールド
000001	1234567890	123456789012345678901	1,234,567,890
000002	1234567890	AAA123456789012345678	8,901,234,567
000003	ABCDEFGHIJ	++++++++++++++++++++	0
000004	1234567890	++++++++++++++++++++	0

＜文字列成型後のFTP送信結果＞



	半角フィールド	全角フィールド	数値フィールド
000001	1234567890	123456789012345678901	1,234,567,890
000002	1234567890	123456789012345678901	1,234,567,890
000003	ABCDEFGHIJ	全角文字@６７８９	1,234,567,890
000004	1234567890	123 ４５６７８９０	1,234,567,890

畑中 侑

株式会社ミガロ.

システム事業部 システム2課

[Delphi/400]

FireMonkeyの活用 カメラコンポーネントを使ったアプリ

略歴
1983年7月6日生まれ
2006年3月 京都産業大学 法学部卒業
2006年4月 株式会社ミガロ. 入社
2006年4月 システム事業部配属

現在の仕事内容
システムの受託開発を担当しており、要件確認から納品・フォロー、保守作業に至るまで、システム開発全般に携わっている。

1. はじめに

Delphi/400は登場当初より、Windowsネイティブ開発環境としてVCLフレームワークを提供してきたが、XE3から新たにWindows、Mac対応が始まり、現在ではWindows、Mac、iOS、Androidの4つのプラットフォームに対応したFireMonkeyフレームワークが備わっている。

VCLフレームワークはWindows APIをラッピングしたフレームワークであるのに対して、FireMonkeyフレームワークは、OS固有のAPIに依存しないため､マルチデバイス化が可能である。

VCLとFireMonkeyとでフレームワークが異なるといっても、開発者がどちらを採用しても、コンポーネントを画面に配置し、必要なイベントを実装するといった開発手法は同じだ。業務システムにもマルチデバイスが多用される昨今、4つのプラットフォームに対応したマルチデバイス用のFireMonkeyフレームワークを使わない手はない。

今回は、近年どのデバイスにも搭載されているカメラ機能に注目し、FireMonkeyフレームワークを利用したマルチデバイス対応の写真撮影アプリの開発手法を説明することで、FireMonkeyフレームワークの活用方法の一端を紹介する。

2. TActionを利用した写真撮影

TActionを利用することで、簡単に写真が撮影できる。TActionを利用するためにTActionListを用いるが、それ以外には、カメラを起動するTButton、撮影した画像を表示するTImageと、合計3つのコンポーネントをフォームに配置するだけで写真撮影アプリが実現できる｡実装方法も至ってシンプルである。作成方法を以下に紹介する。

作成手順

～TActionを利用した写真撮影～

①新規プロジェクトの作成

まず開発画面のメニューバーより、「ファイル|新規作成|マルチデバイスアプリケーション」を選択し、「空のアプリケーション」を選択する。新規プロジェクトが作成されるので、メニューバーより、「ファイル|すべて保存」にて任意の場所に保管する。

②コンポーネントの配置

TActionを利用するため、TActionListをフォームに配置する。TActionListは非ビジュアルコンポーネントのため、配置位置はどこでも問題ない。

次にカメラを実行するためのTButtonを配置する。ここでは画面レイアウト上、下部に配置するため、Alignプロパティを Bottomに設定し、StyleLookupプロパティを cameratoolbuttonに指定する。

このFireMonkeyのTButtonが持つStyleLookupプロパティは、リスト形式で設定でき、アイコンや見た目、カーソルなど、いろいろなスタイルを自動で適用できる便利なプロパティである。

最後に撮影した画像を表示するため

図1　コンポーネントの配置

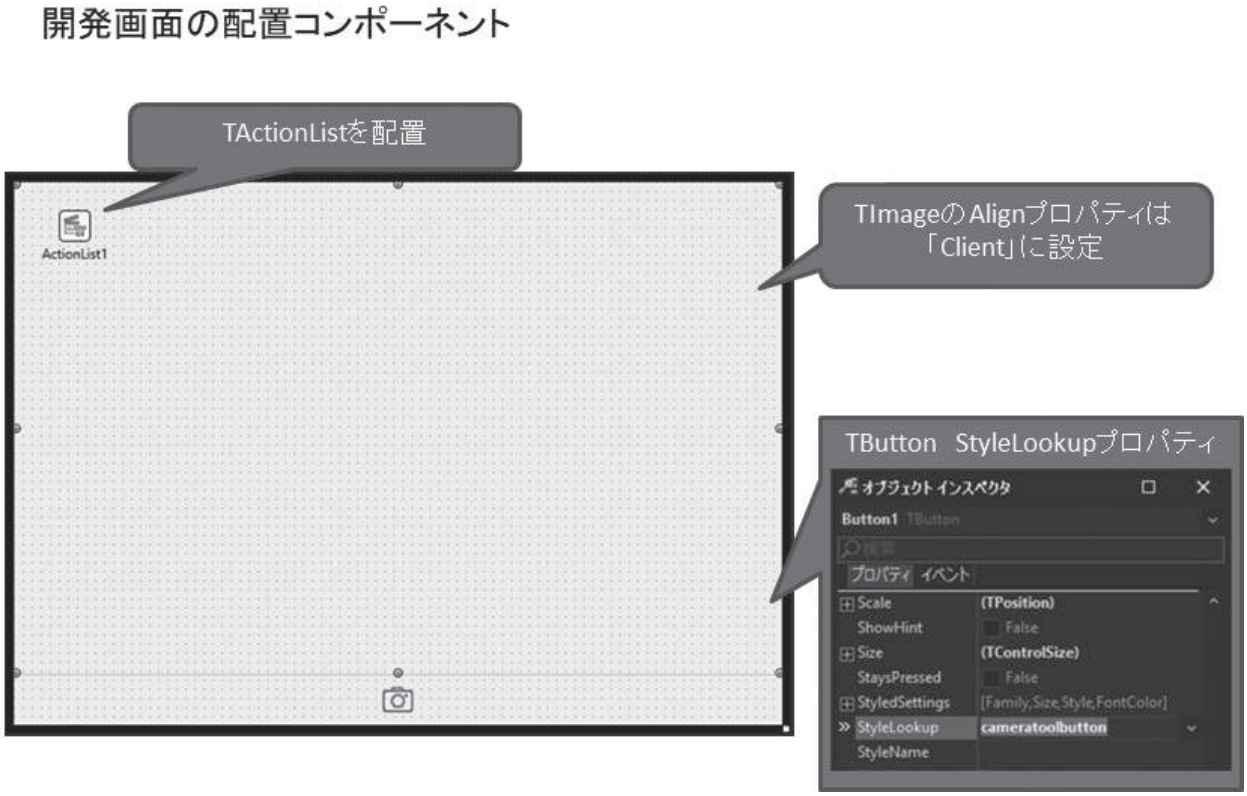


図2　Actionプロパティの設定

ソース1　作成手順～TActionを利用した写真撮影～① OnDidFinishTakingイベントの実装例

```
procedure TForm1.TakePhotoFromCameraAction1DidFinishTaking(Image: TBitmap);
begin
  // 撮影した写真を画面へ表示
  Image1.Bitmap.Assign(Image);
end;

end.
```

の TImage を配置する。画面レイアウトでは画面一面に表示するので、Align プロパティを Client に設定する。【図 1】

③ Action プロパティの設定

フォームに配置した TButton の Action プロパティのリストより「標準アクションの新規作成 | メディアライブラリ | TTakePhotoFromCameraAction」を選択する。これで画面のボタンをタッチすることでカメラが起動する。【図 2】

④イベントの設定

フォームに配置した TActionList をダブルクリック（もしくは「右クリック | アクションリストの設定」）し、カテゴリ＝メディアライブラリから TakePhotoFromCameraAction1 を選択する。オブジェクトインスペクタのイベントタブより、OnDidFinishTaking をダブルクリックし、実装部を作成してコーディングする。

OnDidFinishTaking イベントには、カメラで撮影された画像がその引数 Image（TBitmap 型）に格納されているため、配置した TImage コンポーネントにアサインするだけで実装完了である。【ソース 1】

⑤コンパイル

最後にコンパイルし、エラーがないことを確認し、「ファイル | すべて保存」よりプロジェクトを保管する。

以上の開発手順で、TAction を用いた写真撮影アプリが完成である。iOS や Android OS の端末に配布し実行すれば、開発したアプリで写真撮影ができることを確認できる。

ただし Windows OS 上で実行すると、カメラが起動されない。これについては残念ながら FireMonkey の Action がサポートされていないためである。しかし、Windows OS であっても ShellExecute メソッドを用いることでカメラは起動できる。先ほど作成したプロジェクトをもとに、以下に実装例を挙げる。

Windows OS でカメラを起動する実装例

条件付きコンパイル「{$IFDEF MSWINDOWS}」を用いて、Windows OS の場合は Windows API を uses 節に加えている。また、TButton の Action プロパティには TakePhotoFromCameraAction を紐づけず、On Click イベントにて Windows OS の場合は ShellExecute メソッドでカメラを起動し、それ以外の場合は TakePhotoFromCameraAction を実行するようにする。【ソース 2、ソース 3】

上記の実装例で、Windows OS 用の ShellExecute メソッドによりカメラを起動した場合は注意が必要である。これはあくまでカメラの起動であり、撮影したファイル名等の情報は取得できない。

TakePhotoFromCameraAction を利用した先の例では、OnDidFinishTaking イベントによって、撮影した画像が取得でき、それを利用して TImage に表示していた。しかし Windows OS 用に ShellExecute メソッドでカメラを起動した場合は、カメラロールを監視して、作成されるファイルのタイムスタンプで撮影された画像を判断するなどして画面に表示する必要があり、処理が煩雑になる。そもそも、実行する OS に合わせてコーディングすると手間も増えることになる。

その煩雑さや手間を解決するため、以下に FireMonkey 固有のコンポーネント「TCameraComponent」を紹介する。

3. TCamera Componentを利用した写真撮影

TCameraComponent は、FireMonkey 特有のカメラデバイスに対応するコンポーネントである。保持しているプロパティやイベントも少なく、わずかなコーディングでカメラデバイスを操作できる。

TCameraComponent を使う上でのポイントは、イベント OnSampleBufferReady である。まず TCameraComponent が保持する Active プロパティを True にすることで、カメラが起動する。このアクティブ状態（Active プロパティ＝True）の際に断続的に実行されるイベントが、OnSampleBufferReady である。

このイベントで、ビットマップに出力するために用意された SampleBufferToBitmap メソッドを利用して、アクティブ中のカメラの映像をイメージとして描画し、シャッターのタイミングで画像ファイルとして保存すれば、写真撮影アプリが作成できる。

それでは、実際に TCameraComponent を利用して写真撮影アプリを作成する。写真撮影アプリを作成するために用いる TCameraComponent 以外のコンポーネントは、カメラのシャッター用に TButton、カメラ画像を表示する TImage の 2 つである。

作成方法を以下に紹介する。なおこの作成手順は、先ほど紹介した「TAction を利用した写真撮影」の手順①のとおり、新規プロジェクトを作成した後のものとする。

作成手順

～ TCameraComponent を利用した写真撮影～

①コンポーネントの配置

TCameraComponent をフォームに配置する。TAction コンポーネントと同様、TCameraComponent は非ビジュアルコンポーネントなので配置位置はどこでも構わない。

次にカメラのシャッターとして、TButton を配置する。Align プロパティを Bottom に設定し、StyleLookup プロパティを cameratoolbutton に指定する。さらにカメラの映像を表示するための TImage を配置し、Align プロパティを Client に設定する。【図 3】

②イベントの設定

●フォーム生成時と破棄時

アプリの起動時にカメラ撮影を開始するため、フォームの OnCreate イベントにて TCameraComponent の Active プロパティを True にする。また、フォームの破棄時に生成時に起動したカメラ撮影を終了するため、OnCloseQuery イベントで Active プロパティを False にする。【ソース 4】

●ボタン押下時（シャッター）

カメラの映像からイメージへの描画をいったん停止し、映像を画像ファイルとして保存する。try-finally 構文で TCameraComponent の OnSampleBufferReady イベントをクリアし、再設定する間に、TImage の SaveToFile

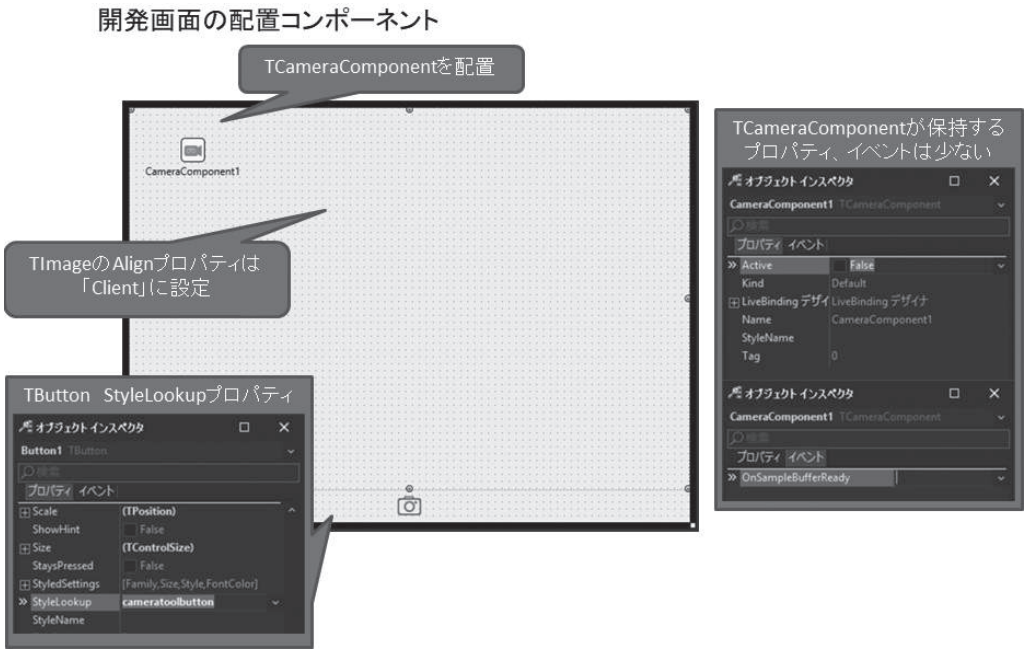
ソース2　作成手順～TActionを利用した写真撮影～② Windows OSでカメラを起動する実装例

```
unit Unit1;

interface

uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.Objects,
  FMX.Controls.Presentation, FMX.StdCtrls, System.Actions, FMX.ActnList,
  FMX.StdActns, FMX.MediaLibrary.Actions
{$IFDEF MSWINDOWS}
  , Winapi.Windows, Winapi.ShellAPI
{$ENDIF}
  ;

type
  TForm1 = class(TForm)
    Image1: TImage;
    ActionList1: TActionList;
    TakePhotoFromCameraAction1: TTakePhotoFromCameraAction;
    Button1: TButton;
    procedure TakePhotoFromCameraAction1DidFinishTaking(Image: TBitmap);
    procedure Button1Click(Sender: TObject);
  private
    { private 宣言 }
  public
    { public 宣言 }
  end;

var
  Form1: TForm1;

implementation        ソース３へ続く
```

ソース3　作成手順～TActionを利用した写真撮影～③ Windows OSでカメラを起動する実装例

```
{$R *.fmx}            ソース２の続き

procedure TForm1.Button1Click(Sender: TObject);
begin

{$IFDEF MSWINDOWS}
  // Windowsの場合、アクションがサポートされていない為、カメラアプリをShellExecute関数で起動
  // この場合、撮影したファイル名等が取得できないため、後続処理で使用したい場合に
  // カメラロールを参照してタイムスタンプ等で判断する必要がある
  ShellExecute(0, 'OPEN', PChar('microsoft.windows.camera:'), nil, nil, SW_SHOWMAXIMIZED);
{$ELSE}
  TakePhotoFromCameraAction1.Execute;
{$ENDIF}
end;

procedure TForm1.TakePhotoFromCameraAction1DidFinishTaking(Image: TBitmap);
begin
{$IF not Defined(MSWINDOWS)}
  // 撮影した写真を画面へ表示
  Image1.Bitmap.Assign(Image);
{$IFEND}
end;

end.
```

図3　コンポーネントの配置



メソッドを利用してファイルとして保存する。【ソース 5】

●カメラがアクティブ状態の時

カメラ映像の撮影はメインスレッドとは別のスレッドで行われるため、画面に配置した TImage へ画像ファイルを描画する際には、Synchronize メソッドを用いる。Synchronize は引数にメソッドが必要なので、「GetImage」というメソッドを別に作成した。

procedure として実装した GetImage メソッドでは、TCameraComponent の SampleBufferToBitmap メソッドを用いて、撮影中の映像を TImage へビットマップファイルとして読み込ませる。【ソース 6】

③コンパイル

最後にコンパイルし、エラーがないことを確認し、「ファイル | すべて保存」よりプロジェクトを保管する。

以上の開発手順で、TCameraComponent を用いた写真撮影アプリが完成である。iOS や Android OS の端末、カメラデバイスがある Windows OS でも、開発したアプリで写真を撮影できる。

4. デバイス搭載カメラの切り替え

前述の開発手順では、基本的な TCameraComponent の利用方法を紹介した。ここではさらに、プロパティについて補足する。

TCameraComponent 自体が持つプロパティは少ないが、開発画面のオブジェクトインスペクタでは「Kind」というプロパティを保持していることがわかる。これはデバイスが搭載しているどのカメラを使うかを指定するプロパティである。選択肢とカメラは、下記のようになる。

Default	：標準
FrontCamera	：前面カメラ
BackCamera	：背面カメラ

仕組みとして、TCameraComponent では使用するカメラを private 宣言の Device 変数で保持しており、Kind プロパティを FrontCamera や BackCamera に変更することで、そのデバイスの適切なカメラが自動的に選択され、Device 変数に格納される。

スマートフォンなどのモバイル端末ではこの選択肢で事足りるが、Windows 端末では USB 外付けカメラが利用できるため、その選択肢を持たない Kind プロパティでは対応できない。Device 変数自体に指定する必要がある。

前述のとおり、Device 変数は private 宣言のため、直接アクセスできないため、ヘルパークラスを作成し、搭載カメラの切り替えを実現する方法を以下に紹介する。これを CameraComponentHelper ユニットとする。

なお、そのデバイスが利用できる対象のカメラは、TCaptureDeviceManager.Current.GetDevicesByMediaType を参照することで取得できるので、合わせてカメラリストのクラスも管理することとする。【ソース 7 ～ 9】

CameraComponentHelper ユニットの概要

● TCameraComponentHelper クラス

TCameraComponent の helper として定義する。UseDevice プロパティを持ち、TCaptureDevice 型の Device プロパティに対して、参照ならびに設定を可能にする。

● TCameraList クラス

TCaptureDeviceList として、GetDevicesByMediaType より取得したカメラを TCaptureDevice 型の Item として、リスト形式で保持する。

CameraComponentHelper ユニットの使い方

具体的な CameraComponentHelper ユニットの使い方を説明する。先ほど紹介した「TCameraComponent を利用した写真撮影」のアプリへの追加実装を前提とし、カメラ切り替えボタンを押すことで、順次、利用可能なカメラに切り替わるものとする。

① uses 節への追加

CameraComponentHelper を uses 節に追加する。これで TCameraList、UseDevice が利用できるようになる。【ソース 10】

②コンポーネントの配置

カメラ切り替え用にボタンを配置する。画面の右下に配置することにする。また Text プロパティを「カメラ切り替え」とする。

③イベントの設定

●フォーム生成時と破棄時

フォームの生成時には、デバイスが利用可能なカメラを TCameraList を用いてリスト化する。コーディングは TCameraList 型のユニット内変数を private 宣言部で定義し、フォーム生成時に TCameraList を Create し、キャストする。

また、取得した利用可能なカメラの数を保持するためインデックス用の Integer 型を private 宣言部で合わせて定義する。フォームの破棄時には、生成した TCameraList 型のユニット内変数を破棄する。【ソース 11】

●ボタン押下時（カメラ切り替え）

カメラの切り替えは、CameraComponentHelper を uses 節に追加したことで利用できるようになった TCameraComponent の UseDevice に、TCameraList 型の Item を指定することで可能となる。またカメラを切り替える際は、TCameraComponent のアクティブ状態を停止してから行う。【ソース 12】

④コンパイル

最後にコンパイルし、エラーがないことを確認し、「ファイル | すべて保存」よりプロジェクトを保管する。

以上の実装で、デバイスが利用可能なカメラを切り替える機能が追加できた。実際に複数のカメラデバイスがあるモバイルや、USB 外付けカメラを接続した Windows 端末で動作を確認すると、カメラ切り替えボタンを押すたびに、撮影しているカメラが切り替わることが確認できる。

5. 画質の指定

TCameraComponent には、オブジェ

ソース4　作成手順～TCameraComponentを利用した写真撮影～①

フォーム生成時と破棄時

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  // カメラの起動
  CameraComponent1.Active := True;
end;

procedure TForm2.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
  // カメラ終了
  if CameraComponent1.Active then
  begin
    CameraComponent1.Active := False;
  end;
end;
```

ソース5　作成手順～TCameraComponentを利用した写真撮影～②

ボタン押下時（シャッター）

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  // カメラの起動状態を確認
  if not CameraComponent1.Active then Exit;

  // カメラからイメージへの描画をストップ
  CameraComponent1.OnSampleBufferReady := nil;
  try
    // 写真をJPG形式で保存
    Image1.Bitmap.SaveToFile('Picture.jpg');
  finally
    // カメラからイメージへの描画を再開
    CameraComponent1.OnSampleBufferReady := CameraComponent1SampleBufferReady;
  end;
end;
```

ソース6　作成手順～TCameraComponentを利用した写真撮影～③

カメラがアクティブ状態の時

```
procedure TForm2.CameraComponent1SampleBufferReady(Sender: TObject;
  const ATime: TMediaTime);
begin
  // メインスレッドでカメラの内容をイメージに描画
  TThread.Synchronize(TThread.CurrentThread, GetImage);
end;

procedure TForm2.GetImage;
begin
  if CameraComponent1.Active then
  begin
    // カメラの内容をイメージに描画
    CameraComponent1.SampleBufferToBitmap(Image1.Bitmap, True);
  end;
end;
```

クトインスペクタで指定できないPublicプロパティも保持している。その中でも写真を撮影したファイルの容量に関する画質の指定について、以下に紹介する。

TCameraComponentのQualityプロパティを用いると、デバイスに設定されている解像度を指定できる。その選択肢は解像度の高いものから順に、「PhotoQuality ＞ HighQuality ＞ MediumQuality ＞ LowQuality」となっている。

指定方法は至ってシンプルである。先ほどのカメラ切り替え用のソースを例として、フォームの生成時に中解像度に指定する方法を紹介する。

Qualityプロパティの指定方法

Qualityプロパティを指定する場合は、FMX.Media.クラスで定義されているTVideoCaptureQuality型からアクセスした各定数を利用する。中解像度は「MediumQuality」である。【ソース13】

これで保存した画像ファイルの用途に応じて、画質の指定が可能となる。

6. おわりに

本稿では、マルチデバイス用のFireMonkeyフレームワークの中からTCameraComponentの開発方法を説明し、少ないコーディングでデバイスのカメラ機能が利用できることを紹介した。

FireMonkeyにはその他にも、VCLフレームワークにない、特有の魅力的な機能やコンポーネントが多数存在する。本稿がきっかけとなり、FireMonkeyフレームワークを活用していただければ幸いである。

M

ソース7　CameraComponentHelperユニット①

TCameraComponentHelperユニット宣言部

```
unit CameraComponentHelper;

interface

uses
  System.Classes, FMX.Media;

type
  { TCameraComponentHelper }
  TCameraComponentHelper = class helper for TCameraComponent
  private
    procedure SetUseDevice(const Value: TCaptureDevice);
    function GetUseDevice: TCaptureDevice;
  public
    property UseDevice: TCaptureDevice read GetUseDevice write SetUseDevice;
  end;

  { TCameraList }
  TCameraList = class
  private
    FCameras: TStringList;
    function GetItems(Index: Integer): TCaptureDevice;
    function GetCount: Integer;
  public
    constructor Create;
    destructor Destroy; override;
    property Count: Integer read GetCount;
    property Items[Index: Integer]: TCaptureDevice read GetItems;
  end;

implementation
```

usesの節に「System.Classes」「FMX.Media」を追加する

TCameraComponentHelperクラス

TCameraListクラス

ソース8　CameraComponentHelperユニット②

TCameraComponentHelper実装部

```
{ TCameraComponentHelper }

function TCameraComponentHelper.GetUseDevice: TCaptureDevice;
begin
  // CameraComponentのデバイスを取得
  with Self do
  begin
    Result := FDevice;
  end;
end;

procedure TCameraComponentHelper.SetUseDevice(const Value: TCaptureDevice);
begin
  // CameraComponentへデバイスを設定
  with Self do
  begin
    FDevice := TVideoCaptureDevice(Value);
  end;
end;
```

TCameraList実装部

```
{ TCameraList }

destructor TCameraList.Destroy;
begin
  FCameras.Free;
  inherited;
end;
```

↓[ソース9]へ続く

ソース9　CameraComponentHelperユニット③

TCameraList実装部

↓[ソース8]の続き

```
{ TCameraList }

constructor TCameraList.Create;
var
  i: Integer;
  CaptureDeviceList: TCaptureDeviceList;
begin
  // カメラデバイスの一覧を保持
  FCameras := TStringList.Create;
  CaptureDeviceList := TCaptureDeviceManager.Current.GetDevicesByMediaType(TMediaType.Video);
  for i:=0 to CaptureDeviceList.Count-1 do
  begin
    FCameras.Add(CaptureDeviceList[i].Name);
  end;
end;

function TCameraList.GetCount: Integer;
begin
  Result := FCameras.Count;
end;

function TCameraList.GetItems(Index: Integer): TCaptureDevice;
begin
  // 指定したカメラデバイスを取得
  if (Index < 0) or (Index >= FCameras.Count) then
  begin
    Result := nil;
    Exit;
  end;
  Result := TCaptureDeviceManager.Current.GetDevicesByName(FCameras[Index]);
end;
```

ソース10　CameraComponentHelperユニットの使い方①

uses節への追加と変数定義

```
uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.Media,
  FMX.Objects, FMX.Controls.Presentation, FMX.StdCtrls, Winapi.MMSystem,
  CameraComponentHelper;                                   uses節に「CameraComponentHelper」を追加する

type
  TForm3 = class(TForm)
    CameraComponent1: TCameraComponent;
    Image1: TImage;
    Rectangle1: TRectangle;
    Button1: TButton;
    Button2: TButton;
    procedure FormCreate(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
    procedure CameraComponent1SampleBufferReady(Sender: TObject;
      const ATime: TMediaTime);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { private 宣言 }
    CameraList: TCameraList;
    FCurrentIndex: Integer;                    TCameraList型、Integer型の変数を定義する

    procedure GetImage;
  public
    { public 宣言 }
  end;
```

ソース11　CameraComponentHelperユニットの使い方②

フォーム生成時と破棄時

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  // デバイス搭載カメラのリスト生成
  CameraList := TCameraList.Create;
  FCurrentIndex := CameraList.Count - 1;

  // カメラコンポーネントの設定
  CameraComponent1.Kind := FMX.Media.TCameraKind.Default;                  // 使用カメラ
  CameraComponent1.UseDevice := CameraList.Items[CameraList.Count - 1];
  CameraComponent1.Active := True;                                         // カメラ起動
end;

procedure TForm3.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
  // カメラのリスト解放
  FreeAndNil(CameraList);

  // カメラ終了
  if CameraComponent1.Active then
  begin
    CameraComponent1.Active := False;
  end;
end;
```

変数CameraListを生成し
インデックスを初期設定する

UseDeviceの指定には
変数CameraListのItemを利用する

ソース12　CameraComponentHelperユニットの使い方③

ボタン押下時（カメラ切り替え）

```
procedure TForm3.Button2Click(Sender: TObject);
begin
  // カメラの切り替え
  CameraComponent1.Active := False; // 一旦カメラを停止

  Inc(FCurrentIndex);
  FCurrentIndex := FCurrentIndex mod CameraList.Count;
  CameraComponent1.UseDevice := CameraList.Items[FCurrentIndex]; // カメラ設定

  CameraComponent1.Active := True; // カメラ再開
end;
```

ソース13　Qualityプロパティの指定方法

フォーム生成時

```
procedure TForm3.FormCreate(Sender: TObject);
begin
  // デバイス搭載カメラのリスト生成
  CameraList := TCameraList.Create;
  FCurrentIndex := CameraList.Count - 1;

