モバイルアプリケーション
開発ガイド
Magic xpa

OUTPERFORM THE FUTURE™

本マニュアルに記載の内容は、将来予告なしに変更することがあります。これらの情報について MSE（Magic Software Enterprises Ltd.）および MSJ（Magic Software Japan K.K.）は、いかなる責任も負いません。

本マニュアルの内容につきましては、万全を期して作成していますが、万一誤りや不正確な記述があったとしても、MSE および MSJ はいかなる責任、債務も負いません。

MSE および MSJ は、この製品の商業価値や特定の用途に対する適合性の保証を含め、この製品に関する明示的、あるいは黙示的な保証は一切していません。

本マニュアルに記載のソフトウェアは、製品の使用許諾契約書に記載の条件に同意をされたライセンス所有者に対してのみ供給されるものです。同ライセンスの許可する条件のもとでのみ、使用または複製することが許されます。

当該ライセンスが特に許可している場合を除いては、いかなる媒体へも複製することはできません。ライセンス所有者自身の個人使用目的で行う場合を除き、MSE または MSJ の書面による事前の許可なしでは、いかなる条件下でも、本マニュアルのいかなる部分も、電子的、機械的、撮影、録音、その他のいかなる手段によっても、コピー、検索システムへの記憶、電送を行うことはできません。

サードパーティ各社商標の引用は、MSE および MSJ の製品に対するコンパチビリティに関しての情報提供のみを目的としてなされるものです。

本マニュアルにおいて、説明のためにサンプルとして引用されている会社名、製品名、住所、人物は、特に断り書きのないかぎり、すべて架空のものであり、実在のものについて言及するものではありません。

Magic は Magic Software Japan K.K. の登録商標です。
Magic xpa は Magic Software Enterprises Ltd. のイスラエルその他の国での商標または登録商標です。
Magic xpa Enterprise Studio、Magic xpa Enterprise Client、Magic xpa Enterprise Server および Magic xpa RIA Server は Magic Software Japan K.K. の商標です。

一般に、会社名、製品名は各社の商標または登録商標です。
MSE および MSJ は、本製品の使用またはその使用によってもたらされる結果に関する保証や告知は一切していません。この製品のもたらす結果およびパフォーマンスに関する危険性は、すべてユーザが責任を負うものとします。

この製品を使用した結果、または使用不可能な結果生じた間接的、偶発的、副次的な損害（営利損失、業務中断、業務情報の損失などの損害も含む）に関し、事前に損害の可能性が勧告されていた場合であっても、MSE および MSJ、その管理者、役員、従業員、代理人は、いかなる場合にも一切責任を負いません。

2023 年 4 月 14 日

はじめに

Android™ や iOS™ 用の Magic RIA クライアントは、各デバイスのネイティブな OS 用のアプリケーションです。そして、Magic RIA のクライアントプロトコルを実装しています。異なるモバイル用デバイスの Magic RIA クライアントを使用することで、様々なモバイルデバイス上で実行する業務用の高度でインタラクティブな RIA アプリケーションを実行させることができます。

Magic xpa を使用したモバイル RIA アプリケーションの開発は、Windows の RIA アプリケーションを開発する場合と同じような技術を必要とします。RIA の概念を理解するには、『RIA チュートリアル』を参照してください。

さらに、モバイルアプリケーションを動作させるには、オフラインで動作する機能が重要なため、『オフラインアプリケーションの開発ガイド』も一読することを推奨します。

本書は、Magic xpa の Ver4.6 を前提に記述されています。説明文によっては、Ver4.6 以前では対応していない項目があります。

開発者向けのガイドライン

デバイスの機能やユーザインタフェースが Windows PC と異なるため、アプリケーションの画面設計やユーザインタフェースの設計の際に、考慮が必要になる場合があります。

例えばカメラや GPS、セキュリティ関連の機能など、様々です。このドキュメントは、開発者が各デバイスの機能をフルに活用できるようにサポートするためのものです。

デバイスの特徴

- **画面サイズと向き** …… モバイル用デバイスは、縦方向と横方向のどちらにおいても様々な解像度や画面サイズがあります。
- **キーボード** …… モバイル用デバイスによっては、フル規格のキーボードを備えているものもあります。キーボードには、さらに専用のメニューキーや ESC キー、トラックパッドやトラックボールなどデスクトップ PC の矢印キーと同じような装置を備えているものもあります。トラックパッドはまた、押下されると専用の機能が割り当てられていることもあります。キーボードのみのデバイスは、一定の方向の画面で固定されており、画面の向きを変更させることはできません。
- **タッチデバイス** …… ほとんどのモバイル用デバイスはタッチスクリーンを備えています。キーボードを備えているものや、備えていないものなど様々です。キーボードが利用できない場合は、タッチデバイスは画面の向きを変更させることができ、画面上に仮想キーボードを提供します。

ユーザ操作

- **ウィンドウのモデル**…… モバイル用デバイスは、ウィンドウを単純に重ねて表示するウインドウ・モデルをサポートします。各アプリケーションは複数のウィンドウを開くことができます。しかし、新規に開くウィンドウは前のウィンドウ上に重なるため、実質的にはモーダルとなります。マウスポインタが存在せず、ウィンドウをエンドユーザが操作（移動したりサイズを変更したり）することはできません。アプリケーションが実行されると、そのメインウインドウ（その後に開くウィンドウを含め）は、デバイスのウィンドウを占有します。
- **トラックパッドを使用したフォーム操作** …… 一般的に、トラックパッドはフォーム上で項目間を移動するために使用されます。Windows PC のキーボードと異なり、Tab キーがないため、項目間を移動するための標準的なキーが存在しません。項目間や（[エディット] コントロールの）項目内での操作は、トラックパッドによる方向性のあるアクションを使用して行われます。
- **タッチキーボードを使用したフォーム操作**……タッチデバイスでは、仮想キーボードを使用します。仮想キーボードにタブ機能がある場合を除き、大部分のデバイスは項目間での操作を行うために項目をタッチします。項目内での操作は、項目の内容の上で長押しすることで有効になります。
- **コンテキストメニュー**……コンテキストメニューは重要で、ユーザ操作ツールの中心になります。画面サイズが比較的小さいため、画面スペースをボタンや画面上のメニューによって「浪費される」代わりに、コンテキストメニューを使用することで大部分のタスクを実行させることができます。
- **入力モード**……［エディット］コントロールは常に挿入モードになります。モバイル用デバイスでは、上書きモードのような機能はありません。
- **バックグラウンドでの実行**…… モバイル用デバイスの OS は、マルチタスク OS です。各アプリケーションが最前面、または、バックグラウンドで実行されます。エンドユーザは実行アプリケーションを参照することができて、アプリケーションを切り替えることができます。バックグラウンドで実行しているアプリケーションは、停止されず実行し続けますが、画面には表示されません。

モバイルデザインモード

上記のように、モバイルデバイスはさまざまな解像度と画面サイズがあります。Magic xpa では、モバイルデザインモードを利用することで容易に画面の定義を行うことができます。

モバイルデザインモードが有効で、コントロールがフォームに追加されると、いくつかの特性は異なるデフォルト値で設定されます。これはモバイルプラットフォームをよりよくサポートします。

ツールバーから［モバイルデザインモード］ボタン をクリックするか、［オプション］メニューを開いて［モバイルデザインモード］を選択することで、この機能が起動されます。

以下の特性は、非モバイルデザインモードと異なります。

- 画面サイズがより小さくなります。これは、最も小さなデバイス用に画面を設計して、より大きなデバイスに対してはコントロールのサイズを拡張するために［位置］特性を使用するが推奨されるからです。
- コントロールのサイズは、より大きくなります。
- テーブルの色は、（[テーブル色の指定］特性内の）カラムの色ではなくテーブルの色で定義されます。
- ［テーブル］ントロールには、スクロールバーが表示されません。
- ［グループ］コントロールには、デフォルトテキストがありません。
- ［ラジオ］コントロールは、［表示］特性のデフォルトが「ボタン」になります。
- コンテナコントロール上に配置される［ライン］コントロールは、コントロールサイズ一杯に水平に拡張されます。
- ［エディット］や［グループ］、［ライン］の各コントロールの［位置］特性の［幅］は、「100」% に設定されます。その結果、デフォルトでは、これらのコントロールはサイズが変更されます。その結果、デフォルトでは、これらのコントロールはサイズが変更されます。
- ［タブ］コントロールはフォーム全体に配置されます。
- ［イメージ］コントロールは、フォームに配置された場合と [テーブル] コントロールに配置された場合でサイズが異なります。

モバイルデザインモードの状態で配置されたコントロールにのみ有効です。既に配置済みのコントロールの表示は変わりません。

［モバイルプレビュー］ペイン

モバイルプラットホーム用に開発する場合、［モバイルプレビュー］ペインを使用することにより開発中の表示フォームの確認することができます。

ツールバーから［モバイル］ボタン をクリックするか、［オプション］メニューを開いて［モバイルプレビューペイン］を選択することで、この機能が起動されます。

これによって、コントロールの位置やサイズを表示させ、さまざまなモバイルデバイス上でコントロールがどのように表示されるかを確認できます。位置とサイズに関するフォームとコントロールの特性は、［モバイルプレビューペイン］ペインでサポートされます。

フローティング・ペインから、プレビューしたいデバイス（例えば iPhone 6）を選択することができます。デバイスの名前の次に、デバイスの寸法が表示されています。また、ツールバーの右側（ または ）をクリックすることでプレビューの傾きを変更することができます。ここで選択できるデバイスオプションは、Magic.ini ファイルの［MAGIC_DEVICES］セクション、または、［モバイル用デバイス］テーブルで定義されます。［モバイル用デバイス］テーブルは、［オプション］メニューから［設定］→［モバイル用デバイス］を選択することでアクセスすることができます。

モバイルデバイスではサポートされないコントロールは、罫線が交差している長方形 として表示されます。

Magic の RIA クライアント

モバイル用デバイス上で実行する Magic RIA クライアントは、スケーラブルで堅固なアプリケーションサーバに接続しながら、モバイル用デバイスの機能を利用するように設計されたネイティブアプリケーションです。

モバイル用デバイスの RIA クライアントは、Windows 用 .NET の Magic RIA クライアント機能のサブセットが実装されており、色々なモバイルデバイス上で動作させることができます。

モバイルデバイスのフォーム

フォームタイプ

すべてのフォームは定義上モーダルウィンドウになります。モーダレスなウィンドウはありません。そして、フォーカスを開いているウィンドウ間で移動することはできません。デフォルトでは、デバイスの画面全体が自動的に拡張されます。フォームの全ての位置とサイズの設定は無視されます。

ポップアップフォーム

［フォーム特性］の［ポップアップ］特性が「Yes」と設定された場合、ポップアップウィンドウ（フルウィンドウではない）としてフォームが表示されます。

ポップアップウィンドウは、ウィンドウを開くプログラムを呼び出すことができません。

モバイル共通

- ポップアップウィンドウの表示位置はフォームの［開始位置］特性によって以下のように定義されます。
  - **カスタム** …… ［X］/［Y］特性の定義内容に基づいて表示されます。
  - **... の中央** …… デバイスの中央に表示されます。
  - **... デフォルト** …… 開いているコントロールの次に開きます（ウィンドウに矢印が表示されます。iPad のみ）。

iOS の場合

- ポップアップウィンドウが表示している間はデバイスの傾きが無視されます。

各アプリケーションは、セッションが継続している間はオープンし続けるフルスクリーンが少なくとも一つはあるように開始する必要があります。追加フォームを開くこともできますが、この最初のフルスクリーンを閉じるとアプリケーションは終了します。

フォームのスクロール

大きなフォームでは、画面にスクロールバーが表示されます。そして、項目間で移動するとフォームがスクロールされます。

- **［テーブル］コントロールでの水平スクロールはサポートされません。**
- **iOS の場合、複数行対応のコントロール上での水平スクロールはサポートされません。**

［位置］特性の使用を推奨します。これによってコントロールがフォームに適合し、スクロールしないようになります。

配置

異なるデバイスの解像度や画面の向きに対応するために、RIA クライアントには、配置メカニズムが独自の方法で実装されています。

配置メカニズムは「フルスクリーン」のフォームのみに関連して、以下のように動作します。

- **オープン時のサイズ変更** ……フォームをオープンする前に、（開発者によって定義された）オリジナルのフォームサイズと現在の画面サイズが比較されます。各軸におけるサイズの違いがあった場合、ウィンドウのサイズを変更する必要があるものとして扱われ、すべてのコントロール上で配置メカニズムが作動します。これによって、小さな画面用に設計されたフォームが、より高解像度の画面では「拡大する」ことが可能になります。
- **回転によるサイズ変更**……タッチデバイスでは、画面の向きを変更（例：480x360 から 360x480 へ）した場合、自動的にその場で解像度を変更することができます。この場合、現在表示されているフォームは、［ウインドウサイズ変更］イベントが処理され、すべてのコントロールで配置メカニズムを作動させます。

フォームのサイズ

RIA クライアントは、フォームのスクロールと配置をサポートしています。しかし、（iPhone と iPad のような）画面サイズが異なるデバイスで実行させる必要のあるアプリケーションを開発する場合、表示させるコントロール数を各デバイスで異なるように、個別のフォームを開発することを推奨します。