  // カメラコンポーネントの設定
  CameraComponent1.Kind := FMX.Media.TCameraKind.Default;                  // 使用カメラ
  CameraComponent1.UseDevice := CameraList.Items[CameraList.Count - 1];
  CameraComponent1.Quality := FMX.Media.TVideoCaptureQuality.MediumQuality; // 画質
  CameraComponent1.Active := True;                                         // カメラ起動
end;
```

福井 和彦

株式会社ミガロ.

システム事業部 プロジェクト推進室

[Delphi/400]

Subversionを使用したDelphiソース管理

略歴
1972 年 3 月 20 日生まれ
1994 年 3 月 大阪電気通信大学 工学部卒業
2001 年 4 月 株式会社ミガロ. 入社
2001 年 4 月 システム事業部配属

現在の仕事内容
主に Delphi/400 を使用したシステムの受託開発を担当しており、要件確認から納品・フォローに至るまで、システム開発全般に携わっている。

1. はじめに

システム開発では、新規機能の追加や既存機能の変更等が発生し、プロジェクトへのファイルの追加やファイル内のコードの変更が発生する。そのためシステムを管理していく上で、「いつ」「誰が」ファイルを追加したのか、「いつ」「誰が」ファイル内のコードの「どこを」変更したのか、その履歴を管理するのは重要なことである。

また複数名でシステムを開発する場合、同じファイルを同じタイミングで、複数名が変更しないように気を付ける必要がある。

そこで本稿では、バージョン管理システムと呼ばれるソフトウェアを使用して、上記の課題を解決する方法を説明していく。

2. バージョン管理システムについて

バージョン管理システム（Version Control System）は VCS とも呼ばれ、「いつ」「誰が」「どこを」ファイルに変更を加えたのかを履歴で管理するソフトウェアである。バージョン管理システムは、大きく以下の 3 つの管理方法に分けられる。

1 ローカル管理方式
2 集中管理方式
3 分散管理方式

本稿では、このうち「2 集中管理方式」でのソース管理方法について説明する。

集中管理方式とは、サーバーにリポジトリ（バージョン管理専用フォルダ）を作成し、各クライアントからリポジトリにアクセスして、ローカル環境とサーバー上のリポジトリとで同期をとる方式である。【図 1】

集中管理方式は仕組みがシンプルでわかりやすいので、初めてバージョン管理システムを使用する場合でも扱いやすい。本稿では、集中管理方式のバージョン管理システムの中で代表的な Subversion を使用する。

以下に、サーバー側とクライアント側に必要なソフトウェアのインストール手順について説明する。

3. Subversionの設定（サーバー編）

3-1. VisualSVN Serverのインストール

サーバーには、VisualSVN Server をインストールする。VisualSVN Server を使用することで、Windows 環境に GUI ベースで手軽に Subversion 環境が構築できる。

インストーラーは、以下の URL から無料でダウンロード可能である。環境に合わせて 32-bit、64-bit のどちらかをダウンロードする。【図 2】

VisualSVN Server ダウンロードサイト
https://www.visualsvn.com/server/download/

図1　バージョン管理システム集中管理方式

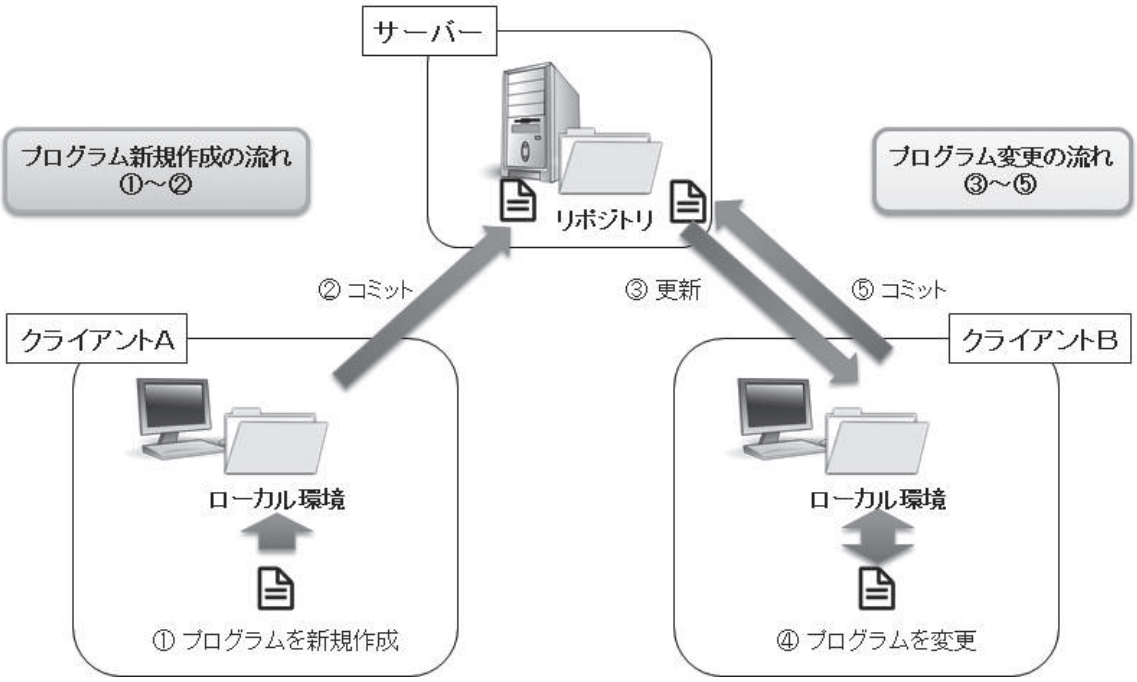


図2　VisualSVN Server ダウンロードサイト

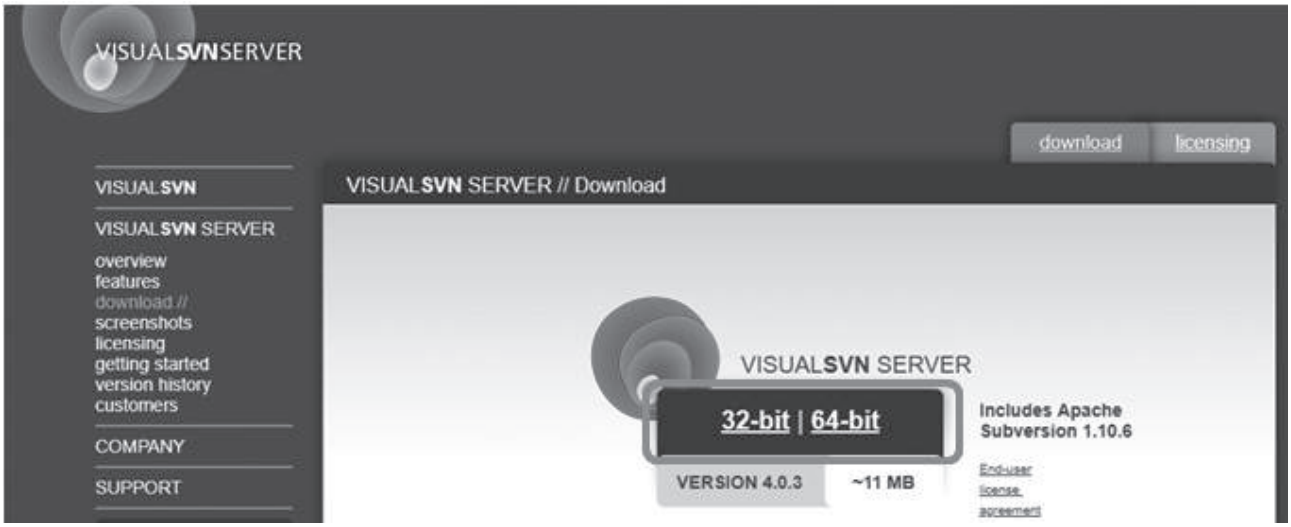
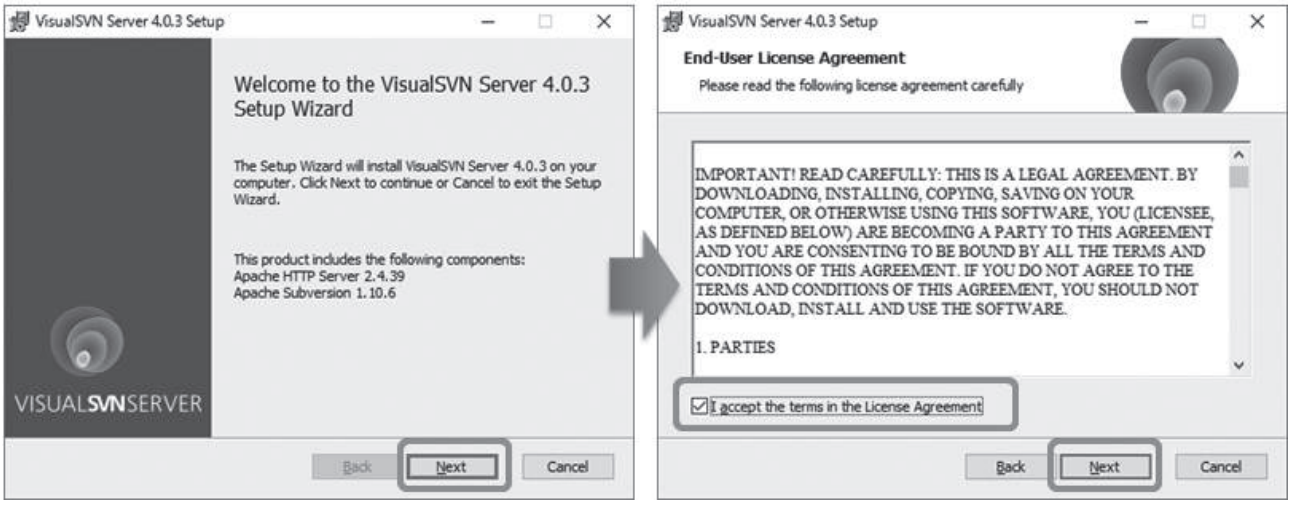


図3　VisualSVN Server インストール①



ダウンロードしたインストーラーを実行して、インストールを開始するとWelcome画面が開く。「Next」ボタンを押下し、ライセンス確認画面へ遷移したら、ライセンス内容を確認し、「ライセンス契約に同意する」チェックをオンにして、「Next」ボタンを押下する。【図3】

コンポーネントの選択画面では、初期値のまま「Next」ボタンを押下する。次画面でインストールするフォルダ、リポジトリフォルダ、サーバーポートを指定して、「Next」ボタンを押下する。【図4】

ユーザー認証を選択する画面では、Subversionのユーザー認証を選択（初期値）して、「Next」ボタンを押下し、次画面で「Install」ボタンを押下して、インストールを開始する。【図5】

インストールが進み、完了画面で「Finish」ボタンを押下して、インストールは完了となる。【図6】

3-2. ユーザーの作成

インストールが完了したら、VisualSVN Server Managerを起動してユーザーを作成する。【図7】

ユーザーは、「誰が」追加や変更を行ったのかを管理するためにも、開発者ごとに作成するのが望ましい。作成方法は、ツリービューより「Users」を右クリックし、「Create User...」を選択する。入力ダイアログが表示されるので、ユーザー名とパスワードを入力する。ここではユーザー名とパスワードを“testuser”と設定する。【図8】

これでユーザーの作成は完了である。

3-3. リポジトリの作成

続いて、リポジトリを作成する。リポジトリは、Delphiの1プロジェクトで1リポジトリ、あるいは複数プロジェクトで1リポジトリというように、ソース管理を行う単位で作成する。

作成方法は、VisualSVN Server Managerのツリービューより、「Repositories」を右クリックして、「Create New Repository...」を選択する。Repositoryタイプは初期値のまま「次へ」ボタンを押下する。【図9】

リポジトリ名を入力する画面でリポジトリ名を入力する。ここでは“TestSample”と設定して、「次へ」ボタンを押下する。初期のリポジトリ構成は、初期値のまま「次へ」ボタンを押下する。【図10】

アクセス許可の設定も、初期値のまま「Create」ボタンを押下して、リポジトリの作成は完了である。【図11】

4. Subversionの設定（クライアント編）

4-1. TortoiseSVNのインストール

クライアントには、TortoiseSVNをインストールする。インストーラーは、以下のURLより無料でダウンロード可能である。環境に合わせて、32-bit、64-bitのどちらかをダウンロードする。【図12】

そして同じページにあるLanguage packsより、日本語化を行う日本語パックをダウンロードする。こちらも環境に合わせて、32bit、64bitのどちらかをダウンロードする。【図13】

TortoiseSVN ダウンロードサイト
https://tortoisesvn.net/downloads.html

ダウンロードしたインストーラーを実行して、インストールを開始すると、Welcome画面が開く。「Next」ボタンを押下して、ライセンス確認画面へ遷移したら、ライセンス内容を確認して、「Next」ボタンを押下する。【図14】

インストールする機能を選択する画面では、初期値のまま「Next」ボタンを押下する。次画面で「Install」ボタンを押下して、インストールを開始する。【図15】

インストールが進み、完了画面で「Finish」ボタンを押下して、インストールは完了となる。【図16】

次に、ダウンロードした日本語パックを実行して、インストールを開始する。ようこそ画面で「次へ」ボタンを押下したら、インストールは完了となる。【図17】

4-2. TortoiseSVNの設定

インストールが完了したら、任意のフォルダを開き、右クリックでショートカットメニューを表示する。「TortoiseSVN」メニューが追加されているので、「TortoiseSVN」→「設定」と開く。【図18】

設定画面のツリービューにある「全般」を選択し、言語の設定を「日本語」にする。【図19】

4-3. 作業フォルダとリポジトリの関連付け

ここでは、クライアントの作業フォルダとサーバーのリポジトリとを関連付ける設定を行う。最初に、クライアントに作業フォルダを作成する。作業フォルダには、後でDelphiのプログラムファイルを保管する。

ここでは、フォルダを以下のように作成する。【図20】

作業フォルダのパス
C:\Work\TestSample

続いて、VisualSVN Server Managerのツリービューより、「Repositories」内の「TestSample」を右クリックして、「Copy URL to Clipboard」を選択し、リポジトリのURLをコピーしておく。【図21】

作業フォルダ「TestSample」で、Shiftキーを押しながら右クリックして、ショートカットメニューを表示する。

メニュー内の「SVNチェックアウト」を選択すると、チェックアウト画面が開く。開いた画面の「リポジトリのURL」に、先ほどコピーしておいたURLを設定し、「チェックアウト先のディレクトリ」に先ほど作成した作業フォルダのパスを設定する。「OK」ボタンを押下して、設定を保存する。【図22】

証明書の確認エラーが出る場合は、「証明書を永久に受け付ける」を選択する。次に認証画面が表示されるので、ユーザー名とパスワードに“testuser”と入力して、「OK」ボタンを押下する。【図23】

チェックアウト終了画面が表示され、作業フォルダには緑色のチェックアイコンが付く。これで、関連付けは完了となる。【図24】

図4　VisualSVN Server インストール②

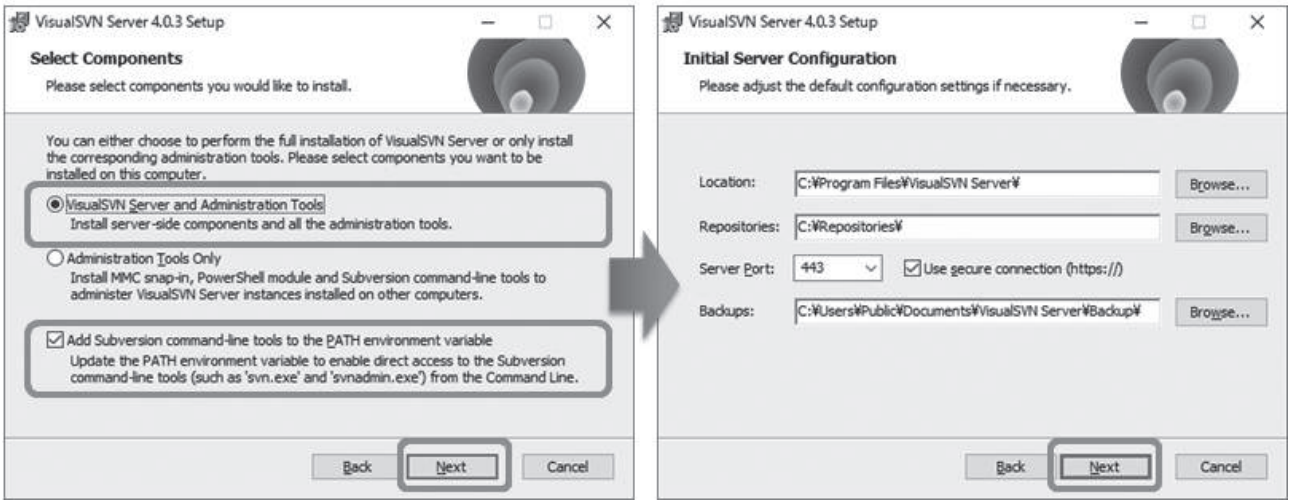


図5　VisualSVN Server インストール③

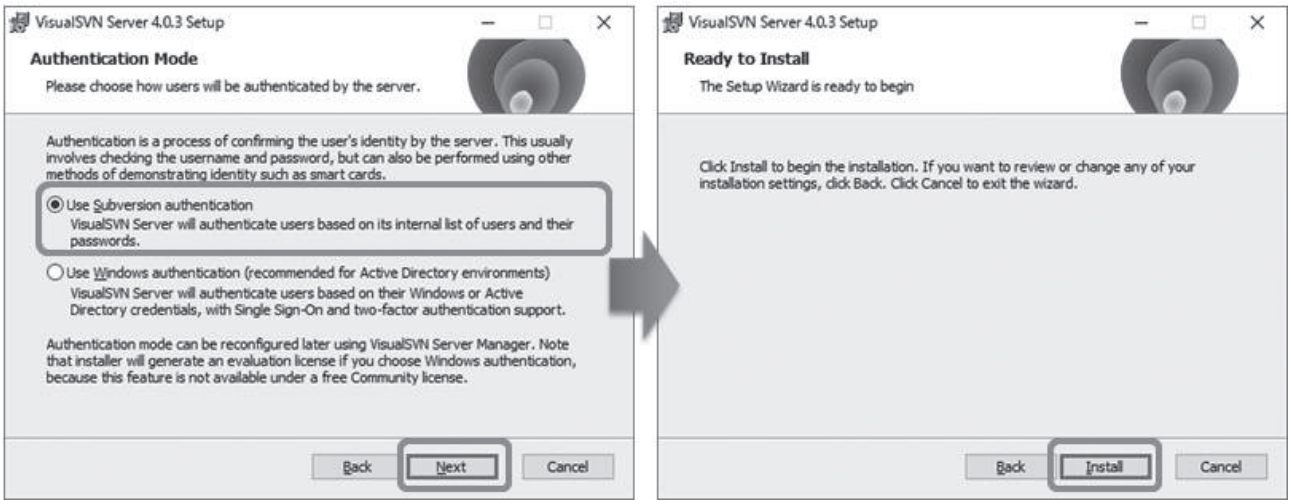
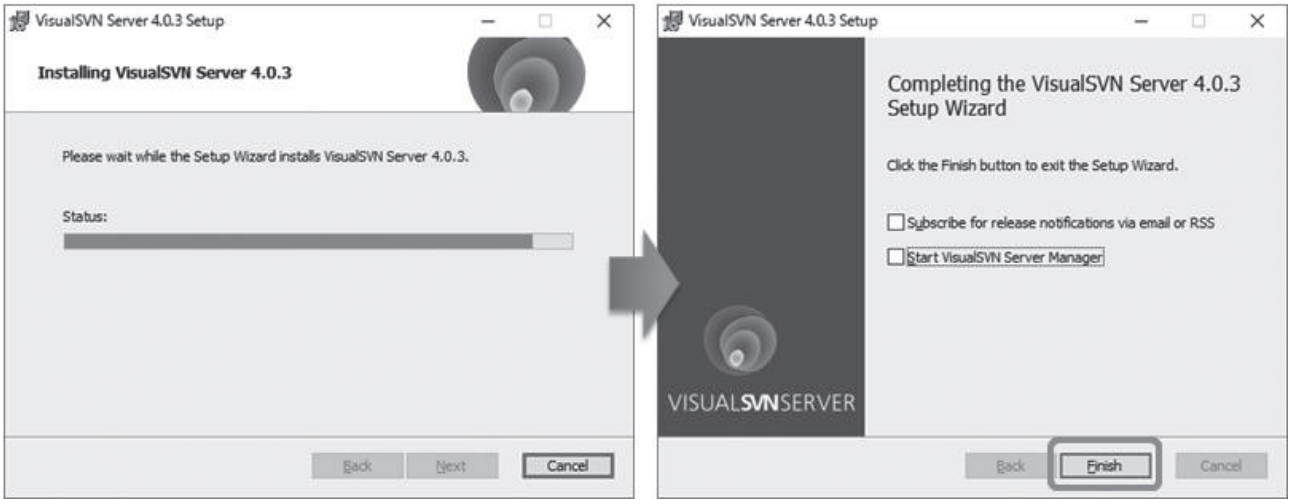


図6　VisualSVN Server インストール④



5. ソース管理について

5-1. 新規・変更ファイルをリポジトリへ反映

ここから、Subversion を使用してソース管理を行う方法について説明する。

まずはソース管理を行うプログラムファイルを Delphi で作成する。Delphi 10.2 Tokyo を起動して、メニューバーより「ファイル」→「新規作成」→「VCL フォームアプリケーション」を選択する。そして、フォームに TButton を配置する。【図 25】

ボタンの配置が完了したら、作成したプロジェクトを保存する。メニューバーより「ファイル」→「すべて保存」を選択し、保存先には先程作成した作業フォルダを指定して保存する。保存した直後の作業フォルダは、【図 26】のようになっている。

では作業フォルダに保管した、Delphi の各ファイルをリポジトリに反映する方法を説明する。【図 27】のように、ソース管理を行うファイルを選択する。そして Shift キーを押しながら右クリックして、ショートカットメニューを表示し、「TortoiseSVN」→「追加」を選択する。

すると【図 28】のように、選択していたファイルのアイコンに、緑色の“+”マークが付く。この時点では、まだリポジトリには反映されていない。

続いて、緑色の“+”マークが付いているファイルを選択する。そして Shift キーを押しながら右クリックをして、ショートカットメニューを表示し、「SVN コミット」を選択する。するとコミットの確認画面が開き、先ほど選択したファイルが一覧に表示されている。「OK」ボタンを押下して、コミットが完了となる。【図 30】

コミット完了後、選択していたファイルのアイコンには緑色の“✓”マークが付く。【図 31】

以上が、新規作成したファイルをリポジトリに反映する方法である。

続いて、変更したファイルをリポジトリに反映する方法について説明する。

Delphi 10.2 Tokyo を起動して、作業フォルダの Project1.dproj を開く。そして Form1 上に TLabel と TEdit を貼り付けて保存する。【図 32】

すると【図 33】のように、変更したファイルのアイコンに赤色の“！”マークが付く。変更したファイルをリポジトリに反映する方法は、【図 29】と同様に、変更したファイルを選択し、ショートカットメニューより「SVN コミット」を選択する。【図 30】と同様にコミットの確認画面が開き、「OK」ボタンを押下して、コミットが完了となる。

変更したファイルのアイコンは【図 31】と同様、緑色の“✓”マークに戻っている。これで、変更したファイルのリポジトリへの反映は完了である。

5-2. リポジトリと同期

ここでは、作業フォルダをリポジトリと同期する操作方法について説明する。最初に、リポジトリに他の開発者が追加・変更したファイルがないかを確認する。作業フォルダで、ファイルを選択せずに右クリックして、ショートカットメニューより「TortoiseSVN」→「変更をチェック」を選択する。【図 34】

画面が開くので、フッター部の「リポジトリをチェック」ボタンを押下する。リポジトリに追加・変更されたファイルが存在する場合、明細に対象ファイルが一覧で表示される。他の開発者により追加または変更されたファイルには、「リモートのテキストの状態」列に“変更”と表示される。「状態」列に表示のないものが、追加されたファイルであることを示す。【図 35】

作業フォルダとリポジトリとのファイルの差分を確認したい場合は、対象ファイルを右クリックして、「最新（HEAD）リビジョンと作業ベースを比較」を選択する。すると TortoiseMerge 画面が開き、差分を確認できる。画面右側がリポジトリのファイル、左側が作業フォルダのファイルを表示しており、差異のある個所には背景色が付いている。【図 36】

特定のファイルのみをリポジトリと同期したい場合は、変更チェック画面で同期したいファイルを選択して、右クリックでショートカットメニューを開き、「更新」を選択することで同期できる。【図 37】

また作業フォルダ全体をリポジトリと同期したい場合は、作業フォルダでファイルを選択せずに、右クリックでショートカットメニューを開き、「SVN 更新」を選択することで同期できる。【図 37】

5-3. ロックについて

複数名でシステムを開発している場合、これから変更するファイルを他の開発者に変更されたくない場合にロックという機能を使用する。ファイルをロックする場合は、作業フォルダで対象ファイルを選択し、右クリックでショートカットメニューを開き、「SVN ロックを取得」を選択する。これでファイルをロックできる。【図 38】

そしてロックしたファイルに対して変更を行い、コミットするとロックは解除される。

また、間違ってロックしてしまった時にロックを解除する場合は、ロックの時と同様に作業フォルダで対象ファイルを選択し、右クリックでショートカットメニューを開き、「SVN ロックを解除」を選択する。これでロックを解除できる。【図 39】

ロックの状況を確認する場合は、【図 34】【図 35】の変更チェック画面を開き、フッター部の「リポジトリをチェック」ボタンを押下することで、自身と他の開発者がロックしているファイルを確認できる。「ロック」列に、ロックしているユーザー名が表示される。【図 40】

他の開発者がロックしているファイルに対してロックを実行する場合、【図 41】の上段のようにロック失敗となる。また、他の開発者がロックしているファイルに対して変更してコミットしようとした場合、【図 41】の下段のようにコミット失敗となる。

【図 41】のコミット時のように、他の開発者がロックしているファイルに対して変更を行うことはできる。その場合、コミットする時に初めて他の開発者がロックしていることに気付くことになる。この際、リポジトリの状態に対して変更した内容をマージしなければならない。

複数名でシステムを開発する場合、「変更するファイルは必ずロックする」というルールにしておけば、ロックする時点で他の開発者がロックしているかど

図7　VisualSVN Server Manager

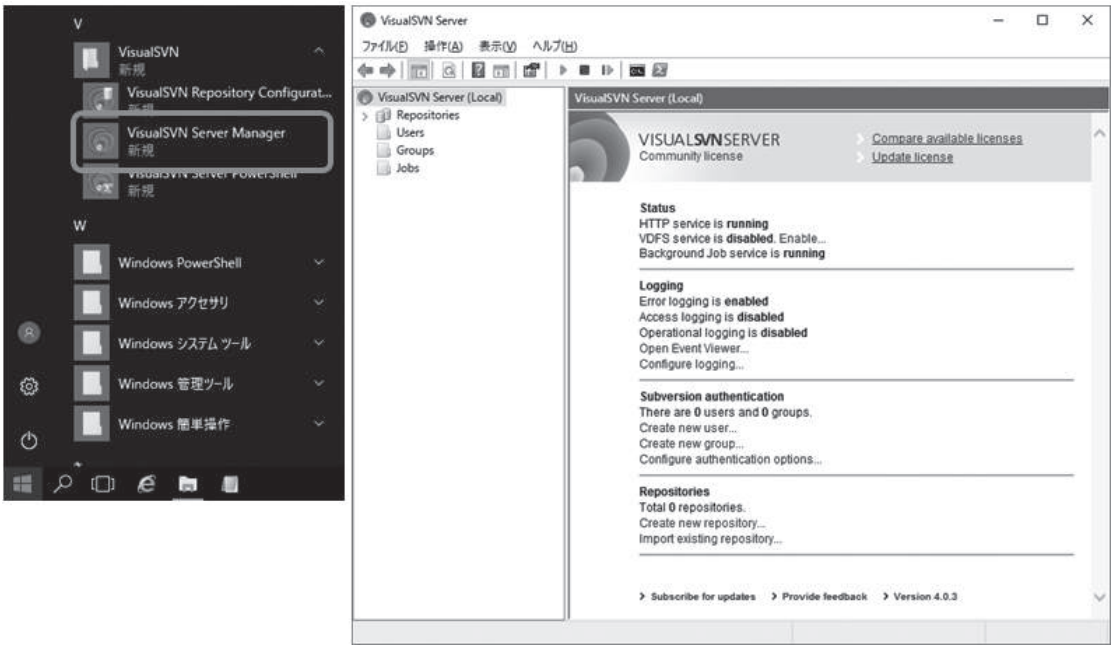


図8　ユーザーの作成

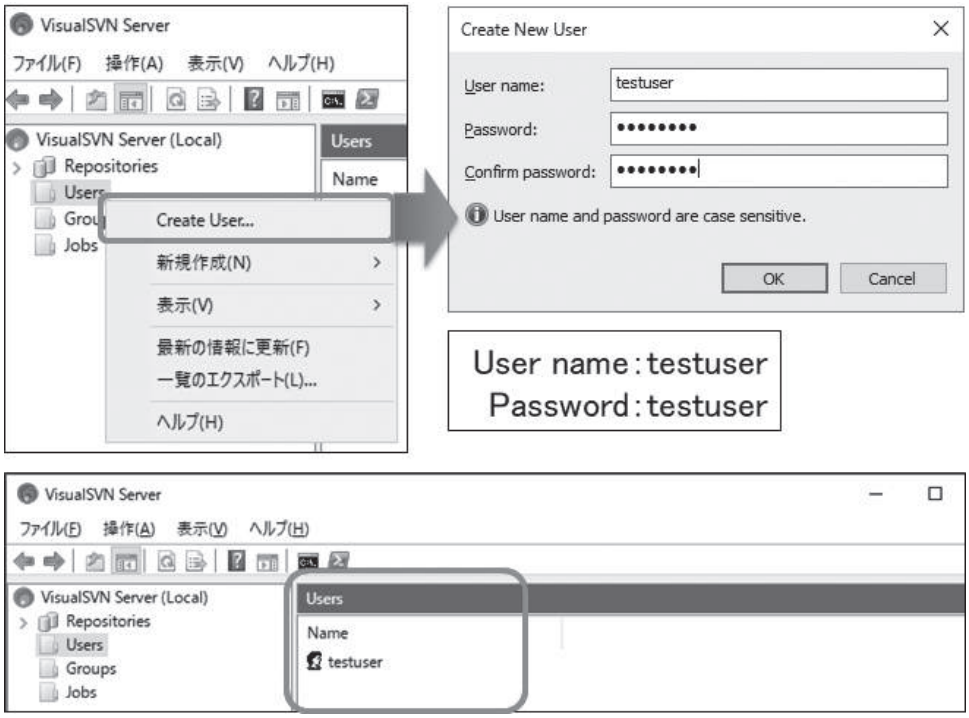
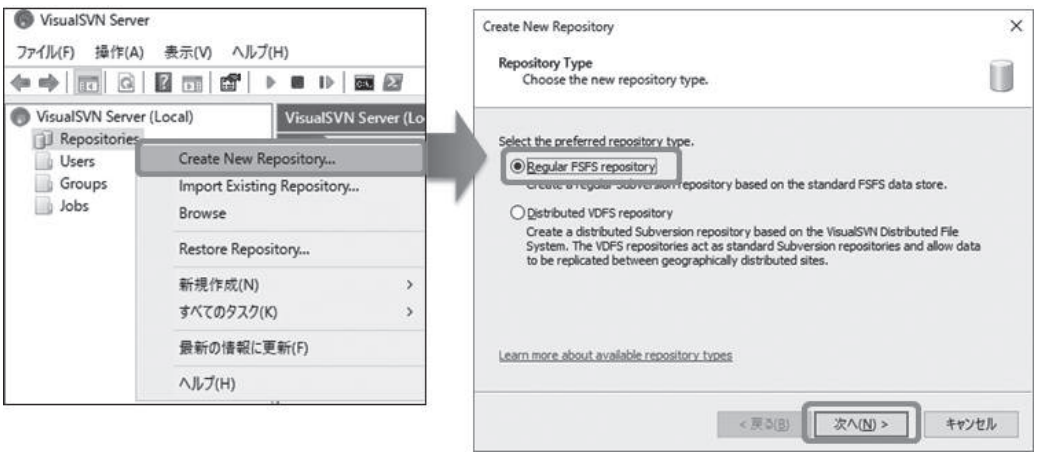


図9　リポジトリの作成①



うかがわかり、同じファイルを同じタイミングで複数名が変更することを防げる。

5-4. 変更履歴を確認

続いて、変更履歴の確認方法について説明する。作業フォルダで変更履歴を確認したいファイルを選択し、右クリックでショートカットメニューを開き、「TortoiseSVN」→「ログを表示」を選択する。【図 42】

するとログメッセージ画面が開き、選択したファイルの変更履歴をリビジョン一覧で確認できる。そしてリビジョン一覧から、ある時点のリビジョンを選択すると、そのタイミングで追加・変更された他のファイルも確認できる。【図 43】

また、コミットをする際に【図 44】のようにログメッセージを記載しておくと、ログメッセージ画面のメッセージ欄に表示される。

5-5. 変更履歴の差分を確認

前節で変更履歴は確認できたので、ここでは変更履歴の差分を確認する方法について説明する。ログメッセージ画面のリビジョン一覧から比較したいリビジョンを 2 つ選択し、右クリックでショートカットメニューを表示し、「リビジョンを比較」を選択する。

すると【図 36】のように TortoiseMerge 画面が開き、リビジョン間の差分を確認できる。

6. まとめ

本稿では、Subversion を使用したソース管理方法について説明した。今回の内容は基本的な使用方法ではあるが、環境構築も簡単で、ソース管理の方法もシンプルでわかりやすいイメージを持っていただけたと思う。

また本稿では取り上げていないが、Subversion は Delphi の統合開発環境からも使用できる。しかし統合開発環境から使用する場合はロック機能が使用できないので、同じファイルを同じタイミングで、複数名が変更しないように気を付ける必要がある。

本稿が、ソース管理について悩んでいる方の参考になれば幸いである。

M

図10　リポジトリの作成②

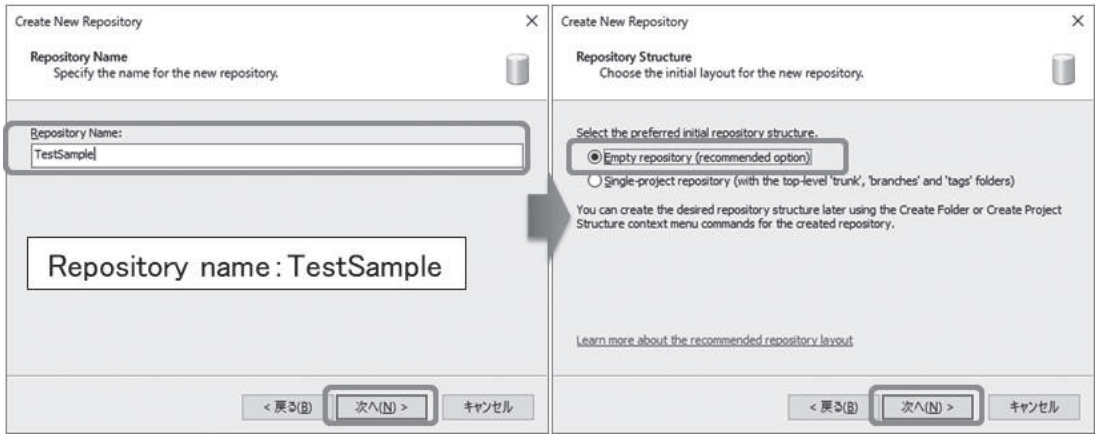


図11　リポジトリの作成③

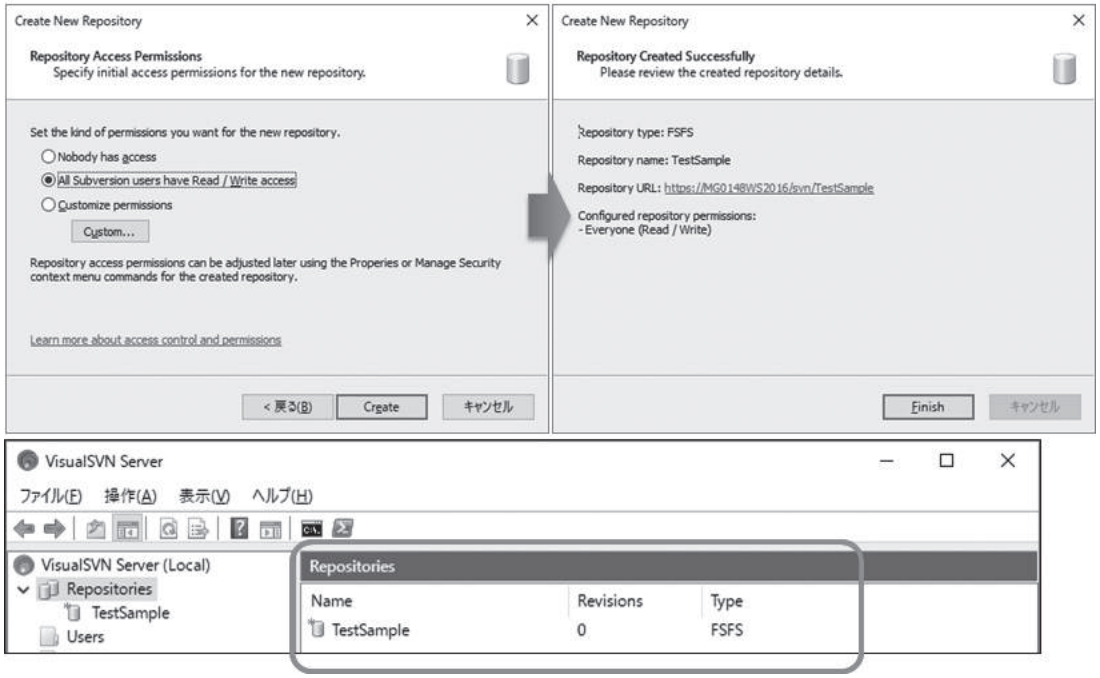


図12　TortoiseSVN ダウンロードサイト①

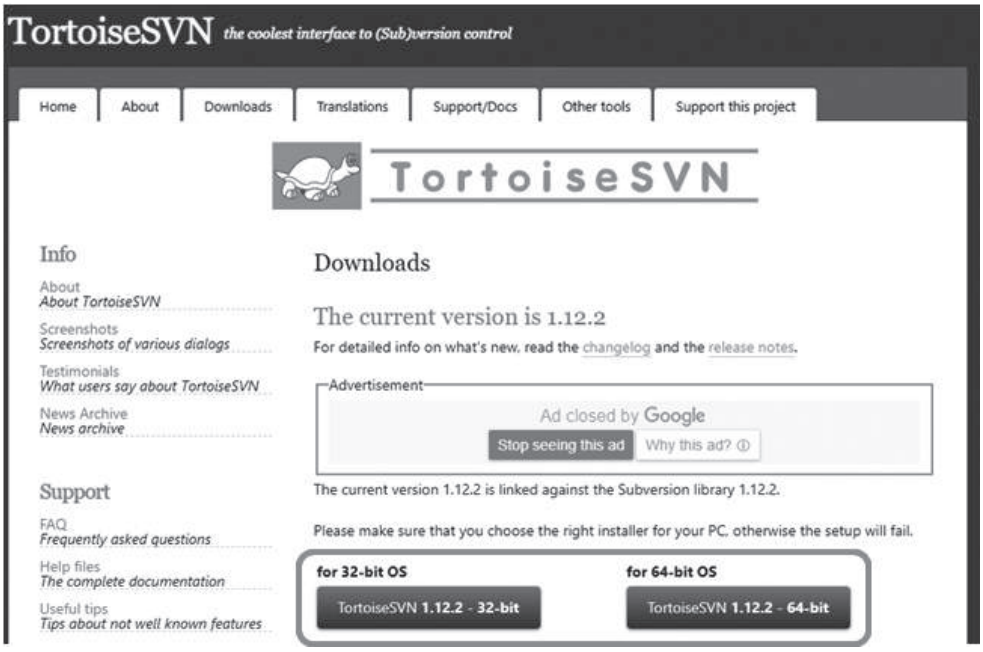
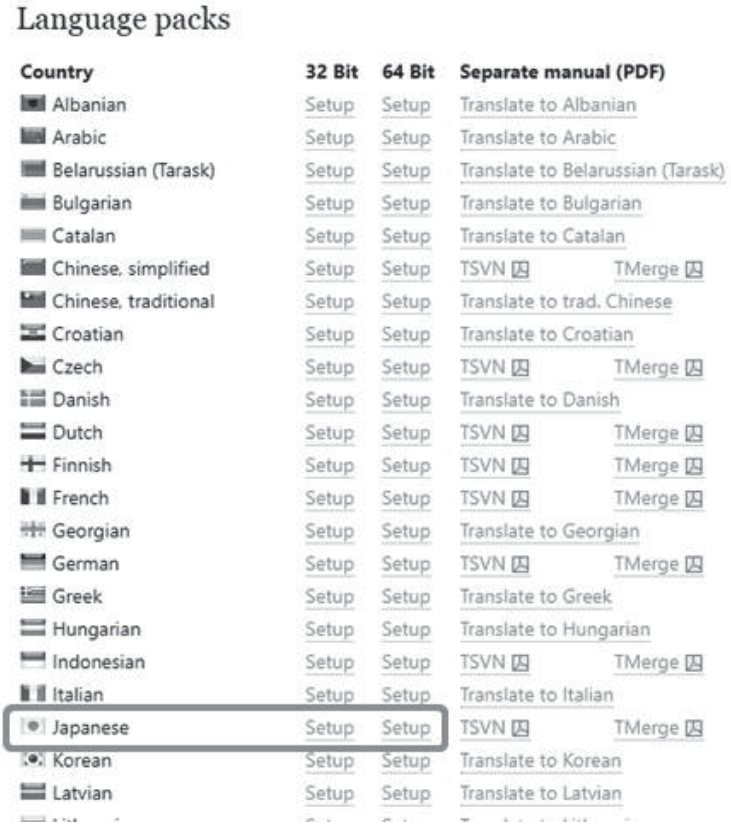


図13　TortoiseSVN ダウンロードサイト②



Language packs

Country	32 Bit	64 Bit	Separate manual (PDF)
Albanian	Setup	Setup	Translate to Albanian
Arabic	Setup	Setup	Translate to Arabic
Belarussian (Tarask)	Setup	Setup	Translate to Belarussian (Tarask)
Bulgarian	Setup	Setup	Translate to Bulgarian
Catalan	Setup	Setup	Translate to Catalan
Chinese, simplified	Setup	Setup	TSVN TMerge
Chinese, traditional	Setup	Setup	Translate to trad. Chinese
Croatian	Setup	Setup	Translate to Croatian
Czech	Setup	Setup	TSVN TMerge
Danish	Setup	Setup	Translate to Danish
Dutch	Setup	Setup	TSVN TMerge
Finnish	Setup	Setup	TSVN TMerge
French	Setup	Setup	TSVN TMerge
Georgian	Setup	Setup	Translate to Georgian
German	Setup	Setup	TSVN TMerge
Greek	Setup	Setup	Translate to Greek
Hungarian	Setup	Setup	Translate to Hungarian
Indonesian	Setup	Setup	TSVN TMerge
Italian	Setup	Setup	Translate to Italian
Japanese	Setup	Setup	TSVN TMerge
Korean	Setup	Setup	Translate to Korean
Latvian	Setup	Setup	Translate to Latvian

図14　TortoiseSVN インストール①

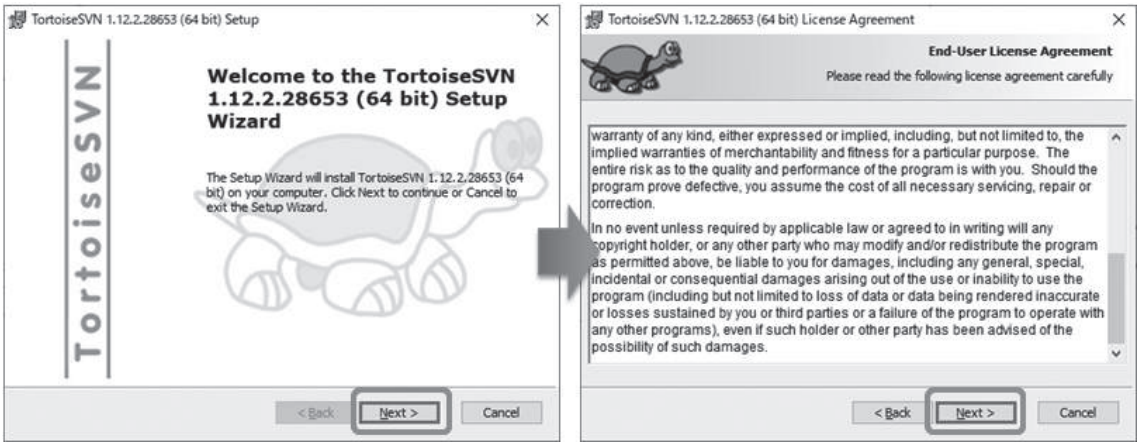


図15　TortoiseSVN インストール②

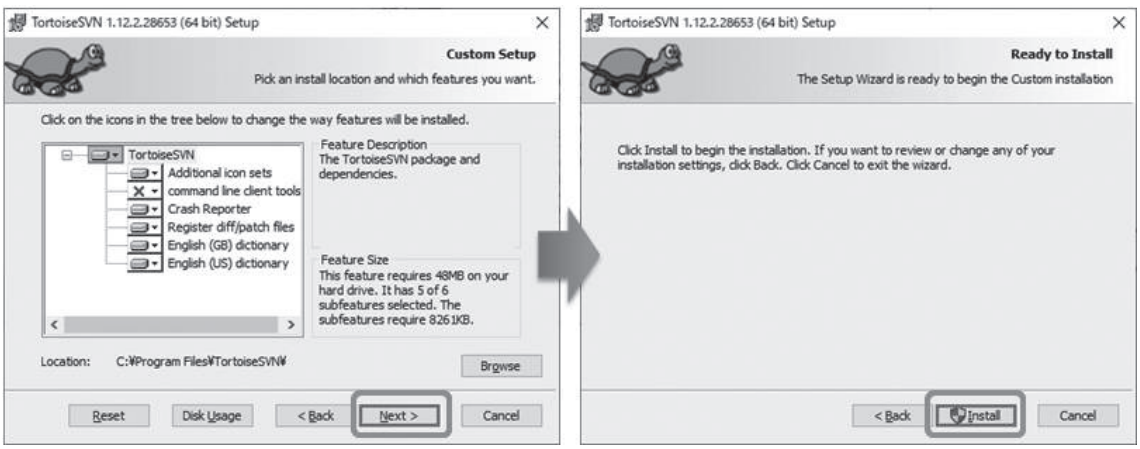


図16　TortoiseSVN インストール③

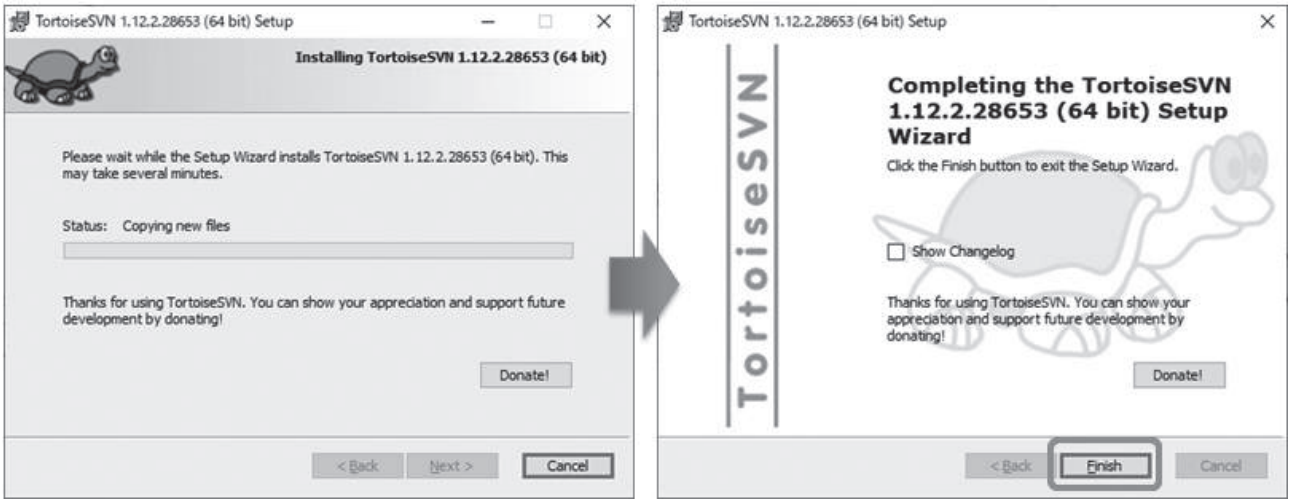


図17　TortoiseSVN インストール④

図18　TortoiseSVN の設定①

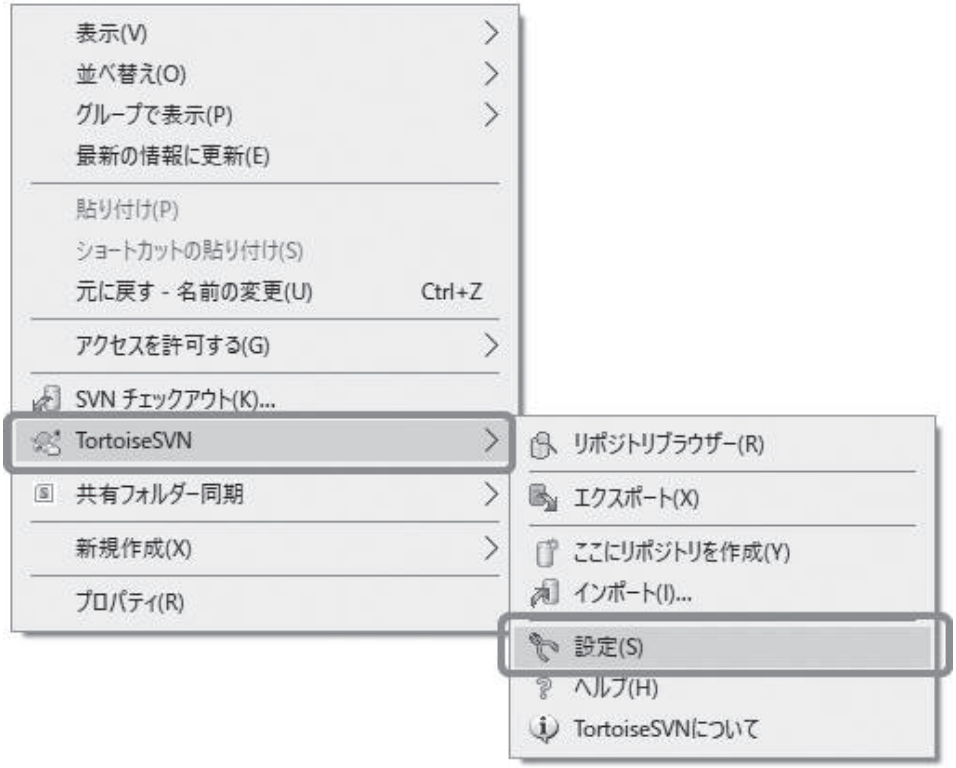


図19　TortoiseSVN の設定②

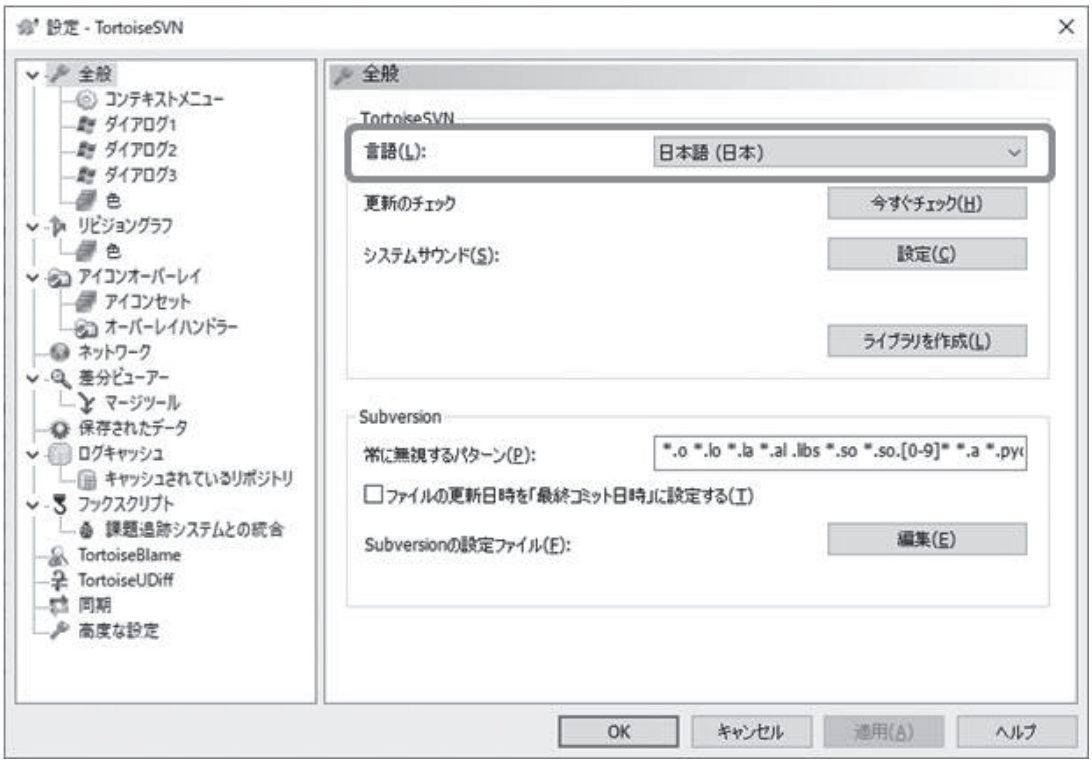


図20　作業フォルダの作成

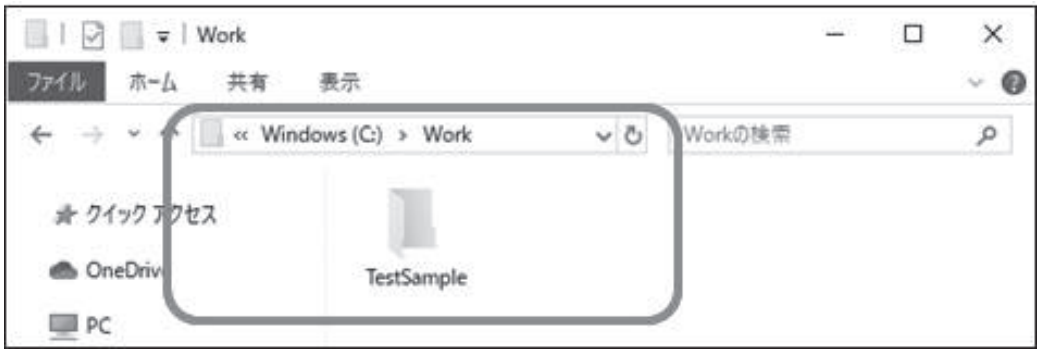


図21　リポジトリのURLをコピー

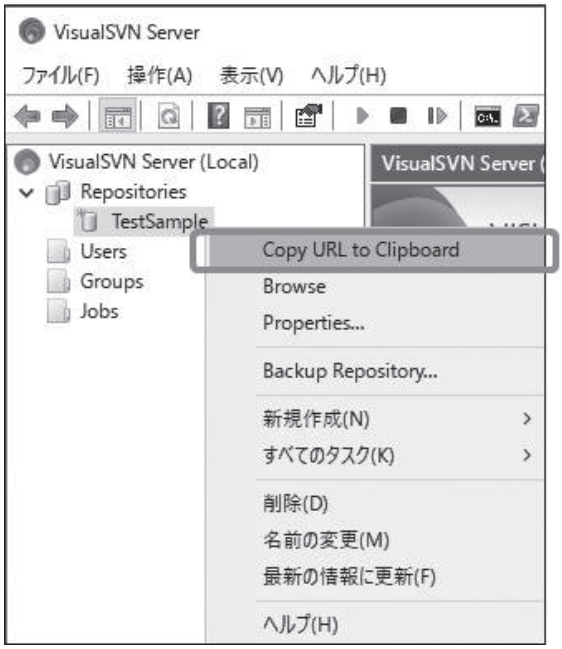


図22　作業フォルダとリポジトリの関連付け①

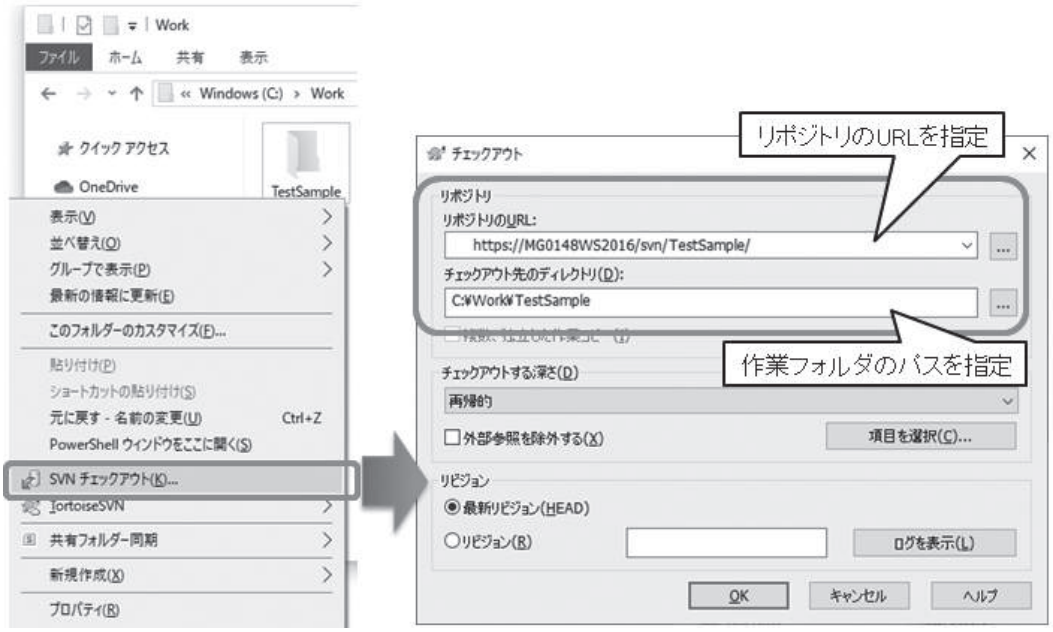


図23　作業フォルダとリポジトリの関連付け②

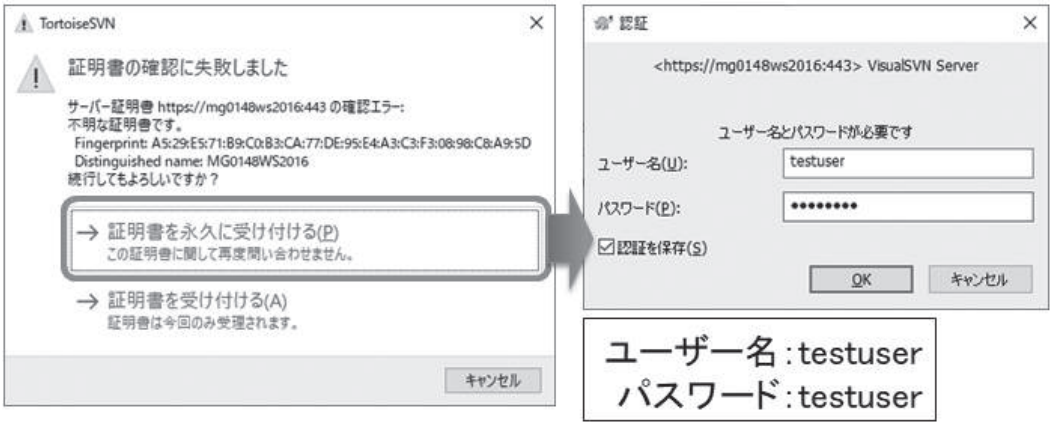


図24　作業フォルダとリポジトリの関連付け③

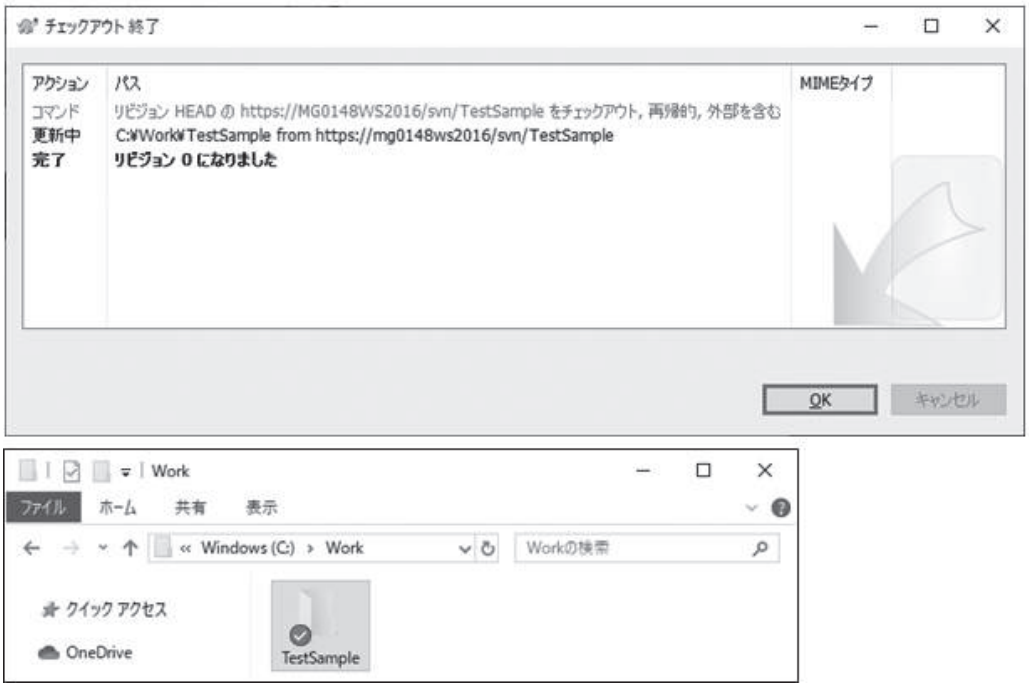


図25　Delphiのプロジェクトを新規作成

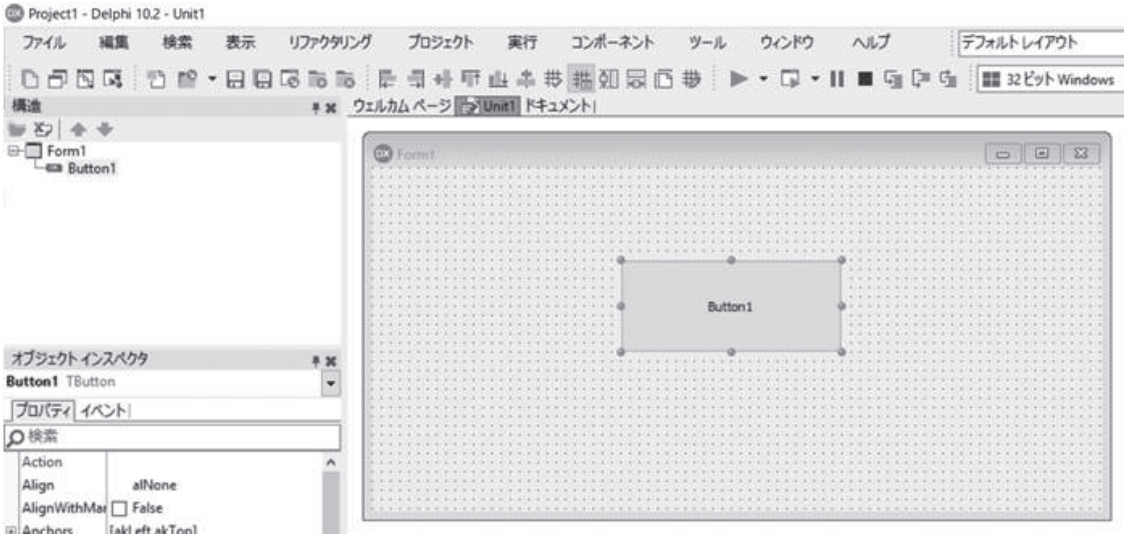


図26　保存直後の作業フォルダ

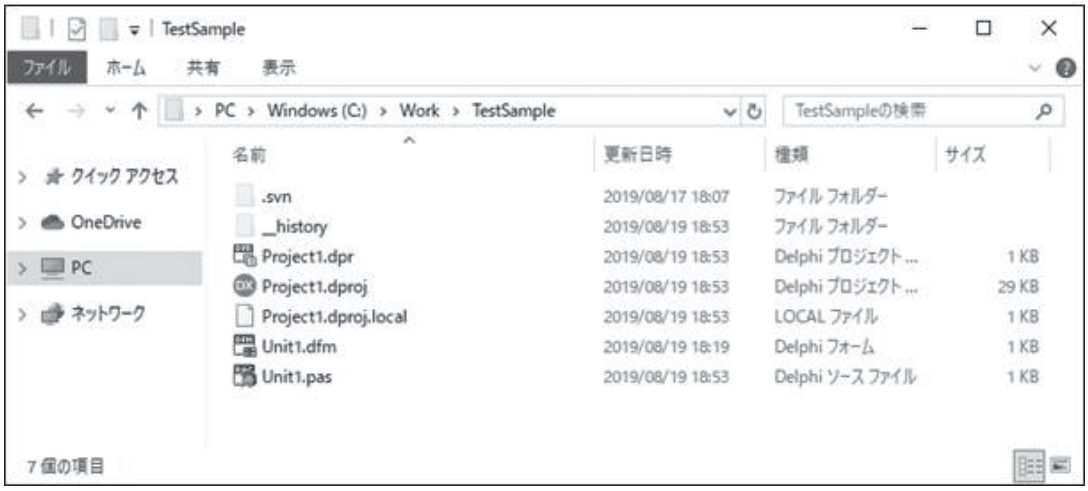


図27　リポジトリへ反映（新規）①

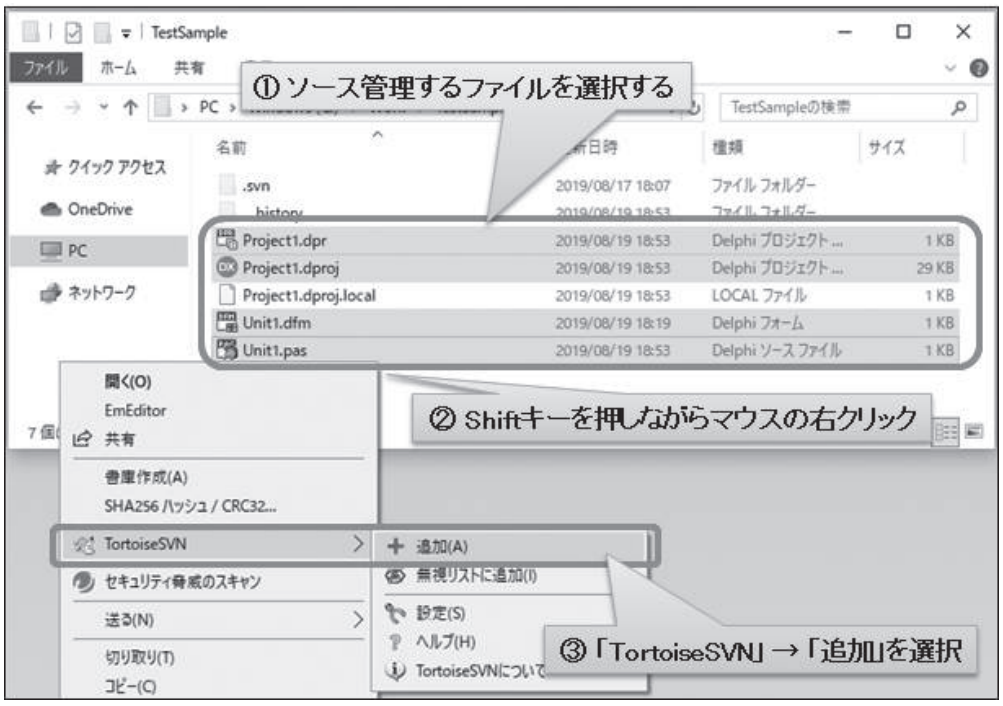


図28　リポジトリへ反映（新規）②

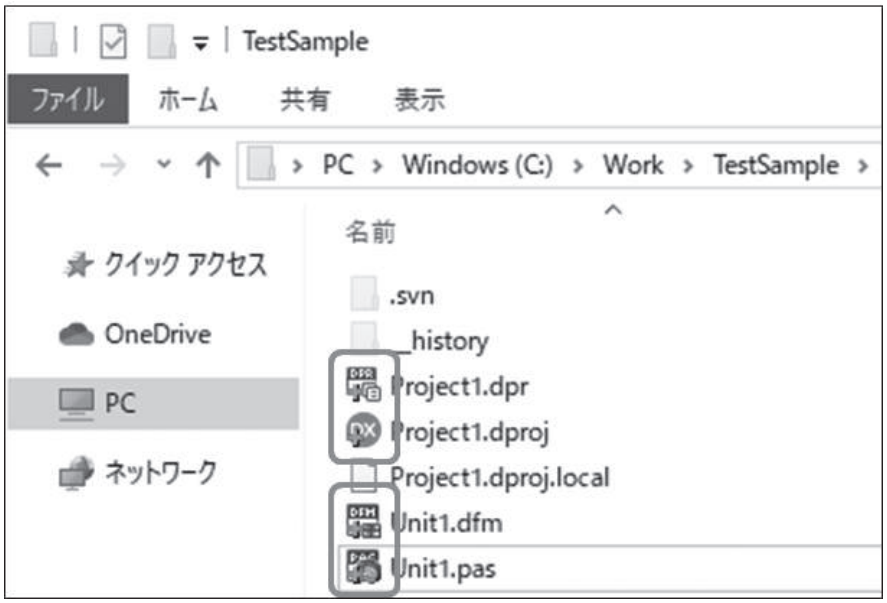


図29　リポジトリへ反映（新規）③

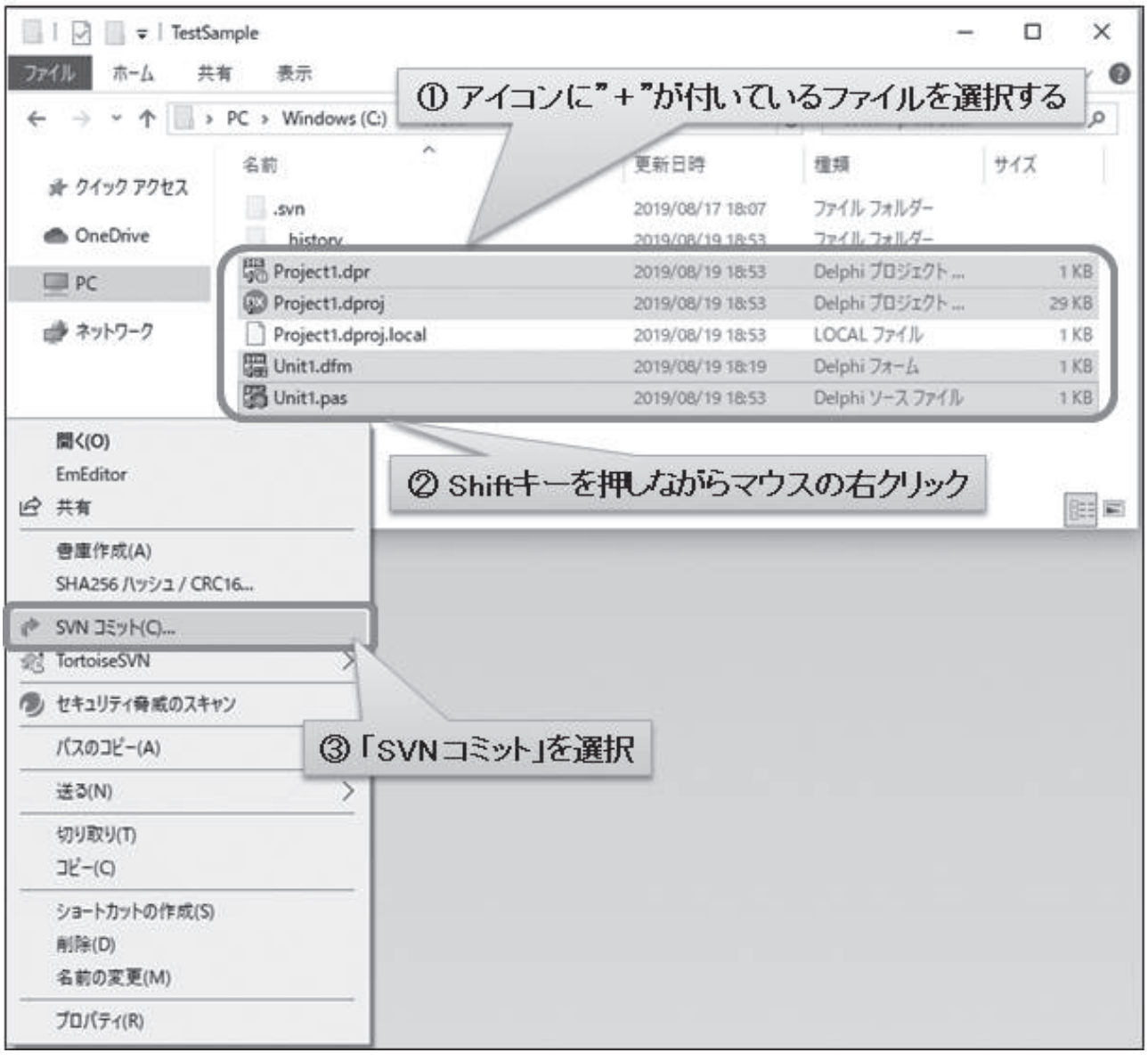


図30　リポジトリへ反映(新規)④

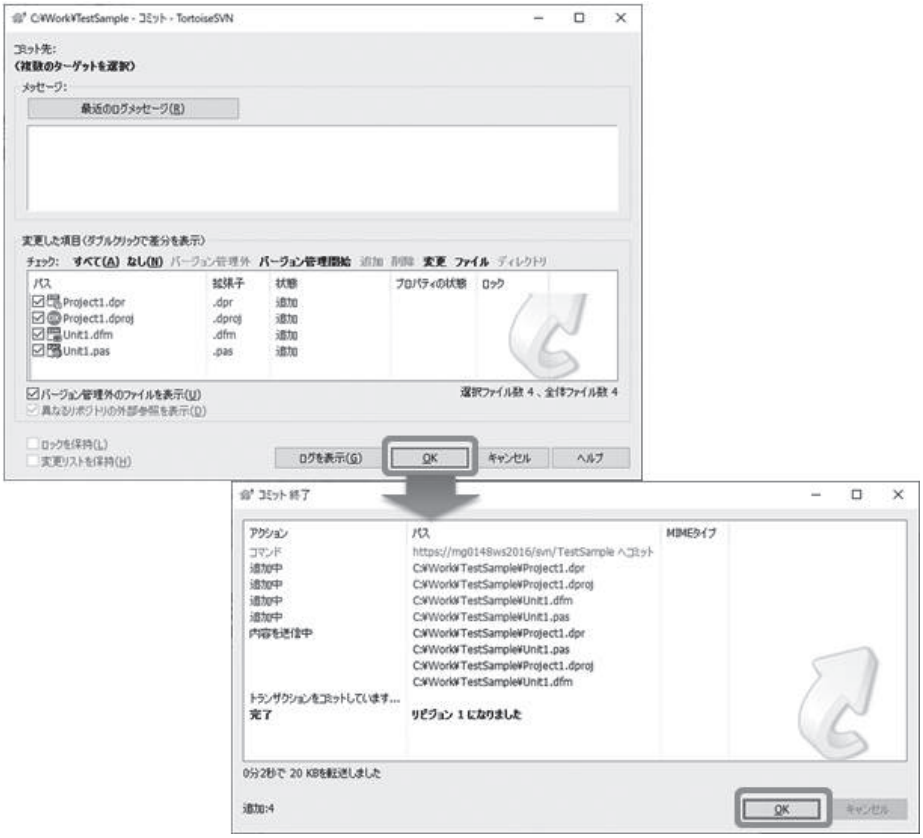


図31　リポジトリへ反映(新規)⑤

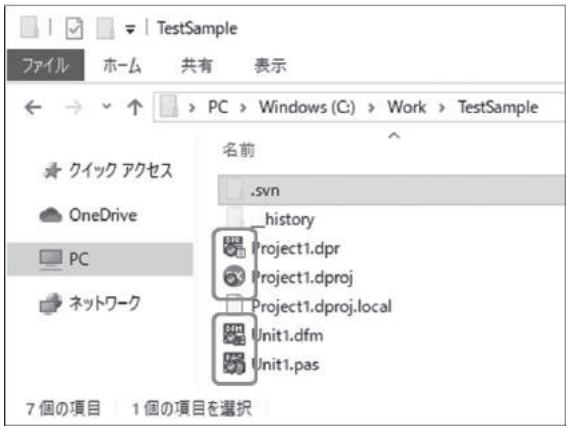


図32　リポジトリへ反映(変更)①

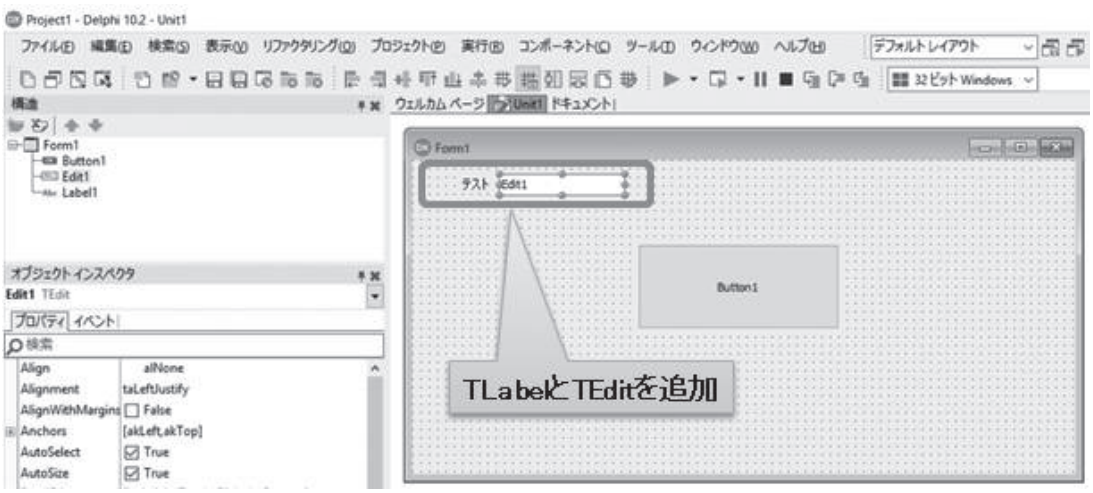


図33　リポジトリへ反映(変更)②

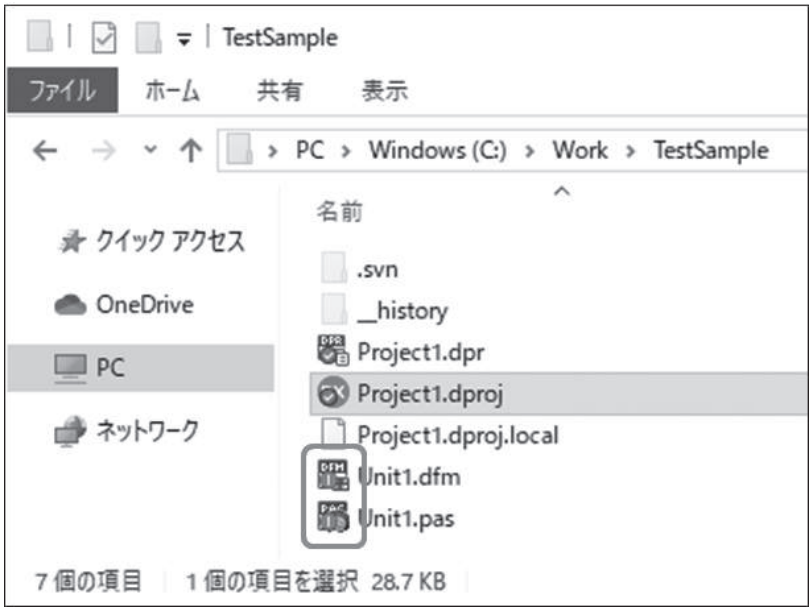


図34　リポジトリと同期①

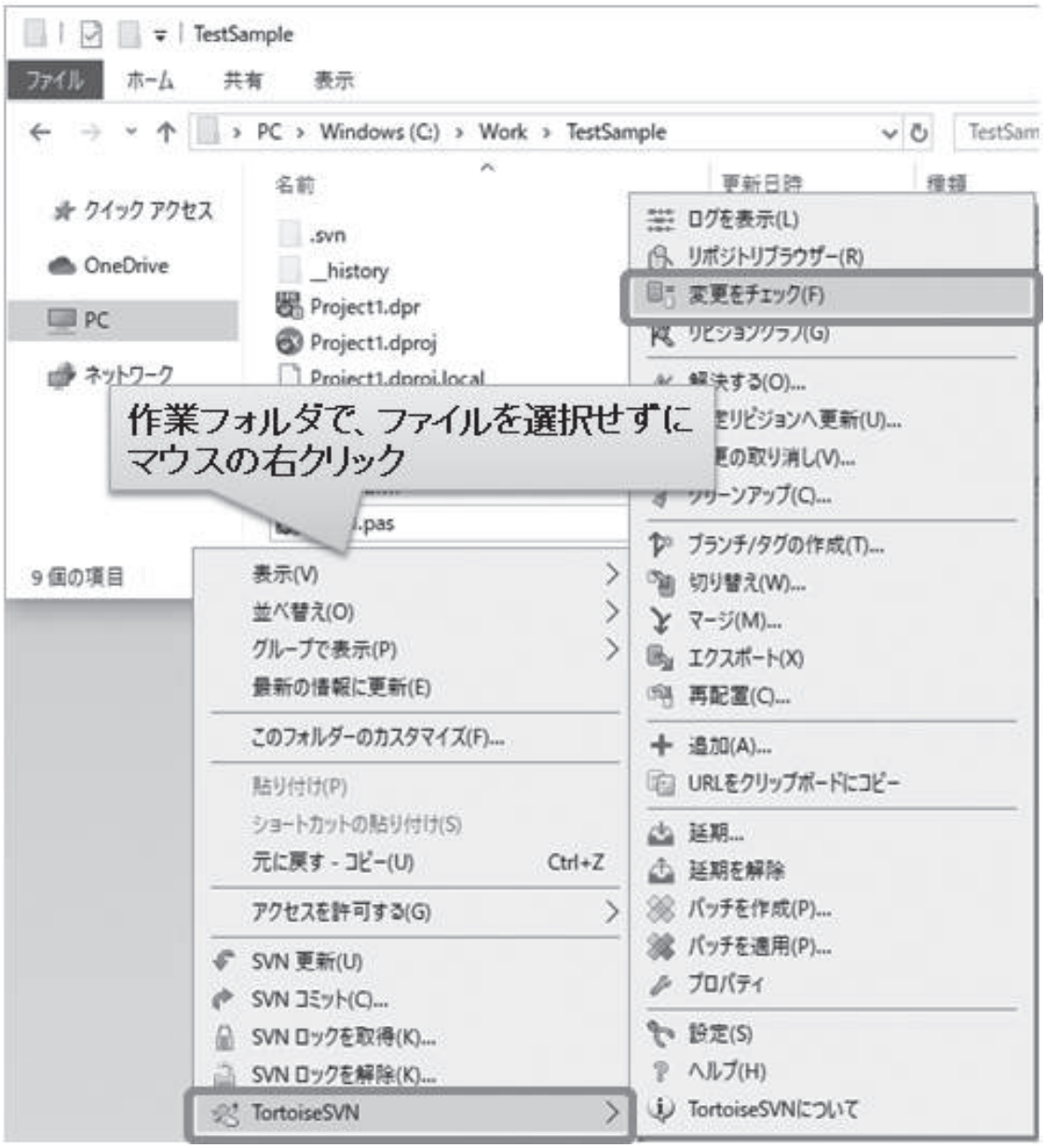


図35　リポジトリと同期②

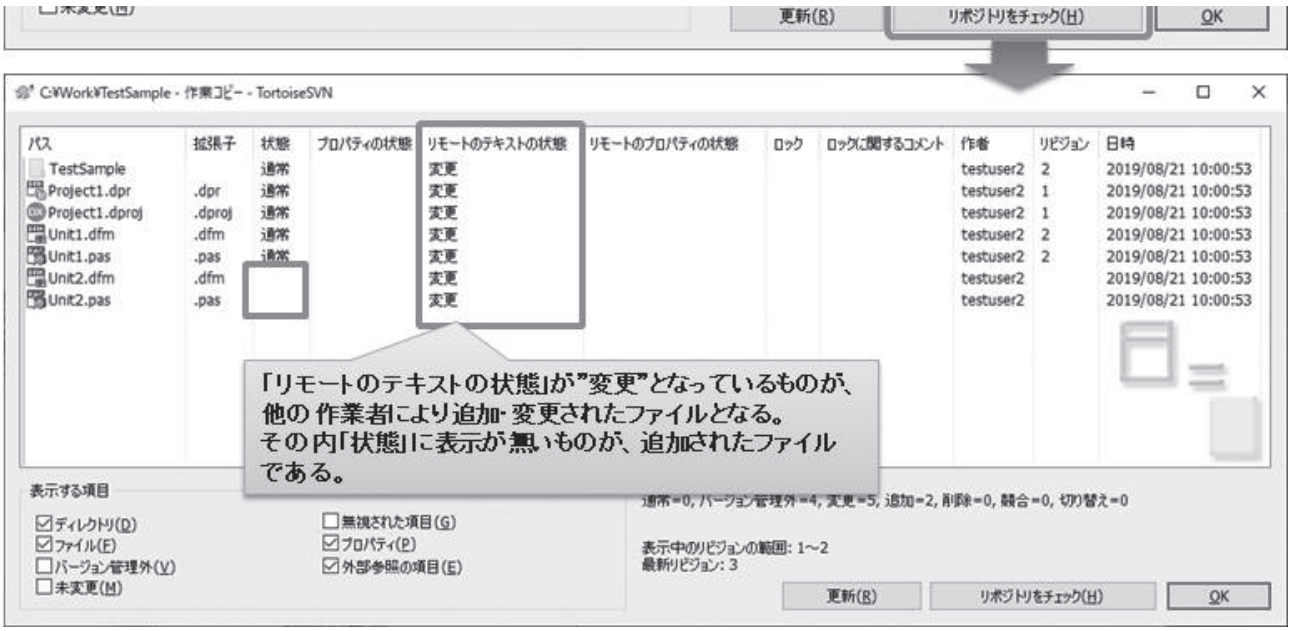


図36　リポジトリと同期③

図37　リポジトリと同期④

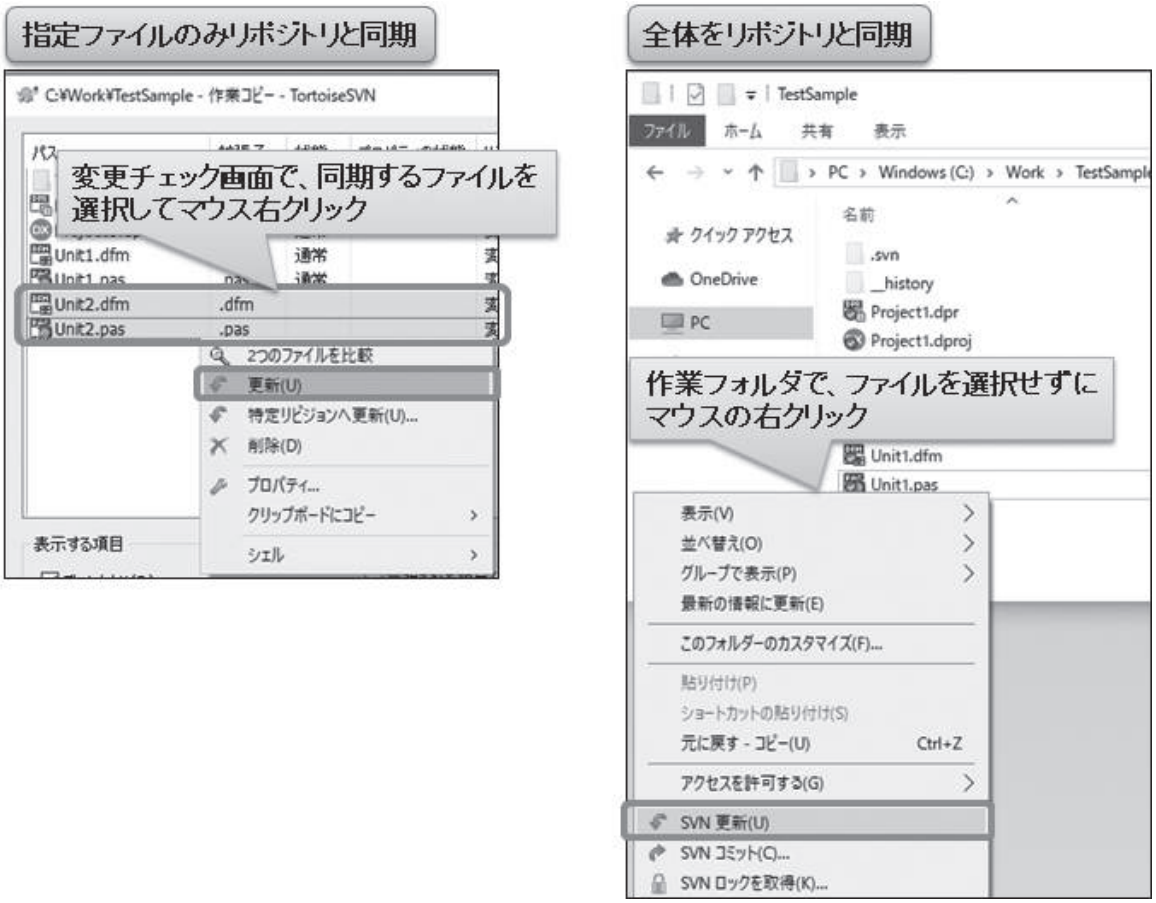


図38　ロックについて①

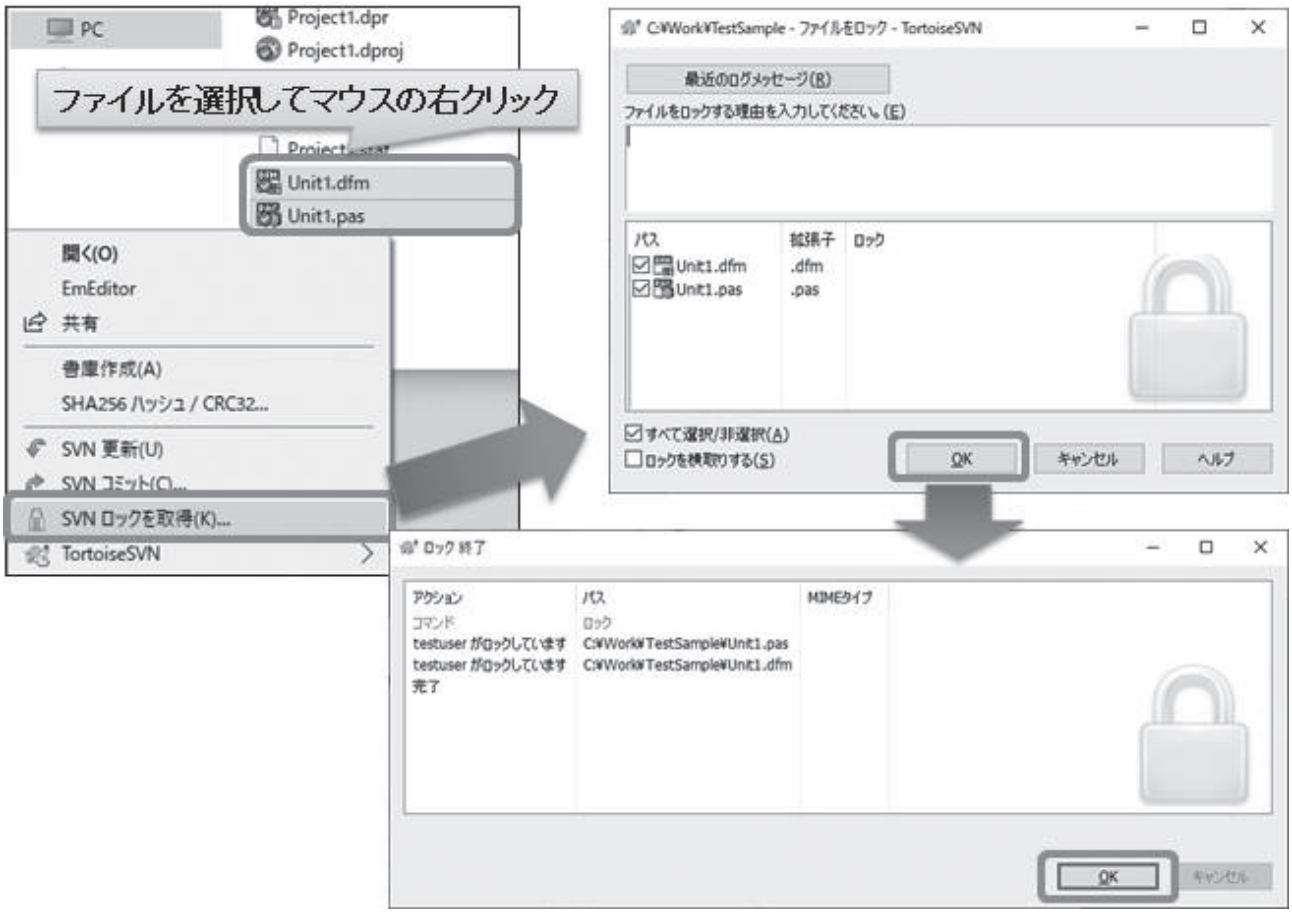


図39　ロックについて②