配置の設定は、フォームタイプ毎に行わなければならないため、各タイプに対して異なるデバイスを使用することができます。

フォームの［寸法単位］として「センチ」を使用する場合、タイトルバー を表示するフル画面のフォームの幅と高さ値は以下の通りになります。

- iPhone （縦方向）
  - iPhone6/6S/7/8/SE 2nd …… 5.9 x 9.2
  - iPhone6 Plus/6S Plus/7 Plus/8 Plus …… 9.5 x 15.8
  - iPhoneX/XS/XS Max/11 Pro/11 Pro Max/iPhone12,13 (Mini/Pro/Pro Max)/iPhone14 (Pro/ProMax) …… 8.5 x 16.5
  - iPhoneXR/11 …… 6.0 x 10.0

  仮想キーボードが表示される場合は、高さが減算されます。
- iPad …… 14.7 x 18.4（縦方向）

  仮想キーボードが表示される場合は、14.7x12.3 程度になります。
- Android ……デバイスによって異なります。

iOS では、以下の理由で、画面はコントロールの大きさに余裕を持たせて設計してください。

- **やや大きめにコントロールが表示されます。**
- **コントロールのサイズが小さいと、タップで扱いにくくなります。**

フォームの傾き

フォームの［傾き固定］特性で「P= 縦」または「L= 横」を入力することによって、傾きをロックさせることができます。

フォームのアニメーション

フォームの［開始アニメーション］特性や［終了アニメーション］特性を使用することで、フォームの開始や終了のアニメーションを定義することができます。

モバイルデバイスのコントロール

サポートされるコントロールと特性

次のコントロールは、モバイル用デバイスの RIA クライアントでサポートされます。

- ラベル
- エディット
- ボタン
- グループ
- ライン
- イメージ
- ブラウザ
- チェックボックス
- コンボボックス
- テーブル
- カラム
- サブフォーム
- タブ
- ラジオボタン
- タブ

リストボックス、リッチテキスト、リッチエディット、ツリーと .NET の各コントロールは、モバイル用デバイスではサポートされません。

フォームとコントロールの各特性のほとんどが、モバイル用デバイスの RIA クライアントでもサポートされます。サポートされた特性の一覧については、Magic xpa のリファレンスを参照してください。

コントロールの表示

モバイル用デバイスのコントロールは、Windows で表示されるコントロールとは、見え方や最小時のサイズ、埋め込まれる空白などが異なります。これは、Magic xpa のサイズ調整機能を使用して定義されたコントロールのサイズでは、（モバイルデバイス上のより大きなパディングのために）入力されたテキストをすべて表示できない場合があることを意味しています。

境界線と角枠

境界と角枠は、Ver2.5 よりコントロール特性の［モバイル］セクションにある以下の特性に値を指定することによってカスタマイズすることができます。コントロールの［色］特性に、システム色を使用すると無効になります。

- 境界幅 …… 境界線の幅（ピクセル）をデバイスの dpi に基づいて指定します。

- 境界色 …… 境界線の色を基本色定義ファイルで指定します。これは、分割線の色にも影響します。
- フォーカス時の境界幅 …… コントロールにフォーカスがある場合の境界の幅。これは、［エディット］コントロールに関連します。
- フォーカス時の境界色 …… コントロールにフォーカスがある場合の境界の色。これは、［エディット］コントロールに関連します。
- 角半径 …… コントロールの角の半径（ピクセル）をデバイスの DPI に基づいて定義します。

iOS の場合、境界幅 やフォーカス時の境界幅、角半径はフォームの［水平精度］特性によって表示幅が変わります。この特性値を大きくすると表示幅が小さくなります。特性値は［寸法単位］特性が「ダイアログ」の場合、「10」前後を目安に設定することを推奨します。

キーボード機能

［エディット］コントロールで以下の特性を使用することにより、キーボードのレイアウトをコントロールすることができます。

- キーボードタイプ …… 表示するキーボードを定義します。
- キーボードリターンキー…… リターンキーの表示とアクションを定義します。
- サジェスト許可 …… サジェスチョン（ヒント）テキストを有効にするかどうかを定義します。

［エディット］コントロール

- 日付型や時刻型の修正可能な［エディット］コントロールでは、デートピッカーやタイムピッカーが表示されます。コントロールの［ピッカー］特性を使用することでこの動作を変更することができます。
- 文字型と Unicode 型の ［エディット］コントロールの書式マスクは（文字を入力しないで) コントロールを抜けた時点で計算されます。これは、モバイルデバイスのビルトインのオートコンプリート機能をサポートします。
- ［エディット］コントロールに対してヒントを定義することができます。ヒントは、入力している間、自動的に削除される［エディット］コントロール上で表示されるテキストです。

Android の場合

［エディット］コントロールで Android のデフォルトの整列規則を使用して整列するようにしたい場合は、［デフォルトの整列］特性を使用してください。これを設定すると、コントロール上の英語のテキストが左寄せで表示されます。この特性に値が設定された場合、Magic xpa の［水平整列］や［垂直整列］特性は無視されます。

［イメージ］コントロール

前記のように、コントロールのサイズは、デバイスの解像度によって変わります。

［イメージスタイル］が「コピー」に設定されたイメージは変更されず、同じサイズで表示されます。このため、異なる解像度のデバイスで表示させた場合、異なった表示になります。

適切に表示させるには、「コントロールスケール」か「フィットスケール」を使用してください。

［タブ］コントロール

［タブ］コントロールは、デバイスのタブバーとして表示されます。タブバーは、フル画面に表示されます。その結果、アプリを開発するときは、フォーム全体に渡って［タブ］コントロールを広げることを推奨します。一般に、タブバーの機能は、デバイスのデフォルト動作に基づいています。

機能	Android	iOS
ナビゲーション	タブバーの項目をクリックするか、タブバーをスワイプします。	タブバーの項目をクリックします。
位置付	フォームの最上位	フォームの底辺
タブ内の水平スクロール	サポートしません。水平スクロールは、タブ間で操作します。 画面サイズを超える（タブ上の) コントロールは表示されません。	サポートされます。
垂直スクロール	サポートされます。	サポートされます。

［タブ］コントロール上のイメージ

タブには、テキストとイメージを含めることができます。イメージは、(Windows でのタブのように）［イメージリストファイル名］と［イメージリストインデックス］の各特性で定義することができます。

Windows のタブとの違いは：

- イメージのサイズは、16x16 に制限されていません。しかし、イメージは四角形でなければなりません。このため、例えば、イメージリストには、100x100 イメージを複数含めることができます。
- イメージファイルは、タブに表示させるすべてのイメージを含める必要があります。
- iOS デバイスの場合は、iOS のヒューマンインタフェースガイドラインにもとづき、25 ピクセル四方のサイズのアイコンファイルを使用することを推奨します。

タブのヘッダの表示（Android のみ）

Android デバイスの場合、空白テキストとイメージを指定しないことで、タブのヘッダを表示させないようにすることができます。この場合、タブはヘッダが表示されずオブジェクト間でスクロールするためだけに使用されます。以下の特性を設定することでテキストやイメージを表示させたり、非表示にしたりすることができます。

表示の組合せ	［表示リスト］特性	［イメージリストファイル名］特性
テキストのみを表示	○	
テキストとイメージを表示	○	○
イメージのみ表示		○
タブバーを非表示 (ペインのスクロールのみ)		

例外

デバイスのデフォルト動作のために、以下の操作は無効です。

- ［タブ］コントロールの外側に配置されるコントロール。
- コントロールが、［タブ］コントロールのレイヤ 0 にリンクされている。
- フォーム上の複数の［タブ］コントロール。
- コンテナコントロール（例えば、［グループ］または別の［タブ］) にリンクされた［タブ］コントロール。
- ［タブ］コントロールが配置されたサブフォームタスク。

フォームの背景（壁紙や色、グラデーション）は、すべてのタブで使用されます。

トグルボタン

トグルが有効なイメージボタンとして表示されるボタンを作成することができます。これは、［イメージリストファイル名］特性を［チェックボックス］コントロールの［モバイル］セクションで設定することで表示されます。

複数のトグル可能イメージボタンを作成することもできます。これは、［イメージリストファイル名］特性を［ラジオボタンス］コントロールの［モバイル］セクションで設定することで表示されます。

コントロール上でのパーク

（プッシュボタンのような）コントロールによっては、OS の定義によってパークできない場合があります。この場合、現在のコントロール上でパークされたままになります。

仮想キーボードのサポート

仮想キーボード付きのタッチデバイスでは、キーボード入力を必要とするコントロール（修正可能な［エディット］コントロール）上にパークすると、仮想キーボードが自動的に表示されます。他のコントロール上では、自動的に隠れます。

仮想キーボードが開いているときは、フォームの見える領域が非常に小さくなる点に注意してください。項目間を移動した場合、パークされたコントロールが可視領域内に表示されるように、RIA クライアントはフォームを自動的にスクロールします。

入力が不要な項目で仮想キーボードが表示されないようにするには、入力不要の［エディット］コントロールの［修正不可］特性を「Yes」に設定するか、タスク自体を「Q= 照会」モードで実行させるようにしてください。

iOS での日本語入力

仮想キーボードでの日本語ひらがなと半角英数字の入力制御は、［エディット］コントロールの［漢字入力］特性に、各モードの番号を設定することによって可能ですが、全角ひらがな（1 または 2)、半角英数字（8）のみ対応します。

全角半角カタカナ、全角英数字モード番号を設定した場合、それぞれ、全角ひらがなモード、半角英数字モードで起動されます。

また、デバイス固有のキーボードの設定によって次のような動作になります。

- 日本語キーボード以外に英語キーボードが追加され、かつ英語キーボードがデフォルトの場合、［漢字入力］特性に「1」～「6」が設定されていても、英語キーボードが起動します。この場合、最初に言語切り替えキー  を押下して日本語入力に切り替えると、それ以後は［漢字入力］特性による制御が可能です。
- 日本語フルキーボード（標準キーボード）と日本語テンキー（あいうえお配列）の両方を追加しても、「1」（ローマ字入力）と、「2」（直接入力）の制御はされません。

インクリメンタルサーチ

インクリメンタルサーチは、モバイル用デバイスではサポートされません。

モバイルデバイスの表示

座標系

フォームの［寸法単位］で「ダイアログ」を使用する場合、ダイアログ単位はフォームのフォントの測定基準を使用してピクセルに変更されます。モバイル用デバイスは、Windows フォントをサポートしないため、デフォルトのフォント（5x13）の測定基準が使用されます。

さらに、モバイル用デバイスはさまざまな解像度や画面サイズで利用できます。全てのデバイスで同ように表示させるため、フォームの寸法とコントロールは、デバイスの DPI（Dot Per Inch）を Windows の DPI（96）で割った値をもとに拡張／縮小されます。

このため、ダイアログ単位をピクセルで変換する際、モバイルクライアントは、次の方法に従います。

- X 座標（ピクセル）＝X 座標（ダイアログ単位）× 5 ×（モバイル用デバイス dpi/96）
- Y 座標（ピクセル）＝Y 座標（ダイアログ単位）× 13 ×（モバイル用デバイス dpi/96）

Windows での計算は 96dpi に基づいていますが、実際の画面の DPI は多少異なります。このため Windows での画面サイズは、モバイル用デバイスと必ずしも同じようには見えません。

デバイス dpi の上書き（Android）

Magic xpa クライアントは、全てのデバイス上で同じサイズのコントロールを表示するためにデバイスに定義された dpi 情報に依存します。しかし、デバイスによっては（例えば、HPSlate21）、正しい DPI 値が返らないものもあります。この場合、コントロールはこれらのデバイス上では正しく表示されない可能性があります。RIAModules¥Android¥Source¥Res¥Values フォルダ内の custom_dpi.xml ファイルを変更することで、デバイスの dpi 値を上書きすることができ、正しい値を定義することができます。

このファイルでは、デバイスモデルを空白なしの小文字で指定し、対応する DPI 値（例：250.8）を定義します。

デバイスモデルは、Android の設定画面や、ClientOSEnvGet('device_model') 関数を実行することで取得できます。

フォント

各 OS は、通常、Windows OS とは異なる独自のフォントが含まれています。モバイル RIA クライアントが他のインターフェースと同じ［フォント］テーブルを使用することができるように、各デバイスで使用する個別のフォントを定義することを推奨します。

Windows 用の［フォント］ダイアログボックスを使用してモバイル用のフォントを選択することができないため、これらのフォントを入力するには、テキストエディタを使用して［フォント］テーブルを編集する必要があります。

例えば、フォントテーブル（fnt_rnt.jpn）に以下のように定義します。

- iOS Helvetica font: iOS font,Helvetica,14,0,0
- iOS Helvetica font: iOS bold font,Helvetica-Bold,14,0,0
- Android Droid Serif font: Android font,Serif,14,0,0
- Android Bold Droid Serif font: Android bold font,Serif,14,0,0,Bold