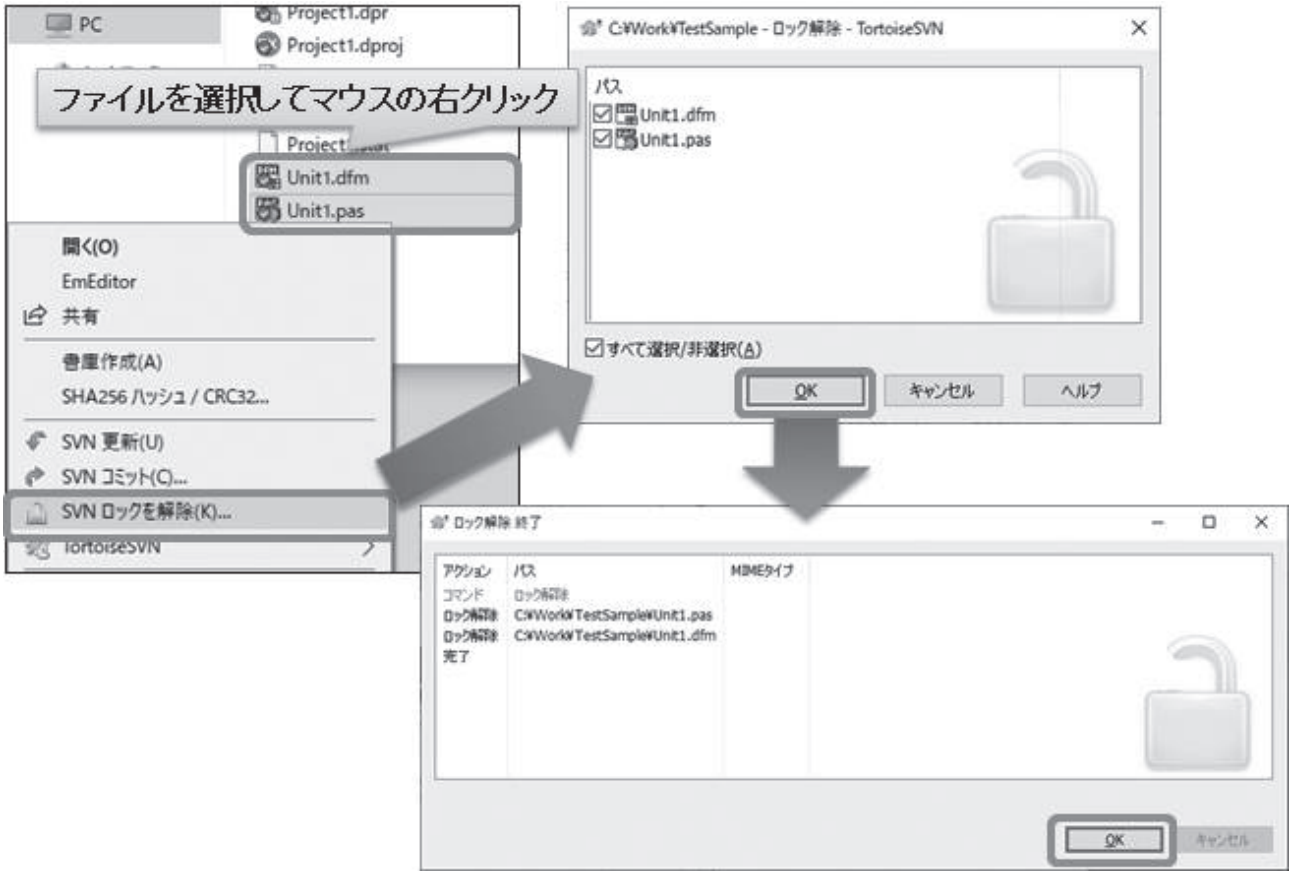
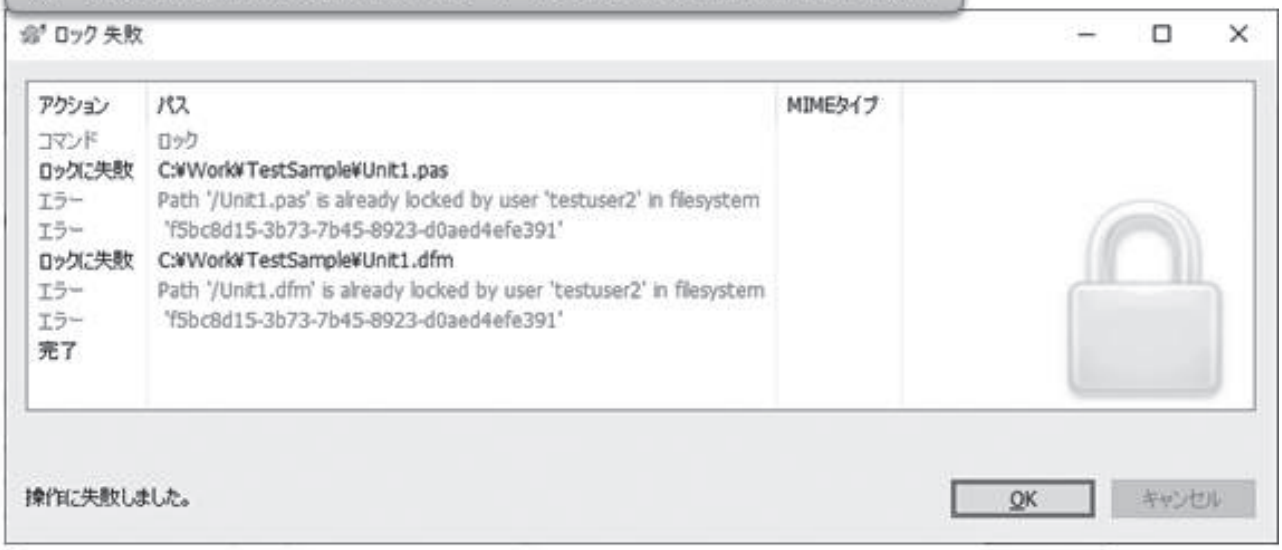


図40　ロックについて③

図41　ロックについて④

他の作業者がロックしているファイルをロックしようとした場合



他の作業者がロックしているファイルを変更してコミットしようとした場合

図42　変更履歴を確認①

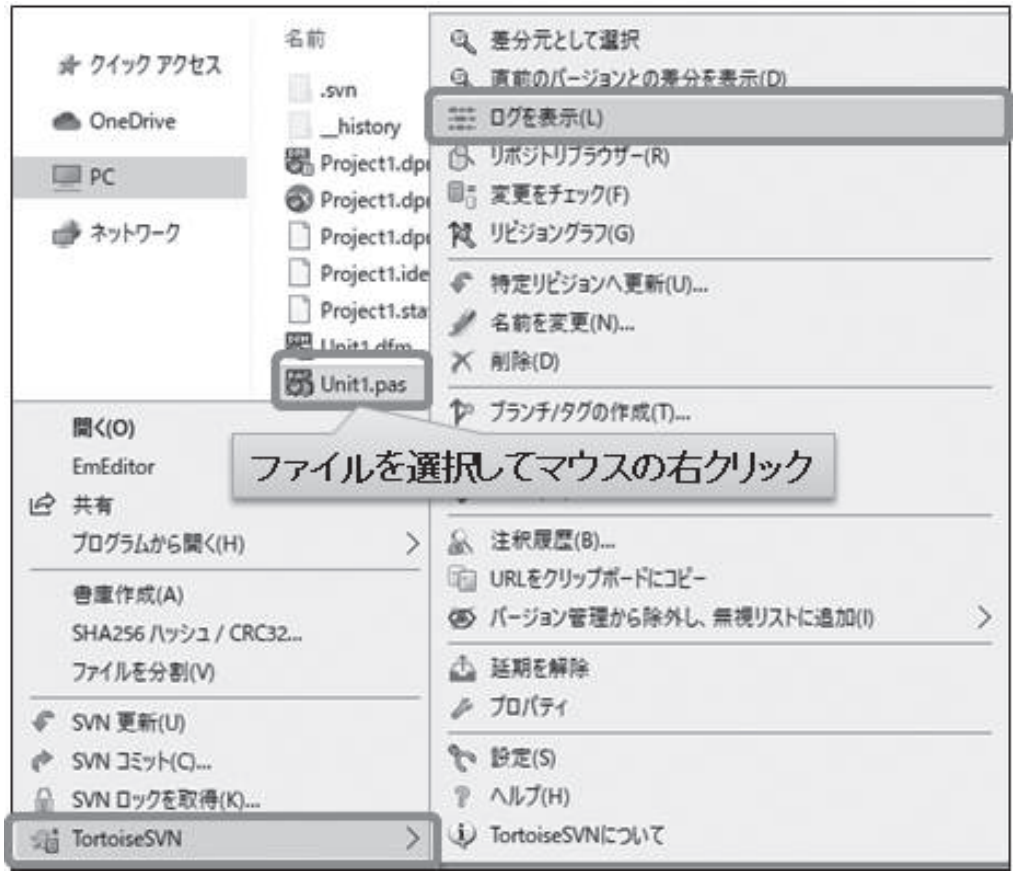


図43　変更履歴を確認②

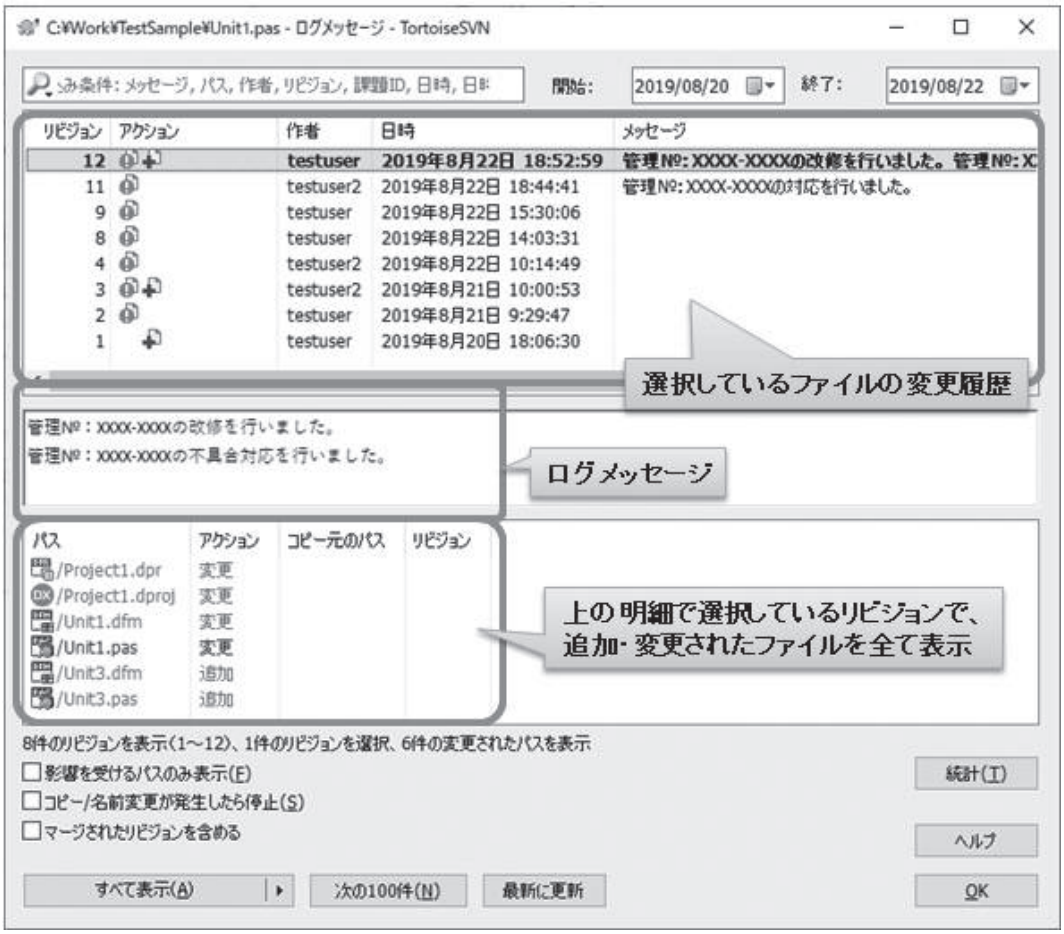


図44　変更履歴を確認③

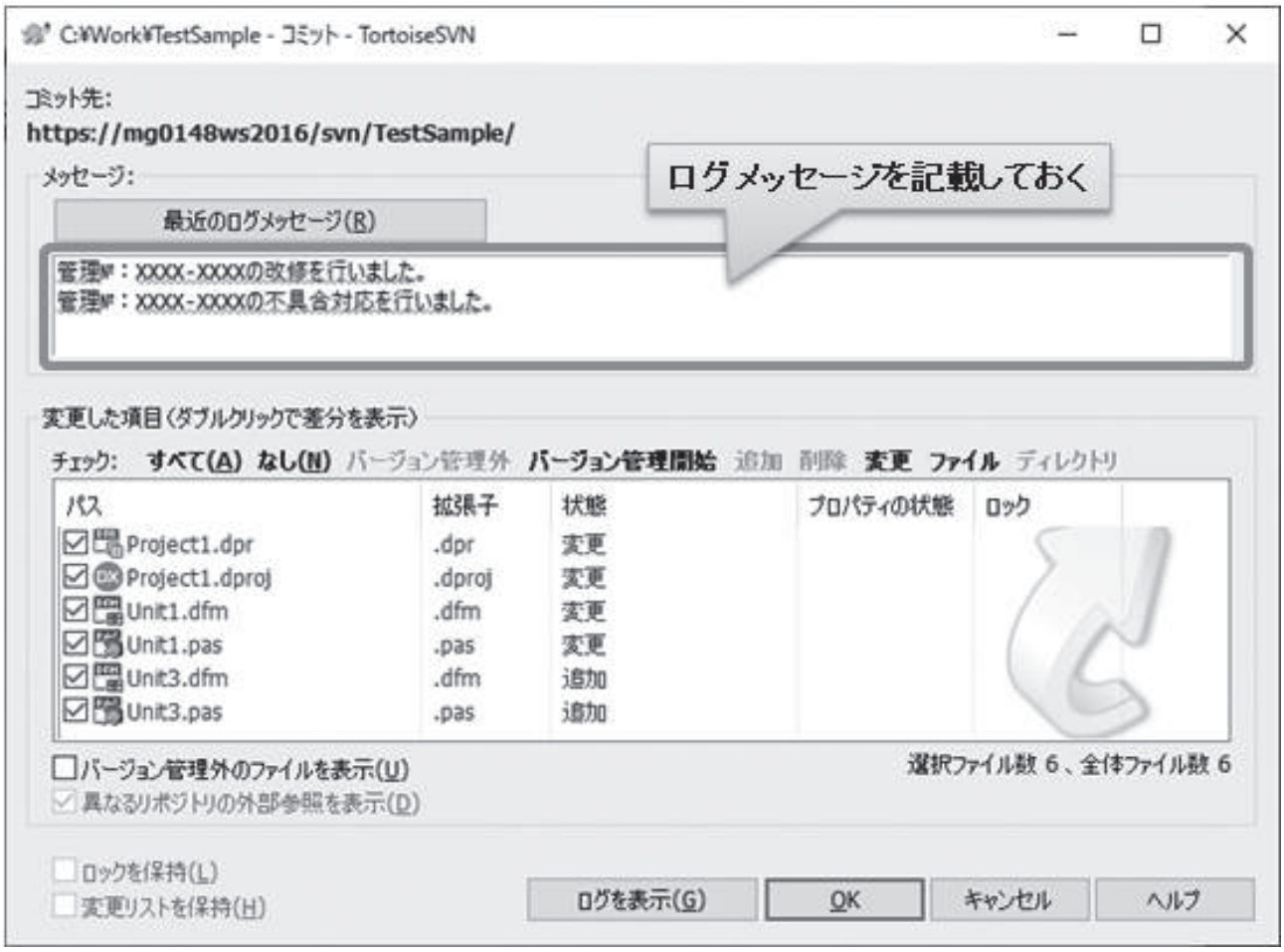
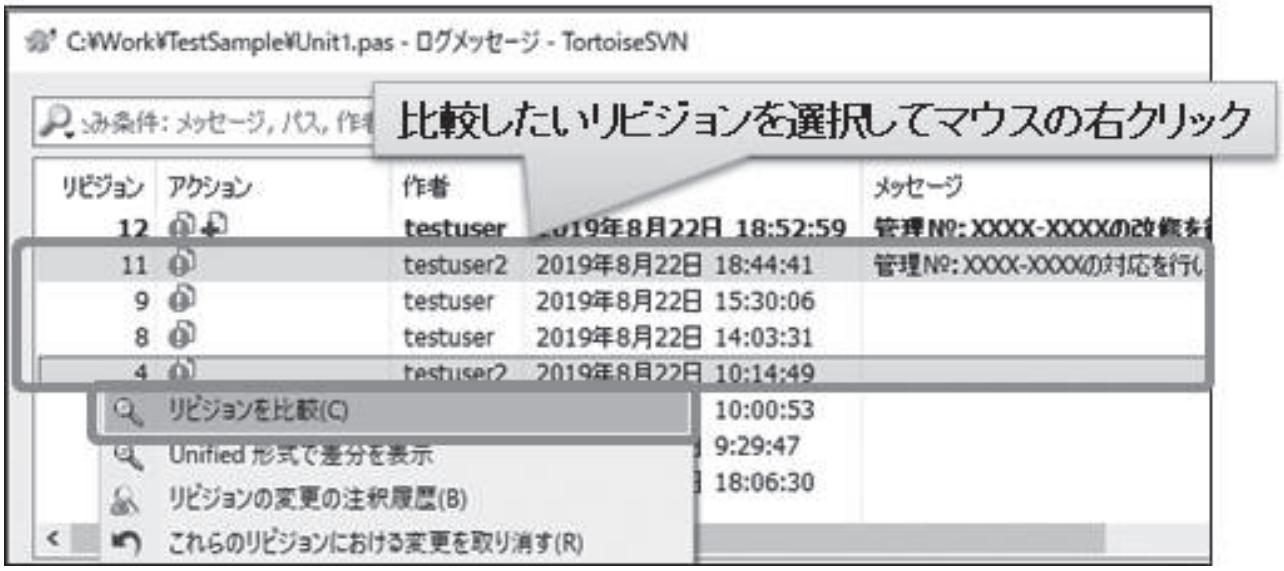


図45　変更履歴の差分を確認



佐田 雄一

株式会社ミガロ.

RAD事業部 技術支援課

[Delphi/400]

Enterprise Connectorsを利用したクラウド連携テクニック

略歴
1985年12月6日生まれ
2009年3月 甲南大学 経営学部卒業
2009年4月 株式会社ミガロ. 入社
2009年4月 システム事業部配属
2019年4月 RAD事業部配属

現在の仕事内容
Delphi/400でのシステム開発や保守作業の経験を経て、現在はサポート業務を担当している。

1. はじめに

基幹系の業務システムは長らく、社内上のサーバーにデータを蓄積するオンプレミス（自社運用）で稼働してきた。Delphi/400のシステムでも、IBM iを中心にオンプレミス環境で稼働していることが多い。

従来はすべてのシステムをオンプレミスで稼働させるのが一般的だったが、近年はWebを介して稼働するさまざまなクラウドサービスが台頭しており、それらを業務で活用することも増えている。オンプレミスのシステムとクラウドサービスは別々に稼働していることが多いのが現状であり、これらの連携が課題になることもあるだろう。このようなオンプレミスとクラウドサービスとの連携でも、Delphi/400が活用できる。

本稿では、Delphi/400からクラウドサービスへのアクセス用に用意されたEnterprise Connectorsを利用して、IBM iのデータとクラウドサービスを連携させる方法について紹介する。

2. Enterprise Connectorsとは

Enterprise Connectorsとは、エンバカデロ・テクノロジーズ社がCData Software社との提携によって、2017年から提供を開始したコンポーネント群で、Delphi/400に搭載されたデータベースエンジンFireDACを使用してクラウドサービスに接続する。

各クラウドサービスには、それぞれのベンダーが提供するWeb APIがあり、それを使用することでDelphi/400等の各種クライアントからクラウドサービスに接続できた。しかし「クラウドサービスごとに接続するAPIの仕様が異なり、個別にAPIの仕様を調査の上、実装しなければいけない」「API側の仕様のアップデート間隔が短いことが多く、仕様変更の都度、個別に修正を実装しなければならない」など、課題が多いのが現状である。

またAPIの基本設計自体が変わる、たとえばデータ形式そのものがXMLからJSONに変更された例も存在する。

Enterprise Connectorsは、こういったベンダー側のAPI仕様をラッピングし、FireDACの共通インターフェースで活用できるようにした。これにより各種クラウドサービスに対して、IBM iやオープン系のDBへのアクセスと同様に、SQLやストアドプロシージャを使用してアクセスできる。【図1】

Enterprise Connectorsのサブスクリプションには、主要な約80種類のクラウドサービスに接続可能な「Enterprise」版と、約150種類のクラウドサービスに接続可能な「Enterprise Plus」版が存在する。

本稿ではこれらの中から代表的なものとして、業務に限らず広く使われているTwitterおよびGoogleスプレッドシートを取り上げ、Delphi/400と連携可能な利用例について紹介する。

これらの連携を使用すれば、たとえば「IBM i上の売上データをもとにGoogle Drive上にGoogleスプレッドシートを作成する」「IBM i上の余剰在庫情報を

図1　Enterprise Connectorsの概要

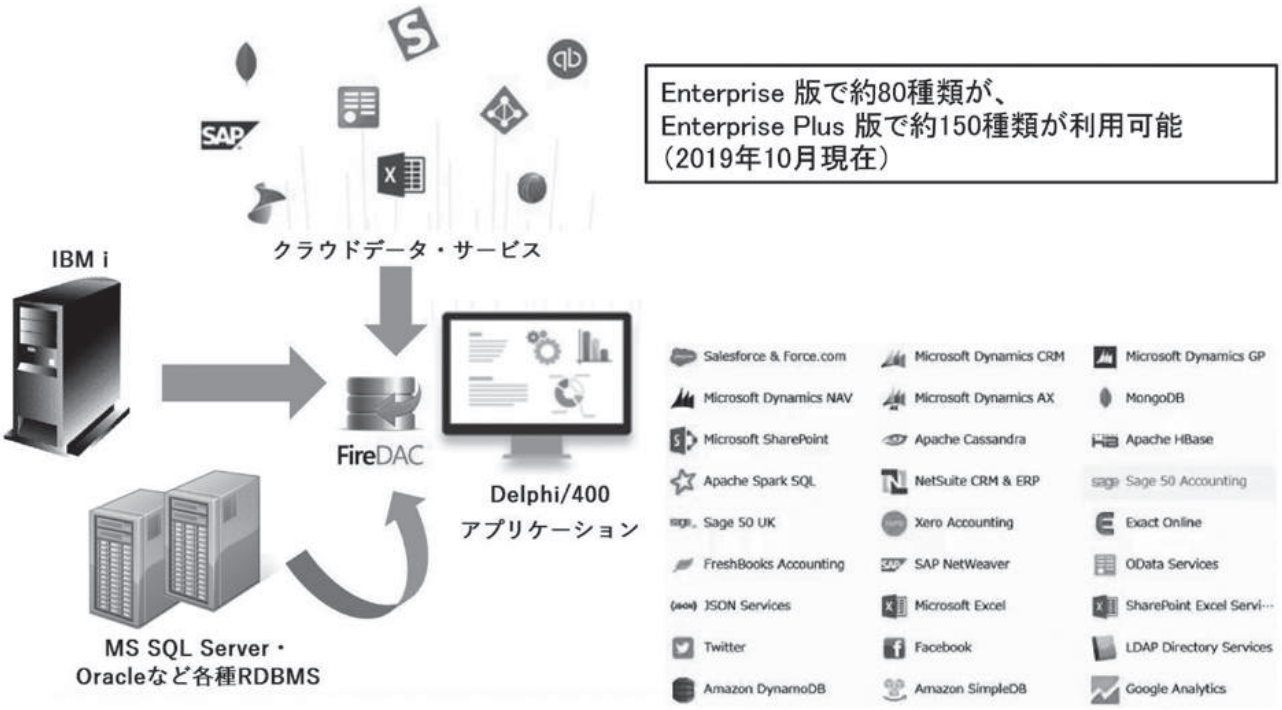
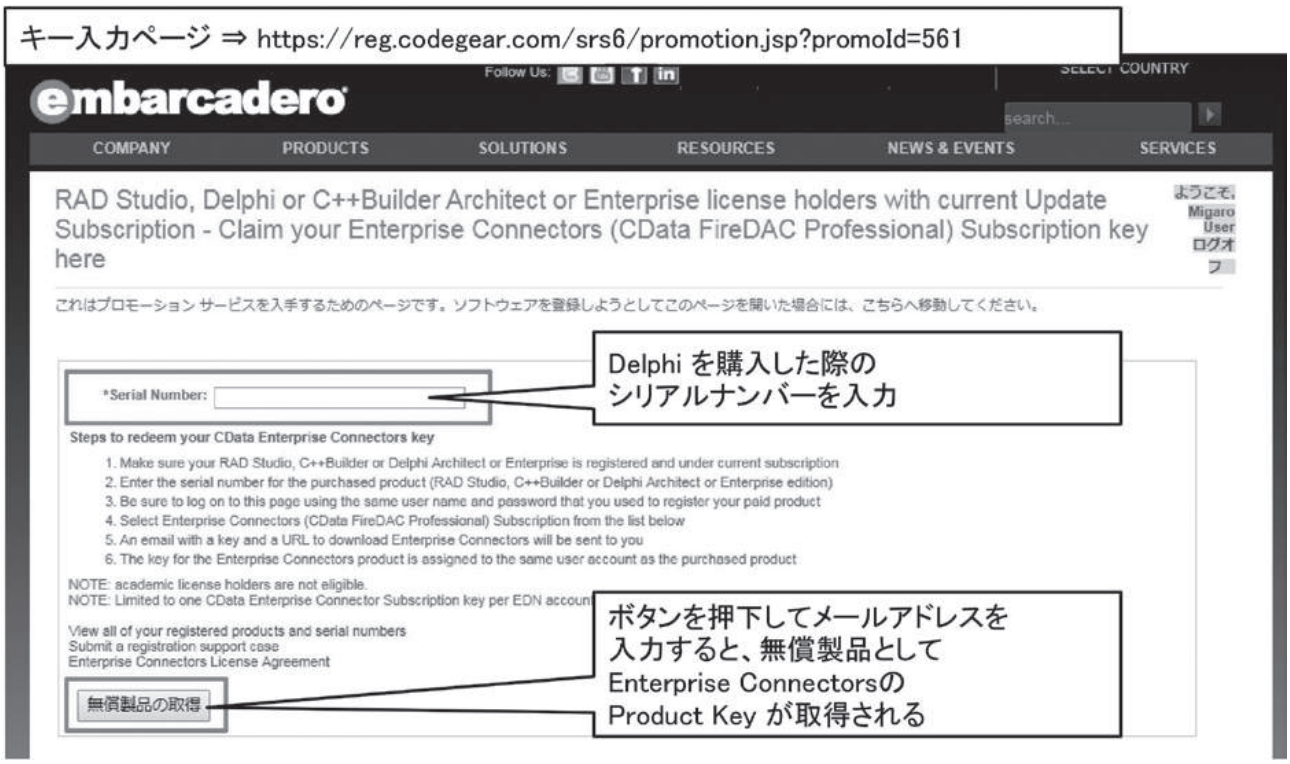


図2　Enterprise Connectorsの取得



もとに、Twitterにキャンペーン情報をツイートする」といった使い方ができる。

なおEnterprise Connectorsは、Delphi/400 Version 10.2 Tokyoからの対応となっており、本稿のサンプルでも同バージョンを使用している。また本稿で扱うそれぞれのコネクターは、「Enterprise」版のサブスクリプションで利用可能である。

3. Enterprise Connectorsのインストール

Enterprise Connectorsは、Delphi/400のオプションのライセンス製品であるが、アップデートサブスクリプションが有効なDelphi/400 10.2 Tokyoをご利用であれば「Enterprise」版を無償で取得できる。

まず、エンバカデロ・テクノロジーズ社のサイト（https://reg.codegear.com/srs6/promotion.jsp?promoId = 561）にアクセスし、Delphi購入時のシリアルナンバーを入力してProduct Keyを取得する。【図2】

次にCData Software社のサイト（https://www.cdata.com/firedac/download/）にアクセスし、必要なコネクターを選択すると【図3】のような画面が表示される。それぞれのコネクターは独立しており、コネクターごとに専用のインストーラと、それによってインストールされる専用のコンポーネントが個別に存在する。

先ほどのProduct Keyとそれを取得するために使用したメールアドレスを入力すると、対象のコネクターのインストーラがダウンロードされる。

インストールが完了すると、ヘルプドキュメントがスタートメニューに登録される。そのコネクター関連の各コマンドや使用方法が英語で記載されているので、開発時の大きな助けになるだろう。【図4】

4. Enterprise Connectorsの開発方法

本章では、各種コネクターの中でもシンプルで扱いやすい「CData CSV FireDAC Components」を使用して、CSVファイルにアクセスする方法を紹介する。

CSVといえば、通常はTStringListなどで読み書きすることが多いが、Enterprise Connectorsの使用によって、CSVデータに対して、一般的なDBアクセスと同様にSQLを使用したデータの抽出や更新が可能になるメリットがある。

それでは、具体的なサンプル作成を通じて開発方法を紹介する。今回のサンプルでは「CSVファイルを読み込み、TDBGridに表示」「IBM iから取得したデータをもとにCSVファイルにレコードを追加」の2つの機能を実現する。

Delphi/400 10.2 Tokyoを起動したら、アプリケーションを新規作成し、新しいフォームの上にコンポーネントを配置する。

まずEnterprise ConnectorsでCSVに接続してデータを読み書きするためのコンポーネントとして、「CData CSV FireDAC Components」をインストールした際に追加されるコネクター「TFDPhysCDataCSVDriverLink」を配置する。

これはIBM iへの接続におけるTFDPhysCO400DriverLinkに相当し、Enterprise Connectorsはコネクターごとのドライバリンクコンポーネントを定義するのがポイントである。また通常のFireDAC接続と同様にTFDConnection・TFDQueryと、取得結果を画面に表示するTDataSource・TDBGridを配置する。

次にIBM iに接続してデータを取得するコンポーネント群として、TFDConnection・TFDQueryおよび、IBM iに接続するためのコネクターとなるTFDPhysCO400DriverLinkを配置する。

あとは処理実行のためにTEditやTButtonなどを配置した結果、【図5】のような画面になる。

そしてTFDConnectionをダブルクリックして、FireDAC接続エディタを開き、ドライバIDに「CDataCSV」を選択する。すると【図6】のように、接続時の各種パラメータが自動でセットされる。

ローカルのCSVファイルに接続する場合は、「URI」パラメータに対象CSVのフォルダを指定する。また「UseRowNumbers」や「RTK」といったパラメータを【図6】のように設定する。

次に、接続するCSVを準備する。本稿のサンプルでは、【図7】のようなレイアウトで「WORKCSV.csv」というサンプルCSVを作成した。CSVのファイル名は､拡張子を除いた「WORKCSV」部分がテーブル名に、1行目のカンマ区切りテキストがそのままフィールドIDに、2行目以降のカンマ区切りテキストがレコードの内容となる。

(1) CSVファイルを読み込み、TDBGridに表示

画面設計が完了したら、【ソース1】のようにロジックを記述する。プログラムを実行して「①SQL SELECT」ボタンを押下することで、CSVの内容がDBとして明細に表示される。【図8】のTDBGridの内容は、単純な全件取得の結果であるが、一部フィールドだけの取得や、WHERE句を使用した絞り込みも可能である。

FireDAC接続エディタの設定で、「UseRowNumbers」をTrueに指定して書き込みを有効にした場合、行番号で各レコードをユニークキーとするための「RowNumber」というフィールドも追加される。【図8】では明示的に表示されたままにしているが、通常は列ごと非表示に設定しても問題ない。

(2) IBM iから取得したデータをもとにCSVファイルにレコードを追加

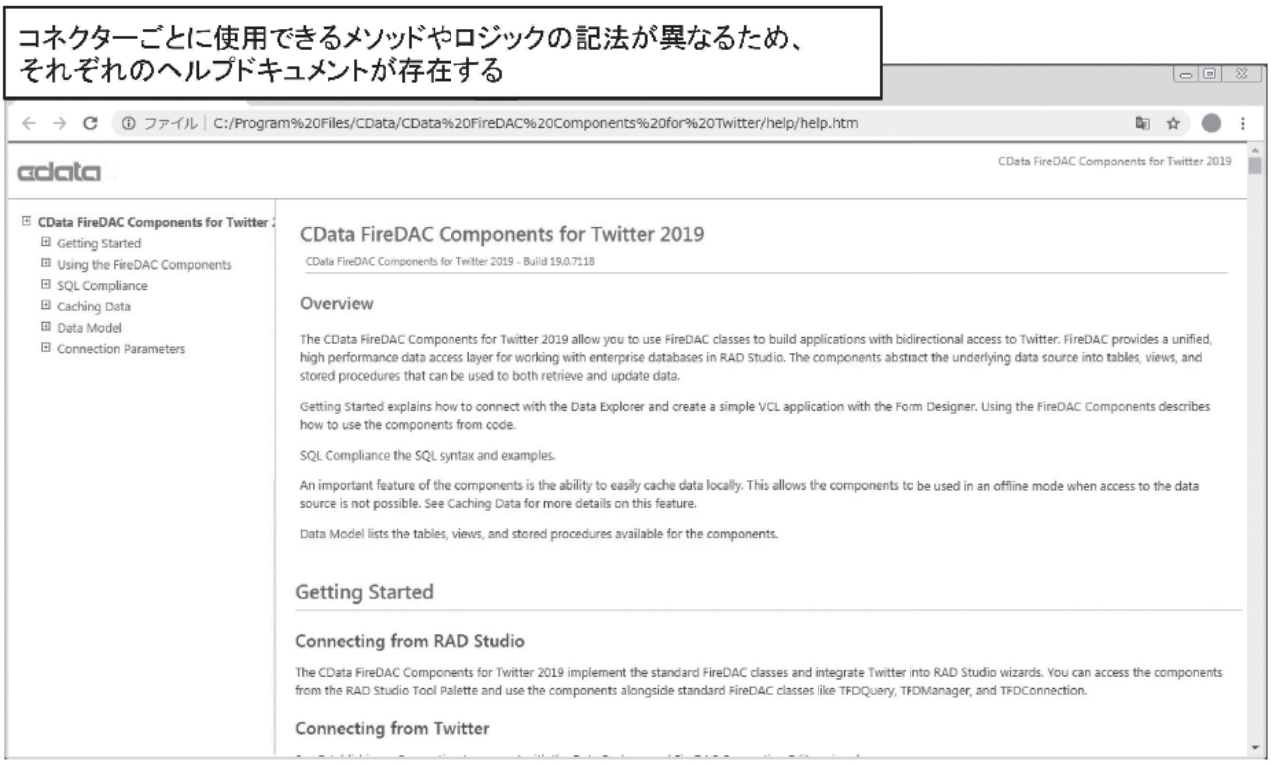
次に、【ソース2】のようにロジックを記述する。顧客コードに値を入力して「②登録」ボタンを押すと、IBM iから条件に合致するデータを取得し、その結果をSQLのINSERT文を使用してCSVに行を追加できる。【図9】

5. クラウドサービスへの接続･活用方法

前章では、Enterprise Connectorsの基本的な開発について紹介したが、ここでは具体的なクラウドサービスの連携例として、業務に限らず広く一般でも使用されるTwitterおよびGoogle Apps（Googleスプレッドシートおよび Google Drive）について紹介する。

図3　Enterprise Connectorsのインストール

図4　Enterprise Connectorsのヘルプドキュメント



5-1. Twitterと連携した情報発信と解析

最初に、有名人・著名人も数多く利用するソーシャルメディアの1つで、Google検索でもヒットしないような口コミをリアルタイムで調べられるTwitterの接続、利用方法を紹介する。

(1) Twitter APIのアプリケーション開発準備

まず、Twitterの開発者サイト（https://dev.twitter.com/apps）から、新しいTwitter Web APIを作成する。開発者サイトにログインしたら、「Create an app」ボタンを押し【図10】、作成するアプリケーションのタイトルや使用目的を英語で入力する。【図11】

入力が完了したら、Twitterアプリケーション開発に必要なAPIキーとアクセストークンを入手できる。【図12】【図13】

なお開発者サイトを利用するにあたっては、Twitterの開発者アカウントの作成が必要である。本稿では作成手順については割愛するが、Google等で「Twitter 開発者アカウント」と検索すれば、その時点での最新情報が参照できるだろう。

(2) Twitter APIを使用したツイート送信方法

APIを作成したら、次はDelphi/400側の開発を行う。ここでは、「CData Twitter FireDAC Components」を利用する。Delphi/400 10.2 Tokyoを起動したら、アプリケーションを新規作成し、新しいフォームに【図14】のようにコンポーネントを配置する。

前章のCSV接続と同様、Delphi/400でIBM iに接続してデータを取得するコンポーネント群と、Enterprise ConnectorsでTwitterに接続するコンポーネント群をそれぞれ配置する。なお、Twitterで使用するコネクターは、「TFDPhysCDataTwitterDriverLink」である。

前章のCSV接続と同じように、TFDConnectionをダブルクリックしてFireDAC接続エディタを開き、ドライバIDに「CDataTwitter」を選択すると、【図15】のように接続時の各種パラメータが自動でセットされる。

ここでセットが必要なパラメータは、接続目的にあわせて【図16】のように設定する。また「②選択行の商品についてツイートする」ボタン押下時の処理を、【ソース3】のように記述する。TFDQueryを使用して、テーブル「Tweets」にSQLでレコードを追加するだけで、ツイートが送信できる。

Delphi/400でIBM iからデータを取得し、【図17】のようにTDBGridに表示した状態から、「②選択行の商品についてツイートする」ボタンを押下すると、【図18】のようにツイートが送信される。送信されたツイートにはユニークキーとなるID（数字18～19桁）が文字型で採番されており、ブラウザのURLで確認できる。

送信時のSQLにより、「In_Reply_To_Status_Id」フィールドに別のツイートのIDをセットすると、そのツイートへのリプライになる。またツイートの本文中に別のツイートのURL「https://twitter.com/XXX/status/（ツイートのID)」を記載すると、そのツイートに対する引用リツイートになる。

本稿の例ではSQLのINSERT文でツイートを送信したが、逆にSQLのDELETE文を発行することで、自分が送信したツイートを削除することも可能である。その際には、対象ツイートのIDをWHERE句に指定する。

なお、それぞれのツイートはFireDAC接続時に認証を受けたユーザーのアカウントで発信されるため、当然のことながら他人のアカウントになりすますことはできない。

(3) Twitter APIを使用した各種データの参照方法

次に、Twitter APIを使用して各種テーブルや、キーワードを指定したツイートを検索するアプリケーションを作成する。

Delphi/400 10.2 Tokyoでアプリケーションを新規作成し、新しいフォームに【図19】のようにコンポーネントを配置する。【ソース4】のようにロジックを記述し、プログラムを実行する。「①タイムラインの表示」ボタンを押下すると、「SELECT * FROM Tweets」というSQLが発行され、自分（ログイン中のユーザー）と自分がフォローしているユーザーのツイートが一覧で表示される。【図20】

「②フォローしているユーザーを表示」「③自分がいいねを押した投稿を表示」「④フォロワーの一覧を表示」の各ボタン押下時の結果についても、【図20】で示したように参照できるので、ぜひ一度試していただきたい。

さて、ここで使用したテーブル「Tweets」は、タイムラインを表示するだけのテーブルではない。キーワードを指定して検索する機能も持っている。

【ソース5】のようにロジックを記述する。WHERE句にフィールド「Search Terms」を指定することで、指定されたキーワードを含む直近の6～9日以内のツイートを検索できる。【ソース5】のロジックでは、影響度が高いツイートを絞り込むため、「リツイート数が一定以上」「いいね数が一定以上」「日本語のツイートに限定」という条件も付けられるようにしている。

【図21】の明細は「Enterprise Connectors」というキーワードで、Twitter検索を行った結果である。

検索結果は、レコードの作成日時が新しい順（ツイートが新しい順、最近フォローされた順など）や、Twitterが独自に判断した影響度順で表示されており、ORDER BY句やインデックスの指定はできない。Twitter APIには、一定時間あたりの通信回数に上限が存在するからである。一定回数を超えて通信を行うと、【図22】のようなエラーが表示され、しばらく接続不能になる。

たとえばORDER BY句をSQLで指定すること自体は可能だが、そのSQLを実行すると、世界中からリアルタイムで流れ込む大量のツイートを取得・選別し続け、結果を表示するより先に、数秒～数十秒のうちに通信回数の上限に到達してエラーになる。

データを並べ替えたい場合は、TStringListやTClientDataSetなどのFireDAC接続から切り離されたローカルキャッシュ内に一度データを格納してから行うとよい。

なお、FireDACのデフォルト設定ではデータを1回の通信で50件ずつ取得するため、キーワードに対する検索結果が多い場合は、数分前までの新しいツ

図5　CSV操作PGM①（画面設計）

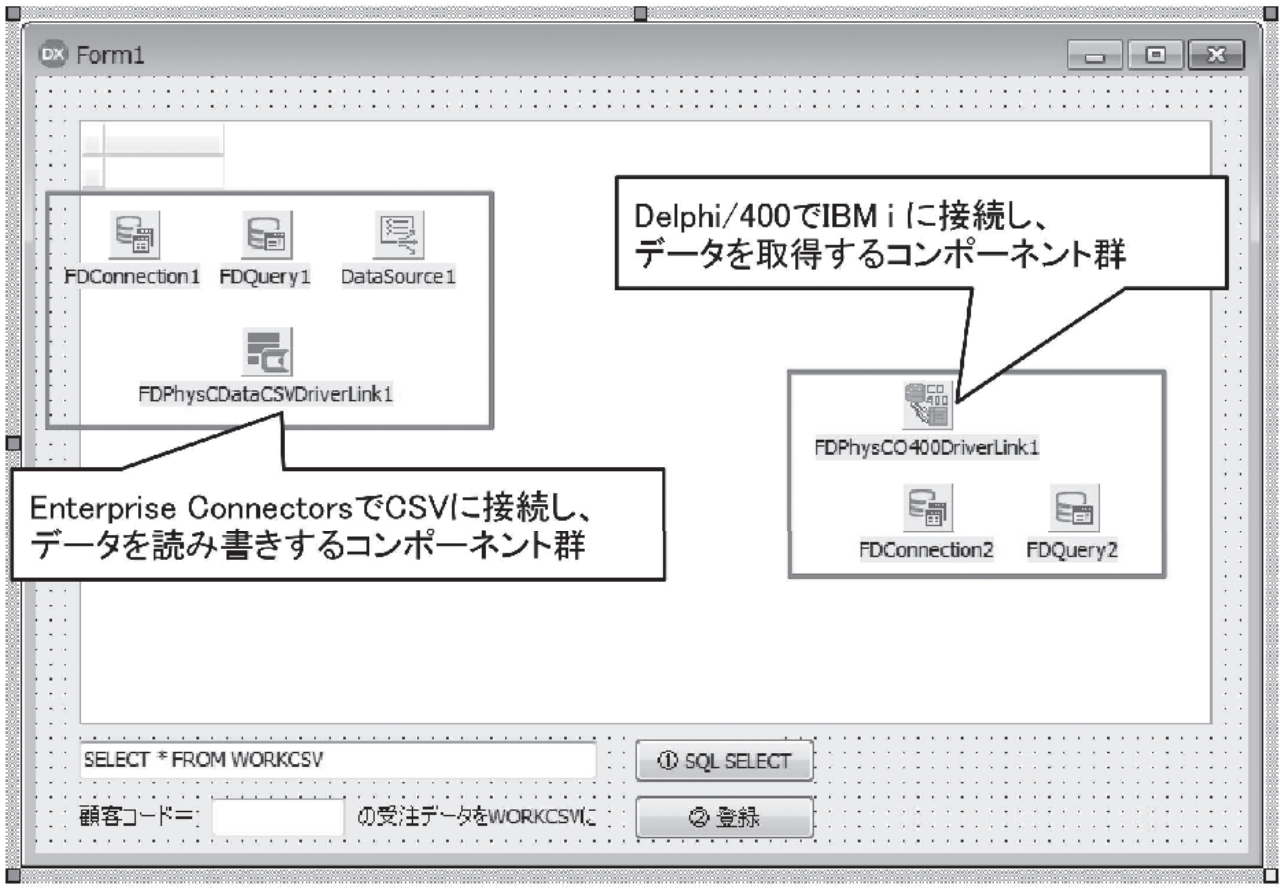
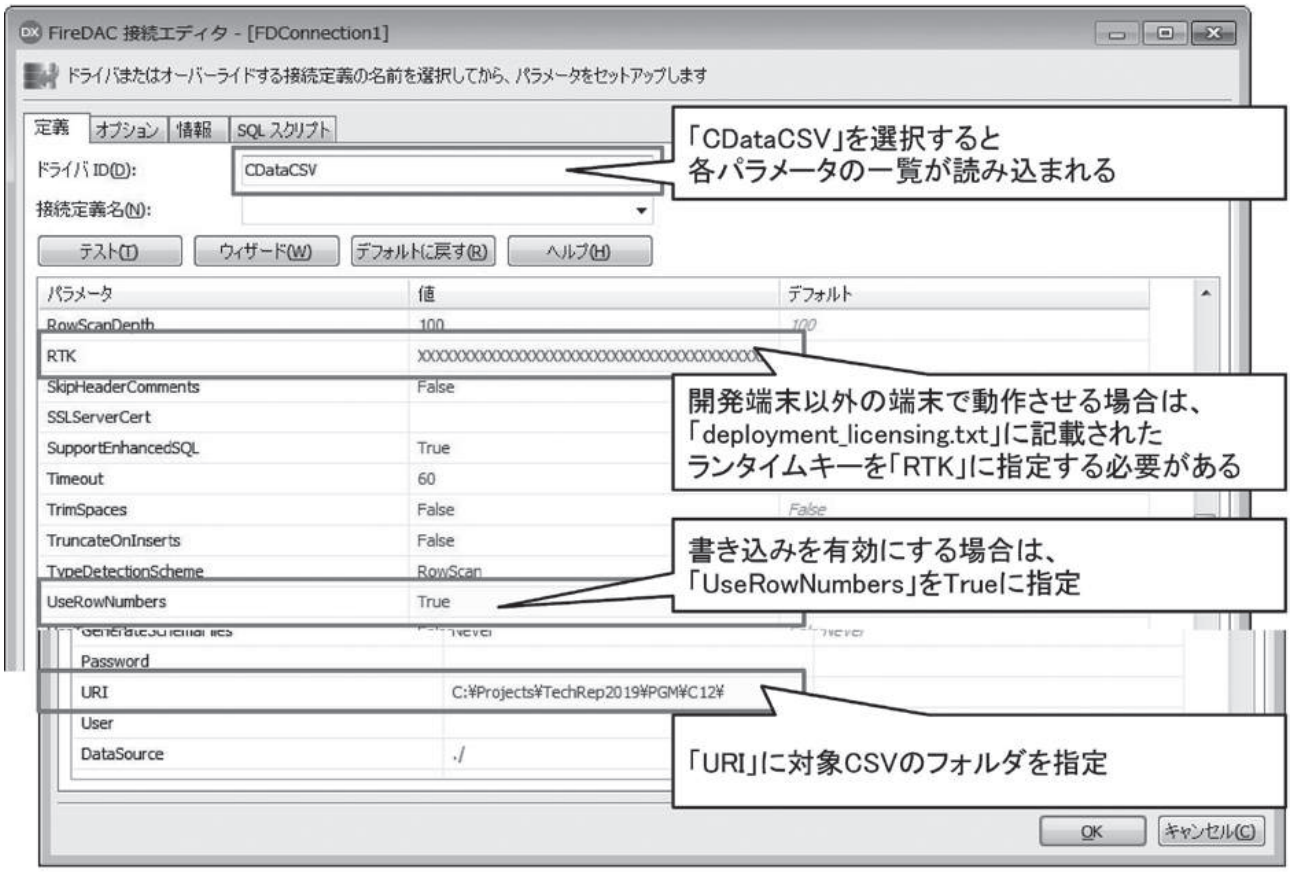


図6　CSV操作PGM②（接続パラメータ設定）



イートしか表示されないこともある。

このような場合には、【ソース 5】で指定したように、TFDQuery の FetchOptions.RowsetSize の設定値を増やすことで、1 回の通信あたりの取得レコード数が増え、より多くの取得結果が得られる。

ここまで Twitter API を利用したデータ送信およびデータ参照の手順を紹介したが、CData Twitter FireDAC Components では、目的に合わせたデータを解析できるよう、さまざまなテーブルやビューが準備されている。代表的なものを、【図 23】に記す。

5-2. Googleスプレッドシートへのデータ出力

次に、「CData Google Sheets FireDAC Components」を使用して、IBM i の売上データを「Google スプレッドシート」に出力する例を紹介する。また、出力した Google ドライブ上のスプレッドシートを、ローカルファイルとしてダウンロードする方法もあわせて紹介する。

これまで IBM i のデータを出力する場合、ローカル PC 上に CSV ファイルや Excel ファイルで出力することが多かったが、クラウド上の Google スプレッドシートにデータが出力できれば、ファイルの共有や同時編集など利便性が向上する。

Delphi/400 10.2 Tokyo を起動したら、アプリケーションを新規作成し、新しいフォームに、【図 24】のようにコンポーネントを配置する。

前章と同様、Delphi/400 で IBM i に接続してデータを取得するコンポーネント群と、Enterprise Connectors で Google スプレッドシートに接続するコンポーネント群をそれぞれ配置する。

なお Google スプレッドシートで使用するコネクターは、「TFDPhysCDataGoogleSheetsDriverLink」である。

前章と同じように、TFDConnection をダブルクリックして FireDAC 接続エディタを開き、ドライバ ID に「CDataGoogleSheets」を選択すると、【図 25】のように接続時の各種パラメータが自動でセットされる。本稿では、現在使用中の PC でログインしている自身の Google アカウントでの接続を前提とするため、【図 25】に記載の設定のみ行えばよい。

FireDAC 接続エディタの画面で、設定変更後に「テスト」ボタンを押すと、ブラウザが起動して【図 26】のような認証画面が表示されるので、このタイミングで Google アカウントへの権限を許可しておく（認証が必要なのは初回接続時のみ）。

この認証により、各処理はログイン中の自身の Google アカウントとして行われる。したがって、完成したアプリケーションを別の Google アカウントでログインした別の PC から起動した場合、初回接続時に同じようにブラウザが立ち上がり、【図 26】の認証画面が表示される。

本稿のサンプルでは、前章で CSV にデータを出力した際と同じデータを使用して、【ソース 6】のようにロジックを記述する。

スプレッドシートの新規作成では、SQL ではなくストアドプロシージャを発行している。Enterprise Connectors では、データの読み書きだけではなく、ファイルの作成や削除、ダウンロードなどの処理が行えるようサブメソッドがコネクターごとに用意されており、使用できる。

TFDQuery でもストアドプロシージャの発行は可能だが、【ソース 6】のように TFDStoredProc で発行するのがシンプルで効率がよい。Google スプレッドシートの操作には、今回使用したスプレッドシートの新規作成以外にも、シートの追加・コピー・削除や書式設定など、ストアドプロシージャで実現できるメソッドが数多く存在するので、ぜひヘルプドキュメントで詳細を参照いただきたい。

また本稿のプログラムでは、出力先スプレッドシートの「A」～「F」という列番号がそのままフィールド ID となっており、SQL で INSERT 文を発行する際にもそのように記述する。

完成したプログラムを実行して、「①出力」ボタンを押すと、ログイン中の Google アカウントが保持する Google スプレッドシートの一覧にファイルが新規作成され、その中に IBM i から取得したデータがセットされる。【図 27】【図 28】

また、【ソース 7】のようにダウンロードボタンのロジックを記述することで、このスプレッドシートをプログラム内から直接（ブラウザを起動せずに）、xlsx や PDF といった形式でダウンロードできる。【図 29】【図 30】

ここでも、ダウンロードの実行にはストアドプロシージャを使用している。

5-3. Google Driveでのデータ検索

最後に、「CData Google Drive FireDAC Components」を使用した、Google Drive のファイル検索について紹介する。

Delphi/400 10.2 Tokyo を起動したら、アプリケーションを新規作成し、新しいフォームに【図 31】のようにコンポーネントを配置する。

このプログラムでは IBM i と接続しないため、Enterprise Connectors で Google Drive に接続するコンポーネント群のみをそれぞれ配置する。なお Google Drive で使用するコネクターは、「TFDPhysCDataGoogleDriveDriverLink」である。

前章までと同じように、TFDConnection をダブルクリックして FireDAC 接続エディタを開き、ドライバ ID に「CDataGoogleDrive」を選択すると、【図 32】のように接続時の各種パラメータが自動でセットされる。

テスト接続時では、前章の Google スプレッドシートと同様に、【図 26】のような Google アカウントの認証画面が初回のみ表示されるので、権限を許可する。

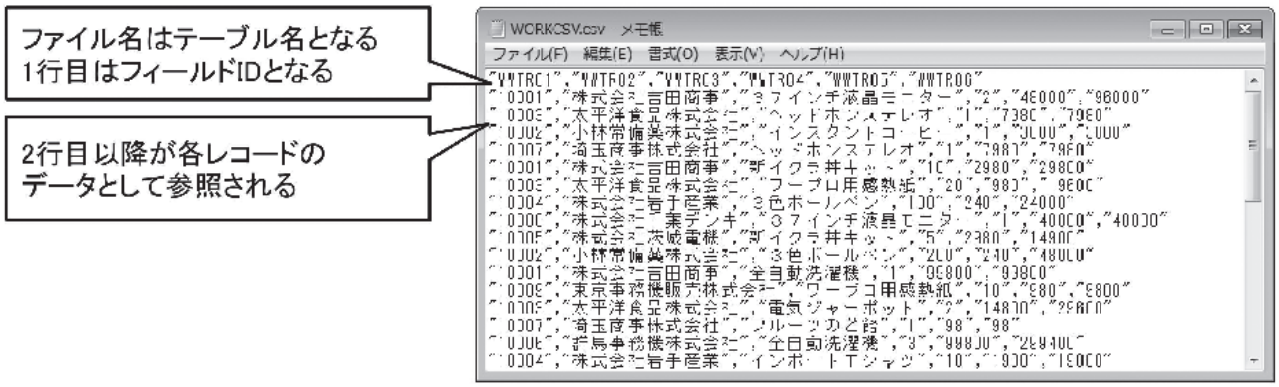
次に、【ソース 8】のようにロジックを記述する。

Google Drive の検索機能では、通常のファイル名やファイルの作成日時などによる絞り込みだけでなく、ファイルの内容を含むフルテキスト検索にも対応している。

たとえば前項で作成した Google スプレッドシートで、顧客名に「株式会社吉田商事」という値をセットしたセルが存在するが、Google Drive 側で検索条件に「吉田商事」と入力することで、セルの内容に「吉田商事」が含まれるスプレッドシートを検索結果として表示させられる。【図 33】

これは Google スプレッドシートのファイル保存先が Google Drive になっ

図7　CSV操作PGM③（CSVファイル設計）



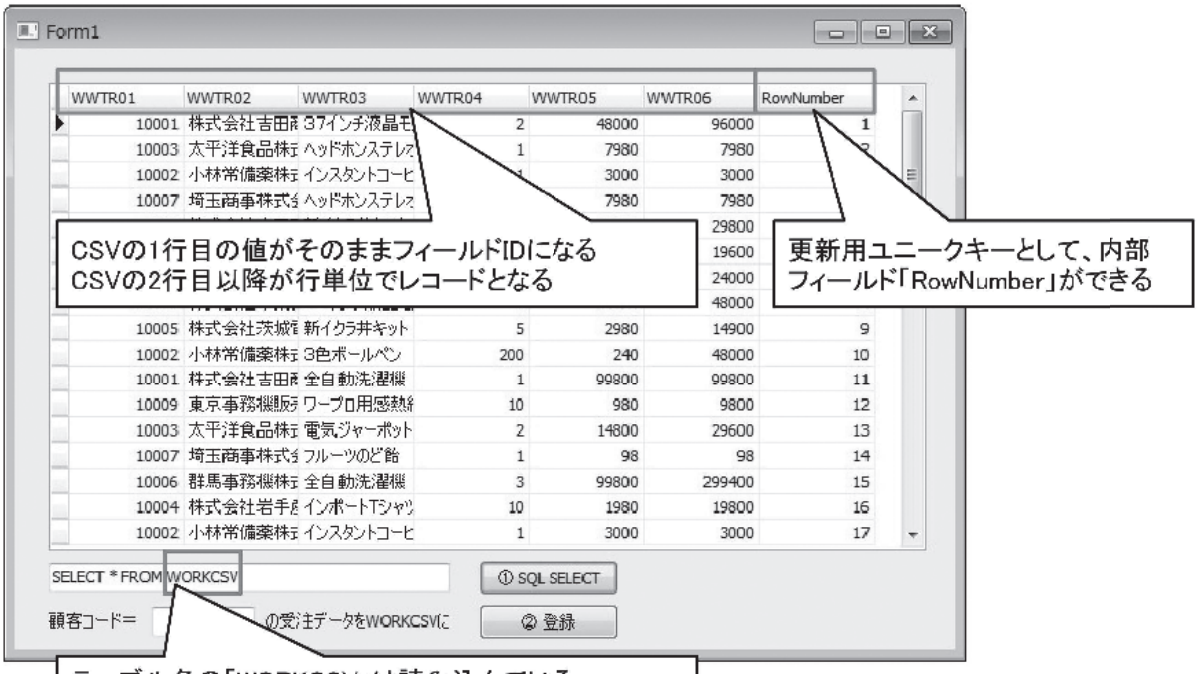
ソース1

FireDAC接続したCSVからSELECTでデータ抽出

```
{*******************************************************************************
 目的：①SQL SELECTボタン 押下時処理
*******************************************************************************}
procedure TForm1.Button2Click(Sender: TObject);
var
  i: Integer;
Begin

  FDQuery1.Close;
  FDQuery1.SQL.Text := Edit1.Text;   // SELECT * FROM WORKCSV
  FDQuery1.Open;

  // 列幅の調整
  for i := 0 to (DBGrid1.Columns.Count - 1) do
  begin
    DBGrid1.Columns.Items[i].Width := 80;
  end;
end;
```

この3行だけでSELECTの
SQLを発行し、DBと同じように
CSVの内容を参照できる

図8　CSV操作PGM④（SELECTで内容表示）



ているためで、同じGoogleアカウントであれば、Google Drive上からもスプレッドシートの内容を参照できる。

前章で、スプレッドシートを新規作成した際に採番されたファイルのIDもGoogle Driveと共通なので、そのIDを含んだGoogle DriveのファイルURLを指定して実行することで、Googleスプレッドシートで対象のファイルが起動する。【ソース9】【図34】

Google Driveではこれ以外にも、各種SQLやストアドプロシージャを使用することで、ファイルのアップロード、ダウンロードやファイル名の変更・削除なども行える。これらの方法もそれぞれヘルプドキュメントに記載されているので、ぜひ一度挑戦していただきたい。

6. まとめ

本稿では、Enterprise Connectorsを使用したクラウドサービスとの連携テクニックについて紹介した。

今回はTwitter、Googleスプレッドシート、Google Driveという一般によく使用されるクラウドサービスを題材に連携方法を紹介したが、Enterprise Connectorsでは他にもSalesforceやkintoneといったエンタープライズ向けクラウドサービスや、MongoDB、Amazon DynamoDBといったようなNoSQLデータベースなど、これからの業務システムに多用する多彩なサービスへのアクセスを可能にしている。

本稿を参考に、さまざまなクラウドサービスとの連携を検討いただきたい。なお、全コネクターが利用可能なEnterprise Connectors Plusの購入をご検討の場合は、弊社営業までお気軽にお問い合わせいただければ幸いである。

M

ソース2

FireDAC接続したCSVにレコードを追加

```
{*****************************************************************************
 目的: ②登録ボタン 押下時処理
*****************************************************************************}
procedure TForm1.Button1Click(Sender: TObject);
var
  iTR01, iTR04, iTR05, iTR06: Integer; // 顧客コード、数量、単価、金額
  sTR02, sTR03: String;                // 顧客名、商品名
begin

  // Delphi/400側のQueryで対象データを抽出
  FDQuery2.Close;
  FDQuery2.SQL.Clear;
  FDQuery2.SQL.Add(' SELECT VFORDP.*, CSCSNM FROM VFORDP ');
  FDQuery2.SQL.Add(' LEFT JOIN VMCSTP ON (CSCSCD = ORCSCD) ');
  FDQuery2.SQL.Add(' WHERE VFORDP.ORCSCD = ' + Edit2.Text); // 顧客コード
  FDQuery2.Open;

  try
    // Enterprise Connectors側のQueryでワークファイルCSVに更新処理
    FDQuery1.SQL.Clear;
    FDQuery1.SQL.Add(' INSERT INTO WORKCSV ( ');
    FDQuery1.SQL.Add(' WWTR01, WWTR02, WWTR03, WWTR04, WWTR05, WWTR06 ');
    FDQuery1.SQL.Add(' ) VALUES ( ');
    FDQuery1.SQL.Add(' :TR01, :TR02, :TR03, :TR04, :TR05, :TR06 ');
    FDQuery1.SQL.Add(' ) ');

    while not FDQuery2.Eof do
    begin
      iTR01 := FDQuery2.FieldByName('ORCSCD').AsInteger;  // 顧客コード
      sTR02 := FDQuery2.FieldByName('CSCSNM').AsString;   // 顧客名
      sTR03 := FDQuery2.FieldByName('ORPRNM').AsString;   // 商品名
      iTR04 := FDQuery2.FieldByName('ORQTY').AsInteger;   // 数量
      iTR05 := FDQuery2.FieldByName('ORUNPR').AsInteger;  // 単価
      iTR06 := iTR04 * iTR05;                             // 金額

      FDQuery1.Close;
      FDQuery1.ParamByName('TR01').AsInteger    := iTR01; // 顧客コード
      FDQuery1.ParamByName('TR02').AsWideString := sTR02; // 顧客名
      FDQuery1.ParamByName('TR03').AsWideString := sTR03; // 商品名
      FDQuery1.ParamByName('TR04').AsInteger    := iTR04; // 数量
      FDQuery1.ParamByName('TR05').AsInteger    := iTR05; // 単価
      FDQuery1.ParamByName('TR06').AsInteger    := iTR06; // 金額
      FDQuery1.ExecSQL; // 更新実行