フォント定義を変更しても実行時に反映されない場合は、サーバ上のキャッシュファイルを一端削除してみてください。

Android の場合

- フォントファミリー名と使用したいフォントのスタイルを指定する必要があります。システムは対応するフォントを検索します。例："Android Serif bold,Droid Serif,12,0,0,Bold"

  デフォルトで利用できるフォントは OS のバージョンによって異なり、以下のようなフォントが利用できます。フォントの種類によっては、Bold と Italic の両方を組み合わせて利用できない場合があります。

フォント	ファミリー名	4.0	4.2
ゴシック体	sans-serif	○	○
Light ゴシック	sans-serif-light		○
Thin ゴシック	sans-serif-thin		○
コンデンス	sans-serif-codensed		○
明朝体	serif	○	○
モノスペース	monospace	○	○

  フォントファイルは、/System/Fonts フォルダにインストールされています。Android 4.0 以上の場合、/System/etc/system_fonts.xml ファイルを参照することで、フォントファミリー名を確認できます。
- ［フォント］テーブルに定義されているフォントがモバイル用デバイスで見つからない場合は、デフォルトのフォントが［フォント］テーブルで定義されているフォントサイズで使用されます。
- 式を使用してフォントを変更することができるため、異なるデバイス用に異なるフォントを使用することができます。

iOS

- 標準の日本語フォントを使用する場合は、以下のフォント名（PostScript 名）を指定してください。

フォント名（フルネーム）	PostScript 名
Helvetica	Helvetica
Helvetica Bold	Helvetica-Bold
Helvetica Oblique	Helvetica-Oblique
Helvetica-BoldOblique	Helvetica-BoldOblique
ヒラギノ角ゴシック W3	HiraKakuProN-W3 / HiraginoSans-W3
ヒラギノ角ゴシック W6	HiraKakuProN-W6 / HiraginoSans-W6
ヒラギノ明朝 ProN W3	HiraMinProN-W3
ヒラギノ明朝 ProN W6	HiraMinProN-W6

  iOS で使用されるフォントに関する情報は、以下を参照してください。

  http://iosfonts.com/

  ダイアログ単位の場合は、7x13 ポイントとして固定されています。これは、MS P ゴシックの 9pt に相当します。

プラットフォーム毎のフォント定義ファイル

モバイルプラットフォーム毎のごとに複数のフォント定義ファイルを保守することができます。プラットフォーム毎に異なるフォント定義ファイルを使用したい場合は、個別の定義ファイルを作成して、プラットフォーム毎にサブフォルダを作成して配置します。

例：

アプリケーションフォント定義ファイルが Support¥fnt_rnt.jpn の場合、iOS 用を Support¥iOS¥fnt_rnt.jpn に、Android 用を Support¥Android¥fnt_rnt.jpn に配置します。このファイルが存在している場合、実行時に、適切なフォント定義ファイルが自動的に使用されます。

このファイルを明示的に定義する必要はありません。

フォントサイズの変換

モバイル用デバイスにフォントを正しく表示させ、Magic の［フォーム］エディタで同じような表示を維持するには、Windows でのフォントサイズをモバイル用に変換するための換算式が適用されます。

（モバイル用デバイス上でのピクセル単位のフォントサイズ）＝（Windows 上でのポイント単位のフォントサイズ）*（モバイルでの dpi）/72

この計算によって、Windows とモバイルの両方でのコントロールを比較して、フォントが同じサイズのように表示されます。

色

モバイル用デバイスの RIA クライアントは、他のインターフェースと同じ［基本色］テーブルを使用しますが、Windows 上で利用できる " システム " 色はサポートされません。

- **システム色がフォームやコントロールで使用された場合、フォームやコントロールに対応するデバイスでのデフォルト色が表示されます。特定の色を使用したい場合は、背景色と前景色を明示的に定義する必要があります。**
- **［エディット］コントロール …… 境界を表示させるには、非システム色を使用してください。**
- **［コンボボックス］コントロール …… 色が使用されると、境界線が表示されないか他の表示になります。矢印を表示するには、（デバイスのデフォルトを取得するため）システム色を使用してください。**

Android では：

- **［テーブル］コントロールやテーブルに配置したコントロール …… パフォーマンスを低下させるため、透過色の使用は推奨しません。**
- **［テーブル］コントロール …… パフォーマンスを低下させるため、代替色の使用は推奨しません。**

基本色定義を変更しても実行時に反映されない場合は、サーバ上のキャッシュファイルを一端削除してみてください。

［タブ］コントロールと色

- タブの色は、（Windows と異なり）［タブ］コントロールのヘッダの前景色と背景色に影響します。
- タブの色は、タブ領域の背景に影響しません。画面の色は、フォームの色やグラデーション、壁紙によって定義されます。
- タブの色がシステム色または透過色の場合、タブの色はフォームの［タイトルバー色］特性（前景色と背景色）と同じ色をになります。
- Android の場合、現在のタブの下線は、タブのテキスト色（つまり、タブの前景色）と同じ色が表示されます。

［テーブル］コントロールと色

システム色が使用されている場合：

- 白色は、コントロールの背景色として使用されます。
- 代替行は、白く表示されます。

プラットフォーム毎の基本色ファイル

モバイルプラットフォーム毎に複数の基本色定義ファイルを保守することができます。プラットフォーム毎に異なる基本色定義ファイルを使用したい場合は、個別の定義ファイルを作成して、プラットフォーム毎にサブフォルダを作成して配置します。

例：

アプリケーション基本色定義ファイルが Support¥clr_rnt.jpn の場合、iOS 用を Support¥iOS¥clr_rnt.jpn に、Android 用を Support¥Android¥clr_rnt.jpn に配置します。このファイルが存在している場合、実行時に、適切な基本色定義ファイルが自動的に使用されます。

このファイルを明示的に定義義する必要がありません。

ネイティブ色のカスタマイズ（Android のみ）

ネイティブ色のコントロールは、OS のデフォルトテーマの色で表示されます。Android 上で、いくつかのオブジェクトに対してこの色を変更するすることができます。

以下のオブジェクトのネイティブ色を変更するには

- フォーカスのある［エディット］コントロールでは、アンダーラインの色
- チェックボックスの色

RIAModules¥Android¥Source¥app¥src¥main¥res¥values フォルダ内にある theme.xml ファイルの AppTheme スタイルセクションに以下の行を追加してください。

```
<item name="colorAccent">#hexColor</item>
```

hexColorb には、指定したい色を RGB で定義します。

例えば、黄色を指定する場合は、以下のようにします。

```
<item name="colorAccent">#ffff00</item>
```

ナビゲーションドロワーの指示アイコンの色を変更するには

RIAModules¥Android¥Source¥app¥src¥main¥res¥values フォルダ内にある theme.xml ファイルの AppTheme スタイルセクションに以下の行を追加してください。

```
<item name="drawerArrowStyle">@style/MyDrawerArrowToggle</item>
```

色を指定する新しいスタイルセクションを追加します。

```
<style name="MyDrawerArrowToggle" parent="Widget.AppCompat.DrawerArrowToggle">
<item name="color">#hexColor</item>
</style>
```

hexColorb には、指定したい色を RGB で定義します。

スピナーイメージ

スピナーイメージを表示するために SetCrsr() 関数を使用することができます。例えば、実行時間が長いバッチまたはインタラクティブでないタスクを呼び出す前に実行させることができます。

代替イメージ

異なる解像度に対応したイメージを定義することができます。

これらを以下のようにサーバ上のサブフォルダに配置します。RIA クライアントは、デバイスの特性に応じて自動的にイメージを読み込みます。

- **ServerFileToClient() 関数は同じアルゴリズムを使用しています。このため、イメージ（または任意のファイルファイル）を取得する必要がある場合、同じ規則に従って実行されます。**
- **ClientOSEnvGet ('device_resource-folder') を実行することで、アクセス可能なフォルダ名を取得できます。**

iOS の場合：

検索処理は、プラットフォーム（iOS）とデバイス修飾子に（例：~iPad、~iPhone）、そして Retina を示す指標（例：@2）に基づいて実行されます。

イメージが指定したサブフォルダ内で見つからない場合、親のフォルダを検索し、元のフォルダまで検索を継続します。

たとえば、画像ファイルが “c:¥images¥a. jpg″ と定義されている場合、iPad 2（Retina）で実行すると、サーバ上の” C:¥images¥iOS¥@2x~iPad¥” のフォルダ内の ”a.jpg” ファイルを取得します。ファイルが見つからない場合、″C:¥images¥iOS¥” を検索し、さらに ”c:¥images¥” からファイルを検索します。

Android の場合：

検索は、プラットホーム（Android）と画面サイズ（たとえば：normal、large）、アスペクト比（たとえば：long、not long）、そして、解像度（たとえば：hdpi）に基づいて実行されます。

検索は、Android のアルゴリズムに基づいてこれらのパラメータの異なる組合せによって実行されます。このため、これらのパラメータの全てを使用してイメージを定義する必要があるわけではありません。

たとえば、イメージファイルが ″C:¥images¥a. jpg″ と定義されている場合、そして、Large_notlong_hdpi の Android デバイスで実行すると、デバイスの特徴に対応していれば様々なサブフォルダにイメージファイルを配置することができます。

パラメータの例

画面	サイズ	サイズ	解像度	アスペクト比
QVGA	240 × 320	small	ldpi	notLong
HVGA	320 × 480	small	mdpi	notLong
WQVGA400	240 × 400	normal	ldpi	Long
WQVGA432	240 × 432	normal	ldpi	Long
WSVGA	1024 × 552	large	mdpi	Long
WVGA800	480 × 800	normal	hdpi	Long
WXGA	1280 × 752	xlarge	mdpi	notLong

設定例：

WSVGA の場合、以下のフォルダのどれかが存在すると、以下の優先度でアクセスします。

“c:¥images¥Android¥lage_long_mdpi”

“c:¥images¥Android¥large¥”

“c:¥images¥Android¥long¥”

“c:¥images¥Android¥mdpi¥”

“c:¥images¥Android¥”

テーマのサポート（Android）

Android クライアントの表示スタイルは、Android OS のテーマに依存しています。Magic xpa では、ActionBar に対応するため、デフォルトのテーマとして AppCmpat.Light を使用するようになりました。

ActionBar で利用できるテーマは、以下の３つだけになります。

- Theme.AppCompat
- Theme.AppCompat.Light
- Theme.AppCompat.Light.DarkActionBar

RIAModules¥Android¥Source¥app¥src¥main¥res¥values フォルダ内にある themes.xml ファイルを修正することでテーマを変更することができます。

モバイルデバイスの操作

ウィンドウ操作

先述したように、モバイル用デバイスは単純に積み重なるウインドウ・モデルをサポートします。各アプリケーションは複数のウィンドウをオープンすることができます。しかし、新しいウィンドウは前のウィンドウの上に積み重なって表示されるため、実質的にはモーダルとなります。

現在のウィンドウを閉じて、前のウィンドウに戻るには、［終了］の内部イベントを発行するか、モバイル用デバイスの組み込み機能を使用します。

- Android …… Back ボタンの使用
- iOS …… ウィンドウのタイトルバーに表示される Back ボタンの使用（ウィンドウがシステムメニューを開くために定義されていれば）

クライアントがビジーの場合や、スピナーが表示されている場合は、Back ボタンが無効になります。

アプリケーションの終了

RIA アプリケーションは、［Magic 終了］の内部イベントを発行したり、モバイル用デバイスの組み込み機能を使用したりすることで終了させることができます。

- Android …… アプリケーションの最初の画面に表示される BACK ボタンを使用します。
- iOS …… （ウィンドウがシステムメニューをオープンするために定義されていれば）アプリケーションの初期画面のタイトルバーの "X" ボタンを使用します。

デバイスの“Home”ボタンを押下した場合、アプリケーションはバックグラウンドに移動します。

上記外の方法（タスクマネージャから）でアプリケーションを終了しても、サーバのコンテキストは自動的に解放されません。コンテキストタイムアウトを過ぎた時点で解放されます。

メニュー

コンテキストメニューは、フォームのレベルでのみ利用できます。

メニューは起動時の定義内容で表示され、実行中に変更させることはできません。

最初のレベルのメニューのみサポートされます。

新規プロジェクトには MDI アプリケーション用に設計されたデフォルトプルダウンメニューが作成されます。モバイルデバイスには MDI がないため、モバイルアプリケーションを開発する場合、このデフォルトエントリを削除することを推奨します。

フォームの［コンテキストメニュー］特性を設定することで、Android と iOS デバイス上でメニューボタンをクリックすると表示されるうメニューエントリを定義することができます。

iOS デバイスでは、コンテキストメニューが定義されているとツールバーにアイコンが表示されます。

ナビゲーションドロワー

フォームの［ナビゲーションドロワーメニュー］特性を設定することでナビゲーションドロワー（Google Play のような左側のメニュー）として表示されルメニューエントリを定義することができます。

Android デバイス：

ナビゲーションドロワーを表示する画面では、画面のタイトルの左側に指示アイコンが表示されます。この指示アイコンのネイティブ色を変更することができます。詳細は、「ナビゲーションドロワーの指示アイコンの色を変更するには」（ページ 10）を参照してしてください。