      FDQuery2.Next;    // Delphi/400側のQueryを次行へ
    end;

  finally
    FDQuery1.Close;
    FDQuery2.Close;
  end;
end;
```

IBM i からデータを抽出

CSVにデータを
登録するためのSQL文作成

CSVにデータを書き込み

図9　CSV操作PGM⑤（CSVへのINSERT）

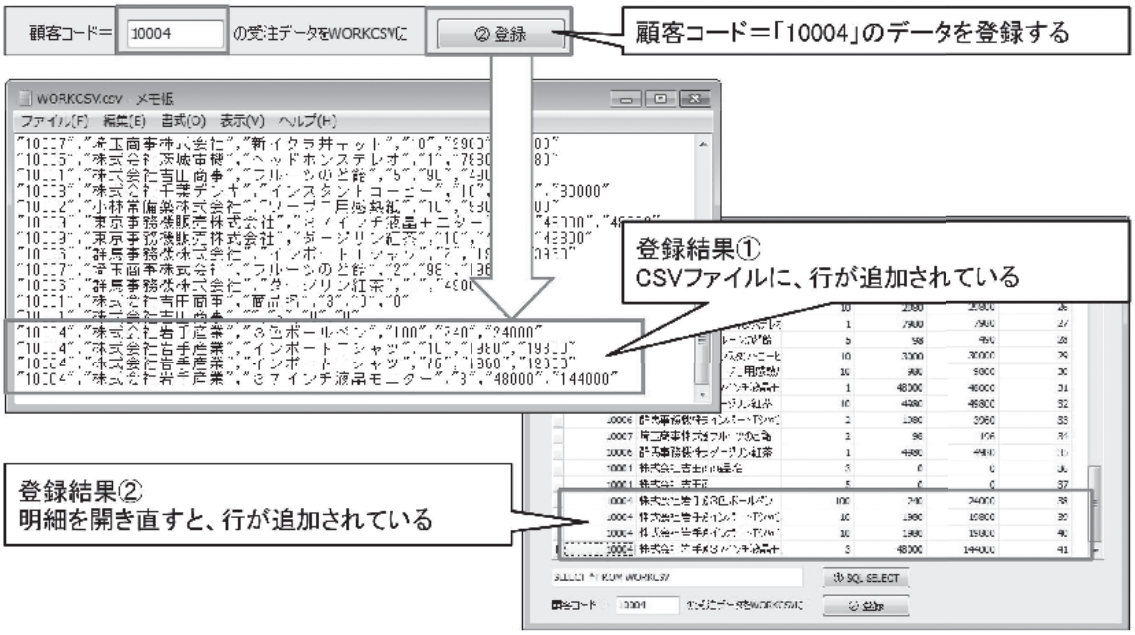


図10　Twitter Web APIの利用開始①

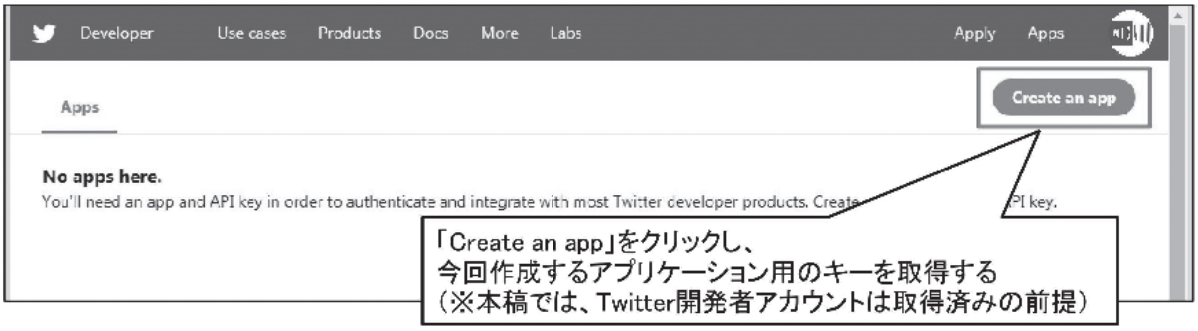


図11　Twitter Web APIの利用開始②

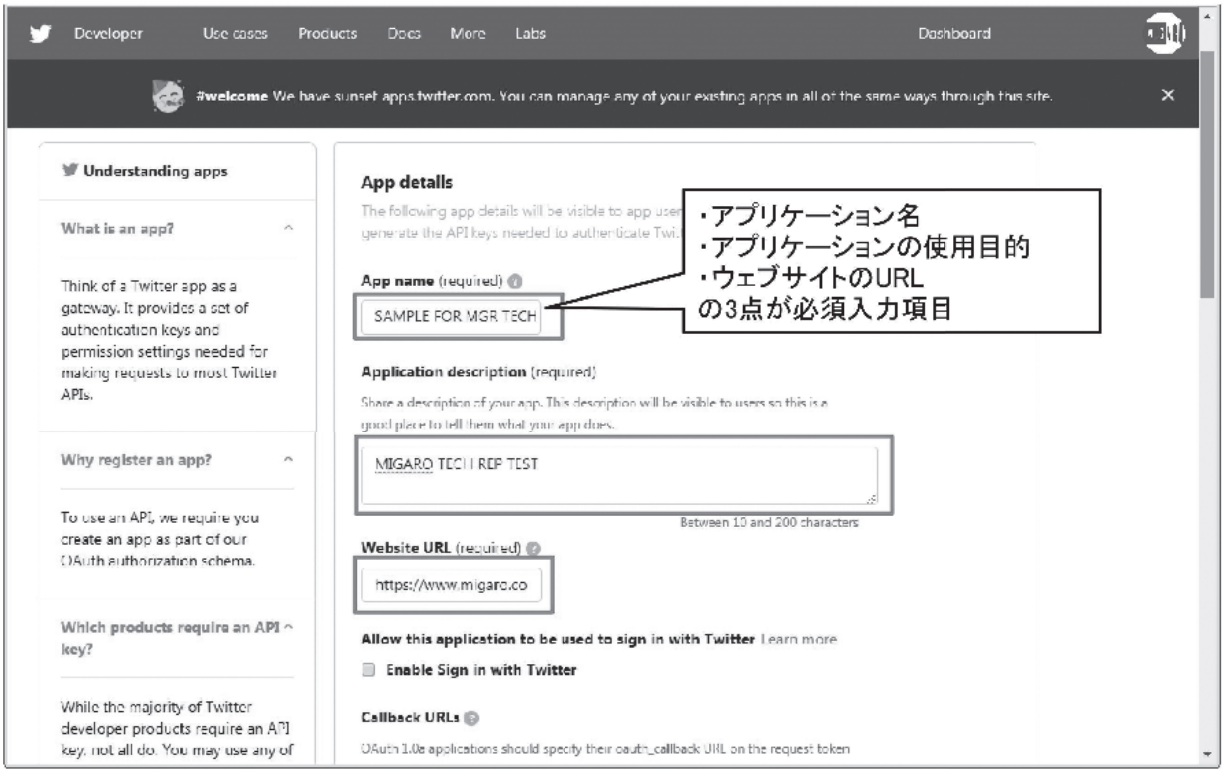


図12　Twitter Web APIの利用開始③

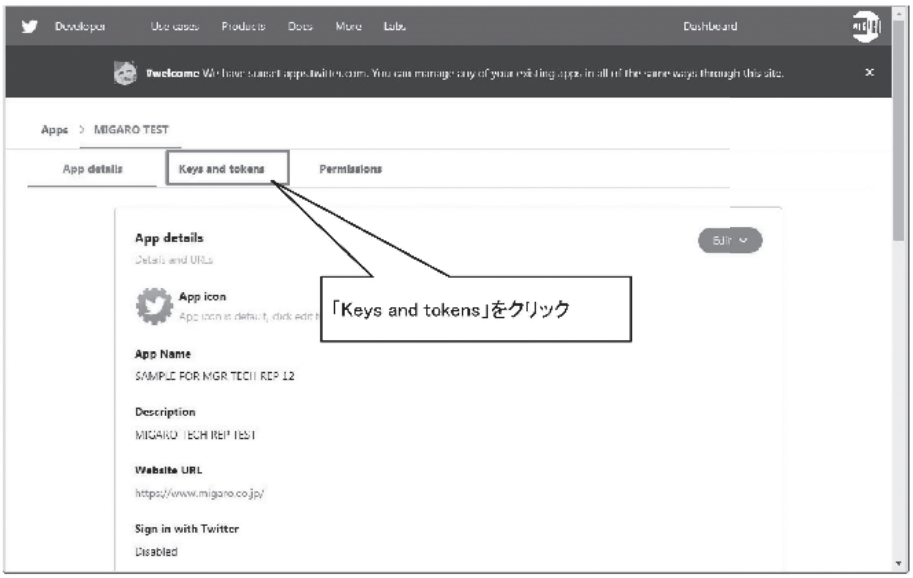


図13　Twitter Web APIの利用開始④

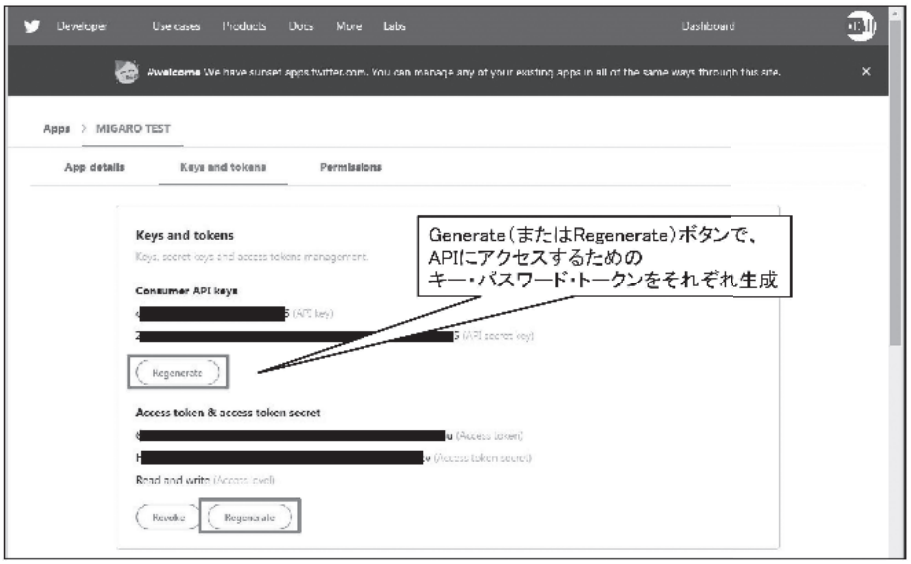


図14　Twitterのデータ送信①（画面設計）

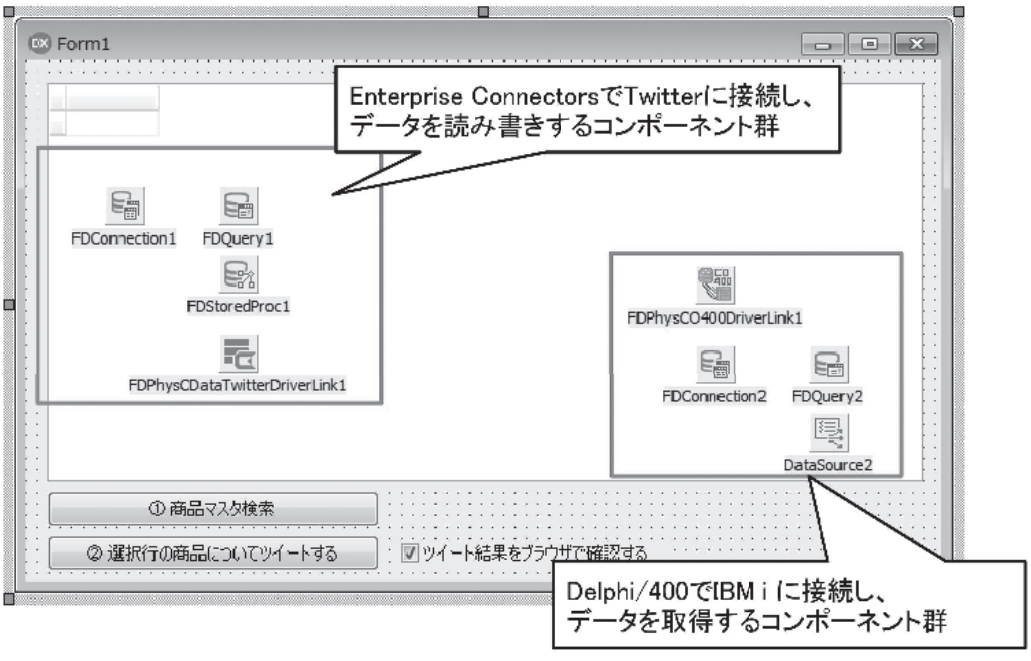


図15　Twitterのデータ送信②（接続パラメータ設定）

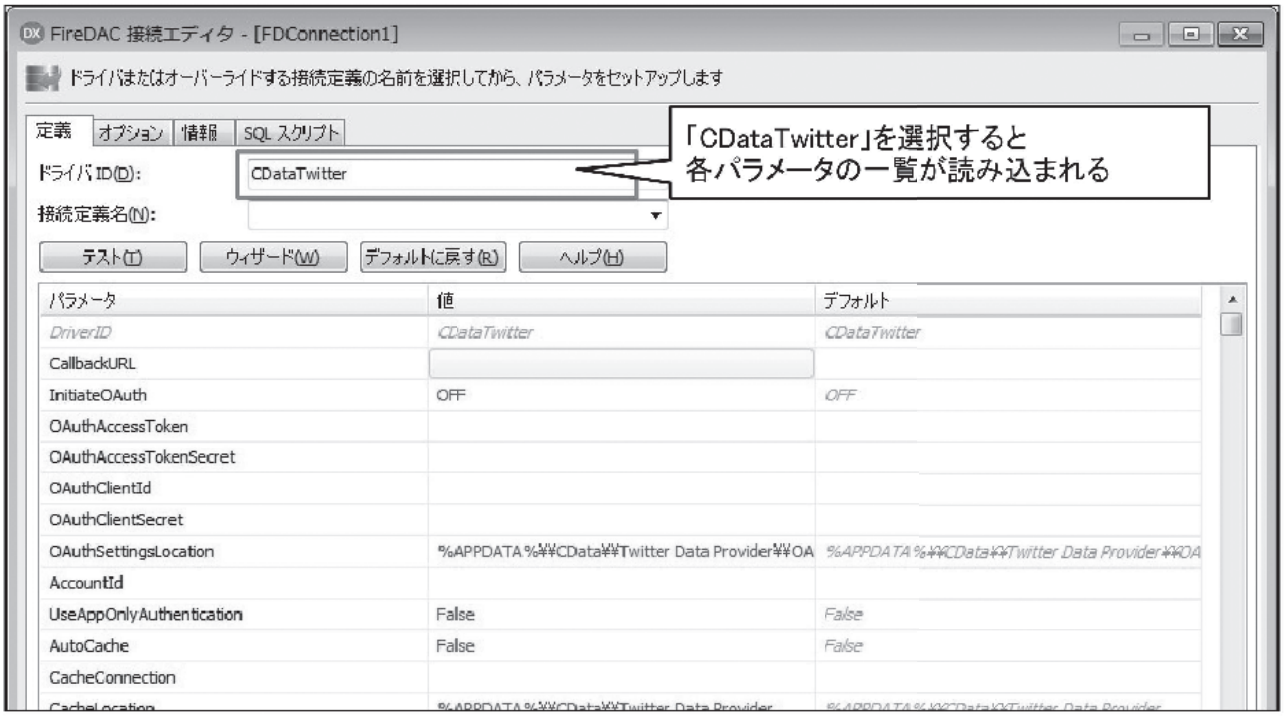


図16　Twitterのデータ送信③（接続パラメータ設定）

＜目的別 TFDConnectionのパラメータ設定値＞　　※「－－－」はデフォルト値のまま

接続目的 / パラメータ名	書き込み／削除を行う		照会のみ（書き込み／削除不要）
	開発者自身のアカウントでのみ書き込み／削除を行う	あらゆるアカウントでログインして書き込み／削除を行う	
OAuthClientId	APP作成時に取得した「API key」		
OAuthClientSecret	APP作成時に取得した「API secret key」		
RTK	開発端末以外の端末で動作させる場合は、以下の場所に記載されたランタイムキーを指定 C:¥Program Files¥CData¥CData FireDAC Components for Twitter¥deployment_licensing.txt		
UseAppOnlyAuthentication	－－－	－－－	True
CallbackURL	－－－	APP作成時に指定する「Callback URL」	－－－
InitiateOAuth	－－－	GETANDREFRESH	－－－
OAuthAccessToken	APP作成時に取得可能な「Access token」	接続時に都度関数で取得（詳細はヘルプドキュメントを参照）	－－－
OAuthAccessTokenSecret	APP作成時に取得可能な「Access token secret」		－－－
結果	開発者自身のアカウントでログインし、読み書きする	ブラウザが起動してAPIの認証画面が表示され、そこでログインしたアカウントで読み書きする	照会専用アカウントでログイン（ユーザー固有の情報は読み取れない）

ソース3

Twitter APIを利用したツイートの作成

```
{*****************************************************************************
 目的：②選択行の商品についてツイートする
*****************************************************************************}
procedure TForm1.Button2Click(Sender: TObject);
var
  sTEXT: String;
begin

  // 未検索時は何もしない
  if (not FDQuery2.Active) or (FDQuery2.Bof and FDQuery2.Eof) then
    Exit;

  // 本文の作成
  sTEXT := '今週のおすすめ商品！！『' +
           FDQuery2.FieldByName('SYHNMM').AsString +
           '』　単価：¥' +
           FormatFloat('#,0', FDQuery2.FieldByName('TANKA').AsInteger) +
           '　詳細はぜひ、ホームページまでアクセスしてください。 '+
           'http://technical-report.jp  #Delphi #Technical_Report';

  // つぶやく
  FDQuery1.SQL.Text := 'INSERT INTO Tweets (Text) VALUES (:TEXT)';
  FDQuery1.ParamByName('TEXT').AsMemo   := sTEXT;    // 本文
  FDQuery1.ExecSQL;

  // 自身の最新ツイートを取得してリンクを開く
  if CheckBox1.Checked then
  begin
    GetMyNewest;
  end;
end;

{*****************************************************************************
 目的：自分の最新ツイートをブラウザで開く
*****************************************************************************}
procedure TForm1.GetMyNewest;
var
  sMe, sID, sURL: String;
begin

  FDQuery1.Close;
  FDQuery1.SQL.Clear;
  FDQuery1.SQL.Add(' SELECT A.ID AS TWID, A.Text, A.From_User_Screen_Name AS MYNAME ');
  FDQuery1.SQL.Add(' from Tweets A ');
  FDQuery1.SQL.Add(' inner join AccountSettings B ');  // 自分のアカウント設定ビューをJOINする
  FDQuery1.SQL.Add(' on (B.Screen_Name = A.From_User_Screen_Name) ');  // 送信者＝自分に絞る
  try
    // 取得結果は常に新しい順なので、ORDERは不要(通信回数制限の原因になる)
    FDQuery1.Open;
    if not(FDQuery1.Bof and FDQuery1.Eof) then
    begin
      sID  := FDQuery1.FieldByName('TWID').AsString;
      sMe  := FDQuery1.FieldByName('MYNAME').AsString;
      sURL := 'https://twitter.com/' + sMe + '/status/' + sID;

      // ShellExecuteによって、通常使うブラウザでURLを開く
      // (※uses節に Winapi.ShellAPI が必要)
      // (※ツイートを直接表示するには、ブラウザ側がログイン状態になっている必要あり)
      ShellExecute(Application.Handle, 'OPEN', PChar(sURL), PChar(''), '', SW_SHOW);
    end;
  finally
    FDQuery1.Close;
  end;
end;
```

※①商品マスタ検索ボタンの処理は通常のDelphi/400の検索処理と同様のロジックのため、省略

本文の文字列に指定した内容が送信される

ハッシュタグやURLは本文中にそのまま記述する（複数ハッシュタグ使用時はスペースを空ける）

この行をエラー無く抜けたら、送信完了

今送信したツイートのIDと自分のアカウントIDを取得し、リンクさせるURLを生成する

図17　Twitterのデータ送信④（商品マスタ検索）

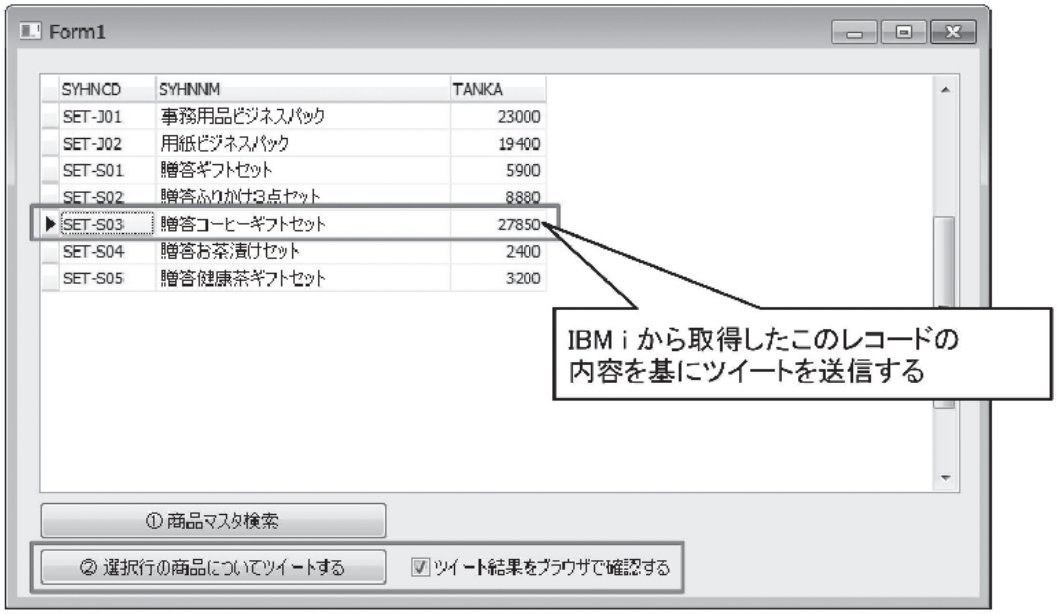


図18　Twitterのデータ送信⑤（ツイート送信結果）

図19　Twitterのデータ参照①（画面設計）

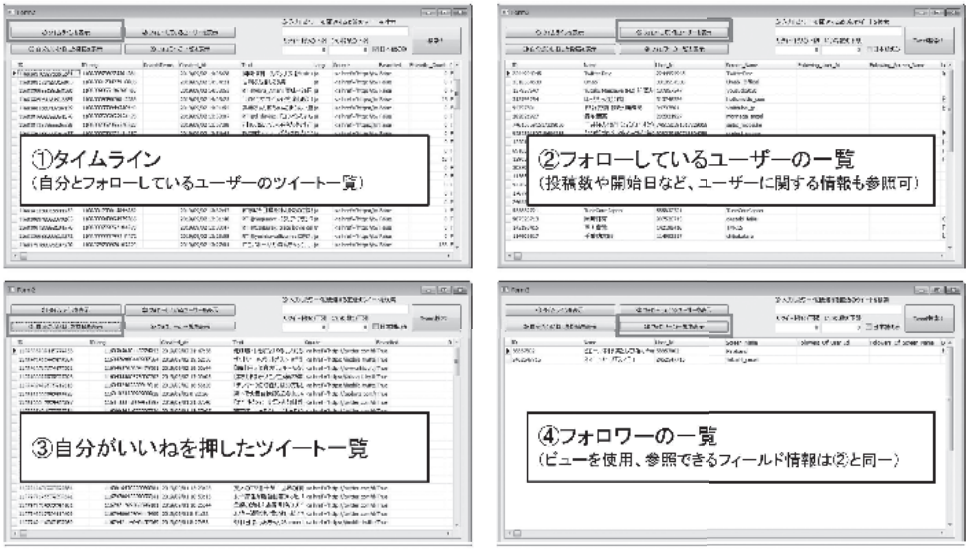
ソース4

Twitter APIを利用したツイートやアカウント情報の取得

```
{**************************************************************************
 目的：①タイムラインを表示
**************************************************************************}
procedure TForm2.Button1Click(Sender: TObject);
begin
  // 自分および自分がフォローしているユーザーのツイートを一覧表示
  FDQuery1.Close;
  FDQuery1.SQL.Text := 'SELECT * FROM Tweets';
  FDQuery1.Open; // SQL実行
  SetColWidths;  // DBGridの列幅設定
end;

{**************************************************************************
 目的：②フォローしているユーザーを表示
**************************************************************************}
procedure TForm2.Button2Click(Sender: TObject);
begin
  // 自分がフォローしているユーザーの一覧表示
  // (※MaxUserLookup=最大取得人数、初期値はわずか100人なので拡張しておく)
  FDQuery1.Close;
  FDQuery1.SQL.Text := 'SELECT * FROM Following Where MaxUserLookup = 20000';
  FDQuery1.Open; // SQL実行
  SetColWidths;  // DBGridの列幅設定
end;

{**************************************************************************
 目的：③自分がいいねした投稿を表示
**************************************************************************}
procedure TForm2.Button3Click(Sender: TObject);
begin
  // 自分が「いいね」を送ったツイートを一覧表示
  FDQuery1.Close;
  FDQuery1.SQL.Text := 'SELECT * FROM Favorites';
  FDQuery1.Open; // SQL実行
  SetColWidths;  // DBGridの列幅設定
end;
```

「④フォロワーの一覧を表示」ボタンの処理は、
②の「Following」部分をビュー「Followers」に変更する
だけで、他の部分は同じとなるため省略

図20　Twitterのデータ参照②（参照結果）



※ 照会専用アカウントでログイン時は、③は参照不可。①②④についても対象ユーザーの指定が別途必要。

図21　Twitterのデータ参照③（参照結果）

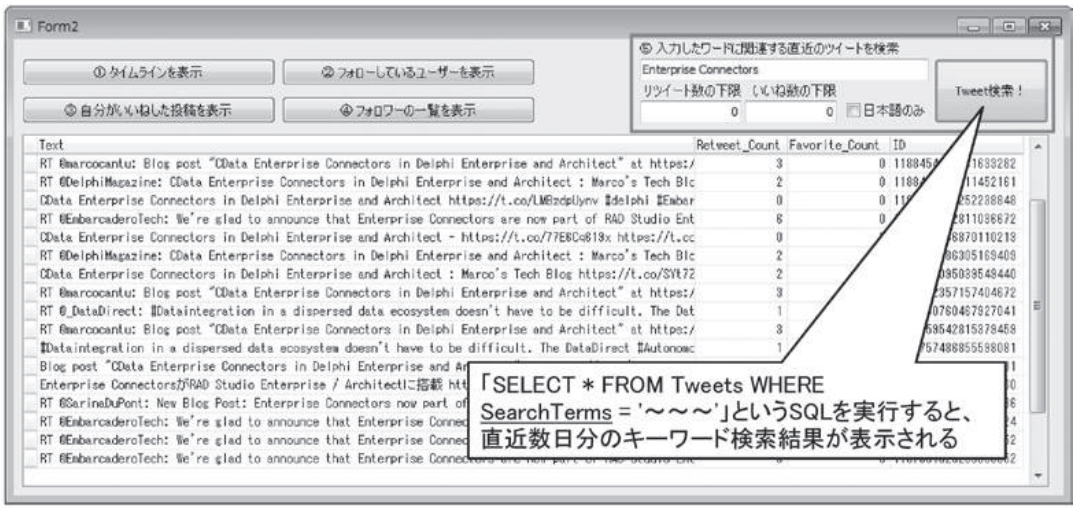
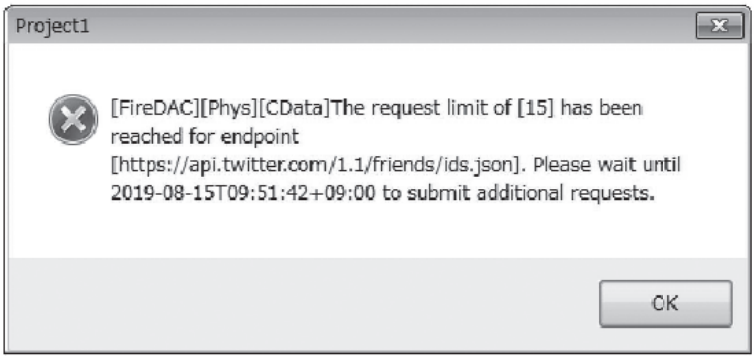


図22　Twitterのデータ参照④（通信回数上限について）



通信回数が規定数を超過すると、エラーでしばらく（最大15分）接続できなくなる

ソース5

Twitter APIを利用したツイートのキーワード検索

```
[*************************************************************************
 目的：⑤入力したワードに関連する直近のツイートを検索
*************************************************************************]
procedure TForm2.Button5Click(Sender: TObject);
begin
  if (Trim(LabeledEdit1.Text) = '') then
    raise Exception.Create('検索ワードは必ず指定して下さい。');

  // 対象データを抽出
  FDQuery1.Close;
  FDQuery1.SQL.Clear;
  FDQuery1.SQL.Add(' SELECT Text, Retweet_Count, Favorite_Count, ID');
  FDQuery1.SQL.Add(' FROM Tweets ');

  // 検索ワード（※SearchTermsをWHERE句に入れることで【ソース4】①とは検索条件が変わる）
  FDQuery1.SQL.Add(' WHERE SearchTerms = ''' + Trim(LabeledEdit1.Text) + ''' ');

  if (StrToIntDef(LabeledEdit2.Text, 0) > 0) then // リツイート数の下限指定時
  begin
    FDQuery1.SQL.Add(' AND Retweet_Count >= ' + LabeledEdit2.Text);
  end;
  if (StrToIntDef(LabeledEdit3.Text, 0) > 0) then // いいね数の下限指定時
  begin
    FDQuery1.SQL.Add(' AND Favorite_Count >= ' + LabeledEdit3.Text);
  end;
  if (CheckBox1.Checked) then  // 日本語のツイートに限定する時
  begin
    FDQuery1.SQL.Add(' AND Lang = ''ja'' ');
  end;

  FDQuery1.FetchOptions.RowsetSize := 100; // １回の通信で取得する件数
  FDQuery1.Open; // SQL実行
  SetColWidths;  // DBGridの列幅設定
end;
```

ツイートのキーワード検索を行うためのSQLを作成

FireDACの初期設定では50件ずつデータを取得するため、1回の通信（50件のみ取得）で必要十分なデータが得られない場合は、増やす
（※増やし過ぎるとレスポンスが遅くなったり、通信回数制限の原因になるため、注意）

図23　Twitterのデータ参照⑤（テーブル・ビューの一覧）

＜代表的なテーブルの一覧＞

テーブル名	概要
Favorites	・自分が「いいね」を送ったツイートを一覧表示 ・「いいね」の追加および解除
Following	・自分がフォローしているユーザーの一覧表示およびフォロー解除
Tweets	・自分および自分がフォローしているユーザーのツイートを一覧表示 ・WHERE句にSearchTermsを指定することで、特定の検索ワードで検索 ・ツイートの作成と削除

＜代表的なビューの一覧＞

ビュー名	概要
AccountSettings	自分のアカウントに関する設定を取得
AdInsights	自分が発信した広告に対する視聴者などの情報を取得
Followers	自分または指定したユーザーをフォローしているユーザーのリストを取得
Mentions	自分が受けた最新のメンション（自分宛ての返信も含む）を取得
Retweets	自分のリツイートのリストを取得
Trends	日時や国や地域（WoeID）を指定してトレンドトピックのリスト（50件）を取得
TweetStream	Twitterを流れるパブリックデータ（全ての公開ツイート）を取得
Users	キーワードや名前を基にユーザーの一覧を取得

※ 自分 = ログインによって認証されたユーザーを指す。
※ 照会専用アカウントでログイン時はユーザーを指定すれば参照できるものと、参照不可のものがある。更新は不可。

図24　Googleスプレッドシート①（画面設計）

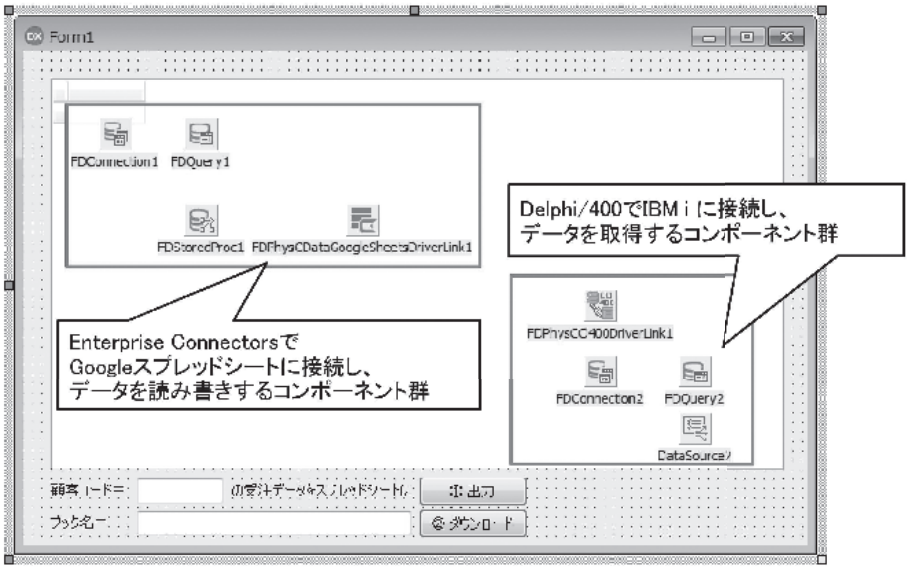


図25　Googleスプレッドシート②（接続パラメータ指定）

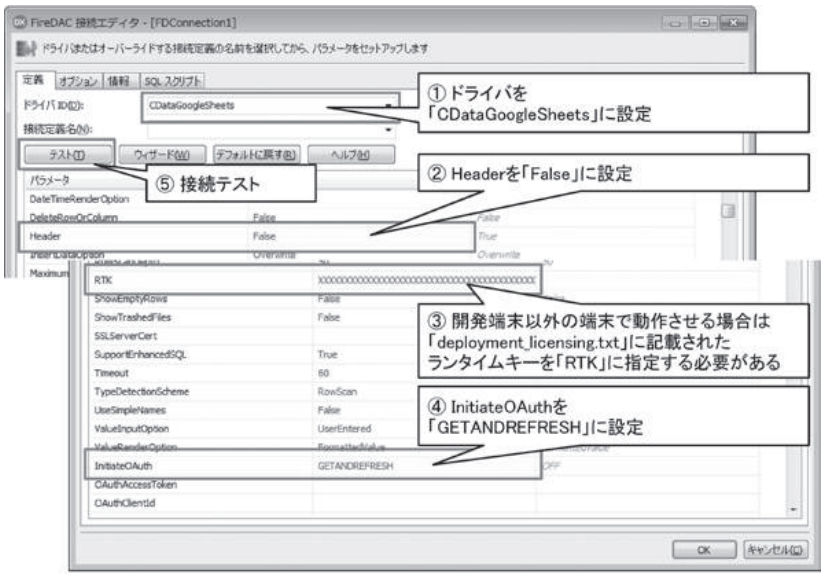


図26　Googleスプレッドシート③（初回接続時の認証）

現在ログイン中のGoogleアカウントで
APIを使用する権限を求められるため、許可する

ソース6

Google Sheets APIを使用したスプレッドシートへのデータ登録

```
{*******************************************************************************
 目的: ①出力ボタン 押下時処理
*******************************************************************************}
procedure TForm1.Button1Click(Sender: TObject);
var
  sXLSX: String;   // スプレッドシート名
  sSHTX: String;   // シート名
  iTR01, iTR04, iTR05, iTR06: Integer;  // 顧客コード、数量、単価、金額
  sTR02, sTR03: String;                 // 顧客名、商品名
begin

  // スプレッドシート名とシート名を定義（シート名は今回は初期値のままとする）
  sXLSX := 'レポート_' + FormatDateTime('YYYYMMDD_HHNNSS', Now);
  sSHTX := 'シート1';

  // 新しい空のスプレッドシートを作成
  FDStoredProc1.StoredProcName := 'CreateSpreadsheet';
  FDStoredProc1.Prepare;
  FDStoredProc1.ParamByName('Title').AsString := sXLSX; // スプレッドシート名
  FDStoredProc1.ExecProc;

  // 採番されたスプレッドシートのIDをグローバル変数に保持
  sXLSD := FDStoredProc1.ParamByName('Id').AsString;

  // Delphi/400側のQueryで対象データを抽出
  FDQuery2.Close;
  // (※この部分は【ソース2】と同一のため省略)
  FDQuery2.Open;

  // Enterprise Connectors接続
  if (not FDConnection1.Connected) then
  begin
    FDConnection1.Open;
  end;

  // Enterprise Connectors側のQueryでスプレッドシートに更新処理
  FDQuery1.Close;
  FDQuery1.SQL.Clear;
  FDQuery1.SQL.Add(' INSERT INTO ' + sXLSX + '_' + sSHTX + ' ( '); // テーブル名は「スプレッドシート名_シート名」形式
  FDQuery1.SQL.Add(' A, B, C, D, E, F ) VALUES ( ');              // 列IDの「A」～「F」がフィールドIDになる
  FDQuery1.SQL.Add(' :TR01, :TR02, :TR03, :TR04, :TR05, :TR06 ) ');

  // スプレッドシートの1行目にタイトル情報を登録
  FDQuery1.ParamByName('TR01').AsString := '顧客コード';
  FDQuery1.ParamByName('TR02').AsString := '顧客名';
  FDQuery1.ParamByName('TR03').AsString := '商品名';
  FDQuery1.ParamByName('TR04').AsString := '数量';
  FDQuery1.ParamByName('TR05').AsString := '単価';
  FDQuery1.ParamByName('TR06').AsString := '金額';
  FDQuery1.ExecSQL;   // INSERTのSQL実行

  // スプレッドシートの2行目以降に各レコード情報を登録
  while not FDQuery2.Eof do
  begin
    iTR01 := FDQuery2.FieldByName('ORCSCD').AsInteger;  // 顧客コード
    // (※この部分は【ソース2】と同一のため省略)
    iTR06 := iTR04 * iTR05;                             // 金額

    FDQuery1.ParamByName('TR01').AsString := IntToStr(iTR01); // 顧客コード
    FDQuery1.ParamByName('TR02').AsString := sTR02;           // 顧客名
    FDQuery1.ParamByName('TR03').AsString := sTR03;           // 商品名
    FDQuery1.ParamByName('TR04').AsString := IntToStr(iTR04); // 数量
    FDQuery1.ParamByName('TR05').AsString := IntToStr(iTR05); // 単価
    FDQuery1.ParamByName('TR06').AsString := FormatFloat('#,0', iTR06); // 金額
    FDQuery1.ExecSQL; // INSERTのSQL実行

    FDQuery2.Next;    // Delphi/400側のQueryを次行へ
  end;
end;
```

ストアドプロシージャを使用して
新しいスプレッドシートを作成

【ソース7】で使用するため、
ストアドプロシージャの戻り値として返る
スプレッドシートのIDを変数に保持する

スプレッドシートにデータを
登録するためのSQL文作成

1レコード目にタイトル情報の文字列を
セットすることで、結果のスプレッドシートも
1行目がタイトル行になる
（※接続時パラメータ「Header」＝Falseを指定）

1行目で全フィールドを文字型として定義した
ため、2行目以降も全て文字としてセット

図27　Googleスプレッドシート④（PGM実行時画面）

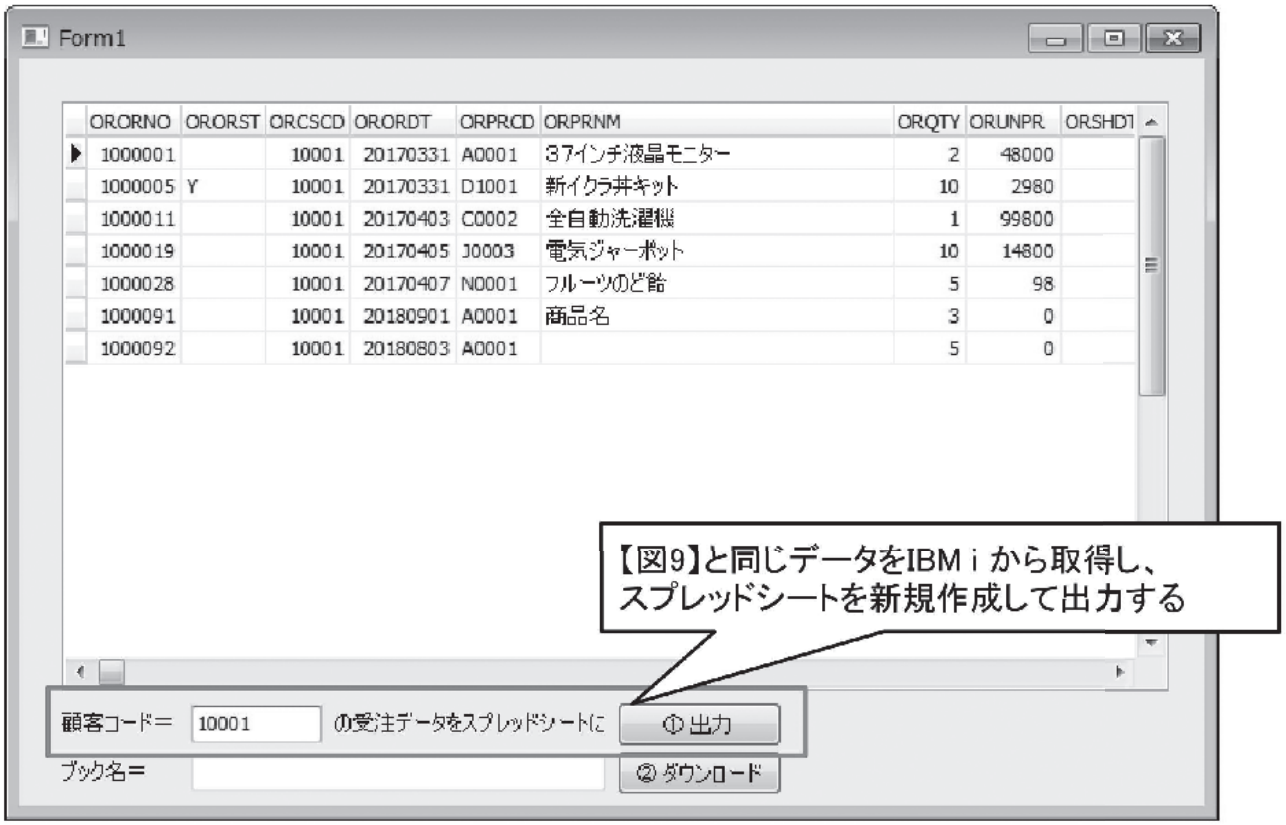
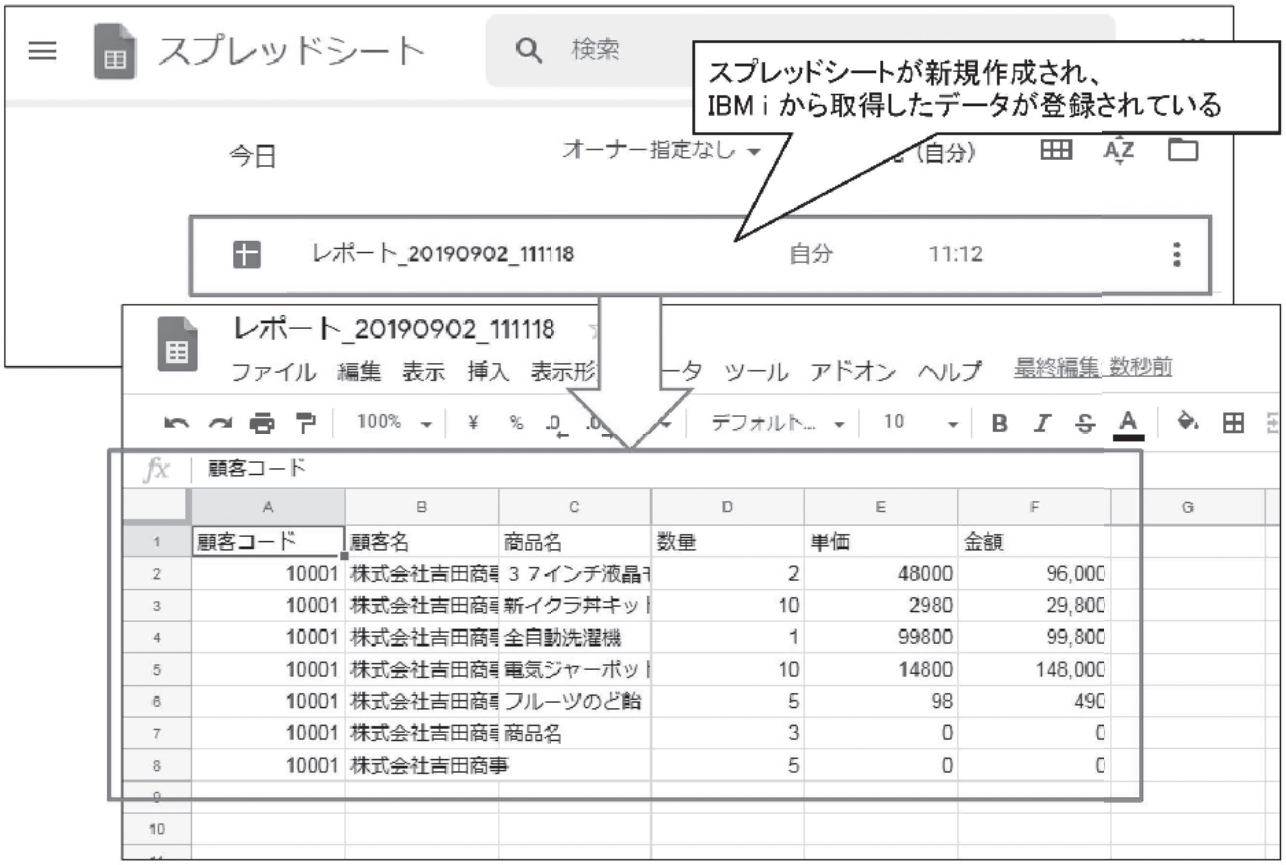


図28　Googleスプレッドシート⑤（出力結果）



ソース7

Google Sheets APIを使用したスプレッドシートのダウンロード

```
{*************************************************************************
 目的: ②ダウンロードボタン 押下時処理
*************************************************************************}
procedure TForm1.Button3Click(Sender: TObject);
var
  sID: String;
  sEXT: String;
begin

  // ダウンロード用に、ストアドプロシージャのパラメータを指定
  FDStoredProc1.StoredProcName := 'DownloadDocument';
  FDStoredProc1.Prepare;
  FDStoredProc1.ParamByName('Id').AsString := sXLSD;        // スプレッドシートのID
  FDStoredProc1.ParamByName('LocalFile').AsString := Edit3.Text; // ダウンロード先

  // ダウンロード先の拡張子によってファイル形式(MIME)を指定
  // (※xlsxの場合はデフォルト値なので指定不要)
  sEXT := UpperCase(ExtractFileExt(Edit3.Text));
  if (sEXT = '.PDF') then
  begin // 例: PDFの場合のMIME値
    FDStoredProc1.ParamByName('FileFormat').AsString := 'application/pdf';
  end;
  FDStoredProc1.ExecProc;     // ダウンロード実行
end;
```

【ソース6】で、スプレッドシートを
新規作成時に保持していた値

以下の4種類が指定可能

XLSX（初期値）：「application/vnd.openxmlformats-officedocument.spreadsheetml.sheet」

PDF：「application/pdf」

CSV（カンマ区切り）：「text/csv」

TSV（タブ区切り）：「text/tab-separated-values」

図29　Googleスプレッドシート⑥(ダウンロード)

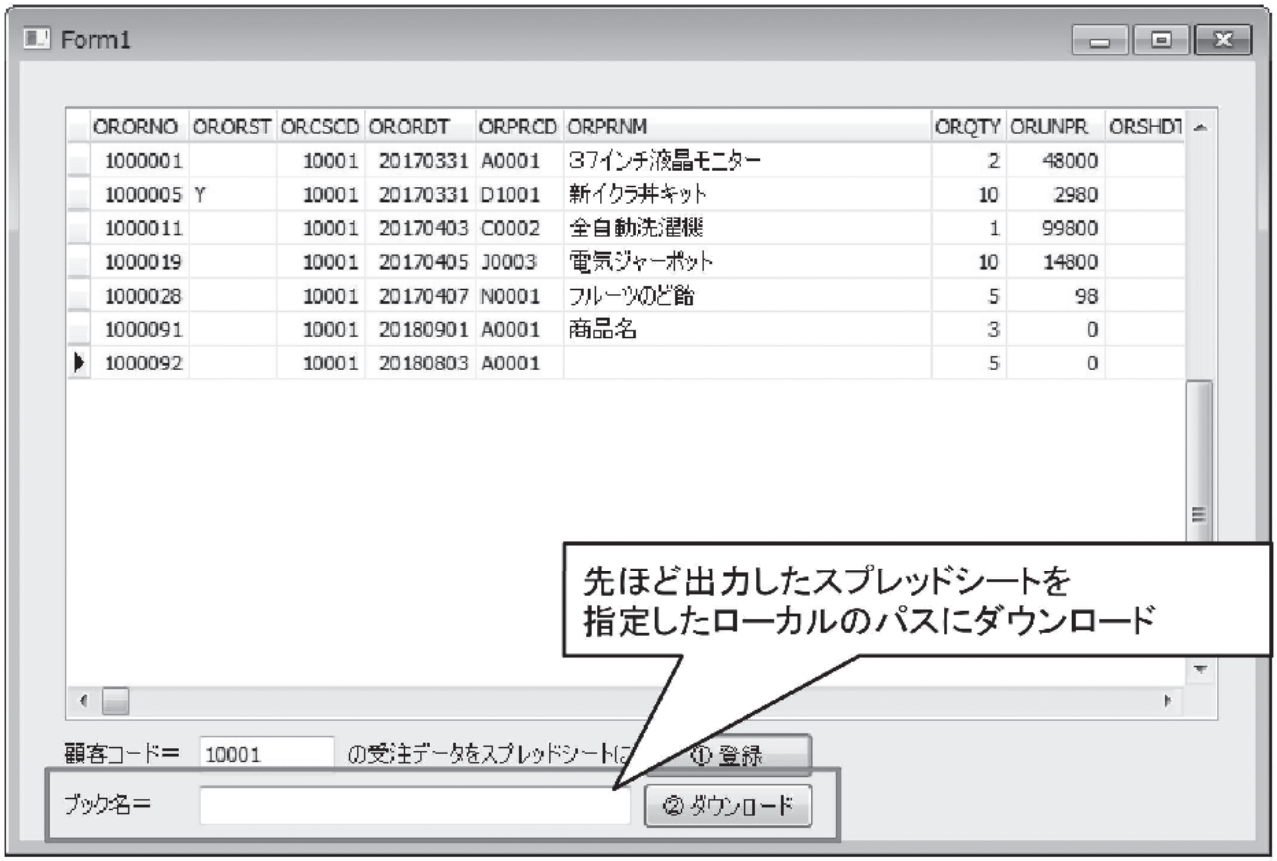


図30　Googleスプレッドシート⑦（ダウンロード結果）

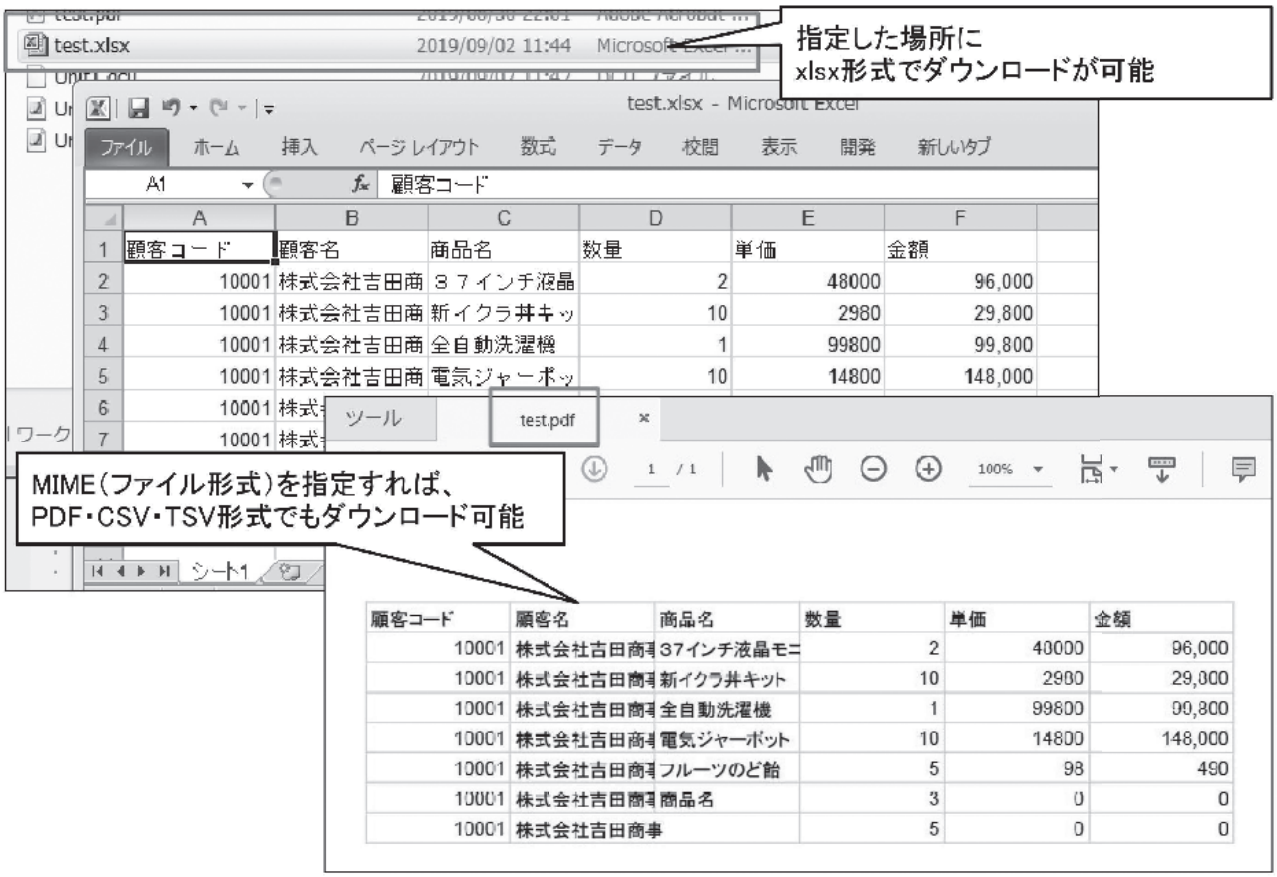


図31　Google Drive①（画面設計）

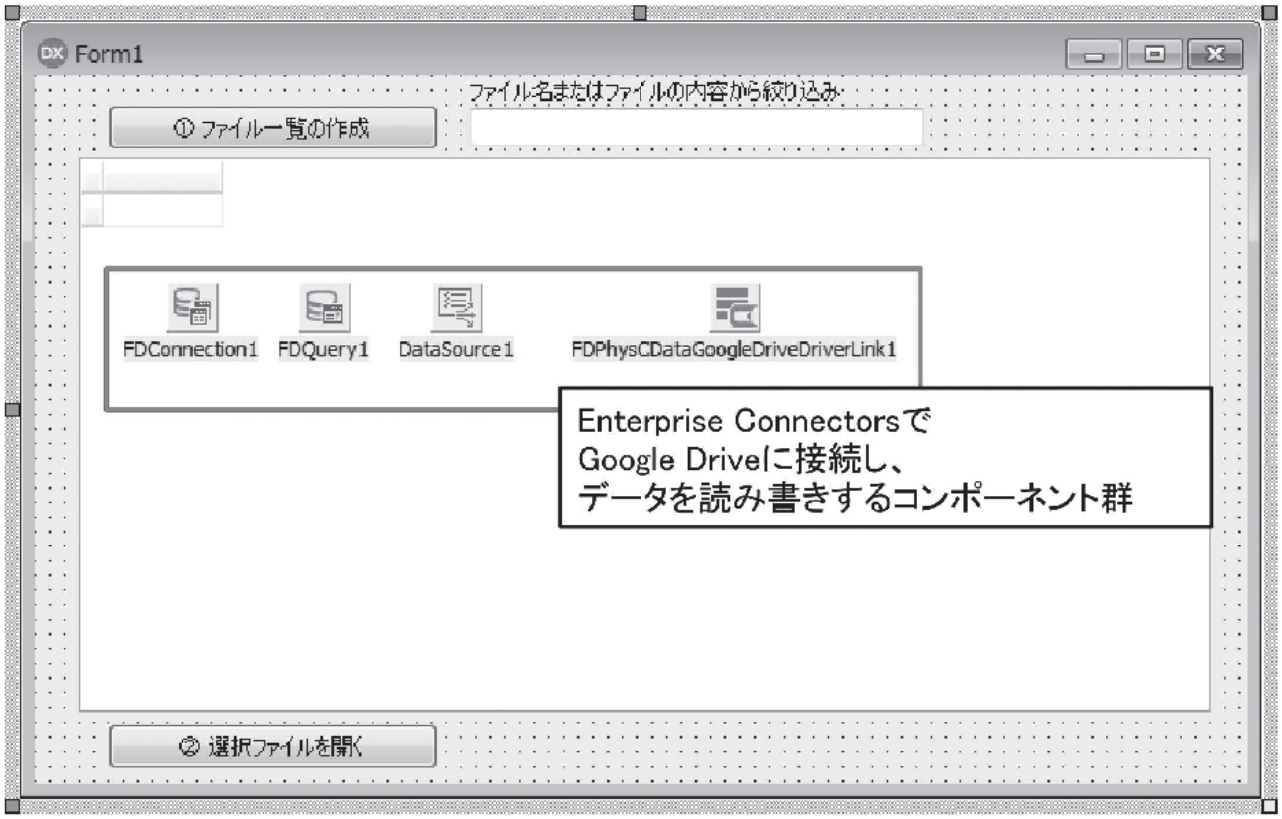


図32　Google Drive②（接続パラメータ設定）

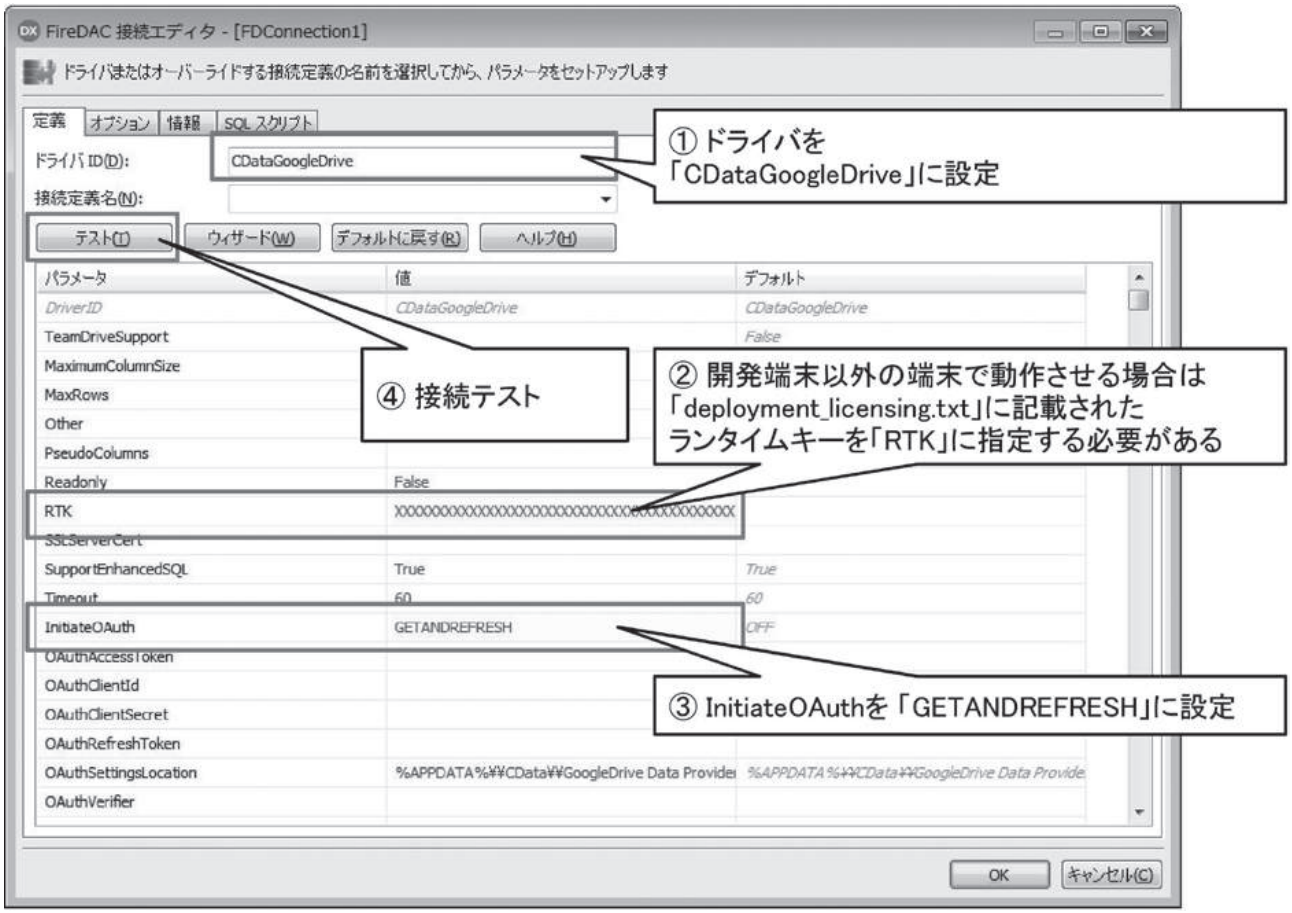
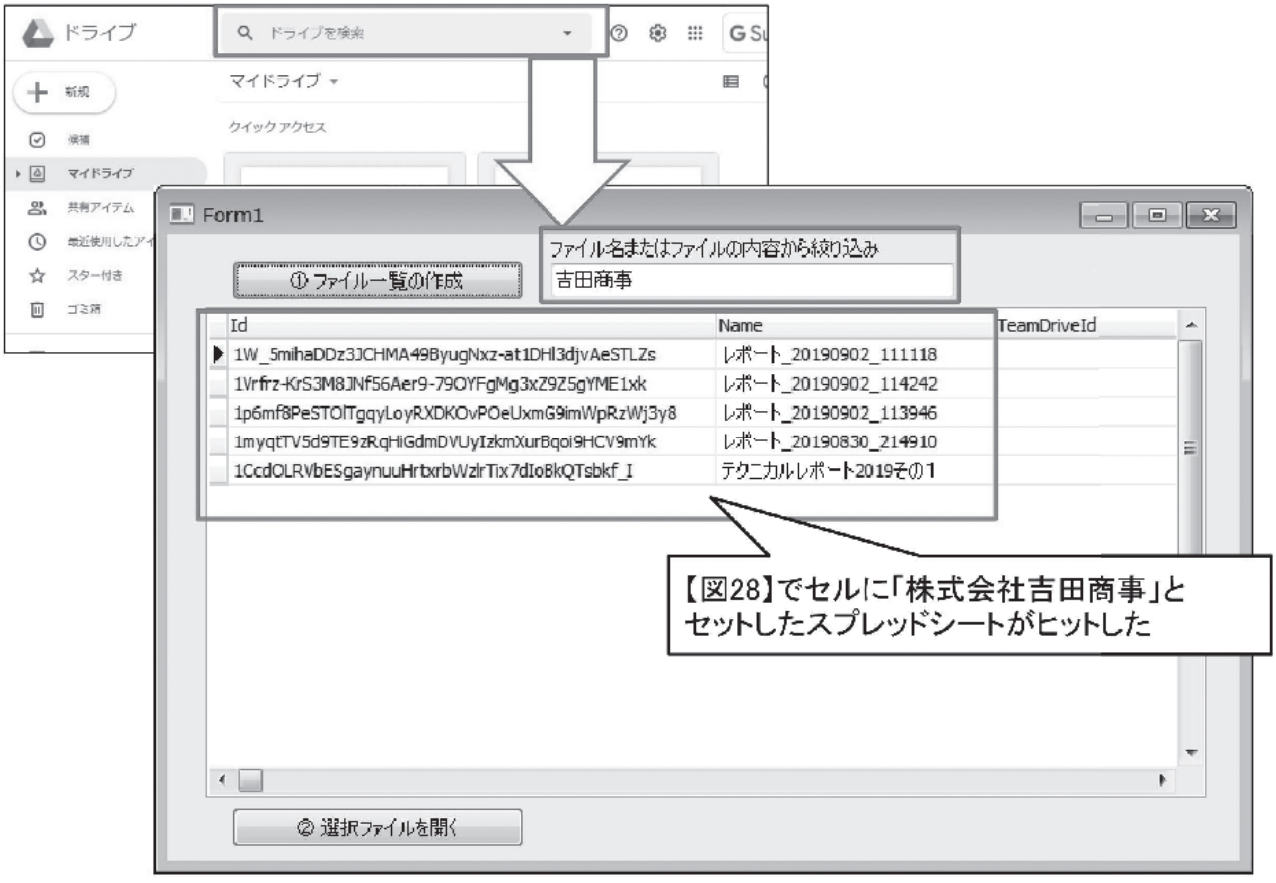


図33　Google Drive③（フルテキスト検索）



ソース8

Google Driveのファイル検索

```
{*************************************************************************
 目的: ①ファイル一覧の作成 押下時処理
*************************************************************************}
procedure TForm1.Button1Click(Sender: TObject);
begin
  // ファイルの一覧
  FDQuery1.Close;
  FDQuery1.SQL.Clear;
  FDQuery1.SQL.Add(' SELECT * FROM Files ');
  if (LabeledEdit1.Text <> '') then
  begin
    FDQuery1.SQL.Add(' WHERE Query = '' (fullText contains ¥''' + LabeledEdit1.Text + '¥'')'' ');
  end;
  FDQuery1.Open;
end;
```

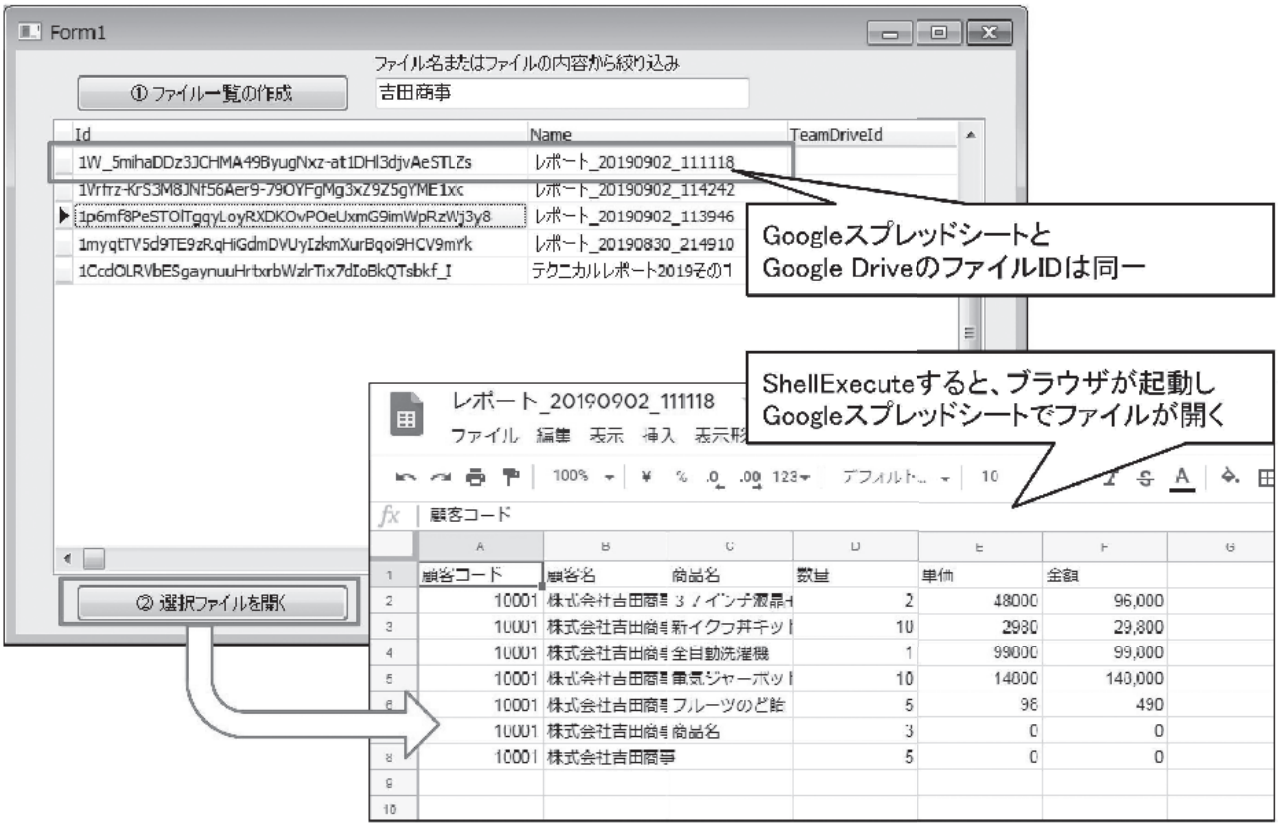
ファイル内の文字列を含めて
フルテキスト検索

ソース9

Google Driveで検索したファイルのオープン

```
{*************************************************************************
 目的: ②選択ファイルを開く 押下時処理
*************************************************************************}
procedure TForm1.Button2Click(Sender: TObject);
var
  sURL: String;
begin
  // 共有可能なURLを実行する (※uses節に Winapi.ShellAPI が必要)
  sURL := 'https://drive.google.com/open?id=' + FDQuery1.FieldByName('ID').AsString;
  ShellExecute(Application.Handle, 'OPEN', PChar(sURL), PChar(''), '', SW_SHOW);
end;
```

図34　Google Drive④(ファイルのオープン)



國元 祐二

株式会社ミガロ.

RAD事業部 技術支援課

[SmartPad4i]

SmartPad4iのインターフェース機能拡張

略歴
1979年3月27日生まれ
2002年3月 追手門学院大学 文学部アジア文化学科卒業
2010年10月 株式会社ミガロ.入社
2010年10月 RAD事業部配属

現在の仕事内容
SmartPad4i（JC/400）、Business4Mobile、Valenceの製品試験やサポート業務、導入支援などを行っている。

1. はじめに

SmartPad4i はIBM iのモダナイゼーションツールである。既存のRPG／COBOLの資産や知識を活用して、Webやモバイルアプリケーションが作成できる。

IBM iのグリーン画面で作成されたアプリケーションでは、DDS表示装置ファイルで画面を定義するが、SmartPad4iではHTMLで画面を定義する。画面への入出力処理を変更することで、既存の業務ロジックを活用しながら、簡単にSmartPad4iアプリケーションへ置き換えられる。

画面定義のHTMLは、最新のHTML、JavaScript、CSSを活用することで自由なインターフェースが構築できるのも魅力である。80桁×24行の制限がある、CUIのグリーン画面では実現できないインターフェースが作成可能である。

SmartPad4iのアプリケーションは、簡単なHTMLとRPG／COBOLの知識があれば作成できる。SmartPad4iの標準機能や、簡易なHTMLで作成できない機能が必要な場合には、JavaScriptを使用して機能拡張に対応する。

本稿では、サポートに問い合わせのあった内容の中から、SmartPad4iをご利用いただいているお客様にぜひ活用していただきたいと感じた、SmartPad4iの標準機能だけでは作成できないインターフェース機能拡張方法の手順を説明する。

2. JavaScriptについて

2-1.JavaScriptとは

JavaScriptは1995年、アメリカのプログラマー、Brendan Eich（ブレンダン・アイク）氏によって設計されたプログラミング言語だ。当初はHTMLへ動きを与えるために開発された言語である。現在では、Webブラウザ上で動作するだけでなく、サーバーサイドで動作する実行環境もあり、Web開発の領域で広く活用されている。

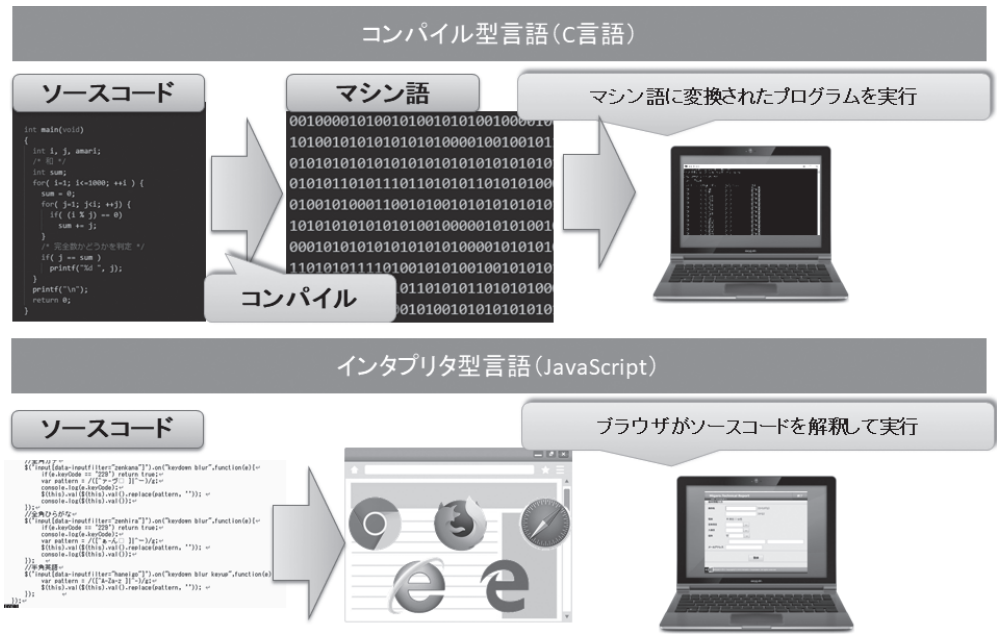
SmartPad4iでも、機能を追加する場合にはJavaScriptを使用する。JavaScriptはプログラミング言語であるが、難しく考える必要はない。JavaScriptは比較的簡単な言語で、初めてプログラミングをする初心者であっても、さまざまなことが実現可能となる。

JavaScriptが比較的簡単な言語とされている理由は2つある。1つは、変数にあらゆる型の値を代入できる点である。JavaScriptには型はあるが、型の制約は弱く、変数にどのような値も設定可能なので、プログラミング時に型を強く意識する必要がない。

もう1つは、プログラム環境を構築するための敷居が低い点である。JavaScriptのプログラミングには特別な環境は必要なく、PCにインストールされているブラウザで動作を確認できる。そのため、JavaScriptは初めてのプログラム作成に適している。

またC言語やRPG／COBOLなどは、コンパイル（プログラム言語で記述されたソースコードをコンピュータ上で実行

図1　JavaScriptはインタプリタ型言語



ソース1

HTMLに<script>タグを追加して処理する方法

```
<script>
    //JavaScriptの処理を記述する
</script>
```

外部ファイルとしてJavaScriptを読込み処理する方法

```
<script src = "/smartpad4i/html/js/sample.js"></script>
```

src属性にファイルパスを記述
/smartpad4i/html/js/sample.js
に配置した場合

図2　JavaScriptは</body>タグ前に記述

headタグ内に記述

```
<html>
<head>
<script>
    //複雑なJavaScriptの処理
</script>
</head>
<body>
   <!--  画面の要素表示 -->
</body>
</html>
```

bodyタグの閉じタグ前に記述

```
<html>
<head>
</head>
<body>
   <!--  画面の要素表示 -->
<script>
    //複雑なJavaScriptの処理
</script>
</body>
</html>
```

可能な形式に変換する処理）が必要である。しかしJavaScriptは、HTMLに記述したソースコードを即座にブラウザ上で実行できる。記述した言語をそのまま読み込み、動作するインタプリタ型言語であるからだ。たとえ誤ってソースコードを記述したとしても、書き直すだけで即座に動作へ反映される。【図1】

次は、JavaScriptの使い方について説明する。JavaScriptはHTMLにscriptタグを追加して処理する方法と、外部ファイルとしてJavaScriptを読み込み処理する方法の2種類ある。【ソース1】

scriptタグ自体はHTMLのどこに記述してもよいが、通常「headタグ内」/「bodyタグ内」/「bodyタグの閉じタグ前」に記述する。以前は、headタグ内に記述されることが多かった。しかし現在では、「bodyタグの閉じタグ前」に記述することが多い。これはブラウザがJavaScriptを処理する際、HTMLの表示処理を停止してしまうためだ。

つまり、「headタグ内」に記述したJavaScriptの処理で時間がかかる場合、JavaScriptの処理が完了するまで、画面は表示されない。そのため、ページを読み込ませる前にJavaScriptを読み込む必要がある場合には、「headタグ内」にscriptタグを記述し、それ以外の場合は画面の表示速度を考慮し、「bodyタグの閉じタグ前」にscriptタグを記述する。【図2】

JavaScriptは、scriptタグを記述するだけで使用できるので非常に簡単である。記述したscriptタグのプログラムはブラウザ上で動作するが、ブラウザの種類により、JavaScriptは方言のように少し違った記述が必要となる。

ブラウザごとに対応するJavaScriptを記述するのは骨が折れるので、記述の違いを吸収してくれるライブラリを使用すると便利である。ライブラリとして最も使用されているのが、次に解説するjQueryである。

2-2. jQueryとは

jQueryとは、米国のプログラマー、John Resig（ジョン・レッシグ）氏によって開発・公開されたJavaScript用のライブラリである。jQueryは著作権表示を消さなければ、商用・非商用を問わず、誰でも自由に利用できる。jQueryを使用すると、HTMLの要素取得やイベント処理、アニメーション、Ajax処理などを簡単に実装可能となる。

jQueryについては、「2015年 No.8 テクニカルレポート」にある『スマートデバイス開発で役立つ　画面拡張テクニック』でも取り上げたので、ご一読いただけると幸いだ。

2-3. jQueryの入手

jQueryは、SmartPad4iをインストールするとWebサーバー上に展開される。そのため、インターネット上からjQueryのライブラリをダウンロードしなくても使用できる。もちろん、SmartPad4iのインストーラーに含まれていない、最新のjQueryをインターネットからダウンロードして使用することも可能である。最新のjQueryは公式サイトからダウンロードできる。【図3】

jQuery 公式サイト
https://jquery.com

リンク先では、圧縮版（compressed, production_jQuery）をダウンロードする。jQueryのデバッグを行いたい場合には、非圧縮版（uncompressed, development_jQuery）をダウンロードすればよい。

2-4. jQueryの設定

jQueryを使用するための設定方法は簡単だ。HTML内にスクリプトタグを追加するだけで使用できる。スクリプトタグでは、src属性にjQueryファイルへのパスを設定するだけである。

また、jQueryはCDN（コンテンツデリバリネットワーク）からも読み込める。CDNはWebコンテンツをインターネット経由で配信するために最適化されたネットワークである。CDNを利用する場合のメリットは、サーバーにjQueryファイルを配置する必要がなくなる点と、CDN経由で直接ダウンロードすることにより、サーバーの負荷を減らせる点だ。デメリットは、クライアント端末がインターネット接続可能でなければならない点である。

jQueryのCDNは、「jQuery CDN」「Google Hosted Libraries」「Microsoft Ajax Content Delivery Network」などで公開されている。この中から、jQuery公式のCDNの使用方法について説明する。jQuery CDNの使用方法は、ブラウザで「jQuery CDN」のサイトにアクセス後、利用したいjQueryバージョンのリンクをクリックする。表示されたscriptタグをコピーしてHTMLに追加すればよい。【図4】

jQuery CDN
https://code.jquery.com/

なお本稿で紹介する機能は、SmartPad4iデフォルトで配置されるjQueryで実装している。【ソース2】

次に、jQueryを使用したSmartPad4iインターフェース機能拡張の実装方法を紹介する。

3. インターフェースの機能拡張

3-1.入力制限機能付きの入力欄

3-1-1. 入力欄の機能拡張

SmartPad4iではHTMLの作成後、SmartPad4i Designerというツールで入力欄の型や長さを設定すると、自動的にエラーチェックする入力欄が作成できる。

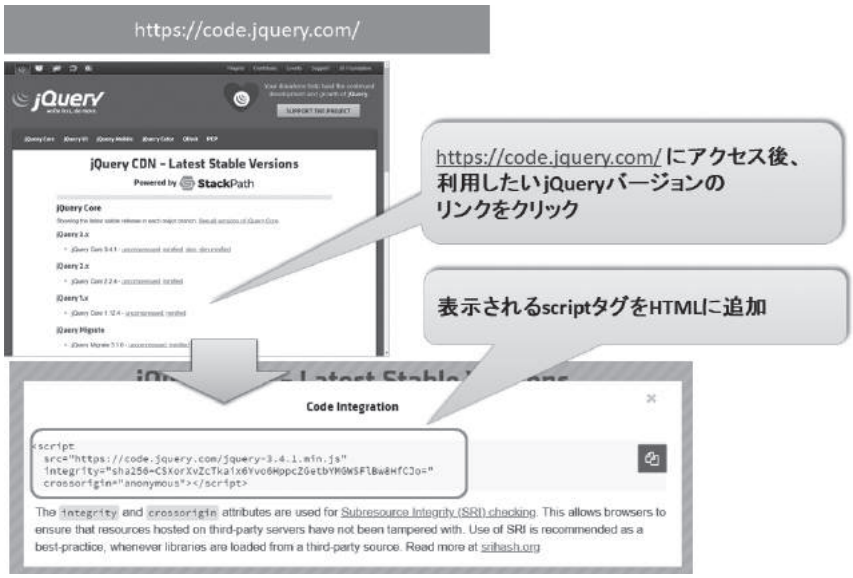
たとえば、IBM iのAタイプフィールドのように、シングルバイト文字のみ入力可能な入力欄や、Oタイプフィールドのようにシングルバイト文字とマルチバイト文字の両方が入力可能で、シフト文字を考慮した文字数チェックを行う入力欄が簡単に作成可能である。

それだけでも充分便利なチェック機能であるが、全角カナのみの入力を許可したい場合や、ひらがなだけを許可したい場合などは、SmartPad4iの通常機能では実現できない。機能を追加するには、JavaScript（jQuery）を使用する。

ここでは、入力欄の入力制限機能の作成方法を紹介する。入力欄の入力制限機能として、全角カナ以外の入力値はクリアする、IBM i TypeがOtypeの入力欄を例に挙げる。【図5】

図3　jQueryのダウンロード

図4　jQuery CDN



ソース2

SmartPad4i デフォルトで配置されるjQueryの読込み

```
<script src="/smartpad4i/APP_SP4i/AR/theme/jquery-1.10.2.min.js"></script>
```

ダウンロードしたjQueryの読込み

```
<script src="/smartpad4i/lib/jquery-3.4.1.min.js"></script>
```

/smartpad4i/lib/jquery-3.4.1.min.js に配置した場合

CDNjQueryの読込み

```
<script
  src="https://code.jquery.com/jquery-3.4.1.min.js"
  integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo="
  crossorigin="anonymous">
</script>
```

3-1-2　実装方法

HTMLの入力欄定義には、HTML5のマークアップ仕様で追加されたカスタムデータ属性を使用する。カスタムデータ属性は、HTMLタグに独自の属性を追加できる機能で、すべてのHTMLタグに設定できる。

設定方法は、タグの属性に「data-」から始まり、ハイフン以後は、アルファベットで任意の名前を開発者が自由に定義できる。入力制限機能のため、カスタムデータ属性の名称を「data-inputfilter」とした。カスタムデータ属性の値には、「zenkana」を設定している。

また同種のカスタムデータ属性を、複数のタグに設定できるので、カスタムデータ属性data-inputfilter = "zenkana"が設定されているすべてのinputタグに、全角カナのみが入力できるフィルタを適用できる。

JavaScriptは外部ファイルにして読み込むため、外部ファイルfilter.jsの読み込み設定を＜/body＞前に追加した。＜/body＞前に追加した理由は、ページのレンダリング完了後にJavaScriptファイルを読み込むことで、画面の表示速度を向上させるためである。なおfilter.jsは、［DocumentRoot］/SmartPad4i/html/js/に配置している。【ソース3】

ここで使用しているjQuery/JavaScriptの関数・メソッドは、【図6】になる。入力制限の機能は、4つのメソッドの使用方法をおさえるだけで作成が可能である。【ソース4】

外部ファイルfilter.jsのスクリプトは、$（function（）で書き始めている。$（function（）｛｝）;の記述方法は、HTMLが読み込まれた後、function関数を実行する。つまりfunction（）の後に記述された、｛｝の処理が実行される。

全角カナのみ許可したい入力欄には、カスタムデータ属性にdata-inputfilter = "zenkana"を設定した。

まず、カスタムデータ属性が設定された要素を取得する必要がある。jQueryでは、要素の取得に$()関数を利用する。$（）関数ですべてのinputタグを取得したい場合には、$（'input'）と定義する。

inputタグの中からカスタムデータ属性が設定された要素を取得するには、[]を使用して、プロパティで絞り込む。つまり、$（'input［data-inputfilter = "zenkana"］'）の記述は、すべてのinputタグからdata-inputfilterカスタムデータ属性が設定されていて、かつ値がzenkanaの要素だけ取得するという意味になる。

取得した要素にイベントを設定するには、onメソッドを使用する。onメソッドでは、第1引数に対象となるイベントの文字列、第2引数に実行する関数を設定できる。第1引数のイベント文字列は、半角スペース区切りで複数設定できる。

注意が必要なのは、keyイベントのkeyCodeが229（IME入力中）の場合に、処理を抜けるよう記述している点である。キーボードが押下されると、keyイベントが発生し、keyCodeにはどのキーを入力したかの値が設定される。

しかしIME入力時には、どのキーが入力されてもkeyCodeに229が設定されるため、入力の確定が行われるまで、処理しないようにしている。

以降の処理は単純で、キー入力（keydownイベント）、またはフォーカスが外れた（blurイベント）場合に、入力された値を正規表現でチェックして文字置換するだけである。

正規表現とは、複数ある文字列の集合を1つの形式で表現する手法だ。正規表現を使用すれば、特定の文字列パターンが出現するかどうか、またどこに出現するかを特定できる。設定した/（[^ァ-ヴ　］|^ー）/gの正規表現は、全角カナと半角スペース、全角スペース以外の文字列を特定できる。

全角カナの場合は、小文字の“ァ”から“ヴ”以外の文字を特定するようにパターンを作成している。【図7】は文字コード順に、カタカナ、ひらがなを並べた表である。ひらがなの場合は、“ぁ”から“ん”ですべてを網羅できるが、カタカナの場合は、“ン”ではなく、“ヴ”までになる点に注意する。小文字の“ヵ”や“ヶ”に関しては、1ヵ所や1ヶ月などの使われる文字で、カタカナではないので含めていない。

パターンに一致した値、つまり全角カナと半角スペース、全角スペース以外の文字列を空文字（''）に置き換えることで、入力制限を実現している。jQueryを使用すると、少しの記述で入力制限できる入力欄が作成できる。

また正規表現のパターンを変えることで、さまざまな入力制限が作成できる。たとえば、全角ひらがなであれば/（[^ぁ-ん　］|^ー）/gの正規表現パターンで入力制限が可能である。【ソース5】

全角カナのみ許可したい入力欄には、inputタグのカスタムデータ属性にdata-inputfilter = "zenkana"、全角ひらがなのみ許可したい入力欄には、inputタグのカスタムデータ属性にdata-inputfilter = "zenhira"を設定する。

外部JavaScriptのfilter.jsを読み込み、HTMLのカスタムデータ属性を設定するだけで、機能が追加できるので非常に便利だ。

さらに半角文字の入力制限であれば、keyupイベントを追加することで、半角英字以外入力できない状態にもできる。さまざまな入力制限が作成できるので、ぜひ活用してほしい。【図8】

3-2.入力可能なコンボボックス

3-2-1 コンボボックスの機能拡張

HTMLのコンボボックス（selectタグ）は、リスト内から選択するだけで、任意の文字は入力できない。ここではリストから選択でき、入力欄としても利用可能なコンボボックスの機能拡張方法を紹介する。

具体例として、フォーカス移動するとIBM iファイルに登録されているJANコード情報をリスト表示しつつ、入力欄としても利用できる入力可能なコンボボックスを作成する。【図9】【図10】

3-2-2 実装方法

ここで追加している、jQuery/JavaScriptの関数・メソッドは、【図11】のとおりである。

入力可能なコンボボックスの機能は、3-1で使用したjQuery／JavaScriptのメソッドと、追加9つのメソッドの使用方法により作成が可能である。

リストの一覧からの選択＋文字入力の機能を実現するには、入力欄の要素とコンボボックスの要素を組み合わせて作成する必要がある。HTMLの定義では、入力欄（inputタグ）とコンボボックス（selectタグ）の2つを使用する。

またカスタムデータ属性を利用して、関連付けを行っている。入力欄（input

図5　入力画面例

ソース3

入力欄のHTML定義

```
<input data-inputfilter="zenkana" type="text" name="INNM2" id="INNM2"> (カナ)
```

HTML5のマークアップ仕様で追加された
カスタムデータ属性を使用

JavaScript 外部ファイルの定義

```
～　省略　～
</form>
</div>
<script src="/smartpad4i/html/js/filter.js"></script>
</body>
</html>
```

外部ファイルの読込み

図6　3-1で使用するjQuery、JavaScriptのメソッド

関数/メソッド	機能
$()	要素の取得
.on	要素にイベント処理を追加
.val	要素のValue値を取得、設定
.replace	文字列の置換処理

タグ）には、カスタムデータ属性 data-customlist を、コンボボックス（select タグ）にはカスタムデータ属性 data-customlistc を定義している。

カスタムデータ属性の値で、入力欄とコンボボックスを関連付けるため、それぞれ値に“1”を設定した。さらに、コンボボックスは入力欄にフォーカスが移動した時に表示するため、css の属性 display:none を使用して、画面表示時は非表示に設定している。【ソース 6】【ソース 7】

入力欄にフォーカスが移動した場合の処理を追加するため、jQuery の on メソッドを使用した。最初に、カスタムデータ属性 data-customlistc が設定されている要素（コンボボックス）を hide メソッドで非表示にしている。

次に、data-customlistc が設定されている要素（コンボボックス）位置を css メソッドで横 0、縦 0 の位置に移動させた。“this”の記述は、イベント発生元の要素を指すため、フォーカスが移動した入力欄となる。

入力欄に設定されているカスタムデータ属性を取得するには、data メソッドを使用する。data メソッドで、入力欄のカスタムデータ属性、data-customlist の値を取得し、その値をもとに、filter メソッドで入力欄の下に表示させたいコンボボックスを絞り込んでいる。

またコンボボックスでリストから選択した値を、どの入力欄へ戻すかを設定するため、コンボボックスに data-returnkey というカスタムデータ属性を追加した。値には入力欄の id 属性を attr メソッドで取得し、設定している。リストとなるコンボボックスの位置が入力欄下になるよう調整後、コンボボックスを show メソッドで表示している。

【ソース 7】のプログラムは 10 行程度になったが、JavaScript のメソッドチェーンを使用すると 1 行で記述可能である。jQuery はとくにメソッドチェーンが効果的に使えるよう設計されているため、効率的にプログラムを作成できる。メソッドチェーンで記述したのが【ソース 8】である。【ソース 7】【ソース 8】は同じ処理になる。

【ソース 9】はコンボボックスが選択された場合や、別の入力欄にフォーカスが移動した場合の処理である。

カスタムデータ属性 data-customlist が、設定されていない入力欄にフォーカス移動した場合、コンボボックスを非表示にしている。また入力欄以外の要素がクリックされた場合にも、コンボボックスを非表示にしている。

リストから選択された場合には、選択値を入力欄に設定する。その入力欄を特定するため、入力欄にフォーカスが移動した際に、カスタムデータ属性 data-returnkey という属性をコンボボックスへ動的に設定している。

例では、data-returnkey ＝“INP1”が設定されるため、値を設定する入力欄（id 属性＝“INP1”）が特定できる。

これで入力欄にフォーカスが移動後、入力欄の下に関連するコンボボックスが表示され、リストから選択でき、入力欄としても利用可能なコンボボックスが完成した。

3-3. 選択リストが絞り込み可能なコンボボックス

3-3-1　絞り込み可能なコンボボックス

自動入力補完機能のように、コンボボックスのリストを絞り込めば操作性がよくなる。3-2 で実装したコンボボックスをさらに便利に利用できるよう、入力欄に値を入力すると、リストが絞り込まれるように実装する方法を紹介する。【図 12】

3-3-2　実装方法

ここで解説する jQuery ／ JavaScript の関数・メソッドは、【図 13】のとおりである。この機能は、これまでに使用した jQuery ／ JavaScript のメソッドと、追加 5 つのメソッドの使用方法により作成が可能である。

実装方法は、3-2 で実装した JavaScript に入力欄の keyup イベントを追加するだけである。【ソース 10】

$(function() {}); の中に、コンボボックスのリストをバックアップするための変数、clonecmb を宣言している。コンボボックスのリストは、入力欄へ入力された値をもとに絞り込む。絞り込みの際、一致しないリストを削除するため、初回に全リストを clone メソッドでバックアップしている。

そして、keyup と focus のイベント処理を記述した。イベントの中の処理は、【ソース 11】のとおりである。

jQuery では、$('［属性 ^ = " 値 "］') とセレクタを記述すると、属性値を前方一致で検索して対象の要素が取得できる。コンボボックスの全リストを入力された値の前方一致で検索して、not メソッドで一致しないリストを取得後、remove メソッドで削除している。

Chrome、FireFox、Safari、Microsoft Edge では、コンボボックスのリスト（option タグ）にスタイルシートを適用することで、リストを表示・非表示にできるが、Internet Explorer では残念ながら動作しない。そのため、コンボボックスのリストを直接削除する方法で実装した。

4. まとめ

本稿では、サポートに問い合わせのあった内容をレポートの題材とした。SmartPad4i は自由なインターフェースを作成できるモダナイゼーションツールであるが、jQuery を活用することでさらなる機能拡張も可能であることが実感できた。

実装が難しいと予想した問い合わせでも、JavaScript を使用してカスタマイズすることで、多くの機能は実現ができた。工夫次第で SmartPad4i アプリケーションは素晴らしいインターフェースを作成できるので、ぜひ本稿の内容を活用いただきたい。

M

ソース4

Filter.jsの記述内容

```
$(function(){
      //全角カナ
      $('input[data-inputfilter="zenkana"]').on("keydown blur",function(e){
            if(e.keyCode == '229') return true;
            var pattern = /([^ァ-ヴ　]|^ー)/g;
            $(this).val($(this).val().replace(pattern, ''));
      });
});
```

正規表現で入力制限

正規表現の記述

```
var pattern = /([^ァ-ヴ␣␣]|^ー)/g;
```

全角スペースと半角スペースを設定

図7　文字コード順について

ひらがな文字コード順

ぁ	あ	ぃ	い	ぅ	う	ぇ	え	ぉ	お	か	が	き	ぎ	く	ぐ	け	げ	こ	ご
さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た	だ	ち	ぢ	っ	つ	づ	て	で	と
ど	な	に	ぬ	ね	の	は	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ
ぽ	ま	み	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ	ゐ
ゑ	を	ん																	

カナ文字コード順

ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク	グ	ケ	ゲ	コ	ゴ
サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト
ド	ナ	ニ	ヌ	ネ	ノ	ハ	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ	ボ
ポ	マ	ミ	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ	ヰ
ヱ	ヲ	ン	ヴ	ヵ	ヶ														

ソース5

Filter.jsの記述内容

```
$(function(){
      ～省略～
      //全角ひらがな
      $('input[data-inputfilter="zenhira"]').on("keydown blur",function(e){
            if(e.keyCode == '229') return true;
            var pattern = /([^ぁ-ん　]|^ー)/g;
            $(this).val($(this).val().replace(pattern, ''));
      });
});
```

図8　入力画面例

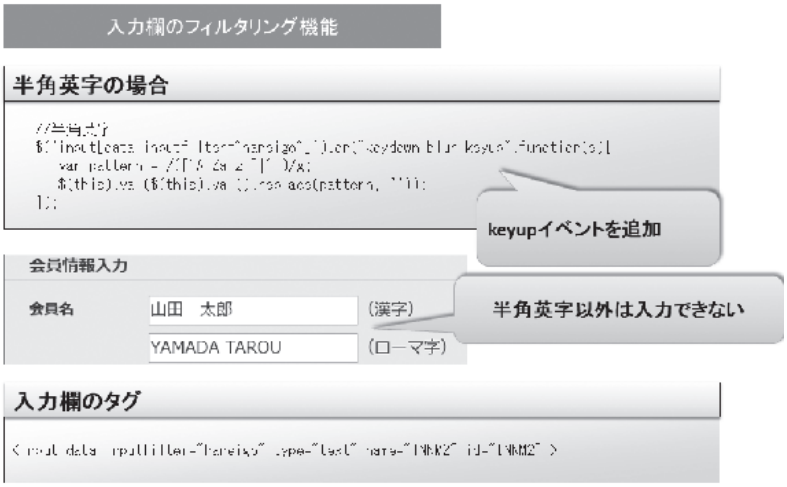


図9　入力可能なコンボボックス①

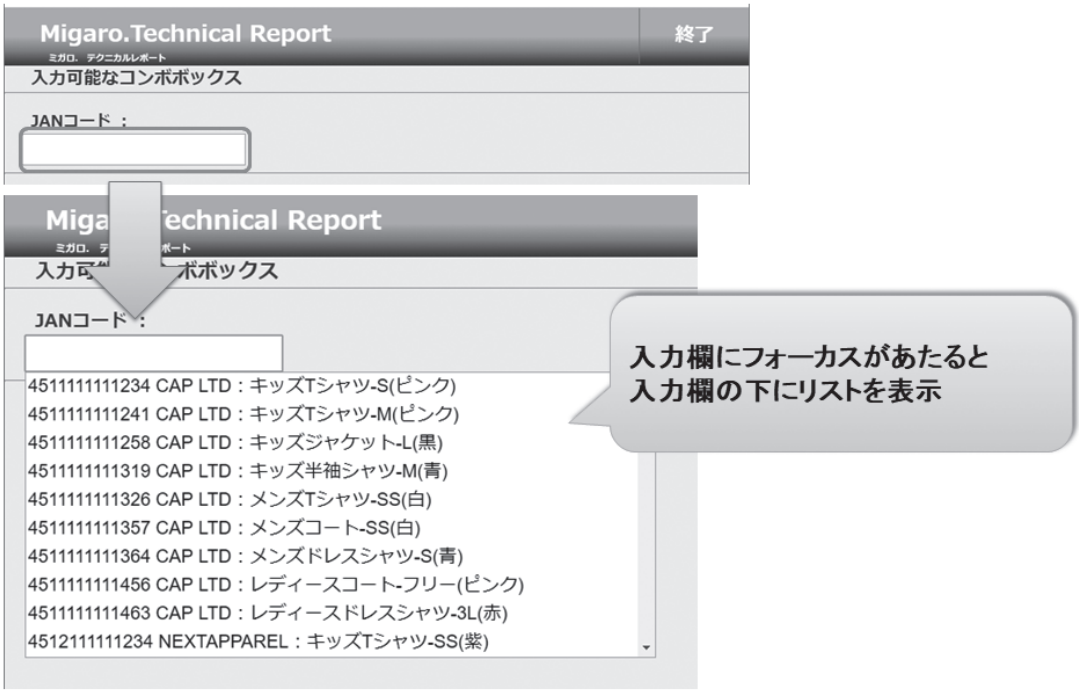


図10　入力可能なコンボボックス②

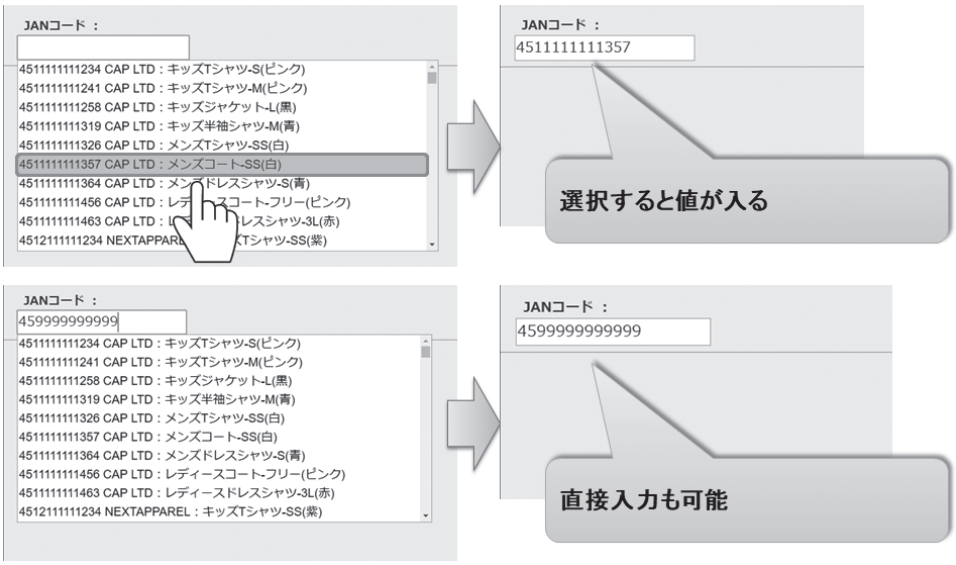


図11　3-2で使用するjQuery、JavaScriptのメソッド

関数/メソッド	機能
.hide	要素の非表示
.show	要素の表示
.css	css属性の取得、設定
.filter	要素の絞り込み
.data	要素のカスタムデータ属性を取得、設定
.attr	要素の属性を取得、設定
.offset	要素の位置を取得、設定
.outerHeight	要素の高さを取得
.find	要素の子要素を取得

ソース6　入力欄とコンボボックスのHTML定義