RIAModules¥Android¥Source¥app¥src¥main¥res¥layout フォルダに配置されている navigation_drawer.xml ファイルを変更することでナビゲーションドロワーの外観をカスタマイズすることができます。

タッチイベント

タッチデバイスでは、タッチイベントが適切な Magic のイベントに割り当てられています。

- "タッチ"、"クリック" または "タップ" ……デスクトップ PC の［クリック］イベントと同じように動作します。これらのイベントは、フォーム上の項目間でフォーカスを移動します。セレクトコントロールでタッチした場合、コントロールは選択肢を変更します。ボタンをタッチした場合、ボタンのイベントが発行されます。
- “スワイプ”……フォームまたはコントロールでのスクロールで使用されます。
- “長押し”……［押下］の内部のイベントが発行されます。コントロールの OS によるコンテキストメニューを表示させることができます。コントロールにコンテキストメニューが定義されている場合は、Magic xpa のコンテキストメニューが表示されます。

デバイスの操作

ファイルシステムとファイル処理

モバイル用デバイスは、適切な機能を使用することでローカルファイル処理をサポートします。ローカルフォルダやファイル名のための命名規則は、各 OS に依存します。

- 名前は、大文字と小文字の区別ができません。
- フォルダの区切りは、スラッシュ（/）を使用して定義されます。

異なるデバイスのフォルダへのアクセスは、デバイスに依存します。

例：

- **iOS** …… OS によって提供される一時フォルダのみアクセスできます。
- **Android** ……異なるフォルダにアクセスすることができます。

ClientFileXXX 数を使用してクライアントのファイル名を取得する場合、一時フォルダのファイルが対象になります。フルパスでこれらの関数を使用する場合、ClientOSEnvGet('temp') を実行して一時フォルダ名を取得することを推奨します。

デバイスの特性を取得

ClientOSEnvGet 関数を使用することで、開発者は指定したデバイスの特性や機能を取得し、ロジックの実行条件として使用することができます。

モバイル用デバイスでは、以下の定義済みキーを利用することで、デバイス情報に取得することができます。

- ClientOSEnvGet (‘device_os’) …… デバイスの OS（Android、iOS）を返します。
- ClientOSEnvGet (‘device_screen-width’) ……縦方向の画面の幅をピクセル単位で返します。
- ClientOSEnvGet (‘device_screen-height’) …… 縦方向の画面の高さをピクセル単位で返します。
- ClientOSEnvGet (‘device_physical-width’) ……縦方向の物理画面の幅をインチで返します。
- ClientOSEnvGet (‘device_ physical -height’) …… 縦方向の物理画面の高さをインチで返します。
  デバイス OS が不正確な dpi 値を返す場合があるため、物理画面のサイズはすべてのデバイスで必ずしも正確にならないことに注意してください。
- ClientOSEnvGet (‘device_orientation’) …… デバイスの画面方向を返します。
- ClientOSEnvGet (‘device_os-version’) …… デバイスの OS の詳細バージョンを返します。
- ClientOSEnvGet (‘device_model’) ……デバイスのモデル番号または名前を返します。
- ClientOSEnvGet (‘device_touch’) …… デバイスにタッチスクリーンが備えられている場合は、「true」を返します。
- ClientOSEnvGet (‘temp’) …… デバイスの一時フォルダを返します（iOS のみ）。
- ClientOSEnvGet (‘device_location’) …… 現在のデバイスの場所を返します。「カメラサポート」（ページ 14）を参照してください。
- ClientOSEnvGet (‘device_magic-version’) …… RIA クライアントのバージョンを返します。
- ClientOSEnvGet (‘device_udf|getargs’) …… 下記の「別のアプリケーションから起動する」（ページ 15）で詳述されるように、アプリケーションが別のアプリケーションから起動されたときの問合わせパラメータを返します。
- ClientOSEnvGet ('device_udf|getpushid ') …… Push 通知を以下のようにデバイスに送るために使用するデバイス ID を返します。（対応バージョン : 2.4）
- ClientOSEnvGet (‘device_udf|my_string’) ……デバイス OS のネイティブコードを呼び出す場合に渡されるパラメータを返します。「ネイティブ OS コードの利用」（ページ 16）を参照してください。
- ClientOSEnvGet(‘device_resource-folder’) …… RIA クライアントが利用するリソース（イメージ）にアクセスできるフォルダ名を返します。

  Android では、Java の定義済みキーを使用することもできます（たとえば : ClientOSEnvGet('java.io.tmpdir')) は、追加情報を取得します。詳細は、以下を参照してください。
  http://developer.android.com/reference/java/lang/System.html#getProperty(java.lang.String)

デバイスの位置検索（GPS）

ClientOSEnvGet() 関数は、内蔵または、接続されている GPS 装置を使用して、現在の場所を問い合わせることができます。関数の構文は以下の通りです。

- ClientOSEnvGet (‘device_location’) …… 現在のデバイスの位置を返します。利用可能な位置オプション（GPS、ネットワークなど）を使用します。結果は、以下のフォーマットの文字列で返されます。
  - OK ……実行処理の成否を返す固定部分です。
  - 緯度と経度……現在の位置の座標が数値で返ります。位置が取得できない場合は、エラーメッセージが返ります。

GPS 装置は衛星を探すため、位置情報の問合わせに対する反応に時間がかかる場合があります。この間に、クライアントはブロックされ、応答を待ちます。開発者は、ユーザのために、処理中であることを表示するようにする必要があります。位置情報の問合わせには、20 秒のビルトイン・タイムアウトがあります。

カメラサポート

ClientImageCapture 関数を使用して、カメラを使用したり、モバイルクライアントのイメージギャラリーから、イメージファイルを取得することができます。

ソース値（最初のパラメータ）で以下の値を指定することができます。

- 0 …… カメラ機能が起動します。
- 1 …… イメージ（ギャラリー）が開きます。

- **カメラは、ビデオのキャプチャには利用できません。**
- **V3.2 以前は、ClientFileOpenDialog 関数を使用していました。**

モバイル用デバイスの機能にアクセス

［コール OS］処理コマンドと特定の URL や電話番号を使用することで、デバイスの機能（電話をかける、SMS の送信、ブラウザの起動）を利用することができます。

［コール OS］処理コマンドの［実行］特性は、「クライアント」にします。

例：

- 電話 …… tel:1-408-555-5555
- SMS …… SMS:1-408-555-1212
- E メール …… mailto:support@magicsoftware.com
- E メール …… mailto:support@magicsoftware.com?cc=me@magicsoftware.com&subject=test&body=test
- ブラウザ …… http://magicsoftware.com

また、以下も参照してください。

http://developer.apple.com/library/safari/#featuredarticles/iPhoneURLScheme_Reference/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007891-SW1

メールの送信時に、件名と内容で利用できない文字があります。例えば iOS では、アンパサンド（&）は '%26' に置き換える必要があります。Android の場合は、次の文字は削除する必要があります。#,%,&, : , and =.

さらに、この方法を使用して URL フォーマットでアクセスすることでサードパーティ製アプリケーションを呼び出すことができます。

- たとえば、'waze: //?q=New York' を使用して Waze GPS をオープンしてニューヨークへのナビゲートを行うことができます。
- 'geo: 59.915494,30.409456’では、ビゲーションアプリ（Google Maps や Waze）で、緯度：59.915494、経度：30.409456 を開きます。

PDF の印刷

iOS の場合：

PDF の印刷は、［コール OS コマンド］処理コマンドで、"Print: ファイル名 " を実行することで実行させることができます。ファイル名は、以下のように定義することができます。

- ローカルファイルのパスは、ServerFileToClient() 関数から受け取ることができます。

- Web サーバ上のファイルへの URL は、例えば、" http：//1.1.1.1/myfile.pdf というように指定します。

- **'Print' の先頭は大文字で指定してください。**
- **［コール OS コマンド］処理コマンドの［実行］特性は、「C= クライアント」に設定してください。**

印刷は、iOS の Air print プロトコルを使用して実行されます。

［コール OS コマンド］処理コマンドが実行されると、iOS の［プリンタオプション］ダイアログが表示されます。ここから出力先のプリンタを指定し、［プリント］をクリックすると印刷が実行されます。［コール OS］処理コマンドの前に［エラー］処理コマンドで、［メッセージ］ダイアログを表示させた場合［プリンタオプション］ダイアログがすぐに閉じてしまい、印刷できない場合があります。

Android の場合：

PDF の印刷は、［コール OS］処理コマンドで、"Print: ファイル名 " を実行することで実行させることができます。ファイル名は、以下のように定義することができます。

ローカルファイルのパスは、ServerFileToClient() 関数から受け取ることができます。

- **'Print' の先頭は大文字で指定してください。**
- **［コール OS コマンド］処理コマンドの［実行］特性は、「C= クライアント」に設定してください。**

印刷は、Google Cloud Print プロトコルを使用して実行されます。

PDF の表示

iOS の場合：

Magic xpa の［ブラウザ］コントロールを使用して PDF を表示させることができます。

PDF は、デバイスのローカルではなくサーバに配置してください。

Android の場合：

Android の場合、ほとんどのブラウザは、PDF を表示しません。したがって、［ブラウザ］コントロールを使用して PDF を表示させることはできません。

しかし、サードパーティの PDF ビューアアプリケーションを使用することで表示させることができます。［コール OS コマンド］処理コマンドで "pdf:filename" というコマンドを実行させることで表示させることができます。"filename" は、デバイスの公開フォルダに保存されたファイルのパスとファイル名を指定します。例：/SDCARD フォルダ。

例えば、以下のコマンドは、ReleaseNotes.pdf ファイルをサーバ上の C:¥Magic フォルダからクライアントの SDCAD フォルダにコピーして、PDF ビューワで開きます。

1. ClientFileCopy(ServerFileToClient('C:¥Magic¥ReleaseNotes.pdf'),'/sdcard/tmpfile.pdf') を実行
2. ［コール OS コマンド］処理コマンドで 'pdf:/sdcard/tmpfile.pdf' を実行します。([実行］特性は、「クライアント」に設定します)。

- **URL で指定することはできません。サーバ上のファイルは、一旦クライアントにコピーして表示させるようにしてください。**
- **デバイスに複数のビューワがインストールされている場合は、表示するビューワを選択するダイアログが表示されます。**

別のアプリケーションから起動する

URL スキーマを使用することで他のアプリケーションや e メールから Magic のクライアントアプリケーションを起動させることができます。

ClientOSEnvGet() 関数で 'device_udf|getargs' を指定することでアプリケーションに渡されたパラメータの値（URL の？以降の文字列）を取得することもできます。

Android

以下のようなリンクが添付された e メールからアプリケーションを起動することができます。

```
clienttitle://?parameters
```

“clienttitle” はアプリケーション名で、Setting.properties ファイル内の “clienttitle” で指定された名前です。

参考：

Magic xpa Ver3.0 以前までは、以下のように指定するようになっていました。

```
http://com.mycompany.myapp?parameters
```

この書式を利用したい場合は、AndoridManifest.xml ファイル内の <intent-filter> エレメント内の <data android:scheme の値を以下のように変更してください。

```
<data android:scheme="http" android:host="com.mycompany.myapp" android:path="/" />
```

iOS

以下のようなリンクが添付された e メールからアプリケーションを起動することができます。

```
myapp://?parameters
```

“myapp” は、. アプリケーション ID で、MagicApp-Info.plist ファイル内の “CFBundleURLTypes” キーで定義されている配列で、” CFBundleURLSchemes” キーの配列値になります。

Magic xpa 3.1 より、標準で URL スキーマを利用して複数のサーバアプリケーションに接続できるようになりました。
詳細は、「複数の実行プロパティを利用した汎用アプリケーション」（ページ 26）」を参照してください。

ネイティブ OS コードの利用

ネイティブ OS コードを呼び出す

Magic アプリケーションからモバイル OS のネイティブコードを使用する必要があるかもしれません。たとえば、カスタムメイドのハードウェアを使用したり、デバイスの情報（例：連絡先）にアクセスする必要がある場合などが考えられます。

ClientNativeCodeExecute() 関数を使用することで、これを実現することができます。

ネイティブ OS コードからユーザイベントを発行する

Magic アプリケーションによって処理されるイベントをネイティブ OS コードから発行する必要がある場合があります。

以下のようにコードを記述することで、コードからユーザイベントを発行することができます。

Android

1. 宣言文を追加します：import com.magicsoftware.core.CoreApplication;
2. イベントを発行します：CoreApplication.getInstance(). invokeUserEvent(event_name,param1,param2);
   - event_name …… **ユーザイベント名**
   - param1 と param2 …… **ユーザイベント・ハンドラに渡される値**

iOS

1. 宣言文を追加します：#import "Magicxpa.h"
2. nil 値で終了するパラメータの値の全てを保持する配列を追加します：NSArray *params = [NSArray arrayWithObjects:param1, param2, nil];
   - param1 と param2 …… **ユーザイベント・ハンドラに渡される値**
3. イベントを発行します：[Magicxpa invokeUserEvent:event_name Params:params];

- event name …… ユーザイベント名

- **パラメータの数に制限はありません。**
- **パラメータは、String タイプのみ使用できます。**

ネイティブコントロールの使用

ネイティブコントロールを Magic のフォームに追加する必要がある場合があります。これは、ネイティブコードを使用してネイティブコントロールを Magic コントロールに割り当てることで可能になります。これを行うには、最初に Magic コントロールに対する参照符を取得する必要があります。