```
<!-- 入力欄の定義 -->
<input type="text" name="INP1" id="INP1" style="margin-left:20px;position:relative"
    data-customlist="1">

<!--コンボボックスの定義 -->
<select id="CMB01" style="display:none;position:absolute;font-size:18px"
    data-customlistc="1" multiple size="10"></select>
```

ソース7　入力欄のイベント定義

```
<script>
  $(function(){
    $('[data-customList]').on({
      //入力欄にフォーカスが設定されると、関連するコンボボックスを表示
      'focus' : function(){
        //コンボボックスを非表示
        $('[data-customlistc]').hide();
        //コンボボックスの位置を初期化
        $('[data-customlistc]').css('left','0');
        $('[data-customlistc]').css('top','0');
        //入力欄のカスタムデータ属性に設定された値のコンボボックスを選択
        var combo =$('[data-customlistc]')
           .filter('[data-customlistc='+ $(this).data('customlist')+']');
        //コンボボックスで選択した値をどの入力欄に戻すのかを設定
        combo.data('returnkey',$(this).attr('id'));
        //コンボボックスの位置を入力欄と同じ座標に設定
        combo.offset($(this).offset());
        //コンボボックスの位置を入力欄の下に設定
        combo.css('top',$(this).outerHeight() + $(this).offset().top);
        //入力欄の値がコンボボックスの要素にある場合は選択状態に設定
        combo.val($(this).val());
        //コンボボックスの表示
        combo.show();
    /* ～省略 ソース9 の内容～ */
</script>
```

ソース8　入力欄のイベント定義 メソッドチェーンで記述

```
<script>
  $(function(){
    $('[data-customList]').on({
      //入力欄にフォーカスが設定されると、関連するコンボボックスを表示
      'focus' : function(){
        $('[data-customlistc]').hide().css('left','0').css('top','0')
         .filter('[data-customlistc='+ $(this).data('customlist')+']')
         .data('returnkey',$(this).attr('id')).offset($(this).offset())
         .css('top',$(this).outerHeight() + $(this).offset().top)
          .val($(this).val()).show();
      },'click' : function(){
        //入力候補を表示しないための設定
        return false;
      }
    });

    /* ～省略 ソース9 の内容～ */
</script>
```

見やすいように改行を含めているが
1行で記述できる

ソース9　コンボボックスのイベント定義

```
<script>
  $(function(){
    /*  ～省略 ソース7 の内容～ */

    //他の要素にフォーカスが当たるとコンボボックスを非表示
    $('input:not([data-customlist])').on('focus',function(){
      $('[data-customlistc]').hide();
    });

    //入力欄以外の要素をクリックしたらコンボボックスを非表示
    $(document).on('click',function(){
      $('[data-customlistc]').hide();
    });

    //コンボボックスが選択されると入力欄に値を設定
    $('[data-customlistc]').on('change',function(){
      $('#' + $(this).data('returnkey')).val($(this).val());
    });
  });
</script>
```

図12　絞り込み可能なコンボボックス

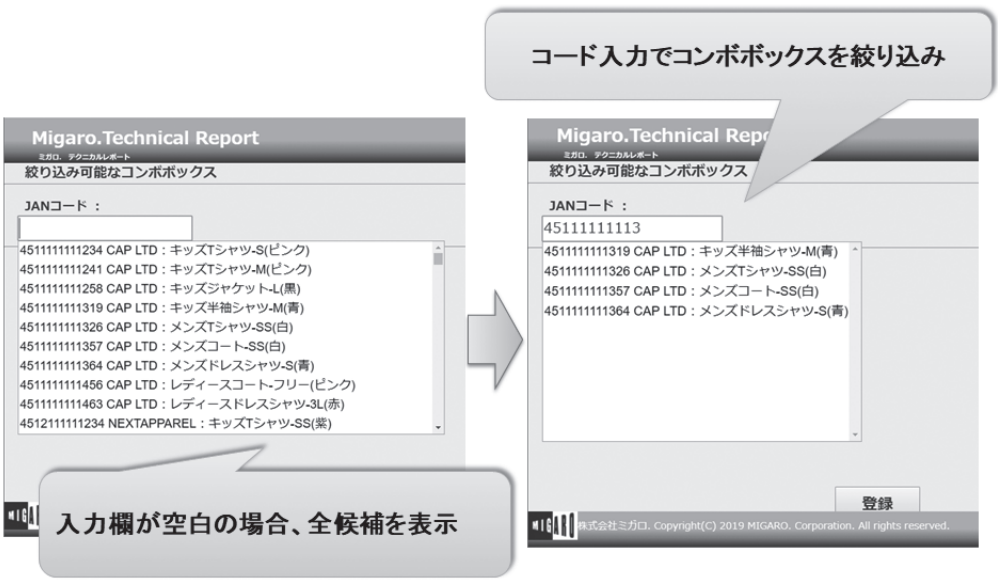


図13　3-3で使用するjQuery、JavaScriptのメソッド

関数/メソッド	機能
.remove	要素の削除
.append	要素の追加
.clone	要素の複製
.not	指定した要素の除外
.length	要素の数を取得

ソース10　コンボボックスの絞り込み処理イベント定義

```
<script>
  $(function(){
    //コンボボックス項目のバックアップ
    var clonecmb = null;
    $('[data-customList]').on({
      //入力欄にフォーカスが設定されると、関連するコンボボックスを表示
      'focus' : function(){
        $('[data-customlistc]').hide().css('left','0').css('top','0')
         .filter('[data-customlistc='+ $(this).data('customlist')+']')
         .data('returnkey',$(this).attr('id')).offset($(this).offset())
         .css('top',$(this).outerHeight() + $(this).offset().top)
          .val($(this).val()).show();
      },'click' : function(){
        //入力候補を表示しないための設定
        return false;
      },//イベント追加
      'keyup focus' : function(){
          /* ～省略 ソース11 の内容～ */
      }
    });
    /* ～省略 ソース9 の内容～ */
</script>
```

コンボボックス項目のバックアップ

keyup と focusイベント

ソース11　コンボボックスの絞り込み処理

```
        var inputValue = $(this).val().toUpperCase();
        var findval = '[value^="' + inputValue + '"]';
        var cmb = $('[data-customlistc]')
          .filter('[data-customlistc='+ $(this).data('customlist')+']');
        if(!clonecmb){
          clonecmb = cmb.find('option').clone(false);
        }else{
          //全項目を再作成
          cmb.find('option').remove();
          cmb.append(clonecmb.clone(false));
        }
        if(inputValue==''){
          return;
        }else{
          //入力値がコンボボックスの項目に含まれない場合、項目を削除
          cmb.find('option').not(findval).remove();
          if(cmb.find('option').length == 0){
            //項目が存在しない場合
            cmb.hide();
          }else{
            //項目が存在する場合
            cmb.show();
          }
        }
```

尾崎 浩司

Valence

株式会社ミガロ.

RAD事業部 技術支援課

[Valence]

Valence App Builder RPG連携テクニック

略歴
1973年8月16日生まれ
1996年3月 三重大学 工学部卒業
1999年10月 株式会社ミガロ. 入社
1999年10月 システム事業部配属
2013年4月 RAD事業部配属

現在の仕事内容
Delphi/400を中心としたテクニカルサポート対応や製品セミナーの講師などを担当している。

1. はじめに

Valenceは、IBM iをWeb環境で活用するモダナイゼーション開発・運用ツールである。Valenceには、次のような特長がある。

・ローコード開発ツールによりアプリの高速開発が可能
・ビジュアルな運用管理ツール群を搭載
・PCのWebブラウザ、モバイル両方に対応

Valenceの特長の1つであるローコード開発ツールが、「Valence App Builder」である。Valence App Builderを使用すれば、基幹システム開発に必要な入力・更新・照会アプリケーションが、ほぼノンコーディングで作成できる。

また、その作成方法はウィザードを使用したシンプルな手法なので、操作に慣れると、簡単なものであれば、ものの数分でアプリケーションを作成することが可能である。

Valence App Builderはノンコーディングで簡単にアプリケーションを開発できるが、複雑な業務ロジックを伴うアプリケーションを開発する場合には、RPGを組み合わせることも可能だ。

本稿では、Valence App BuilderとRPGとを組み合わせて連携するためのテクニックについて紹介する。

2. Valence App Builderアプリ開発方法

はじめに、Valence App Builderのアプリケーション開発方法を確認しておこう。Valence App Builderでは、次の3つのステップでアプリケーションを作成する。

1　データソースの作成
2　ウィジェットの作成
3　アプリケーションの作成

「1　データソースの作成」では、アプリケーションで使用するデータ元となるIBM i上のファイルを選択し、「2　ウィジェットの作成」では、データソースをもとに、表（グリッド）やグラフ等のデータを表現するための部品を定義する。そして、「3　アプリケーションの作成」では、1つあるいは複数のウィジェットを画面上に配置し、アクション（動作）を定義することで、1つのアプリケーションとして完成させるのである。【図1】

実際の開発手順は、次のとおりだ。まず、データソースを作成する。データソースの作成は、IBM iのQUERYユーティリティで、QUERYの定義を作成するのと似ている。データソース作成ウィザードを使用して、アプリケーションが使用するIBM i上のファイルとカラム（フィールド）を選択・保存すればよい。

ウィザードではほかにも複数ファイルを使用した時の結合条件や、データの絞り込み条件、並び順なども指定できる。【図2】

次に、保存したデータソースをもとにウィジェットを作成する。ウィジェット

図1　Valence App Builder　3つの開発ステップ

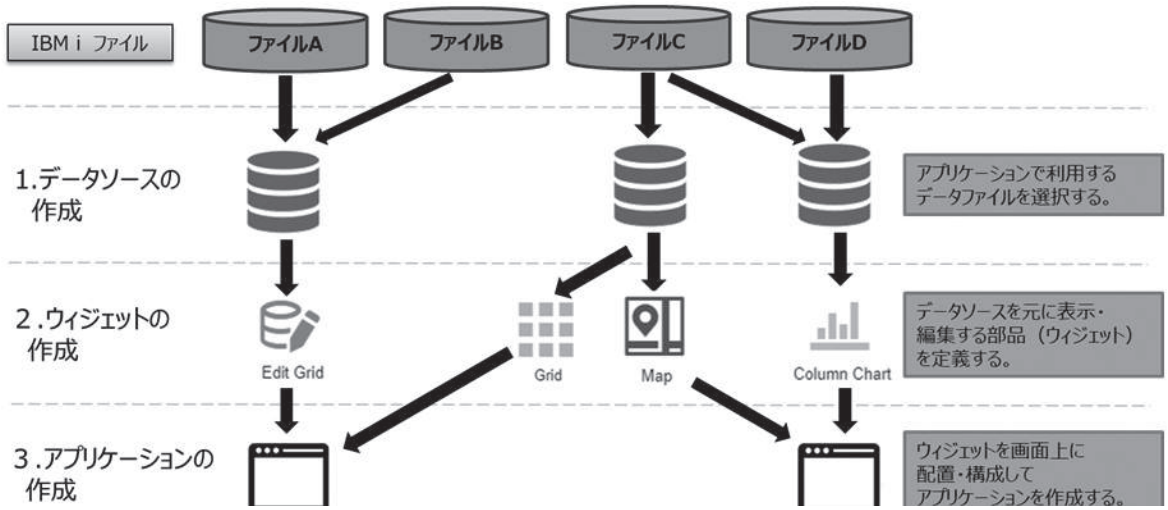
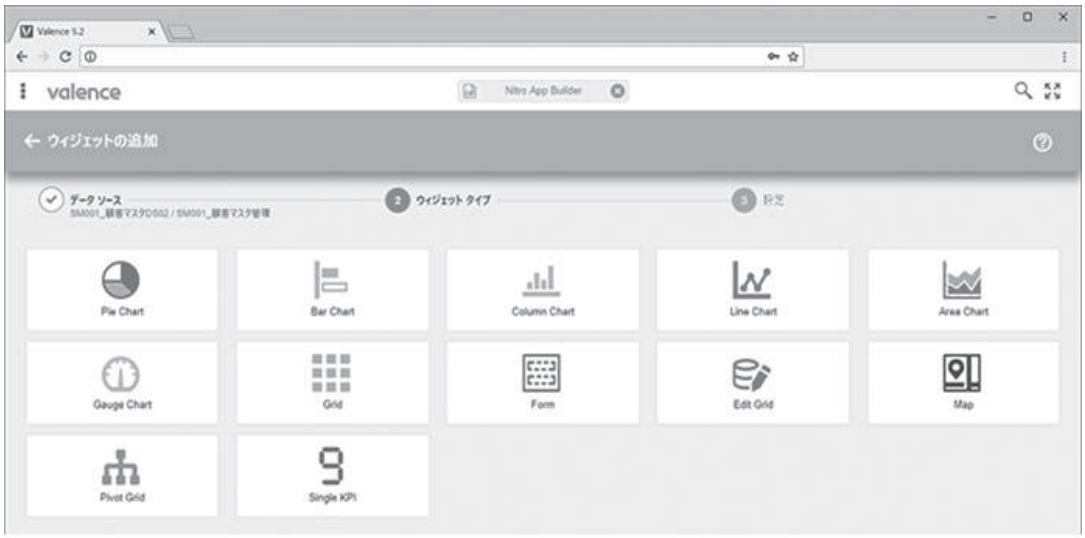


図2　データソース作成ウィザード

図3　ウィジェット一覧



グラフ　　：　円グラフ、 棒グラフ（横、縦）、折れ線グラフ、 面グラフ、 ゲージ（メーター）
グリッド　：　グリッド（表）、 編集用グリッド、 クロス集計グリッド、
その他　　：　フォーム、 指標表示（シングルKPI）、地図

は、抽出したデータをさまざまな見せ方で表示するための部品である。2019年8月現在の最新版であるValence5.2には、表（グリッド）やグラフ、地図など全部で12種類のウィジェットが用意されている。【図3】

選択したウィジェットに応じた設定画面が用意されているので、データソース上のカラム（フィールド）を関連付ける。たとえばグリッドであれば、表示させる列の指定や書式の設定を行えばよい。また、グリッドやグラフのウィジェットでは、データソース上のレコードを絞り込むためのフィルタ設定が可能である。これにより、実行時にユーザーによる条件指定が行える。【図4】

必要なウィジェットが揃ったら、1つあるいは複数のウィジェットを組み合わせてアプリケーションを作成する。複数のウィジェットを1つのワークスペースに配置した単一画面のアプリケーションや、複数のセクションを定義してワークスペースを増やすことで、複数画面をもつアプリケーションを作成できる。【図5】

ワークスペース上に複数のウィジェットを配置した場合や、複数のアプリケーションセクションを定義した場合には、ユーザーの操作に対するアクションを設定できる。たとえば、グリッドに表示された受注一覧の行をクリックした時に、選択した受注NOでデータの絞り込み（フィルタ）を行い、結果を詳細フォームに表示するといったアクションが設定できる。【図6】

以上が、Valence App Builderでアプリケーションを開発する手順である。基本的な部分はウィザードを使った簡単な定義で済み、一切のコーディングを行わずにアプリケーションが作成できる。

3. Valence App BuilderとRPG連携の基本

Valence App Builderでは、編集グリッド（Edit Grid）やフォーム（Form）の入力欄に関して、カラムの属性に合わせた入力可能文字の制御や項目の必須チェックなどを、ウィザードで設定できる。

しかし、たとえば複数項目間の相関エラーチェック等はウィザードでは設定できない。また、データソース上のカラム（フィールド）以外のファイルやフィールドを内部的に更新することはできない。こうしたValence App Builderの設定だけでは実現できない処理は、RPGロジックを追加することで対応する。

Valence App Builderでは、下記のタイミングからRPGプログラムを呼び出して実行できる。

1 グリッドの行やグラフをクリック、あるいはウィジェットやアプリケーションセクションに追加したボタンをクリックした時
2 編集グリッド（Edit Grid）でレコードの追加／更新／削除を行う時
3 グリッド等でユーザーがフィルタ（絞り込み）を行う時

1については、アプリケーションの動作内容（アクション）設定画面でRPGプログラム呼び出しのアクションを追加できる（なお、ウィジェットやアプリケーションセクション上にボタンを追加するのも、この動作内容設定画面で行う）。【図7】

2と3は、それぞれウィジェットの設定画面で呼び出したいRPGプログラムのIDを追加すればよい。【図8】【図9】

Valence App Builderでは、実行するRPGプログラムをテンプレートプログラムからコピーして作成する。テンプレートプログラムは、Valenceライブラリ上にあるQRPGLESRCの中に含まれている。【図10】

1 クリック：VALENCE52/QRPGLESRC（EXNABBTN）
2 Edit Grid：VALENCE52/QRPGLESRC（EXNABVAL）
3 フィルタ：VALENCE52/QRPGLESRC（EXNABFLT）

1のクリック時に実行されるプログラム（EXNABBTN）のテンプレートプログラムは、【ソース1】である。テンプレートプログラムは、フリーフォームRPGで記述されている。

【1-①】が実行するプログラムの定義（プロトタイプPRおよびプロシージャーインターフェースPI）となっており、アプリケーションを作成する際には、ここにそれぞれプログラムIDを指定する。

【1-②】がプログラムのメイン処理で、この中からProcessサブプロシージャーを呼び出している。呼び出されたProcessサブプロシージャー【1-③】の中に、アプリケーションに必要なロジックを記述すればよい。

具体例を2つ紹介する。1つ目は、商品マスタメンテナンスの登録画面である。【図11】

フォームウィジェットを使用した簡単なアプリケーションで、商品CD、商品名、単価を入力して、登録ボタンをクリックすると、商品マスタに新規レコードを追加する。この登録ボタンをクリックした時に呼び出すRPGプログラム（TEC010）が、【ソース2】である。

【2-①】のプログラム定義では、プログラムIDの“TEC010”を指定している。そしてProcessサブプロシージャーの中に必要な処理を記述している。【2-②】は、Processサブプロシージャーの中で使用する変数の定義である。【2-③】は、フォームウィジェット上の入力項目の値を取得して、【2-②】で定義した変数に代入する処理である。

GetFormChar（文字）/GetFormNum（数値）は、ウィジェットに定義されたカラム（フィールド）の値を取得するために用意されたValenceのAPIである。

【2-④】がこのプログラムの主処理で、この中でキーフィールドの重複チェックおよび新規レコードの登録を行っている。その中にある【2-⑤】【2-⑥】が、処理結果をValenceアプリケーション側に返却する処理である。vvOut_toJsonPairというValenceのAPIにより、処理結果が返却される。

この命令のパラメータにある“success:true（false）”は、処理結果の正常／エラーを表し、“msg”はエラー等のポップアップメッセージを、“info”は画面下部に表示されるお知らせメッセージを表す。

このプログラムを見るとわかるように、Valence App BuilderにおけるRPGにはフリーフォームRPGを使用するのだが、ValenceのAPIを使用する部分のみ、フリー形式（/free ～ /end-free）で記述しており、その他のロジック部分は、従来どおりの固定長形式で記述している。

図4　ウィジェットのフィルタ設定

絞り込み対象とするフィールドを指定

ウィジェットの中に、絞り込み条件欄が追加される

図5　アプリケーションの作成

画面（ワークスペース）上に一つあるいは複数のウィジェットを配置できる

つまりテンプレートプログラムの仕組み（構成）と Valence の API の使用方法さえマスタすれば、これまでどおりの RPG スキルをそのまま活用して開発できる。

なお、作成した RPG ソースは通常どおりにコンパイルしてオブジェクトを生成すればよいが、コンパイル時に RPG プリプロセッサ・オプションを“*LVL2”に変更する点に注意してほしい。【図 12】

2つ目は、編集グリッド（Edit Grid）を使用したユーザーマスタメンテナンス画面である。【図 13】

このアプリケーションでは、入力画面上の連絡区分の値により、異なる必須項目エラーチェックを行う。レコード新規追加／更新／削除のタイミングで呼び出す RPG プログラム（TEC020）が、【ソース 3-1】【ソース 3-2】である。このプログラムは、テンプレートプログラム EXNABVAL をコピーして作成したものである。

【3-①】のプログラム定義では、プログラム ID の“TEC020”を指定している。【3-②】がプログラムのメイン処理となっている。パラメータ inMode は、Edit Grid の実行状態を表しており、レコードの新規追加時は“ADD”、編集時は“EDIT”、削除時は“DELETE”がセットされるようになっている。

そして実行状態に応じて、新規追加時であれば ProcesssAdd サブプロシージャーが、編集時であれば ProcessEdit サブプロシージャーが、削除時であれば ProcessDelete サブプロシージャーが呼び出されるので、それぞれに必要なアプリケーションロジックを各サブプロシージャーに記述すればよい。

【3-③】がこの ProcesssAdd サブプロシージャーの主処理で、連絡区分の値に応じたエラーチェック処理を実行している。その中にある【3-④】【3-⑤】が Valence の API を使用した部分でフリー形式である（【3-④】【3-⑤】以外の箇所は、固定長形式で記述している）。

【3-④】が、Edit Grid 上の編集画面で更新された項目の値を取得して変数に代入する処理である。値の取得には、GetValue を使用する。これは、パラメータに指定したファイル、フィールドの値を取得する Valence の API である。【3-⑤】に記述された SendError は、データの更新処理を中断し、エラーメッセージを出力する API である。

今回は2つの具体例をもとに、Valence App Builder から RPG プログラムを連携する方法を紹介した。

本章の最後に、Valence App Builder の各 RPG テンプレートプログラムから使用できる API 一覧を紹介する。【図 14】

なお各 API の詳細は、Valence Portal にある「Valence API ドキュメント」メニューに説明とサンプルが記載されているので、確認してほしい。

4. Valence RPG ToolKit活用テクニック

前章では、Valence App Builder から RPG プログラムを呼び出す方法を説明したが、他にも Valence には、RPG を活用して機能拡張を行う仕組みが用意されている。それが、RPG ToolKit である。ここでは RPG ToolKit の活用例として、ファイルのダウンロード機能、メール送信機能の実装を紹介する。

まず、ファイルのダウンロード実装例を紹介する。ファイルのダウンロードといっても大きく分けて、「1　あらかじめ保存しているファイルをダウンロードする」「2　動的にファイルを生成してダウンロードする」の2つが考えられる。今回は両方の手法を紹介する。

1 の保存ファイルダウンロードについて、具体例として商品カタログ PDF のダウンロード機能を紹介する。【図 15】

フォームウィジェット上にある［カタログダウンロード］ボタンをクリックすると、IFS 上に保存された選択商品 CD に関するカタログファイル（［商品 CD］.pdf）をダウンロードする。このボタンをクリックした時に実行される RPG プログラム（TEC030）が、【ソース 4】である。

【4-①】が、IFS 上に保存された PDF ファイルの保管先パスを取得するロジックである。vvUtility_getValenceSetting は、Valence の設定情報を取得する API で、パラメータに“ROOT_PATH”を指定することで、IFS 上の Valence ルートフォルダを取得できる。今回は取得したルートフォルダの配下にある "resources/pdfs" の中に、PDF ファイルがあらかじめ保存されている想定である。

【4-②】では、対象となる商品 CD の PDF ファイルが存在するかどうかを判定している。vvIfs_pathExists は、パラメータに指定したファイルが存在するかどうかを確認するための API で、ファイルが存在する場合は“*ON”、存在しない場合は“*OFF”が返却される。今回はファイルが存在しない場合、エラーメッセージを Valence に渡している。

【4-③】が、PDF ファイルを Valence 側に受け渡す処理である。vvout 構造体は出力するファイルを指定するもので、download フィールドに“1”をセットすると、クライアントへファイルを受け渡すモードとなる。そして file フィールドにダウンロードしたいファイルをセットし、vvOut_file という API を実行すると、ファイルが Valence 側へ渡される。

なおファイルのダウンロード機能を実装する場合は、Valence App Builder の RPG プログラム呼び出し設定画面にて、［ファイル返却時の処理］に“ダウンロード”を指定する。【図 16】

次に、2 の動的に作成したファイルダウンロードについて、具体例として担当者別受注一覧 Excel のダウンロード機能を紹介する。【図 17】

フォームウィジェット上にある［受注ダウンロード］ボタンをクリックすると、IBM i 上の受注ファイル（F_JUCHU_H）から、選択した担当者 CD の受注データを取得し、その結果をもとに動的に作成した Excel ファイルをダウンロードする。ボタンをクリックした時に実行される RPG プログラム（TEC040）が、【ソース 5】である。

【5-①】が、受注ファイルを取得するためのデータ抽出用 SQL 文である。フォーム上の担当者 CD を取得し、その値を使った SQL 文字列を作成している。【5-②】が Excel ファイルのダウンロード処理である。vvOut_execSqlToSS は、SQL で取得した結果をもとに Excel ファイルを出力する API である。

なお今回の方法では、SQL をもとに Excel ファイルを作成しているが、データ構造体（DS）をもとに Excel ファイ

図6　動作内容（アクション）の設定

図7　クリック時のRPG呼び出し

図8　Edit Grid　編集時のRPG呼び出し

ルを生成するvvOut_toSSというAPIも用意されている。READ命令などファイル操作で取得したデータを出力する場合には、データ構造体に値をセットした上で、こちらのAPIを使用すればよい。

ダウンロードされたExcelファイルを見ると、SQLで取得した結果が出力されているのがわかる。【図18】

ただし、このままではSQLにより取得されたすべてのフィールドが出力されており、行タイトルには、フィールドIDがそのまま出力される。ロジックを追加することにより、出力するExcelの列設定をカスタマイズすることもできる。カスタマイズしたTEC040プログラムが、【ソース6】である。

【6-①】で、列の設定を行うためのvvSSCol構造体の配列を定義している。そして【6-②】の部分で、Excelの列ごとに出力したいカラム（フィールド）および列のタイトルを設定している。【6-③】のvvOut_execSqlToSSにて、出力するSQL文字列とともに、列定義の構造体配列をパラメータにセットすることで､列定義を含むExcel出力が行える。【図19】

最後に、メール送信機能の実装例を紹介する。ここでは、ファイルダウンロードで作成したExcelと同じものを添付してメール送信する機能を紹介する。【図20】

フォームウィジェット上にある［メール送信］ボタンをクリックすると、メール送信先アドレスを入力するダイアログが表示される。RPGプログラム呼び出しのアクションを設定する際に、任意のパラメータを追加できるオプション設定があり、今回はメールアドレスとして、パラメータ“MAILTO”を追加している。【図21】

メールアドレスを入力して、［OK］ボタンをクリックすると、動的に作成したExcelファイルを添付したメールを、指定したアドレスに送信する｡この［メール送信］ボタンをクリックした時に実行されるRPGプログラム（TEC050）が、【ソース7】である。

メール送信のAPIを使用する場合､【7-①】のようにD仕様書に“/define includeEMAIL”を追加する。【7-②】では、vvIn_charを使用している。これは、Valence側で指定された文字列を取得するAPIで、追加パラメータに指定した“MAILTO”と“sid”を取得している。“sid”はセッションIDのことで、Valenceを実行しているブラウザごとに一意になる文字列が取得できる。

【7-③】が、Excelファイルを生成する部分である。今回はダウンロードではなく、IFS上にExcelファイルを一時保管している。vvOut構造体のdownloadフィールドに“F”を設定すると、IFS上へファイルが作成できる。今回は、セッションIDを使用した一意な名称のExcelファイルを生成している。

【7-④】が、メール送信内容を作成する部分である。vvMail構造体に宛先、題名、本文、添付ファイル等の送信内容をセットすればよい。

【7-⑤】が、実際にメールを送信する処理である。vvMail_sendを実行し、メールが正常に送信できた場合には“*ON”が、エラーとなった場合は“*OFF”が返却される。

メール送信処理が終わったら、IFS上に一時保管したExcelファイルを【7-⑥】で削除している。vvIfs_deleteFileは、IFS上のファイルを削除する処理である。

今回のプログラムにより、指定されたメールアドレス宛てに、添付ファイル付きのメール送信が行える。【図22】

なお、メール送信機能を使用する場合、あらかじめValenceの設定画面にて、SMTPサーバーに関する設定をしておく必要がある。

5. さいごに

本稿では、Valence App BuilderからRPGを連携する手法について紹介してきた。Valence App Builderは、3つのステップで簡単にアプリケーションが作成できる。

ローコード開発プラットフォームであるValence App Builderは、一切のコーディングを行わずに開発できる。もちろん業務ロジックなど複雑な処理にはプログラムの追加が必要となるが、テンプレートプログラムと、ValenceのAPI使用方法をマスタすれば、RPGスキルで簡単に処理が追加できる。

また今回紹介したとおり、RPG ToolKitを活用すれば、Valence App Builderのアプリケーションに対し、さまざまな機能拡張も行える。ぜひ本稿を参考に、さらなるValence App Builderの活用をご検討いただきたい。

M

図9　フィルタウィジェットのRPG呼び出し

図10　テンプレートRPGソース

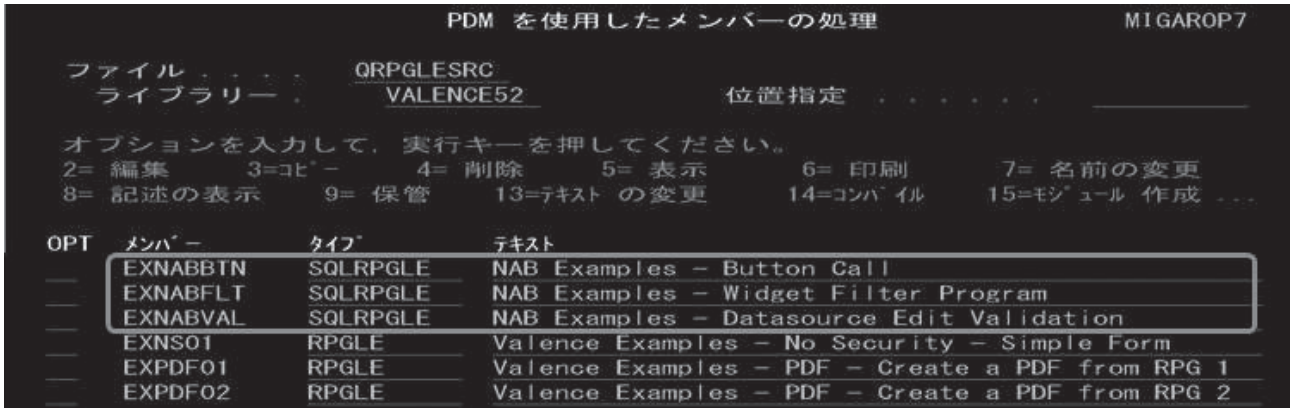


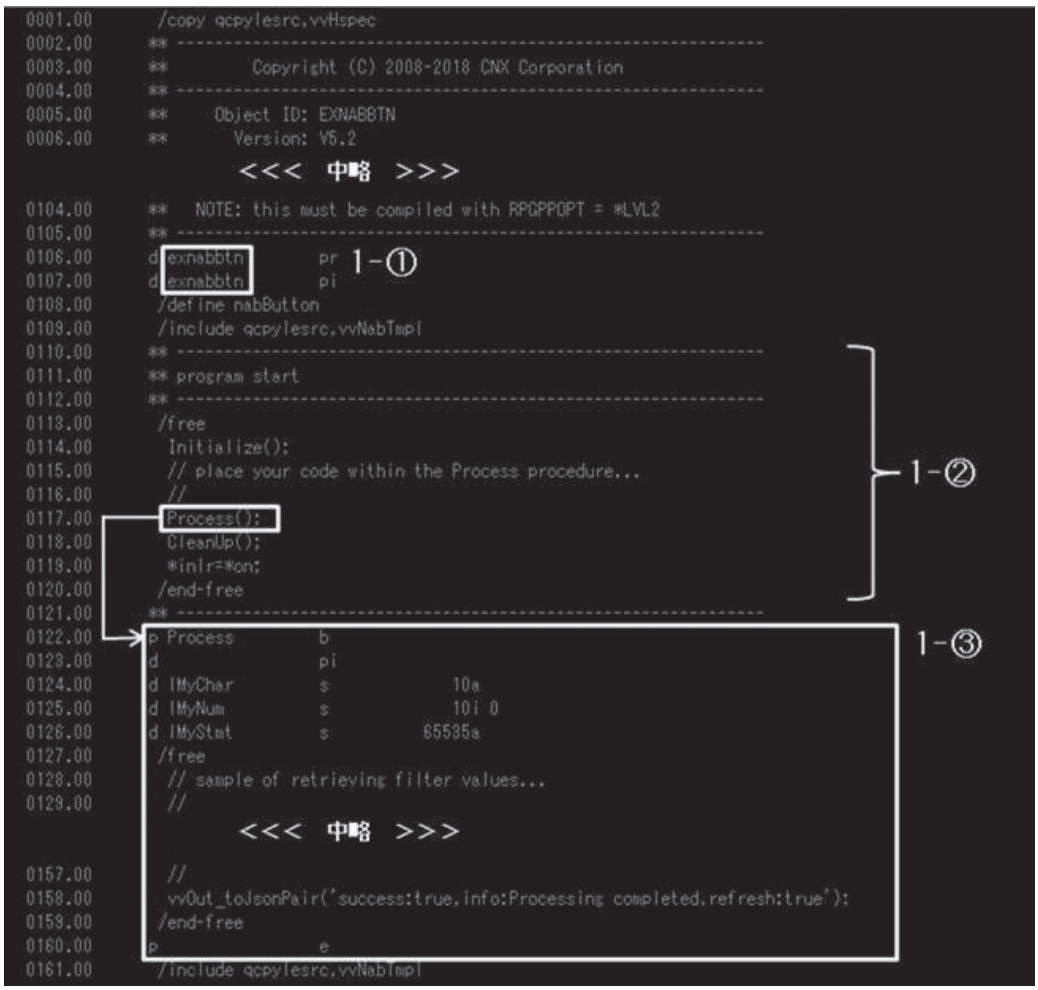
図11　ボタンクリック　サンプルプログラム

実行画面イメージ

重複エラーの場合、ポップアップメッセージを表示

登録が正常終了した場合、画面下部にお知らせメッセージ表示

ソース1　テンプレートプログラム(EXNABBTN)



```
0001.00      /copy qcpylesrc,vvHspec
0002.00     ** -------------------------------------------------------------
0003.00     **         Copyright (C) 2008-2018 CNX Corporation
0004.00     ** -------------------------------------------------------------
0005.00     **      Object ID: EXNABBTN
0006.00     **        Version: V6.2
                     <<< 中略 >>>
0104.00     **   NOTE: this must be compiled with RPGPPOPT = *LVL2
0105.00     ** -------------------------------------------------------------
0106.00     d exnabbtn        pr                  1-①
0107.00     d exnabbtn        pi
0108.00      /define nabButton
0109.00      /include qcpylesrc,vvNabTmpl
0110.00     ** -------------------------------------------------------------
0111.00     ** program start
0112.00     ** -------------------------------------------------------------
0113.00      /free
0114.00       Initialize();
0115.00       // place your code within the Process procedure...
0116.00       //
0117.00       Process();
0118.00       CleanUp();
0119.00       *inlr=*on;
0120.00      /end-free
0121.00     ** -------------------------------------------------------------
0122.00     p Process         b                   1-②
0123.00     d                 pi
0124.00     d lMyChar         s             10a
0125.00     d lMyNum          s             10i 0
0126.00     d lMyStat         s          65535a
0127.00      /free
0128.00       // sample of retrieving filter values...
0129.00       //
                     <<< 中略 >>>
0157.00       //
0158.00       vvOut_toJsonPair('success:true,info:Processing completed,refresh:true');
0159.00      /end-free
0160.00     p                 e                   1-③
0161.00      /include qcpylesrc,vvNabTmpl
```

ソース2　商品マスタメンテナンス新規登録(TEC010)

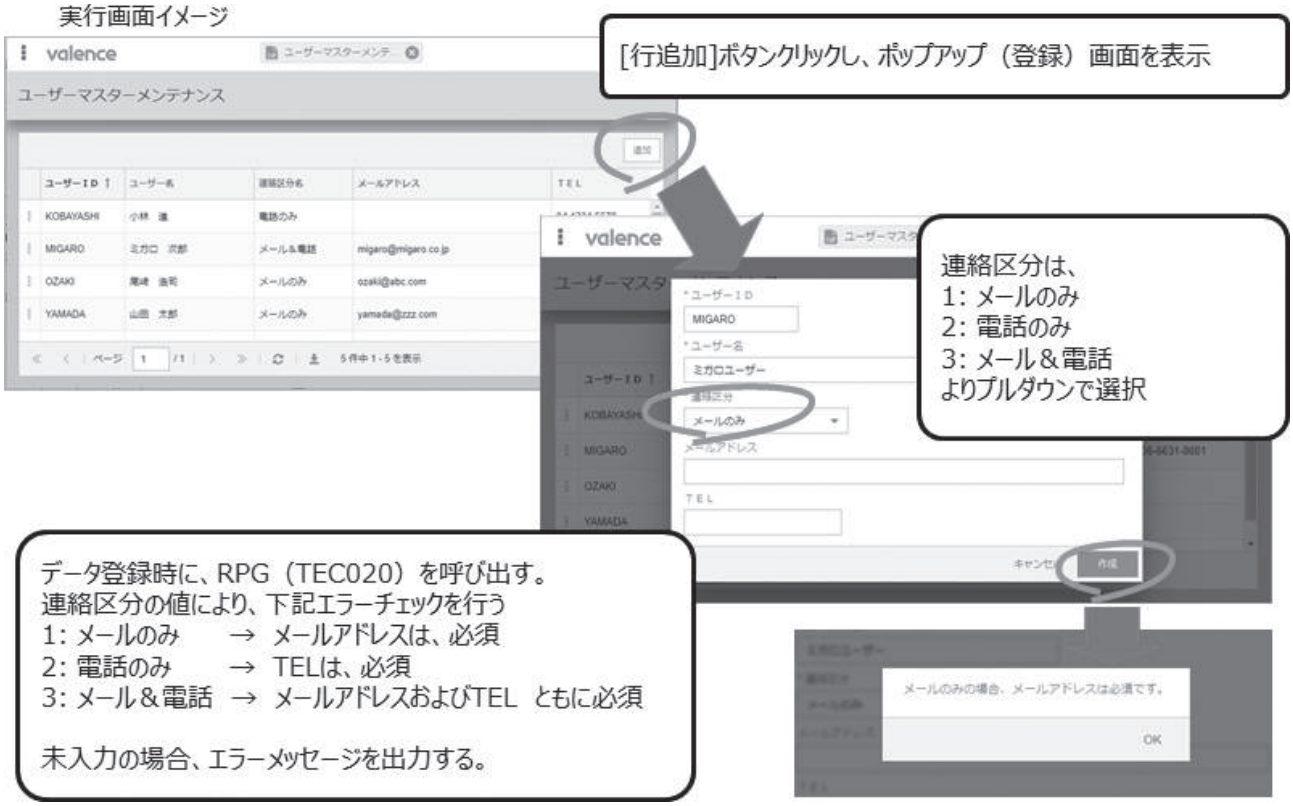
```
0001.00      /copy qcpylesrc,vvHspec
0002.00     ** ------------------------------------------------------------
0003.00     ** TEC010:商品マスタメンテナンス新規登録
0004.00     ** ------------------------------------------------------------
0005.00     F* ------------------------------------------------------------
0006.00     F* ファイル定義
0007.00     F* ------------------------------------------------------------
0008.00     F*<商品マスタ>
0009.00     FMSYOHIN   UF A E           K DISK
0010.00     F*
0011.00     d TEC010          pr
0012.00     d TEC010          pi
0013.00      /define nabButton
0014.00      /include qcpylesrc,vvNabTmpl
0015.00     ** ------------------------------------------------------------
0016.00     ** program start
0017.00     ** ------------------------------------------------------------
0018.00      /free
0019.00       Initialize();
0020.00       Process();
0021.00       CleanUp();
0022.00       *inlr=*on;
0023.00      /end-free
0024.00     ** ------------------------------------------------------------
0025.00     p Process         b
0026.00     d                 pi
0027.00     D VSYHNCD         S             10A
0028.00     D VSYHNNM         S             32A
0029.00     D VTANKA          S              9  0
0030.00     C*
0031.00      /free
0032.00       //フォーム上の値を取得
0033.00       VSYHNCD = GetFormChar('F1_SYHNCD');          //商品ＣＤ
0034.00       VSYHNNM = GetFormChar('F1_SYHNNM':'O');      //商品名
0035.00       VTANKA  = GetFormNum('F1_TANKA');            //単価
0036.00      /end-free
0037.00     C*
0038.00     C*-----キー重複チェック
0039.00     C     VSYHNCD       CHAIN     MSYOHIR                        81
0040.00     C     *IN81         IFEQ      *OFF
0041.00     C*
0042.00      /free
0043.00        //エラーメッセージを送信
0044.00        vvOut_toJsonPair('success:false,'
0045.00          + 'msg:入力した商品ＣＤは既に存在しています')
0046.00      /end-free
0047.00     C*
0048.00     C                   ELSE
0049.00     C*-----新規レコード登録
0050.00     C                   MOVEL     VSYHNCD       SYHNCD
0051.00     C                   MOVEL     VSYHNNM       SYHNNM
0052.00     C                   Z-ADD     VTANKA        TANKA
0053.00     C*
0054.00     C                   WRITE     MSYOHIR
0055.00     C*
0056.00      /free
0057.00        //正常終了メッセージを送信
0058.00        vvOut_toJsonPair('success:true,refresh:true,'
0059.00          + 'info:登録が終了しました');
0060.00      /end-free
0061.00     C*
0062.00     C                   END
0063.00     p                 e
0064.00      /include qcpylesrc,vvNabTmpl
```

2-①
2-②
2-③
2-④
2-⑤
2-⑥

図12　コンパイル時のオプション指定

図13　Edit Grid　サンプルプログラム

実行画面イメージ



ソース3-1　ユーザーマスタメンテナンス更新(TEC020)

```
0001.00      /copy qcpylesrc,vvHspec
0002.00     ** -------------------------------------------------------------
0003.00     ** TEC020:ユーザーマスタメンテナンス
0004.00     ** -------------------------------------------------------------
0005.00     F*-----------------------------
0006.00     F* ファイル定義
0007.00     F*-----------------------------
0008.00     F* <メッセージマスタ>
0009.00     FMMSGP     IF   E           K DISK
0010.00     F*
0011.00     d TEC020  3-① pr
0012.00     d  inMode                       10a
0013.00     d  inDataPtr                     *
0014.00     d  outStopProcess...
0015.00     d                                n
0016.00     d TEC020  3-① pi
0017.00     d  inMode                       10a
0018.00     d  inDataPtr                     *
0019.00     d  outStopProcess...
0020.00     d                                n
0021.00      /define nabValidation
0022.00      /include qcpylesrc,vvNabTmpl
0023.00     ** -------------------------------------------------------------
0024.00     ** program start
0025.00     ** -------------------------------------------------------------
0026.00      /free
0027.00       Initialize();
0028.00
0029.00       // perform validations based on the mode...
0030.00       //
0031.00       if inMode = 'ADD';
0032.00         ProcessAdd();
0033.00       elseif inMode = 'EDIT';
0034.00         ProcessEdit();
0035.00       elseif inMode = 'DELETE';
0036.00         ProcessDelete();
0037.00       elseif inMode = 'POSTADD';
0038.00         ProcessPostAdd();
0039.00       elseif inMode = 'POSTDELETE';
0040.00         ProcessPostDelete();
0041.00       elseif inMode = 'POSTEDIT';
0042.00         ProcessPostEdit();
0043.00       endif;
0044.00
0045.00       CleanUp();
0046.00       *inlr=*on;
0047.00      /end-free
0048.00     ** -------------------------------------------------------------
0049.00     p ProcessAdd      b
0050.00     d                 pi
0051.00     **
```

3-②

ソース3-2　ユーザーマスタメンテナンス更新(TEC020)

```
0052.00     D LUSRKBN         S              1A
0053.00     D LUSEMAL         S             40A
0054.00     D LUSTEL          S             14A
0055.00     D LERR            S              5A
0056.00     **                                                    3-④
0057.00      /free
0058.00       //----- 画面上の値を取得
0059.00       LUSRKBN = GetValue('MUSER':'USRKBN'):  //--連絡区分
0060.00       LUSEMAL = GetValue('MUSER':'USEMAL'):  //--メールアドレス
0061.00       LUSTEL  = GetValue('MUSER':'USTEL'):   //--ＴＥＬ
0062.00      /end-free
0063.00     C*-----連絡区分＝１（メールのみの場合）
0064.00     C     LUSRKBN       IFEQ      '1'
0065.00     C     LUSEMAL       IFEQ      *BLANK
0066.00     C                   MOVEL     'E0010'       LERR
0067.00     C                   ENDIF
0068.00     C                   ENDIF
0069.00     C*-----連絡区分＝２（電話のみの場合）
0070.00     C     LUSRKBN       IFEQ      '2'
0071.00     C     LUSTEL        IFEQ      *BLANK
0072.00     C                   MOVEL     'E0020'       LERR
0073.00     C                   ENDIF
0074.00     C                   ENDIF
0075.00     C*-----連絡区分＝３（メール＆電話の場合）
0076.00     C     LUSRKBN       IFEQ      '3'
0077.00     C     LUSEMAL       IFEQ      *BLANK
0078.00     C     LUSTEL        OREQ      *BLANK
0079.00     C                   MOVEL     'E0030'       LERR
0080.00     C                   ENDIF
0081.00     C                   ENDIF
0082.00     C*-----エラーメッセージ取得
0083.00     C     LERR          IFNE      *BLANK
0084.00     C     LERR          CHAIN     MMSGR                          31
0085.00     C     *IN31         IFEQ      *OFF
0086.00      /free
0087.00        //エラーメッセージを送信                    3-⑤
0088.00        SendError(MMSGTX);
0089.00      /end-free
0090.00     C                   ENDIF
0091.00     C                   ENDIF
0092.00     P                   e
0093.00     ** ------------------------------------------------------------
0094.00     p ProcessDelete   b

                <<<  中略  >>>

0125.00     p                   e
0126.00      /include qcpylesrc,vvNabTmpl
```

3-③

図14　Valence App Builder API一覧

1．クリック　（EXNABBTN）

I/O	API	概要
I	vvIn_char	処置パラメータの取得
I	GetFormChar	フィールド値取得(文字列)
I	GetFormNum	フィールド値取得(数値)
I	GetSelectionChar	選択したレコードのフィールド値取得(文字列)
I	GetSelectionNum	選択したレコードのフィールド値取得(数値)
I	GetFilterValue	フィルタ条件の取得
-	GetSqlStatement	関連するデータソースで実行されている SQL ステートメントの取得
O	vvOut_toJsonPair	ブラウザへ結果の返却
O	vvOut_data	ブラウザへ結果の返却(フィルターウィジェット実行)

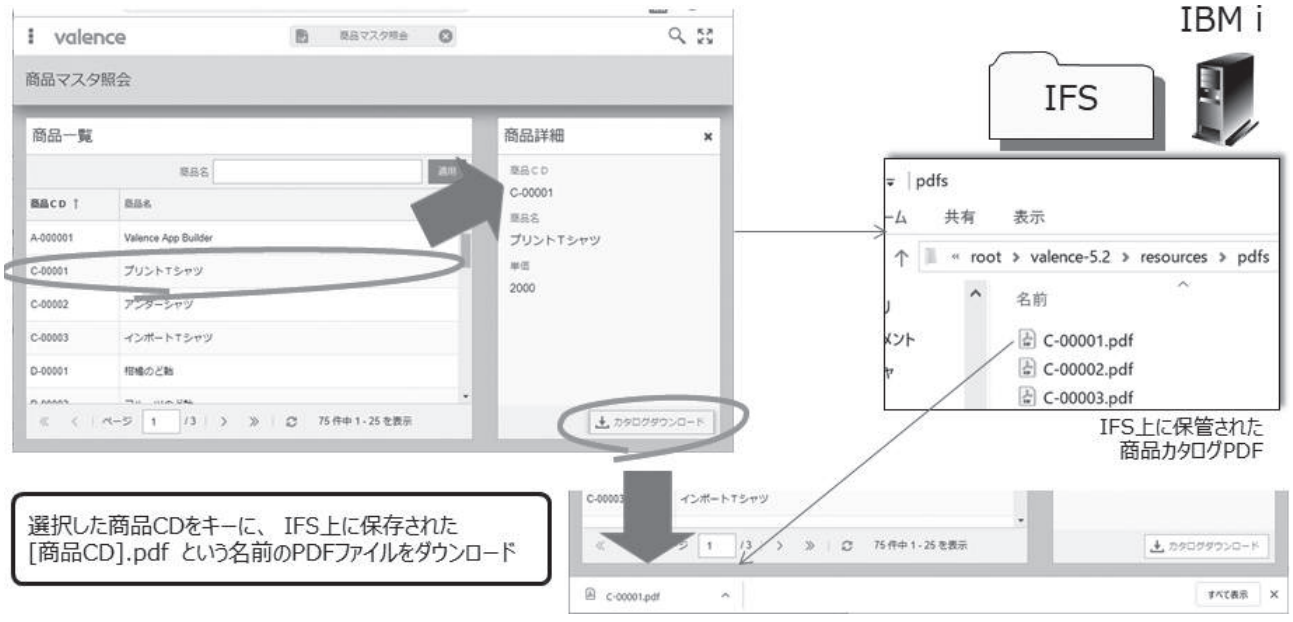
2．Edit Grid（EXNABVAL）

I/O	API	概要
I	GetValue	入力値の取得
I	GetFilterValue	フィルター条件の取得
I	IsChanged	指定されたファイル/フィールドが変更されているかどうかを判断
-	SetValue	ファイル名、フィールド名の引数で設定、対象の値を更新
O	SetDoNotProcess	指定されたファイルへの追加、更新、削除処理をスキップ
O	SendError	処理の中断とメッセージの表示

3．フィルタ（EXNABFLT）

I/O	API	概要
I	GetValue	引数のフィルタ値の取得
-	SetValue	引数に設定したフィールドに対してフィルタ値を設定
O	AddFilterString	指定されたフィルタフィールドにフィルタを追加
O	WriteAllFilters	すべてのフィルタフィールドに対してフィルタを追加
O	SendError	処理の中断とメッセージの表示

図15　PDFファイル　ダウンロード



ソース4　商品カタログPDFダウンロード(TEC030)

```
0001.00      /copy qcpylesrc,vvHspec
0002.00     ** ------------------------------------------------------------
0003.00     ** TEC030:PDFダウンロード
0004.00     ** ------------------------------------------------------------
0005.00     d TEC030          pr
0006.00     d TEC030          pi
0007.00      /define nabButton
0008.00      /include qcpylesrc,vvNabTmpl
0009.00     ** ------------------------------------------------------------
0010.00     ** program start
0011.00     ** ------------------------------------------------------------
0012.00      /free
0013.00       Initialize();
0014.00       Process();
0015.00       CleanUp();
0016.00       *inlr=*on;
0017.00      /end-free
0018.00     ** ------------------------------------------------------------
0019.00     p Process         b
0020.00     d                 pi
0021.00     D SYHNCD          S             10A
0022.00     D ROOTPATH        S             20A
0023.00     D PDFPATH         S             64A
0024.00     D FILENAME        S             14A
0025.00     D*
0026.00      /free
0027.00        //フォーム上の値を取得
0028.00        SYHNCD = GetFormChar('F1_SYHNCD');        //商品ＣＤ
0029.00
0030.00        //PDF保存先PATH取得                                          4-①
0031.00        ROOTPATH = vvUtility_getValenceSetting('ROOT_PATH');
0032.00        PDFPATH  = %trim(ROOTPATH) + 'resources/pdfs/';
0033.00
0034.00        //PDFファイル名取得
0035.00        FILENAME = %trim(SYHNCD) + '.pdf';
0036.00
0037.00        //PDFファイルの存在チェック
0038.00        if not vvIfs_pathExists(%trim(PDFPATH) + FILENAME);   4-②
0039.00          //ファイルが存在しない場合エラー
0040.00          vvOut_toJsonPair('success:false,msg:PDFが存在しません。');
0041.00        else;
0042.00          //PDFファイルダウンロード                                  4-③
0043.00          vvOut.download = '1';
0044.00          vvOut.file     = FILENAME;
0045.00          vvOut_file(%trim(PDFPATH) + FILENAME:vvOut);
0046.00        endif;
0047.00       /end-free
0048.00     p                 e
0049.00      /include qcpylesrc,vvNabTmpl
```

図16　ファイル返却時の処理

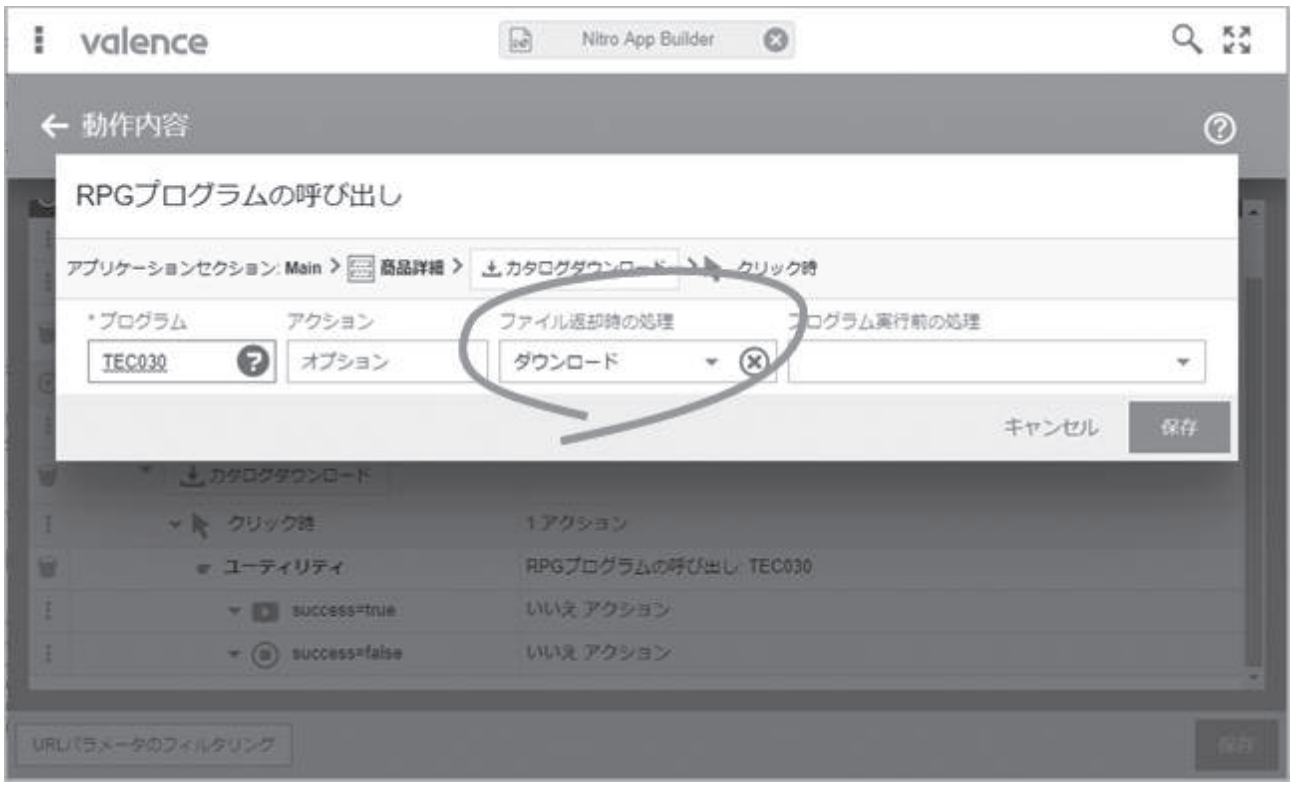
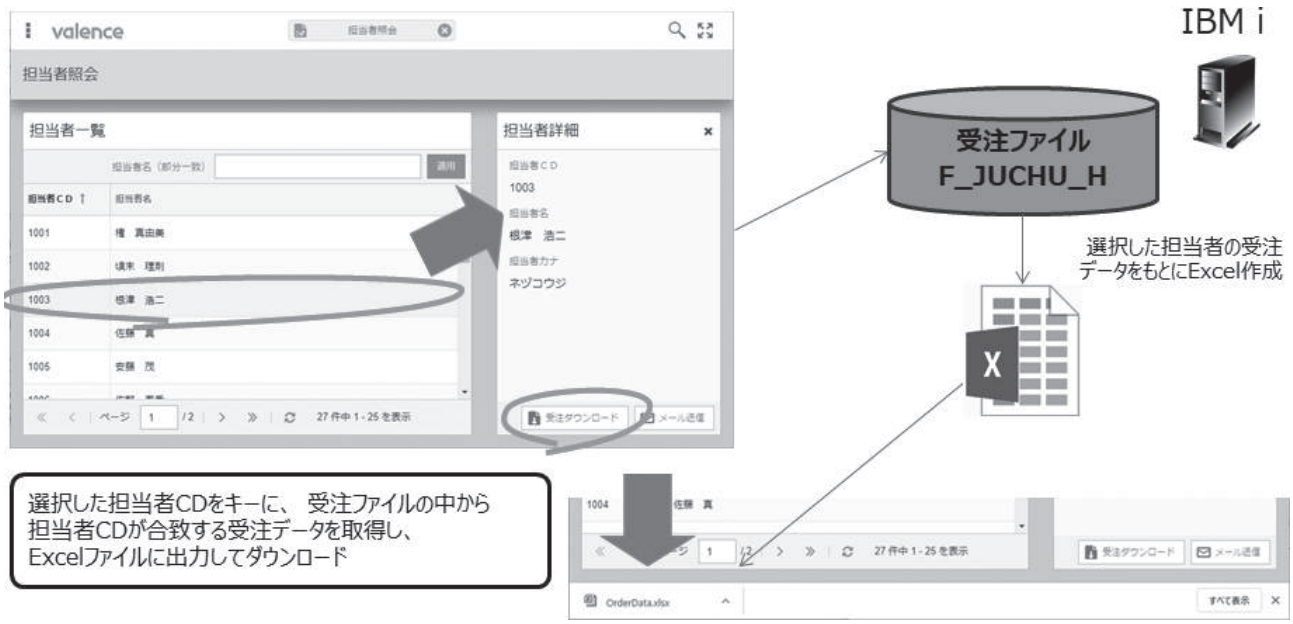


図17　動的に作成したExcelファイル　ダウンロード



ソース5　受注データExcelダウンロード（TEC040）

```
0001.00      /copy qcpylesrc,vvHspec
0002.00     ** ------------------------------------------------------------
0003.00     ** TEC040:受注データダウンロード
0004.00     ** ------------------------------------------------------------
0005.00     d TEC040          pr
0006.00     d TEC040          pi
0007.00      /define nabButton
0008.00      /include qcpylesrc,vvNabTmpl
0009.00     ** ------------------------------------------------------------
0010.00     ** program start
0011.00     ** ------------------------------------------------------------
0012.00      /free
0013.00       Initialize();
0014.00       Process();
0015.00       CleanUp();
0016.00       *inlr=*on;
0017.00      /end-free
0018.00     ** ------------------------------------------------------------
0019.00     p Process         b
0020.00     d                 pi
0021.00     D TANTOCD         S              4A
0022.00     D SQLSTR          S          32766A
0023.00     D*
0024.00      /free
0025.00        //フォーム上の値を取得
0026.00        TANTOCD = GetFormChar('F1_T_TANTO_CD');        //担当者ＣＤ
0027.00
0028.00        //SQL抽出条件
0029.00        SQLSTR = 'SELECT * FROM F_JUCHU_H'
0030.00               + ' WHERE H_TANTO_CD = ''' + TANTOCD + ''''
0031.00               + ' ORDER BY H_JUCHU_NO';
0032.00
0033.00        //SQL抽出結果をエクセルダウンロード
0034.00        vvOut.download = '1';
0035.00        vvOut.file     = 'OrderData.xlsx';
0036.00        vvOut_execSqlToSS(vvOut:SQLSTR);
0037.00      /end-free
0038.00     p                 e
0039.00      /include qcpylesrc,vvNabTmpl
```

5-①

5-②

図18　動的に作成したExcelファイル

OrderData.xlsx - Microsoft Excel

ファイル　ホーム　挿入　ページレイアウト　数式　データ　校閲　表示　開発

A1　H_JUCHU_NO

	A	B	C	D	E	F	G	H	I
1	H_JUCHU_NO	H_JUCHUBI	H_TANTO_CD	H_KKYAK_CD	H_JUCHUKIN	H_YOTEIBI	H_SYUKABI	H_SYUKABIN	H_SYUKA_NO
2	10,000,077	2018-03-26	1003	100020	4,408,230	2018-03-27	1900-01-01		
3	10,000,120	2018-04-21	1003	100006	780	2018-04-23	1900-01-01		
4	10,000,158	2018-07-22	1003	100017	106,800	2018-07-23	2018-07-25	郵便小包	1569143347
5	10,000,175	2018-06-06	1003	100012	1,445,500	2018-06-12	2018-06-06	FEDDEX	1656110686
6	10,000,183	2018-05-06	1003	100026	183,400	2018-05-08	1900-01-01		
7	10,000,233	2018-07-29	1003	100014	3,044,000	2018-08-05	2018-08-02	郵便小包	1925618472
8	10,000,270	2018-06-01	1003	100005	503,840	2018-06-07	2018-06-05	長谷川運輸	1645224477
9	10,000,300	2018-08-09	1003	100025	301,880	2018-08-10	1900-01-01		
10	10,000,302	2018-04-23	1003	100016	2,689,470	2018-04-27	2018-04-29	長谷川運輸	1506412851
11	10,000,346	2018-06-15	1003	100011	702,930	2018-06-20	2018-06-22	郵便小包	1606839667
12	10,000,381	2018-07-09	1003	100020	366,412	2018-07-12	2018-07-12	FEDDEX	1217493302
13	10,000,408	2018-05-12	1003	100009	1,692,200	2018-05-14	1900-01-01		
14	10,000,409	2018-08-06	1003	100025					
15	10,000,432	2018-05-15	1003	100008					
16	10,000,465	2018-07-15	1003	100020					
17	10,000,525	2018-04-01	1003	100026					
18	10,000,540	2018-04-19	1003	100011					
19	10,000,572	2018-06-20	1003	100005					
20	10,000,595	2018-03-21	1003	100006	623,600	2018-03-22	2018-03-24	用川急便	1217850569

SQLで抽出したフィールドがすべて出力
行タイトルは、フィールドIDがそのまま出力
されている

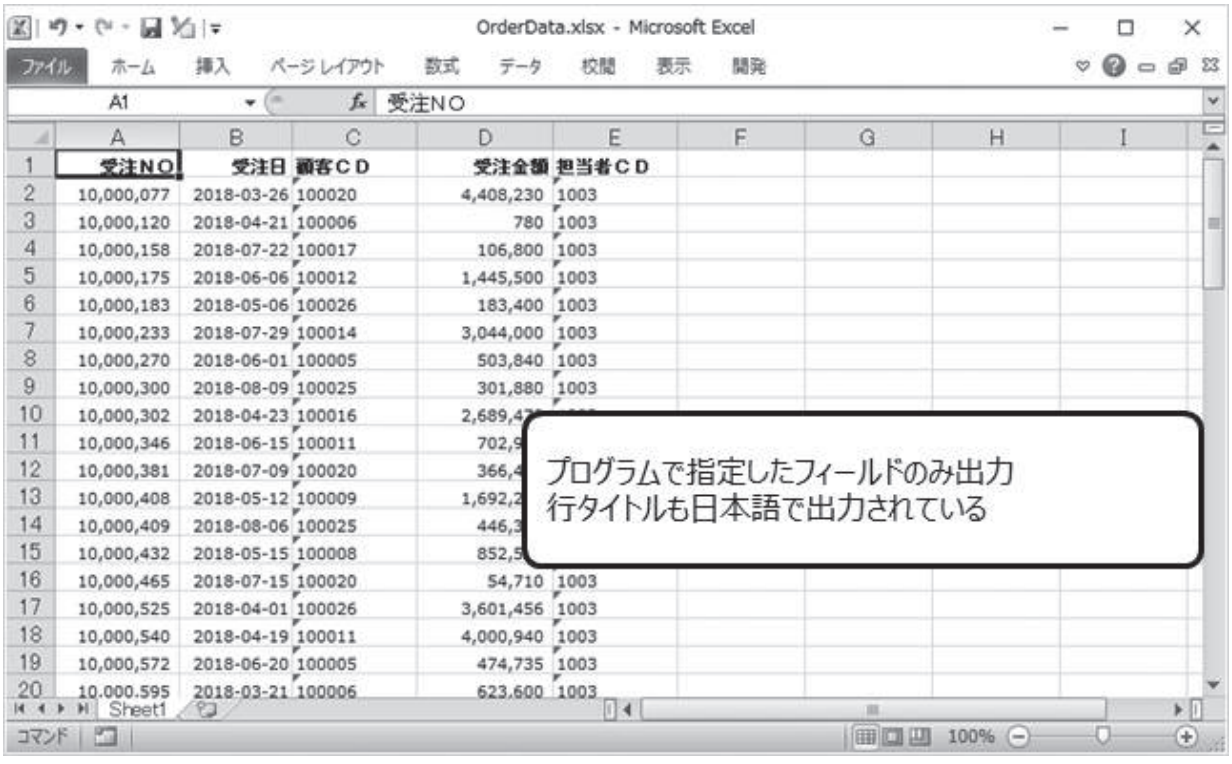
Sheet1

コマンド　100%

ソース6　Excelファイルの加工を追加（TEC040）

```
0018.00     **-------------------------------------------------------------
0019.00     p Process         b
0020.00     d                 pi
0021.00     D TANTOCD         S              4A
0022.00     D SQLSTR          S          32766A                    6-①
0023.00     D COL             DS                   LIKEDS(vvSSCol)
0024.00     D                                      DIM(5) INZ
0025.00     D*
0026.00      /free
0027.00        //フォーム上の値を取得
0028.00        TANTOCD = GetFormChar('F1_T_TANTO_CD');        //担当者ＣＤ
0029.00
0030.00        //SQL抽出条件
0031.00        SQLSTR = 'SELECT * FROM F_JUCHU_H '
0032.00               + ' WHERE H_TANTO_CD = ''' + TANTOCD + ''''
0033.00               + ' ORDER BY H_JUCHU_NO';
0034.00                                                          6-②
0035.00        //表示列および列タイトル指定
0036.00        COL(1).SQLName = 'H_JUCHU_NO';
0037.00        COL(1).heading = '受注ＮＯ';
0038.00        COL(2).SQLName = 'H_JUCHUBI';
0039.00        COL(2).heading = '受注日 ';
0040.00        COL(3).SQLName = 'H_KKYAK_CD';
0041.00        COL(3).heading = '顧客ＣＤ';
0042.00        COL(4).SQLName = 'H_JUCHUKIN';
0043.00        COL(4).heading = '受注金額';
0044.00        COL(5).SQLName = 'H_TANTO_CD';
0045.00        COL(5).heading = '担当者ＣＤ';
0046.00
0047.00        //SQL抽出結果をエクセルダウンロード
0048.00        vvOut.download = '1';
0049.00        vvOut.file     = 'OrderData.xlsx';                6-③
0050.00        vvOut_execSqlToSS(vvOut:SQLSTR:%addr(COL):%elem(COL));
0051.00      /end-free
0052.00     P                 e
0053.00      /include qcpylesrc,vvNabTmpl
```

図19　改良後に作成されたExcelファイル



受注NO	受注日	顧客ＣＤ	受注金額	担当者ＣＤ
10,000,077	2018-03-26	100020	4,408,230	1003
10,000,120	2018-04-21	100006	780	1003
10,000,158	2018-07-22	100017	106,800	1003
10,000,175	2018-06-06	100012	1,445,500	1003
10,000,183	2018-05-06	100026	183,400	1003
10,000,233	2018-07-29	100014	3,044,000	1003
10,000,270	2018-06-01	100005	503,840	1003
10,000,300	2018-08-09	100025	301,880	1003
10,000,302	2018-04-23	100016	2,689,47	
10,000,346	2018-06-15	100011	702,9	
10,000,381	2018-07-09	100020	366,4	
10,000,408	2018-05-12	100009	1,692,2	
10,000,409	2018-08-06	100025	446,3	
10,000,432	2018-05-15	100008	852,5	
10,000,465	2018-07-15	100020	54,710	1003
10,000,525	2018-04-01	100026	3,601,456	1003
10,000,540	2018-04-19	100011	4,000,940	1003
10,000,572	2018-06-20	100005	474,735	1003
10,000,595	2018-03-21	100006	623,600	1003

図20　メール送信

図21　RPGプログラム実行時に追加パラメータをセット

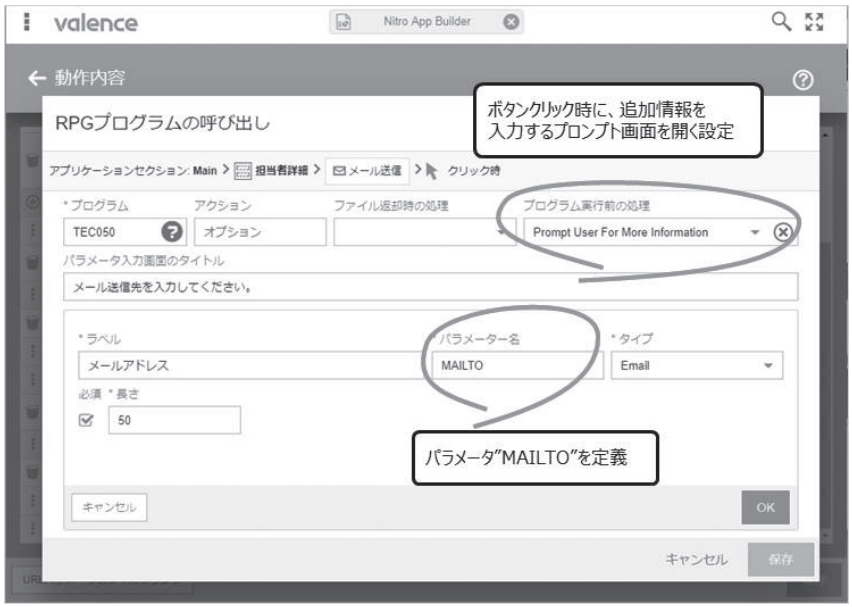
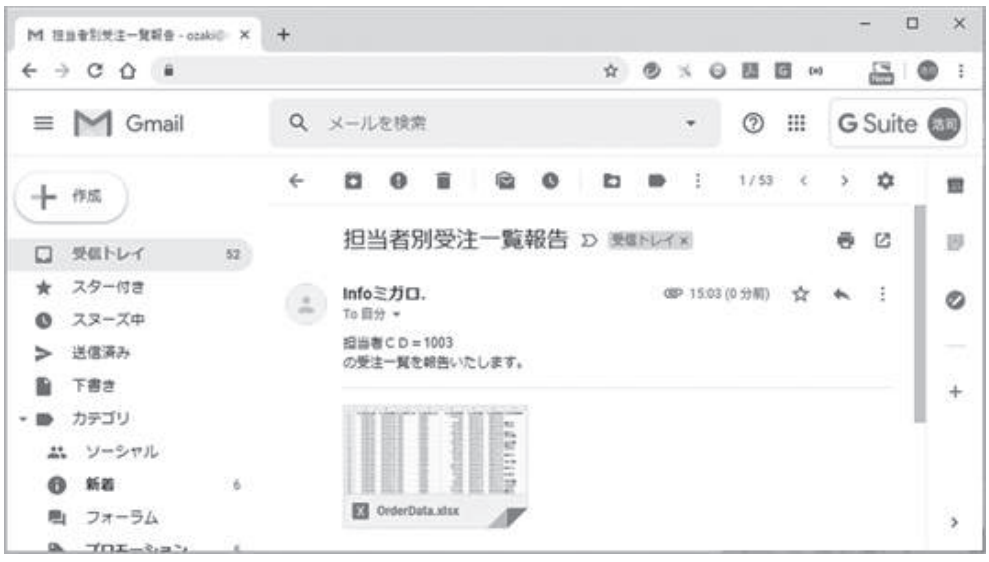


図22　メール受信結果



ソース7　メール送信(TEC050)

```
0001.00      /copy qcpylesrc,vvHspec
0002.00     ** ------------------------------------------------------------
0003.00     ** TEC050:メール送信
0004.00     ** ------------------------------------------------------------
0005.00     d TEC050          pr
0006.00     d TEC050          pi
0007.00      /define nabButton
0008.00      /define includeEMAIL                    7-①
0009.00      /include qcpylesrc,vvNabTmpl
0010.00     ** ------------------------------------------------------------
0011.00     ** program start
0012.00     ** ------------------------------------------------------------
0013.00      /free
0014.00       Initialize();
0015.00       Process();
0016.00       CleanUp();
0017.00       *inlr=*on;
0018.00      /end-free
0019.00     ** ------------------------------------------------------------
0020.00     p Process         b
0021.00     d                 pi
0022.00     D TANTOCD         S              4A
0023.00     D MAILTO          S             50A
0024.00     D TMPPATH         S             20A
0025.00     D SID             S             64A
0026.00     D SQLSTR          S          32766A
0027.00     D*
0028.00      /free
0029.00        //フォーム上の値を取得
0030.00        TANTOCD = GetFormChar('F1_T_TANTO_CD');        //担当者ＣＤ
0031.00        MAILTO  = vvIn_char('MAILTO');                 //送信アドレス   7-②
0032.00        SID     = vvIn_char('sid');                    //セッションID
0033.00
0034.00        //ファイル保存先取得
0035.00        TMPPATH = vvUtility_getValenceSetting('TEMP_PATH');
0036.00
0037.00        //SQL抽出条件
0038.00        SQLSTR = 'SELECT * FROM F_JUCHU_H'
0039.00               + ' WHERE H_TANTO_CD = ''' + TANTOCD + ''''
0040.00               + ' ORDER BY H_JUCHU_NO';
0041.00
0042.00        //SQL抽出結果をIFS上に出力                                  7-③
0043.00        vvOut.download = 'F';
0044.00        vvOut.file     = %trim(TMPPATH) + SID + '.xlsx';
0045.00        vvOut_execSqlToSS(vvOut:SQLSTR);
0046.00
0047.00        //メール内容作成                                            7-④
0048.00        vvMail.from        = 'info@migaro.co.jp';
0049.00        vvMail.to          = MAILTO;
0050.00        vvMail.subject     = '担当者別受注一覧報告';
0051.00        vvMail.body        = '担当者ＣＤ=' + TANTOCD + '<br>'
0052.00                           + 'の受注一覧を報告いたします。';
0053.00        vvMail.attachment  = vvOut.file;
0054.00        vvMail.attachAlias = 'OrderData.xlsx';
0055.00
0056.00        //メール送信                                                7-⑤
0057.00        if not vvMail_send(vvMail);
0058.00          vvOut_toJsonPair('success:false,msg:送信エラー');
0059.00        else;
0060.00          vvOut_toJsonPair('success:true,info:送信しました');
0061.00        endif;
0062.00
0063.00        //出力ファイルを削除                                        7-⑥
0064.00        vvIfs_deleteFile(vvOut.file);
0065.00
0066.00      /end-free
0067.00     p                 e
0068.00      /include qcpylesrc,vvNabTmpl
```

Migaro. Technical Report
既刊号バックナンバー

電子版・書籍(紙)媒体で提供中!
http://www.migaro.co.jp/contents/support/technical_report/

No.1　2008 年秋

お客様受賞論文

●最優秀賞

直感的に理解できるシステムを目指して―情報の“見える化”の取り組み
石井 裕昭様／豊鋼材工業株式会社

●ゴールド賞

運用部間にサプライズをもたらした Delphi/400
春木 治様／株式会社ロゴスコーポレーション

●シルバー賞

JACi400 使用による Web アプリケーション開発工数削減
中富 俊典様／日本梱包運輸倉庫株式会社

Delphi/400 を利用した Web 受注システム
飯田 豊様／東洋佐々木ガラス株式会社

●優秀賞

Delphi/400 による販売管理システム（FAINS）について
藤田 建作様／株式会社船井総合研究所

技研化成の新基幹システム再構築
藤田 健治様／技研化成株式会社

SE 論文

はじめての Delphi/400 プログラミング
畑中 侑／システム事業部 システム２課

Delphi/400 と Excel との連携
中嶋 祥子／ RAD 事業部 技術支援課

連携で広がる Delphi/400 活用術
尾崎 浩司／システム事業部 システム２課

フォーム継承による効率向上開発手法
吉原 泰介／ RAD 事業部 技術支援課

API を利用した出力待ち行列情報の取得方法
鶴巣 博行／ RAD 事業部 技術支援課

Delphi テクニカルエッセンス Q&A 集
吉原 泰介／ RAD 事業部 技術支援課

JACi400 を使って RPG で Web 画面を制御する方法
松尾 悦郎／システム事業部 システム２課

あなたはプラインドタッチができますか？
福井 和彦／システム事業部 システム１課

No.2　2009 年秋

お客様受賞論文

●最優秀賞

JACi400 で 既存 Web サービスの内製化を実現
佐々木 仁志様／株式会社ジャストオートリーシング

●ゴールド賞

.NET 環境での Delphi/400 の活用
福田 祐之様／林兼コンピューター株式会社

●シルバー賞

5250 で動作する「中古車 在庫照会プログラム」の GUI 化
佐久間 雄様／株式会社ケーユー

●優秀賞

Delphi による 輸入システム「MISYS」の再構築
秦 榮禧様／株式会社モトックス

Delphi/400 による 物流システムの再構築
仲井 学様／西川リビング株式会社

Delphi/400 で開発し 3 台のオフコンを 1 台の IBM i へ統合
島根 英行様／シルフ

SE 論文

JACi400 環境でマッシュアップ！
岩田 真和／ RAD 事業部 技術支援課

Delphi/400 を利用したはじめての Web 開発
福岡 浩行／システム事業部 システム２課

Delphi/400 を使用した Web サービスアプリケーション
尾崎 浩司／システム事業部 システム３課

Delphi/400 によるネイティブ資産の応用活用
吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

RPG でパフォーマンスを制御
松尾 悦郎／システム事業部 システム１課

MKS Integrity を利用したシステム開発
宮坂 優大・田村 洋一郎／システム事業部 システム１課

No.3　2010 年秋

お客様受賞論文

●最優秀賞

建物のクレーム情報管理システム「アフターサービス DB」について

大橋 良之様／東レ建設株式会社

●ゴールド賞

Delphi/400 で「写真管理ソフト」と「スプールファイルの PDF 化ソフト」を自社開発

寒河江 幸喜様／日綜産業株式会社

●シルバー賞

Delphi/400 で鉄鋼受発注業務を統一し 鉄鋼 EDI も実現

柿本 直樹様／合鐵産業株式会社

●優秀賞

Delphi/400 で EIS（Executive Information System）の高速化

小島 栄一様／西川計測株式会社

イントラでの PHP-Delphi-RPG 連携

仲井 学様／西川リビング株式会社

Delphi/400 を使った取引先管理システム

大崎 貴昭様／森定興商株式会社

SE 論文

Delphi/400 ローカルキャッシュ活用術

中嶋 祥子／ RAD 事業部 技術支援課

Delphi/400 帳票開発ノウハウ公開

尾崎 浩司／システム事業部 システム３課

Delphi/400 でドラッグ&ドロップを制御

辻林 涼子／システム事業部 システム２課

Delphi/400 のモジュールバージョン管理手法

前田 和寛／システム事業部 システム２課

Delphi/400　Web からの PDF 出力

福井 和彦・清水 孝将／システム事業部システム３課・システム２課

Delphi/400 で Flash 動画の実装

吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

No.4　2011 年秋［創立 20 周年記念号］

お客様受賞論文

●最優秀賞

全社の経費処理業務を効率化した「e 総務システム」

鈴木 英明様／阪和興業株式会社

●ゴールド賞

「Web 進捗管理システム」でリアルタイム性を実現

堀内 一弘様／エスケーロジ株式会社

●シルバー賞

「営業奨励金申請書」をたった２日間で開発

簑島 宏明様／株式会社ケーユーホールディングス

液体輸送における「配車支援システム」の構築

桂 哲様／ライオン流通サービス株式会社

SE 論文

グラフ活用リファレンス

中嶋 祥子／ RAD 事業部 技術支援課

Web サービスを利用して機能 UP ！

福井 和彦・畑中 侑／システム事業部 システム２課

OpenOffice 実践活用

吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

VCL for the Web 活用 TIPS 紹介

尾崎 浩司／システム事業部 プロジェクト推進室

JC/400 で JavaScript 活用

清水 孝将／システム事業部 システム１課

jQuery 連携で機能拡張

國元 祐二／ RAD 事業部 技術支援課 顧客サポート

No.5　2012 年秋［創刊 5 周年記念］

お客様受賞論文

【部門 1】

●最優秀賞

JC/400 による取引先との Web-EDI システム構築

久保田 佳裕様／極東産機株式会社

●ゴールド賞

Delphi と Excel を使用した帳票コストの削減

大久保 治高様／合鐵産業株式会社

もっと見やすく、もっと使いやすい画面を

新谷 直正様／株式会社アダル

【部門 2】

●優秀賞

Delphi/400 で確認業務の効率化

為国 順子様／ベネトンジャパン株式会社

取引先申請システムでの稟議書作成ワークフロー

大崎 貴昭様／森定興商株式会社

Delphi/400 で IBM i のストアードプロシージャを利用し、SQL 処理を高速化

島根 英行様／シルフ

SE 論文

InstallAware を使った Delphi/400 運用環境の構築

中嶋 祥子／ RAD 事業部 技術支援課 顧客サポート

カスタマイズコンポーネント入門 Delphi/400 開発効率向上

前田 和寛／システム事業部 システム 2 課

Delphi/400 スマートデバイスアプリケーション開発

吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

DataSnap を使用した 3 層アプリケーション構築技法

尾崎 浩司／システム事業部 プロジェクト推進室

JC/400 でポップアップウィンドウの制御&活用ノウハウ

清水 孝将・伊地知 聖貴／システム事業部 システム 1 課

【創刊 5 周年記念】

ミガロ.SE 座談会―お客様と共に歩む、お客様への熱い思い

No.6　2013 年秋

お客様受賞論文

【部門 1】

●最優秀賞

自社用開発フレームワークの構築

駒田 純也様／ユサコ株式会社

●ゴールド賞

Delphi/400 で CTI 開発および関連機能組み込み

仲井 正人様／株式会社スマイル・ジャパン

●シルバー賞

IBM WebFacing から JC/400 への移行・リニューアル手法

八木 秀樹様／極東産機株式会社

Delphi/400 と Delphi を利用した IBM i 資源の有効活用

小山 祐二様／澁谷工業株式会社

発注システムを VB から Delphi へ移植しリニューアル

川島 寛様／株式会社タツミヤ

【部門 2】

●優秀賞

5250 画面を使用せずに AS/400 スプールファイルをコントロールする

白井 昌哉様／太陽セメント工業株式会社

Delphi/400 を利用した 承認フロー導入による IT 内部統制構築

塚本 圭一様／ライオン流通サービス株式会社

SE 論文

FastReport を使用した帳票作成入門

尾崎 浩司／ RAD 事業部 営業推進課

Delphi/400 で開発する 64bit アプリケーション

吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

Web コンポーネントのカスタマイズ入門

佐田 雄一／システム事業部 システム 1 課

Indy を利用したメール送信機能開発

辻野 健・前坂 誠二／システム事業部 システム 2 課

Windows テキストファイル操作ノウハウ

小杉 智昭／システム事業部 プロジェクト推進室

JC/400 Web アプリケーションのユーザー管理・メニュー管理活用術

吉原 泰介・國元 裕二／ RAD 事業部 技術支援課 顧客サポート

No.7　2014 年秋

お客様受賞論文

【部門 1】

●最優秀賞

Delphi/400 による生産スケジューラの再構築

柿村 実様／東洋佐々木ガラス株式会社

●ゴールド賞

Delphi/400 および Delphi を利用したオンライン個人別メニューの構築

小山 祐二様／澁谷工業株式会社

●シルバー賞

IBM i と Delphi/400 のコラボレーション

新谷 直正様／株式会社アダル

●シルバー賞

荷札発行システムリプレースについて

仲井 学様／西川リビング株式会社

【部門 2】

●優秀賞

Delphi/400 バージョンアップのためのクライアント環境構築

普入 弘様／株式会社エイエステクノロジー

●優秀賞

外出先からメールでリアルタイム在庫を問い合せ

島根 英行様／シルフ

SE 論文

iOS/Android ネイティブアプリケーション入門

吉原 泰介／ RAD 事業部 技術支援課

ファイル加工プログラミングテクニック

小杉 智昭／システム事業部 プロジェクト推進室

FastReport を使用した帳票作成テクニック

前坂 誠二／システム事業部

大量データ処理テクニック

佐田 雄一／システム事業部

スマートデバイス WEB アプリケーション入門

尾崎 浩司／ RAD 事業部 営業推進課

國元 祐二／ RAD 事業部 技術支援課

No.8　2015 年秋

お客様受賞論文

【部門 1】

●最優秀賞

iPod Touch の業務利用開発と検証

石井 裕昭様／豊鋼材工業株式会社

●ゴールド賞

ブランク加工図管理システムの構築

小山 祐二様／澁谷工業株式会社

●シルバー賞

Delphi/400 でスプールファイル管理（WRKSPLF コマンドの活用）

三好 誠様／ユサコ株式会社

●シルバー賞

予算管理システムの構築

川島 寛様／株式会社タツミヤ

●シルバー賞

送状データ送信システムの Web 化について

仲井 学様／西川リビング株式会社

【部門 2】

●優秀賞

繰り返し DB 参照時の ClientDataSet の First 機能について

牛嶋 信之様／株式会社佐賀鉄工所

●優秀賞

IBM i のカレンダーを基準に他のシステムを稼働

福島 利昭様／株式会社ランドコンピュータ

SE 論文

フレームを利用した開発手法

前坂 誠二／システム事業部 システム 2 課

Windows タブレット用にカスタムソフトウェアキーボードを実装

福井 和彦／システム事業部 プロジェクト推進室

マルチスレッドを使用したレスポンスタイム向上

尾崎 浩司／ RAD 事業部 営業・営業推進課

Android アプリケーションの NFC 機能活用

吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

スマートデバイス開発で役立つ画面拡張テクニック

國元 祐二／ RAD 事業部 技術支援課 顧客サポート

No.9　2016 年秋

お客様受賞論文

【部門 1】

●最優秀賞

IBM i の見える化で実現するアジャイル開発

吉岡 延泰様／日本調理機株式会社社

●ゴールド賞

Windows Like 5250 への道のり

小山 祐二様／澁谷工業株式会社

●シルバー賞

Delphi プログラム管理ソフトの開発

牛嶋 信之様／株式会社佐賀鉄工所

【部門 2】

●優秀賞

Delphi/400 を利用した定型業務の PDF 化

佐藤 岳様／ライオン流通サービス株式会社

●優秀賞

ちょい足しモバイル

仲井 正人様／株式会社スマイル・ジャパン

●優秀賞

AS/400 の受注データを Web で社員に公開

福島 利昭様／株式会社ランドコンピュータ

SE 論文

iOS モバイルアプリ開発のデザイニングテクニック

前坂 誠二／システム事業部 システム 2 課

新データベースエンジン FireDAC を使ってみよう！

福井 和彦／システム事業部 プロジェクト推進室

Delphi/400 最新プログラム文法の活用法

尾崎 浩司／ RAD 事業部 営業・営業推進課

FastReport を活用した電子帳票作成テクニック

宮坂 優大／システム事業部 システム 1 課

Beacon 技術による IoT 活用の第一歩

吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

Web & ハイブリッドアプリ開発で役立つ IBM i & ブラウザデバッグテクニック

國元 祐二／ RAD 事業部 技術支援課 顧客サポート

No.10　2017 年秋

Migaro.Technical Report 創刊 10 周年記念

パートナー様からの祝辞

武藤 和博様／日本アイ・ビー・エム株式会社

藤井 等様／エンバカデロ・テクノロジーズ日本法人

Serge Charbit 様／ SystemObjects Corporation

飯田 恭子様／アイマガジン株式会社

お客様からの祝辞・お客様の声

石井 裕昭様／豊鋼材工業株式会社

牛嶋 信之様／株式会社佐賀鉄工所

大崎 貴昭様／森定興商株式会社

川島 寛様／株式会社タツミヤ

久保田 佳裕様／極東産機株式会社

駒田 純也様／ユサコ株式会社

小山 祐二様／澁谷工業株式会社

寒河江 幸喜様／日綜産業株式会社

佐々木 仁志様／株式会社ジャストオートリーシング

佐藤 岳様／ライオン流通サービス株式会社

白井 昌哉様／太陽エコブロックス株式会社

仲井 学様／西川リビング株式会社

福島 利昭様／株式会社ランドコンピュータ

お客様座談会

石井 裕昭様／豊鋼材工業株式会社

駒田 純也様／ユサコ株式会社

寒河江 幸喜様／日綜産業株式会社

仲井 学様／西川リビング株式会社

上甲 將隆／株式会社ミガロ.

司会　飯田 恭子様／アイマガジン株式会社

お客様受賞論文

【部門 1】

●最優秀賞

貸金庫と鍵のマッチング業務を Delphi/400 で実現
―文字認識データと基幹システムデータを統合

佐藤 正様／株式会社富士精工本社

●ゴールド賞

Windows タブレット導入による工作部材料受入業務改革

小山 祐二様／澁谷工業株式会社

【部門 2】

●優秀賞

Delphi/400 を利用した各拠点 PING コマンド簡素化

松垣 秀昭様／ライオン流通サービス株式会社

汎用的な帳票出力画面

牛嶋 信之様／株式会社佐賀鉄工所

バーコードリーダー読み取り後、次の入力位置にカーソルを自動遷移させる技術

上総 龍央様／キョーラクシステムクリエート株式会社

IBM i のスプールファイル参照機能を Web で構築

福島 利昭様／株式会社ランドコンピュータ

SE 論文

デスクトップアプリケーション開発でも
役立つ FireMonkey 活用入門

尾崎 浩司／ RAD 事業部 営業・営業推進課

Delphi/400 バージョンアップに伴う文字コードの違いと制御

宮坂 優大／システム事業部 システム 1 課

FastReport への効率的な帳票レイアウトコンバート

畑中 侑／システム事業部 システム 2 課

IBM i トリガー機能を活かしたセキュリティログ対応

八木沼 幸一／システム事業部 プロジェクト推進室

アプリケーションテザリングを利用した PC &モバイルアプリケーション連携

吉原 泰介／ RAD 事業部 技術支援課 顧客サポート

SmartPad4i の運用で役立つ WEB サーバー機能

國元 祐二／ RAD 事業部 技術支援課 顧客サポート

No.11　2018 年秋

お客様受賞論文

【部門 1】

●最優秀賞

Excel テンプレートを使用した帳票出力機能の開発

駒田 純也 様／ユサコ株式会社

●ゴールド賞

SP4i の活用による製品検査チェックシステムの構築

八木 秀樹 様／極東産機株式会社

【部門 2】

●優秀賞

配車支援システムを Delphi/400 で再構築

村上 稔明 様／ライオン流通サービス株式会社

一般シール受注入力業務の Delphi/400 化

寺西 健一 様／大阪シーリング印刷株式会社

Delphi/400 による無線ハンディターミナルのデータ集約の仕組みの実装

寺西 健一 様／大阪シーリング印刷株式会社

SE 論文

OLE を利用した Excel 出力のパフォーマンス向上手法

薬師 尚之／システム事業部 システム 2 課

FireDAC 実践プログラミングテクニック

佐田 雄一／システム事業部 システム 1 課

REST による Web サービスを活用した機能拡張テクニック

尾崎 浩司／ RAD 事業部 営業・営業推進課

Google Maps Platform を使用したアプリケーション開発テクニック

福井 和彦・小杉 智昭／システム事業部 プロジェクト推進室

RAD Server を使った新しい多層アプリケーション構築

吉原 泰介／ RAD 事業部 技術支援課

JC/400 から SP4i へのマイグレーションノウハウ

吉原 泰介・國元 祐二／ RAD 事業部 技術支援課

MEMO

MEMO

MEMO

MIGARO.
TECHNICAL
REPORT

Migaro.Technical Report
No.12　2019 年秋
ミガロ.テクニカルレポート

2019 年 12 月 1 日 初版発行

◆発行
株式会社ミガロ.
〒 556-0017
大阪府大阪市浪速区湊町 2-1-57 難波サンケイビル 13F
TEL：06（6631）8601　FAX：06（6631）8603
http://www.migaro.co.jp/

◆発行人
上甲 將隆

◆編集協力
アイマガジン株式会社

◆デザインフォーマット
近江デザイン事務所

MIGARO. TECHNICAL REPORT

ミガロ.テクニカルレポート

株式会社ミガロ.

http://www.migaro.co.jp/

本社
〒556-0017
大阪市浪速区湊町2-1-57
難波サンケイビル 13F
TEL:06(6631)8601
FAX:06(6631)8603

東京事業所
〒100-0013
東京都千代田区霞が関3-7-1
霞が関東急ビル 2F
TEL:03(5510)5701
FAX:03(5510)5702