以下のようにコードを追加することで Magic コントロールに対する参照符を取得することができます。

Android

1. 宣言文を追加します：import com.magicsoftware.core.CoreApplication;
2. Magic コントロールの参照符を保存する変数を追加します：View myCustomView = CoreApplication.getInstance(). getControlByName(ctrlname,generation);
   - ctrlname …… Magic コントロール名
   - generation …… タスク世代番号（現在のタスクの場合は、0）

iOS

1. 宣言文を追加します：#import "Magicxpa.h"
2. Magic コントロール参照符を保存するための変数を追加します：UIView *myCustomView = [Magicxpa getControlByName:ctrlname TaskGeneration:generation];
   - ctrlname …… Magic コントロール名
   - generation …… タスク世代番号（現在のタスクの場合は、0）

ユーザに許可を要求 (Android)

デバイスのカメラを起動したりカレンダ入力を実行するなどのネイティブコードを実行する場合、ユーザに特定の許可を要求する必要があります。

以下の例のように以下の行にネイティブコードに追加することでユーザに承認を得るように依頼することができます。

```
boolean requestResult = CoreApplication.getInstance().requestPermission(Manifest.permission.RECORD_AUDIO);
```

ここで、ユーザは RECORD_AUDIO 許可を承認するように依頼され、結果は requestResult に格納されます。

下位互換性

Ver2.5 までは、以下のようにすることができました。

ClientOSEnvGet() 関数を実行するで、device_udf | my_string の値を使用してネイティブコードを呼び出すことができました。

Android デバイスでは、invokeExternalEvent(“gmy_string”) を記述することで Magic xpa の［外部イベント］を発行させることができました。iOS デバイスでは、[self invokeExternalEvent:@”my_string”]; で発行させることができました。

この機能は、まだサポートされていますが、廃止される可能性があり推奨していません。

プッシュ通知の送信と受信

アプリケーションがモバイルデバイス上にインストールされると、モバイルデバイスにプッシュ通知を送ることができます。

Android

アプリケーションを実行しているデバイスに、プッシュ通知を送るには：

プッシュ通知をアプリケーションを実行しているデバイスに送るには、Google Firebase Developer Console にアプリを定義し、カスタムクライアントアプリをビルドする必要があります。（バージョン 3.2 以降）

1. Google Firebase Developer Console を開きます。
2. ［Project Overview］で［新規プロジェクトを作成］をクリックします。

3. ［プロジェクト名］欄にアプリの名前を入力し、国を選択してください。次に Firebase に新規プロジェクトを作成するために［プロジェクトを作成］をクリックします。
4. ［アプリを追加］をクリックします。
5. ［google-service.json］をクリックします。アプリに対する google-service.json ファイルがダウンロードされます。
6. ［継続］をクリックし、続いて［終了］を行います。
7. RIAModules¥Android¥Source¥app フォルダ内の google-service.json ファイルを置き換えます。
8. カスタムクライアントをビルドします。

通知を送るために必要なサーバキーを取得します。

1. Google Firebase Developer Console の定義されたプロジェクトを開きます。
2. プロジェクト名の右側の［メニュー］アイコンをクリックし、［設定］をクリックします。プロジェクトの設定画面が表示されます。
3. ［クラウドメッセージング］タブをクリックし、サーバキーを取得します。

アプリが適切に設定され、デバイスにインストールされた後で、デバイス ID を取得する必要があります。デバイスが FCM(Firebase Cloud Messaging) サービスに登録されると、ユニークな ID を取得します。デバイス上でアプリを開始すると、FCM への登録処理が開始され、デバイスは ID を取得します。ClientOSEnvGet('device_udf|getpushid') 関数を使用して、アプリからこの ID を取り出すことができます。

デバイス ID をデータベースに保存するためにロジックをアプリケーション (通常は、ログオン処理の後) に追加する必要があります。ID は、後でメッセージをデバイスに送るために使用することができます。

デバイス ID が実行中に変更される可能性があります。通常は、通常は、デバイスがサーバ（ログオン後）に接続されているたびにデバイス ID を取得するため心配する必要はありません。

このシナリオを処理したい場合は、src フォルダ内の MyFirebaseInstanceIDService.Java ファイルの、sendRegistrationToServer メソッド内の最後のイベント行のコメントマークを解除することで、デバイス ID が変更されると Magic xpa アプリにイベントが送られるように変更することができます。また、このイベントのためのロジックユニットをアプリケーションに追加し、デバイスリストのデバイス ID を更新する必要があります。

通知コンソールから通知をアプリのユーザに送ることができます。

1. Google Firebase Developer Console を開きます。
2. 作成したプロジェクトを選択します。
3. 画面の左側の［GROW/Notifications］をクリックします。
4. まだ実行していなければ、［新しいメッセージ］をクリックします。
5. ［メッセージ文］欄に送信するメッセージを入力し、メッセージラベルを入力します。
6. ［ターゲット］で「ユーザーセグメント］を選択し、［アプリ］を選択し、［メッセージを送信］をクリックします。
7. ［メッセージの再確認］ウィンドウが表示されます。内容を確認し、［送信］をクリックします。

プッシュ通知は、様々なサードパーティのソフトウェアパッケージからデバイスに送信されます。

cur を使用して送る場合は、以下のように実行してください。

```
curl -X POST --header "Authorization:key=AAAAA" --Header "Content-Type: application/json" https://
fcm.googleapis.com/fcm/send -d "{¥"to¥":¥"BBBB¥",¥"notification¥":{¥"body¥":¥"CCCC¥"}}"
```

- AAAA…… 最初の段階で取得したサーバーキー（例、"AfdsSyg5ENO6jxUPQvUpIduydn53DlYiiGPTxUM"）
- BBBB …… デバイス ID（例、"AXDSFGFSDSFSGSF"）がカンマ区切りで指定された文字列。
- CCCC …… 送信する文字列（例、"Hello World"）

ファイアウォール上で、ポート 5228、5229 と 5230 を開いておく必要があります。

iOS

アプリケーションを実行しているデバイスにプッシュ通知を送るには、以下のようにします：

1. iOS 開発ポータル上でアプリケーションを定義します。
2. APNs 証明書を作成します。
3. APNs 証明書を使用するアプリケーションに対し、Apple Push Notification サービスを有効にします。

APNs 証明書作成

証明書 CSR ファイルは取得済みであることが前提です。

1. AppleDeveloper にログインし、「Certificates, Identifiers & Profiles」を選択します。
2. 左ペインのメニューから［Identifiers］を選択し、AppID を作成します。この時「Push Notifications」を選択状態にします。
3. 左ペインのメニューから「Certificates」を選択し、右上の「+」ボタンで証明書を作成します。
4. [What type of certificate do you need] の画面が表示されます。評価用の場合は Development セクションの [Apple Push Notificationservice SSL(Sandbox)] を選択し、[Continue] をクリックします。製品用の場合は、[Production] セクションの [Apple Push Notificationservice SSL(Sandbox&Production)] を選択します。
5. [Which App ID would you like to use] の画面が表示されます。#2 で作成した App ID を選択し、[Continue] をクリックします。
6. [About Creating a Certificate Sining Request(CSR)] の画面が表示されます。(CSR ファイルの作成手順の説明が表示されます。）[Continue] をクリックします。
7. [Generate your certificate] の画面が表示されます。[Chose File] をクリックして CSR ファイル (デフォルト名：CertificateSigningRequest.certSigningRequest) を選択します。
8. [Your certificate is ready] の画面が表示されます。
9. [Download] をクリックして証明書ファイル（cer）をダウンロードします。

Push 実行時に必要となる証明書ファイルを用意します。

1. Mac のキーチェーンアクセスを開きます。
2. 「Apple Development IOS Push Services: （AppID)」の証明書を見つけます。
3. 2 つ折りになっているので展開し、証明書と秘密鍵の両方を選択状態にします。
4. 右クリックで「2 項目を書き出す」を選択し、p12 形式でエクスポートします。仮に「push_development.p12」とします。

アプリの設定

プロジェクトでプッシュ通知を使用するための設定を行います。

1. Xcode でプロジェクトを開きます。
2. 「Capabilities」から「Push Notifications」と「Background Modes」を ON にします。
3. 「Background Modes」は「Background fetch」と「Remote Notifications」にチェックを入れます。
4. プロジェクトをビルドします。

アプリケーションが適切に設定され、デバイス上にインストールされた後は、デバイス ID を取得する必要があります。デバイスが APN サービスに登録されると、APN サービスからユニークな ID を取得します。デバイス上でアプリケーションを開始すると、APN への登録処理が開始され、デバイスは ID を取得します。この ID は、ClientOSEnvGet(device_udf|getpushid') 関数を使用しているアプリケーションから読み出すことができます。

データベースにデバイス ID を保存するために、アプリケーション（通常、ログオン処理の後）にロジックを追加することで、後でデバイスにメッセージを送る際に ID を使用することができます。

サンプルプログラム

デバイス ID の取得

リッチクライアント逆引きサンプルの RNC06 をモバイルから呼び出すことで、デバイス ID を取得し、データベースに保存することができます。

メッセージの送信

オンライン逆引きサンプルの CN20、リッチクライアント逆引きサンプル の RNC20 で取得したデバイス ID に対してメッセージを送ることが出来ます。

デフォルトは製品用（Product）の認証ファイルを使用して送ります。開発用 (Saqndbox) の場合は、初期化処理を以下のように変更する必要があります。

```
DotNet.PushSharp.Apple.ApnsConfiguration(DotNet.PushSharp.Apple.ApnsConfiguration.ApnsServerEnvironment.Sandbox, F, G)
```

プッシュ通知は、様々なサードパーティのソフトウェアパッケージからデバイスに送信されます。

例：

- NET コード：https://github.com/Redth/PushSharp
- Java コード：http://code.google.com/p/javapns/

- **ファイアウォール上で、ポート 2195 と 2196 を開いておく必要があります。**
- **Windows からメッセージを送る には、Mac で作成された SSL 証明書を、PKCS12 フォーマットの証明書ファイルに変換する必要があります。**

参考：http://code.google.com/p/apns-sharp/wiki/HowToCreatePKCS12Certificate

サードパーティのライブラリと統合する

モバイル用クライアントアプリケーションにサードパーティ製のライブラリを統合させることができます。

クライアントアプリケーションをビルドする際に、ライブラリの統合を行わなければなりません。

ライブラリを呼びすにアプリケーション内でどのようにネイティブコードを記述するかについての詳細は、「ネイティブ OS コードの利用」（ページ 16）の説明を参照してください。

Android

単にライブラリをアプリケーションの libs フォルダ（例：RIAModules¥Android¥Source¥app¥libs の下）にコピーしてください。

イブラリに追加する許可が必要な場合は、AndroidManifest.xml（RIAModules¥Android¥Source¥app¥src¥main）ファイルでそれらを定義する必要があります。

iOS

Xcode を使用してライブラリをプロジェクトに追加する必要があります。

開発環境

Android と iOS プラットフォームは、モバイル用デバイス上にネイティブアプリケーションをインストールするために、管理され制御された処理を提供します。これらのプラットフォームで実行される Magic アプリケーションは、ユニークな名前とアイコン、アプリケーション特性を持った自己完結型のアプリケーションとしてパッケージ化する必要があります。

アプリケーションを構築するには、設定ファイルを編集して、Magic RIA クライアント・エンジンが埋め込まれた、新しいユニークなアプリケーションを作成する“ビルド”ツールを実行する必要があります。このアプリケーションは、サポートされた流通機構を使用してモバイル用デバイスに配備することができます。

モバイル用デバイスの最低条件

デモや評価を目的としてデバイスを使用する場合は、製品の Readme を参照してください。

モバイルアプリの開発環境

Android

JDK

Magic xpa Studio のインストールの際に <Java> サブディレクトリに Open JDK がインストールされます。

Windows の環境変数 <JAVA_HOME> を <Magic xpa Studio のインストールフォルダ >¥Java¥x64 に設定してください。

Android Studio

Magic xpa 3.1 より、Android Studio の開発環境に合わせた gradle というビルドツールを使用してモジュールのビルドを行うようになりました。以下からダウンロードしてインストールしてください。

http://developer.android.com/sdk/index.html#Other

Android Studio で Connfigure → SDK Manager を選択することで、開発環境を指定することができます。

Android 用クライアントアプリをビルドするには、以下の環境が必要です。(Magic xpa のバージョンによって、変更される場合があります。)

- [SDK Platoform] タブ
  - Android 13.0 (API 33) :
- [SDK Tools] タブ
  - Android SDK Build-tools (revision 33.0.x) …… revision 33.0.x でビルドするように設定されています。[Show Package Details] をチェックすると、インストールするレビジョンを指定することができるので、"Android SDK Build-tools 33.0.x" を選択してください。
  - Android SDK Tools (revision 33.x)
  - Google Play Services
  - Google USB Driver

[Android SDK Location] に表示されている SDK のパスは、Magic 側でビルドする際に指定するものですのであらかじめ記録しておいてください。

- **インストール時に指定したパスと同じになっていない場合があります。**
- **コンポーネントに関する追加情報は、以下を参照してください。**

http://developer.android.com/sdk/installing.html

Gradle の設定

** 上記で定義されたものより新しいバージョンがインストールされている場合、gradle 設定ファイルのバージョン番号を更新する必要があるかもしれません。これを行うには、build.gradle ファイル (RIAModules¥Android¥Source¥app) を開いてください。このファイル内で、compileSdkVersion のバージョン番号と buildToolsVersion パラメータを、インストールしたバージョンに変更してください。

例 :

```
android {
    compileSdkVersion 33
    buildToolsVersion "30.0.3"
```

compileSdkVersion パラメータは SDK プラットフォームのバージョン番号であり、buildToolsVersion パラメータは Android SDK ビルドツールのバージョン番号です。

iOS

iOS モジュールは、Mac 上の Xcode を利用してビルドします。

Magic サーバの実行環境

Magic のサーバエンジンは、モバイル RIA のクライアントからリクエストを処理するためにバックグラウンドで実行させる必要があります。

モバイルデバイスをサーバに接続

モバイル用デバイス上でアプリケーションを実行するには、デバイスがサーバにアクセスできるようにしなければなりません。

これを行うには、以下のような方法があります。

- デバイスをローカル PC と同じネットワークに接続するには、デバイスの Wi-Fi 機能を使用してください。
- デバイスが USB で PC に接続することができるのであれば、USB 接続を有効にし、サーバアドレスとして PC に設定された IP アドレスを使用することができます。
- Wi-Fi 機能を持つ PC で開発する場合は、PC を無線 LAN のホットスポットにして、デバイスを Wi-Fi でそのホットスポットに接続して使用することができます。

詳細は、「モバイル用デバイス上でアプリケーションをテストする」を参照してください。

モバイル用デバイスのシミュレータ

アプリケーションをテストするために、デバイスのシミュレータを使用することができます。しかし、シミュレータは実際のデバイスと比較すると機能が制限されるため、動作が異なる場合があります。したがって、実際のデバイスを使用してアプリケーションをテストすることも必要です。

iOS

iOS シミュレータは Mac OS 上でのみ実行可能です。バージョン 11.0 以上の iOS 開発環境（Xcode）を必要とします。

シミュレータを実行するには、Xcode でプロジェクトをオープンし、（左上側の）スキーマコンボボックスに必要なデバイス（例:iPad Simulator）を設定し、［Run］ボタンをクリックしてください。

Android

※ Android では、”Emulator” と表記されています。

Android エミュレータは、Windows 上で実行させることができます。

Android Studio のメイン画面から、Configure > AVD Manager を選択することで Android エミュレータを実行するための AVD（Android Virtual Device）Manager が起動されます。起動されたら Android エミュレータの環境設定と仮想環境の設定を行います。

Android Virtual Device Manager

1. Android Virtual Device Manager のダイアログで新しいデバイスを追加するには、［Create Virtual Deveice...］ボタンをクリックし以下を設定します。
   - Category …… **仮想デバイスのタイプ**（Phone または Tablet）
   - **機種 …… テーブルからデバイスの機種を選択します。**
2. ［Next］ボタンをクリックします。
3. System Image を選択します。"Downliad" が表示されている場合は、クリックするとイメージ情報がダウンロードされます。
4. ［Next］ボタンをクリックします。
5. ADV Name を入力します。
6. ［Finish］ボタンをクリックします。

デバイス作成が終了すると、デバイスがリストに追加されます。デバイスを選択し、［Action］で矢印ボタンをクリックするとエミュレータが実行されます。

- **Genymotion“ というエミュレータを利用することもできます。以下からダウンロードすることができます（ユーザ登録が必要です）。**

https://cloud.genymotion.com/page/launchpad/download/

このエミュレータは、仮想の Android デバイスとして実行する仮想 PC です。仮想 PC をダウンロードしてインストールした後に、Android デバイスをダウンロードして、ウィザードに従ってインストールしてください。このエミュレータは、ネイティブの Android エミュレータより動作が速くなります。

モバイル開発の考慮事項

モバイル用のアプリケーションを開発する場合、以下のようにすることを推奨します。

- ほとんどの画面を、ローカルデータを表示するオフライン画面として作成する。
- クライアントにデータを転送するための同期プログラムを使用する。

上記のようにすることで、ネットワーク環境やサーバ環境が利用できなくても、エンドユーザがアプリケーションを利用できるようになります。

サーバ (例えば、クライアントに情報を保存しない金融系のアプリケーション) と接続している間に、アプリケーション全体を実行させる必要がある場合、最初の画面はオフラインにする必要があります。

これは、以下のように、よりよいユーザ満足度を提供するために重要です。

- ネットワーク環境やサーバ環境が利用できなくても、最初の画面が表示される。
- ユーザがアプリケーションを開き、コンテキストがサーバ上でクローズされた場合、アプリケーションは、最初の画面にに戻り、アプリが終了することがない。

iTune Connect でアプリケーションをアップデートする際、(ネットワークが利用できないなどで) アプリをロードしない場合、Apple はアプリケーションの公開を拒否する可能性があることに注意してください。

また、ClientSessionSet('EnableCommunicationDialogs'、'FALSE'LOG) 関数を使用して、接続エラーメッセージを表示させず、［無効サーバ］イベントを使用したロジックユニット内で独自のメッセージを表示させることを推奨します。

デバイスまたはシミュレータでアプリケーションを実行する

モバイル RIA クライアントは、アプリケーションの実行特性を変更した後に、コードのコンパイルや署名を行うことなく、モバイル用デバイス上でアプリケーションを実行するための簡単なインターフェースを提供します。

この方法は、開発時の評価目的でのみ使用します。アプリケーションを配布するには、後ほど指定されるような独自のカスタムアプリケーションを作成する必要があります。

Magic アプリケーションをテストするには、以下の手順に従ってください。

1.Magic xpa サーバの準備

(Magic xpa Studio で Ctrl+F7 を押下するか、実行エンジンを起動して) Magic サーバでアプリケーションを実行します。モバイル RIA クライアントからリクエストを処理するためには、Magic サーバをバックグラウンドで実行させる必要があります。

2. アプリケーションの実行情報が定義されたテキストファイルの作成

以下のような行が含まれたファイルが必要です。

```
<properties>
<property key="protocol" val="http"/>
<property key="server" val="192.168.137.1"/>
<property key="requester" val="Magic4xScripts/MGrqispi.dll"/>
<property key="appname" val="XXX"/>
<property key="prgname" val="XXX"/>
</properties>
```

使用する Web サーバやリクエスタの内容に従って修正してください (”3x” の ”x” は、バージョンによって異なります)。

サンプルファイル (DevProps.txt) には、これらの行が含まれています。このファイルは、以下のフォルダにコピーされています。

- %EngineDir%¥RIAModules¥Android
- %EngineDir%¥RIAModules¥iOS

このファイルの拡張子は、“txt” にしてください。(例 : DevProps.txt)

別の拡張子を使用することもできますが、その場合はモバイル用デバイスがファイルを開くことができるように、サーバ側で MIME タイプを定義する必要があります。

ファイルは、エイリアスで公開されているサーバフォルダ (例 : Magic4xScripts または Magic4xRIAApplications) に配置してください。

例 :http: //192.168.137.1/Magic4xRIAApplications/DevProps.txt.

3. アプリケーションのビルド

後ほど説明する方法で、提供しているファイルを使用してアプリケーションをビルドしてください。Andrido 用の実行モジュールはサンプルとして以下のフォルダにコピーされています。

```
%EngineDir%¥RIAModules¥Android¥Myapp.APK
```

リッチクライアントインタフェースビルダを使用してビルドすることもできます。Android 版は、実行モジュールまで作成できますが、iOS 版は、ビルドするために必要なファイルを作成するだけで、実際のビルド作業は、Mac で行います。

4. アプリケーションのインストール

モバイル用デバイスまたはシミュレータにアプリケーションをインストールしてください。第 7 章「モバイル用デバイスまたはシミュレータにアプリケーションをインストールする」(37 ページ) を参照してください。

5. アプリケーションの実行

アプリケーションをインストールした後、これを実行すると、ポップアップダイアログが開きます。

このダイアログボックスには、上述した Web サーバ上の実行特性ファイルをアクセスするための URL を入力してください。

例：http://192.168.137.1/Magic4xRIAApplications/DevProps.txt

6.OK をクリックして、アプリケーションを実行

- 一旦、上記の手順を終了すると、モバイル用デバイス上でアプリケーションを実行することで入力した URL に定義される接続設定が読み込まれます。
- モバイル用デバイス上でアプリケーションを実行すると、これらの設定の変更内容が自動的に適用されます。このため、モバイル用デバイス上にアプリケーションを再インストールすることなく実行環境（例えばプログラムの公開名）を簡単に変更することができます。
- アクセスする URL を変更するには以下のようにします。
  - Android の場合 …… Android デバイスの［設定］－［アカウント］画面で、URL を変更することができます。［アカウントの追加］をクリックするとアプリケーション名が一覧表示されます。Magic xpa のアプリケーション名をクリックすると URL を指定する画面が表示されます。リッチクライアントインタフェースビルダで作成されたアプリでは表示されません。
  - iOS の場合 …… iOS デバイスの［設定］画面で変更できますが、デフォルトのプロジェクトではこの機能は無効になっています。有効にするには、Xcode で iOS 用プロジェクトを開いている状態で［File］メニューから［Add Files to “MagicApp”］を選択し、Settings.Bundle フォルダを選択し、［Add］をクリックしてください。このプロジェクトをビルドすることで有効になります。
- モバイルデバイスに複数インスタンスのアプリケーションをインストールする必要がある場合は、以下で説明するようにユニークな ID になるようにアプリケーションをカスタマイズしてください。

リッチクライアントインタフェースビルダを使用してビルドした場合は、接続先のアプリケーションがクライアントアプリに組み込まれるため URL 指定のダイアログは表示されません、また変更もできません。

複数の実行プロパティを利用した汎用アプリケーション

デバイスにインストールされた汎用アプリを使用して、URL をパラメータとして指定することが可能です。

以下のリンクを持つ HTML 起動サイトを使用することで、異なる動作環境を持つアプリケーションの起動を行うことができます。

<a href="magicxpa4x://?URL=http://192.168.137.1/Magic4xRIAApplications/DevProps.txt">Start My App

"magicxpa4x" はスキーマ名です。インストールするアプリによって異なります。半角英数字の小文字で指定してください。

- Android …… s<Magic xpa Studio>\RIAModules\Android\Source\settings.properties ファイルの "client.title" の値を修正して下さい。
- iOS …… <Magic xpa Studio>\RIAModules\iOS\Source\MagicApp\MagicApp-Info.plist ? の "CFBundleURLSchemes"（Xcode でプロジェクトを開いた場合は、URLTypes >> Item 0 >> URL Schemes）の値を修正してください。

"http://192.168.137.1/Magic4xRIAApplications/DevProps.txt" は実行ファイルにアクセスするための URL です。

アプリがクライアントで起動されると、アプリケーション内の execution.properties ファイルに接続情報がない場合、URL リンクから実行情報が取得されます。この値は、エンドユーザが設定画面に書き込んだように保存されます。

この機能は、例えば、1 つのアプリをコンパイルしクライアントにインストールすると、異なる URL によってこのクライアントアプリが開始され異なるアプリを起動させることができます。

アプリケーションのカスタマイズ

アプリケーションは、表示される名前やアイコン、ロゴをカスタマイズしたり、独自の証明書を使用して署名したりすることができます。

カスタムアプリケーションを作成するには、コードを編集する必要があります。

ここでは、手動でビルドする方法について説明しています。リッチクライアントインタフェースビルダを使用する場合は、リファレンスヘルプを参照してください。

Android

コードをコンパイルするには、上記の Android エミュレータの節で説明したように、JDK と Android Studio（SDK）をインストールしておく必要があります。

Android アプリケーション用のソースコードは、以下のフォルダにコピーされています。

```
%EngineDir%¥RIAModules¥Android¥Source
```

Settings.properties ファイルを以下のように修正することで、アプリケーションの特性を変更することができます。

- sdk.dir …… Android SDK のフォルダ。（例：C:¥¥Users¥¥<UserId>¥¥AppData¥¥Local¥¥Android¥¥Sdk）
- target …… コンパイルが実行される Android のバージョン。実行可能な最も低いバージョンを指定してください。そのバージョンが、PC 上にインストールされていなければなりません。Android SDK マネージャを使用したり、android-sdk¥platforms フォルダを参照したりすることでインストールされたバージョンを確認することができます。
- client.title …… アプリケーションのファイル名およびデバイスに表示されるアプリケーション名。
- client.version.code …… バージョンを比較する際に使用されるアプリケーションコードのバージョンを示す数値。正の整数値で指定してください。
- client.version.name …… ユーザに表示するためのアプリケーションのリリースバージョンを表す文字列
- package.name …… パッケージの識別 ID。パッケージ名は小文字で、Android システム上にインストールされるすべてのパッケージ全体でユニークでなければなりません。大文字を使用した場合、スクリプトで小文字に変換されます。パッケージ名には数字や 'new' などのような予約語として使用できない文字があります。
- key.store, key.alias, key.store.password, key.alias.password ……署名ファイルや ID やパスワードを指定します。（key.store 格納先によっては、正しくビルドできない場合があります。）
- output.dir …… ビルドファイルの出力先を指定します。
- build.dir …… ビルド処理で使用される一時フォルダ。このフォルダは、ビルドが成功した後に削除されます。絶対パスか相対パスを指定します。

以下のファイルも変更することができます。

- アイコン ……RIAModules¥Android¥Source¥app¥src¥main¥res¥drawable-xxxx サブフォルダ内のアイコンファイルを置き換えてください。以下のサイズのアイコンファイルを提供する必要があります。
  - drawable-mdpi …… 48x48 ピクセル
  - drawable-hdpi …… 72x72 ピクセル
  - drawable-xdpi …… 96x96 ピクセル
  - drawable-xxdpi …… 144x144 ピクセル
- 起動画面のロゴ ……RIAModules¥Android¥Source¥app¥src¥main¥res¥drawable-xxx サブフォルダ内の logo.png ファイルを差し替えてください。デバイスの画面サイズに基づいた異なるサイズの画像を置く必要があります。
  - drawable-hdpi …… 高解像度縦用画像（例：480 × 800 ピクセル）
  - drawable-land-hdpi …… 高解像度横用画像（例：800 × 480 ピクセル）
  - drawable-mdpi …… 中解像度縦用画像（例：320 × 480 ピクセル）
  - drawable-land-mdpi …… 中解像度横用画像（例：480 × 320 ピクセル）
  - drawable-xhdpi …… 超高解像度縦用画像（例：640 × 960 ピクセル）
  - drawable-land-xhdpi …… 超高解像度横用画像（例：960 × 640 ピクセル）
  - drawable-large-mdpi …… 大画面縦用画像（例：480 × 800 ピクセル）
  - drawable-large-land-mdpi …… 大画面横用画像（例：800 × 480 ピクセル）
  - drawable-xlarge-mdpi …… 超大画面縦用画像（例：800 × 1280 ピクセル）
  - drawable-xlarge-land-mdpi …… 超大画面縦用画像横用画像（例：12080 × 800 ピクセル）
- 実行特性 ……execution.properties ファイルを開いて、実行値を変更してください。

上記で定義されているような実行特性や参照するファイルの URL を定義する必要があります。

URL の設定を空白にしておくと、ダイアログボックスが開きます。この場合、エンドユーザが URL を入力する必要があります。クライアントモジュールに汎用性を持たせる場合は、この方法が便利です。

変更処理を実行した後、アプリケーションをコンパイルして、署名する必要があります。

参考：

アプリケーション名を日本語で表示させる場合は以下のようにしてください。

1. RIAModules¥Android¥Source¥app¥src¥main¥res¥values フォルダ内の strings.xml ファイルをテキストエディタで開き、<resources> タグ内の、"name="app_name" の行を以下のように変更します。

   ```
   <string name="app_name" translatable="true">MagicDev</string>
   ```

2. RIAModules¥Android¥Source¥app¥src¥main¥res¥values-ja フォルダ内の strings.xml ファイルをテキストエディタで開き、<resources> タグ内に、アプリケーション名となる文字列を以下のように追加します。ファイルは、UTF-8 で保存してください。

   ```
   <string name="app_name">テストアプリケーション</string>
   ```

- **Values フォルダ内の strings.xml ファイルで、name="app_name" が translatable="false" になっている場合は、Values-ja フォルダ内の strings.xml ファイルには、name="app_name" の定義行を含めないでください。この設定の場合は、settings.properties ファイル内の client.title の値がタイトル名になります。**
- **translatable="true" に定義されている場合は、OS の言語設定にもとづいて Values-xx 内の定義内容がタイトル名として表示されます。**
- **リッチクライアントインタフェースビルダを使用する場合は、この設定を利用することができません。build.cmd を使用した場合のみ有効です。**

カスタムアプリケーションをコンパイルして署名するには、以下の手順に従ってください。

1.keystore ファイルの作成

APK ファイルを署名するには、最初に独自のキーと証明書を含んだ keystore を作成する必要があります（この処理は一回だけ必要です）。

テスト用の keystore がインストールされるため、評価目的であれば、これを使用することでこの手順をスキップすることができます。実行環境で使用する場合は、独自の keystore を作成することを推奨します。

keystore を作成するには、Java SDK でインストールされる keytool アプリケーションを使用します。以下のコマンドを実行し、表示される指示に従って、keystore を作成してください。

keytool -genkey -keystore ＜証明書ファイル名＞ -alias ＜証明書エイリアス名＞ -validity ＜証明書の有効日数＞

例：keytool -genkey -keystore my.keystore -alias mykey -validity 8000

2. 署名された APK ファイルの作成

Android プロジェクトのコンパイルは、Gradle tool を使用して行います。

1. 管理者権限でコマンドプロンプトを開き、以下のフォルダに移動します。

   ```
   %EngineDir%¥RIAModules¥Android¥Source
   ```

2. 以下のコマンドを実行します：build.cmd

   ビルド処理が実行され、output ディレクトリ内に”client.title”で指定された名前＋”.apk”でファイルが作成されます。

- **一番最初にビルドする場合は、すべての Gradle コンポーネントをダウンロードするためにインターネットにアクセスする必要があります。**
- **Android Studio の Configure/SDK Manager を開き、「モバイルアプリの開発環境」（ページ：21）で記載されたコンポーネントをインストールする必要があります。**
- **ビルド処理に問題が発生した場合は、コマンドプロンプト内に表示されるエラーメッセージを参照してください。**

これで終了です。これによって作成された apk ファイルをモバイルデバイスにインストールすることができます。

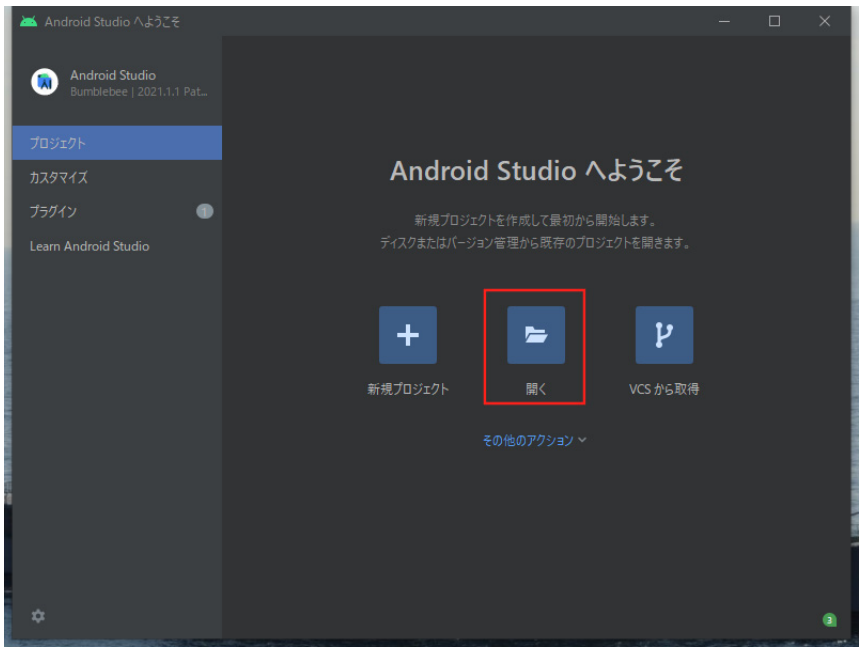
詳細情報は、以下を参照してください。

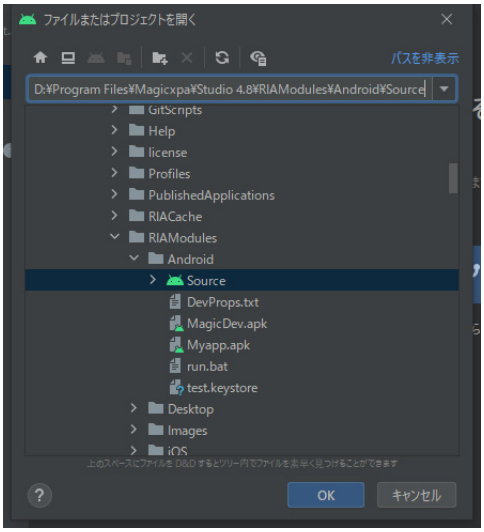
http://developer.android.com/guide/publishing/preparing.html

Android Studio で Signed Bundle を生成する

Signed Bundle は、アプリのコンパイル済みコードと必要なリソースを含むパブリッシングフォーマットです。署名入りバンドルを生成するには、以下の手順で行ってください。

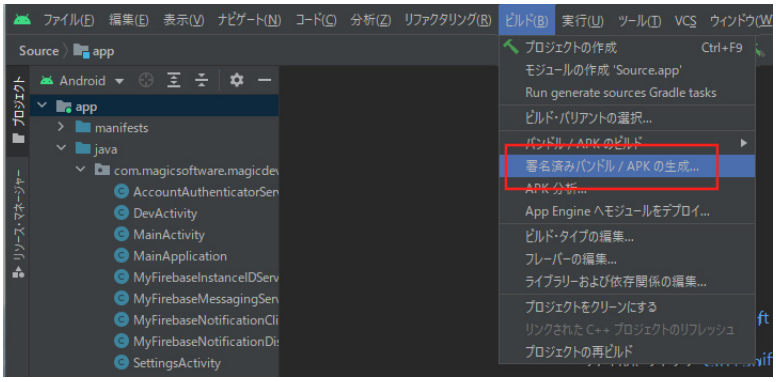
1. Android Studio で Android プロジェクト (RIAModules¥Android¥Source フォルダ) を開きます。



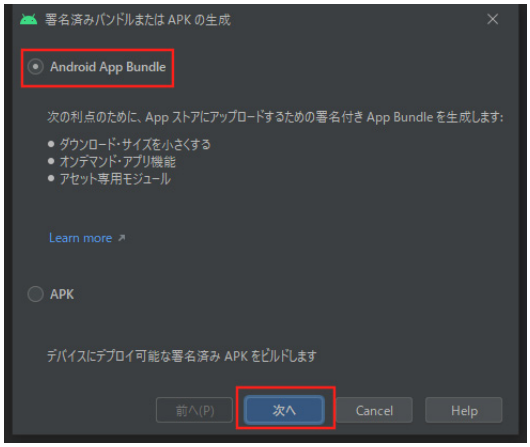


インストールフォルダが "C:¥Program Files (x86)" 内にある場合は読み込めない場合があります。

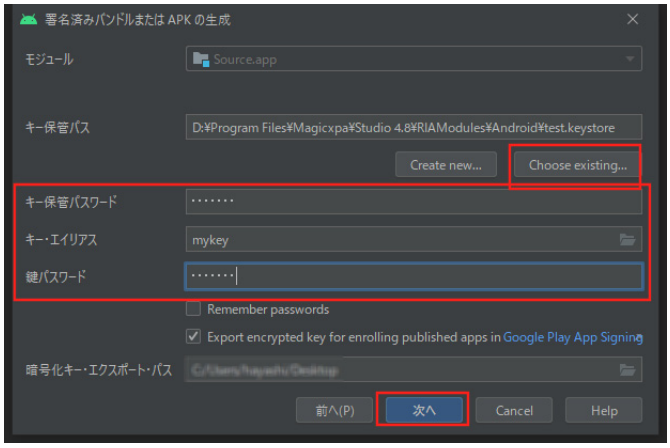
2. ［ビルド > 署名済みバンドル / APK の作成］をクリックします。



3. ［Android App Bundle］を選択します。
4. ［次へ］をクリックします。



5. 証明書の情報を設定するダイアログが表示されます。
6. settings.properties ファイル を参考に設定し、［次へ］をクリックします。



以下は、settings.properties ファイルの内容です。

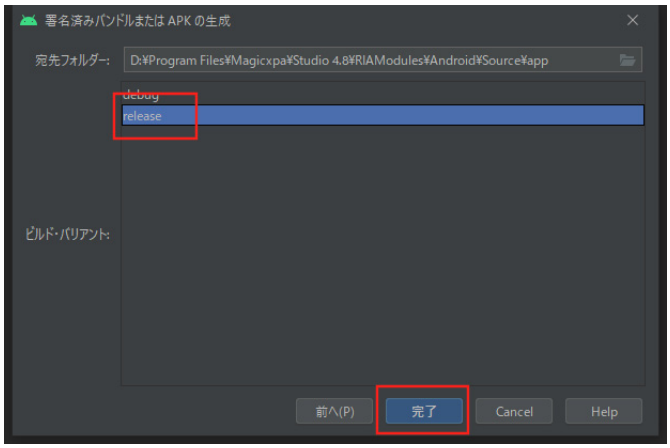
```
sdk.dir=C:¥¥Program Files¥¥Android¥¥android-sdk
target=android-30
client.title=Myapp
client.version.code=1
client.version.name=1.0
package.name=my.application.full

key.store=c:¥¥temp¥¥android¥¥test.keystore
```

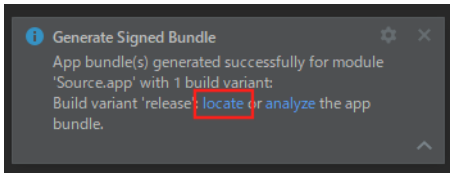
```
key.store.password=test123
key.alias=mykey
key.alias.password=test123


output.dir=output
build.dir=build
httpLibrary=okhttp
webviewJavaScriptEnabled=true
```

7. "Release" を選択して、[完了] をクリックします。



8. [完了] をクリックします。処理が完了すると、バブルが表示されます。



9. "Locate" をクリックすると、aab ファイルが生成されたフォルダが開きます。

詳しくは、https://developer.android.com/guide/app-bundle を参照してください。

アプリケーション権限の定義

Android アプリケーションでは、異なるアクション（メッセージの送信や電話の呼出など）を行う場合、デフォルトで権限設定を行う必要があります。

これらの権限設定は、Magic xpa のビルトイン機能として提供されているため、デフォルトで利用できます。

しかし、AndroidManifest.xml ファイル（RIAModules¥Android¥Source¥app¥src¥main 内にあります）をテキストエディタで開き、許可しない機能に対して tools:node="remove" を追加することで無効にすることができます。

例えば、SEND_SMS を利用できないようにするには、以下のように設定してください。

```
<uses-permission android:name="android.permission.SEND_SMS" tools:node="remove" />
```

- **この権限は、Magic xpa のコアライブラリでも定義されているため、マニフェストファイルで権限設定を定義行をコメントにしたり削除しても無効にはなりません。**
- **アプリでどの権限が利用可能かを確認するには、コマンドプロンプトで以下のコマンドを実行してチェックすることができます。**

aapt d permissions "MyApp.apk"

aapt.exe は、android-sdk\build-tools\xxx\ フォルダ内にあります。

iOS

コードをコンパイルするには、iOS シミュレータの説明で紹介した Xcode インストールし、アプリ ID（AppID）で署名されたプロビジョニングプロファイルが必要です。

このドキュメントは、内部用（in-house）のパッケージを対象としています。App store によって配布する場合は対象外です。Ad-Hoc の場合は、デバイスの UUID を登録したプロビジョニングプロファイルを使用する必要があります。

リッチクライアントインタフェースビルダで iOS 用のファイルを作成するとビルド作業に必要なファイルが作成されますが、実際のビルド作業は Mac OS 上で行う必要があります。

ここでは、シェルスクリプトを使用してビルドする方法について説明していますが、Xcode でプロジェクトを開いてビルドする方法を推奨します。

iOS アプリケーションのソースコードは、以下にコピーされます。

```
%EngineDir%¥RIAModules¥iOS¥Source
```

settings.properties ファイルを以下のように修正することでアプリケーションの特性を変更することができます。

- bundle.display.name …… ユーザに表示されるアプリケーションのタイトル
- build …… アプリケーションコードのバージョンを表す整数値（バージョンを比較する際に使用されます）。
- Version …… ユーザに表示されるアプリケーションコードのリリースバージョンを表す文字列。
- bundle.identifier …… プロジェクトの ID（XCode で確認できます）。ID は、iOS システム上にインストールされるすべてのパッケージ全体でユニークにしなければなりません。

プロビジョニングファイルを作成する際、プロビジョニング内のアプリケーション ID は、ここから入力される値と同じにする必要があります。大文字小文字を区別しています。

- target.name …… AppStore で表示されるアプリケーションのタイトル。
- developer.account.name ……プロビジョニングのタイプに基づいた、Apple での開発者のアカウント名（Apple のポータル認証画面に表示されます）。”iPhone Destribution:” や” iPhone Developer:” などの提供者のタイプに基づいた接頭辞が付加されます。
  たとえば、iOS のディストリビューションポートタルで認証されている証明書の名前が "Magic Software Japan K.K." の場合、開発者用の証明書となり、開発者名は ”iPhone Developer:Magic Software Japan K.K.” となります。
- provisioning.profile.filename …… これは、提供ファイルを作成した後に Apple Developer ポータルからダウンロードしたプロビジョニングファイルです（例：Magic_distribution.mobileprovision）。

ファイルのフォーマットは iOS 互換にする必要があります。これは、独自にファイルを作成する場合、行末のコードはラインフィード（’0A’）のみにする必要があり、Windows のように改行コードを含める（’0D 0A’）ではないことを意味しています。

起動画像とアプリアイコンの設定

Apple によって指定された基準に準拠するために、Magic xpa は source¥Media. xcassets フォルダの下にアセットカタログを提供しています。

このフォルダには 2 つのフォルダがあります。

- AppIcon.appiconset …… アプリのアイコンをカスタマイズするために使用します。
- LaunchImage.launchimage …… 起動イメージをカスタマイズするために使用します。

これらのイメージとアイコンは、Magic xpa が提供するデフォルトのイメージとアイコンであり、同じフォルダ内の contents.json ファイルで参照されます。

これら 2 つのフォルダのいずれかから画像をカスタマイズするには、2 つのオプションがあります。

- 既存の画像を同じ寸法のカスタム画像に置き換えることができます。この場合、画像名は以前のものと同じものが利用できます。
- 画像にカスタム名を付けたい場合は、既存のファイルをカスタム名に置き換える必要があります。さらに、対応する Json エントリを同じカスタム名で更新する必要があります。

iPhone と iPad デバイスでは、以下の画像サイズを選択することができます。。

デバイス	ポートレイト	ランドスケープ
iPhone		
iPhone 5,6		

デバイス	ポートレイト	ランドスケープ
1x	320 × 480 ps	不要
2x	640 × 960 ps	
Retain 4	640 × 1136 ps	
iPhone 7+		不要
2x	640 × 960 ps	
Retain 4	640 × 1136 ps	
iPhone iOS 8+		
Retain HD 5.5	1242 × 2208 ps	2208 × 1242 ps
Retain HD 4.7	750 × 1334 ps	不要
iPhone X/iPhone Xs		
iOS 11+	1125 × 2436 ps	2436 × 1125 ps
iPad		
iPad iOS 5,6		不要
1x	768 × 1024 ps	
2x	1536 × 2048 ps	
iPad Without Status Bar iOS 5,6		
1x	768 × 1004 ps	1024 × 758 ps
2x	1536 × 2008 ps	2048 × 1496 ps
iPad iOS 7+		
1x	768 × 1024 ps	1024 × 768 ps
2x	1536 × 2048 ps	2048 × 1536 ps

以下のファイルも同じように変更することができます。

- 実行特性 …… MagicApp フォルダ内の execution.properties ファイルを開き、実行値を変更します。
  - 実行特性の値か上記の定義情報を参照している URL のどちらかを定義する必要があります。
  - URL 特性を空のままにした場合、ダイアログボックスが開き、エンドユーザが URL を入力する必要があります。
  これは、異なるユーザが異なるサーバにログイン s できるような汎用的なアプリケーションで便利です。
  ダイアログボックスで URL を指定した後、iOS デバイスの設定画面で URL の値を変更することができます。
  build.sh コマンドを実行せずに Xcode を使用してクライアントをコンパイルした場合、設定画面でアプリケーションを表示するためには、Xcode から以下の操作を行う必要があることに注意してください。
  ［File］メニューから、［Add Files to "MagicApp"］を選択します。開いたダイアログボックスで Settings.Bundle フォルダを選択し、［Add］をクリックします。

変更をした後は、アプリケーションをコンパイルして、署名する必要があります。

カスタムアプリケーションをコンパイルして、署名するには、以下の手順を実行します。

1. Apple Developer プログラム / 証明書と配布プロフィール

- "iOS Developer Enterprise Program”のアカウントか、チームメンバーとして既存のアカウントに参加する必要があります。
  詳細は、以下を参照してください：https://developer.apple.com/jp/programs/ios/enterprise/
- “iOS Developer Program” の開発者アカウントでも同じように使用することができますが、開発者アカウントを使用して署名されるアプリケーションは、App ストアか特別な環境でのみ配信できます。
- 登録した後、アカウントに対応した証明書を作成して、Mac にダウンロードする必要があります。
  詳細は、以下を参照してください：https://developer.apple.com/ios/manage/certificates/team/howto.action
- 証明書を作成したら、アプリケーションを内部向けに配布するための配布ファイルを作成する必要があります。アカウントとリンクされた無制限のファイルを作成することができます。
  詳細は、以下を参照してください：https://developer.apple.com/ios/manage/provisioningprofiles/howto.action

2 署名された IPA ファイルの作成

署名された IPA ファイルを作成するには、build フォルダ内にある build.sh という名前のシェルスクリプトファイルを使用します。リッチクライアントインタフェースビルダで作成すると、以下のパス内にこのフォルダが作成されます。

[Magic xpa のインストールフォルダ]¥PublishedApplications¥[アプリケーション名]¥iOS

IPA ファイルを作成するには、以下の手順を実行してください。

- 変更された build フォルダを Mac にコピーします。
- Mac でターミナルシェルを開き、build フォルダに移動します。
- プロビジョニングファイル build フォルダにコピーします。
- 次のコマンドでスクリプトを実行します：sh buildxxxx.sh
  シェルスクリプトは、以下を使用します。それぞれ、利用可能な証明書が異なります。
  - AddHoc 用 …… buildAdHoc.sh
  - Enterprise 用 ……buildEnterprise.sh
  - AppStore 用 …… buildStore.sh

このスクリプトは、xcodebuild が Ver7 の場合のみ有効です。複数の Xcode がインストールされている場合は、Preferences/Location を開いて [Command Line Tools] で Ver7 以上を指定してください。

結果ファイルは、output フォルダに作成されます。

使用するプロビジョニングファイル と App ID や証明書の名前に整合性がない場合、ビルドどが正しく行われません。ビルドが正しく行われなかった場合、build.log ファイル内にエラー内容が記録されます。

例えば：

“Code Sign error: Provisioning profile 'XXX' specifies the Application Identifier 'YYY' which doesn't match the current setting 'ZZZ'”

この場合、Xcode は異なる開発者名で定義されているかもしれません。この名前を変更するには、Xcode でプロジェクトを開き、project properties 内の Code Signing セクションを開き、ここに定義されている開発者名で settings.properties ファイルを修正してください。

リソースファイルのパッケージ化

モバイルパッケージの作成時、リソースファイルをパッケージ化することで、インストール処理やアップグレード処理によってクライアント側のアプリケーション・キャッシュフォルダにリソースを展開することができます。これによって、実行時にリソースファイルをサーバから読み出すことなく Magic xpa プログラムで利用することができます。

この機能の主要な使用法は、最初の実行時にサーバからファイルを読み出す代わりに、パッケージのインストール時に、イメージやローカルデータベースを展開することです。

これらのファイルをパッケージ化するには、有効なフォルダの中にファイルを置く必要があります。これは、以下の「ファイル名の構成」のセクションで説明しているように、RIAModules¥Android¥Source と RIAModules¥iOS¥Source に位置づけられます。

- **他の（パックされない）ファイルについては、ファイルが使用される時に、クライアントがサーバのコピーとキャッシュのコピーの間でファイルの更新日付をチェックします。タイムスタンプが異なると、ファイルはサーバから読み込まれます。これは、実行サーバ上のファイルの更新日付がクライアントをビルドするために使用したマシンでの更新日付と同じにする必要があることを意味します。**
- **キャッシュされたファイルのフォルダ上で起動時に ServerFileToClient() 関数を使用することを推奨します。例：ServerFileToClient('image')。クライアント側のキャッシュファイルが最新かどうかを確認し、さらにクライアントがサーバにアクセスすることを防止します。**
- **（Android/iOS サブフォルダで位置付けられる）プラットホーム独自のリソースをパッケージ化することはができません。**
- **最初の起動をより早くするため、Magic xpa の内部ファイル（例：基本色ファイル）をパッケージ化することもできます。**

ファイル名の構成

assets フォルダにはサブフォルダを含めなければなりません。そして、Magic xpa プログラムで参照する場合と同じ構成のファイルを置く必要があります。例：

- images¥myimage.png の相対パスを使用して定義されるイメージでは、assets フォルダには image と呼ばれるサブフォルダを含めなければなりません。そして、それは myimage.png ファイルを含んでいなければなりません。
  例：assets¥images¥myimage.png
- c:¥images¥myimage.png のフルパスを使用して定義されるイメージの場合、assets フォルダは c_ という名前が付けられたサブフォルダを含まなければなりません。そして、このサブフォルダには image と呼ばれるサブフォルダを含まなければなりません。そして、ここには myimage.png ファイルを含まなければなりません。最初のフォルダ名を作成するとき、ドライブ文字の後のコロン（:）をアンダースコアと置き換えなければなりません。
  例：assets¥c_¥images¥myimage.png

この場合、Magic サーバ側のアプリケーションのフォルダ構成に合わせる必要があり、汎用的なモバイルアプリでは利用できません。

ネストされたサブフォルダを作成する代わりに、アンダースコアによって区切られたサブフォルダ名を連結した 1 つのサブフォルダを作成したり、名前の一部としてこのパスを持つようにイメージのファイル名を変更することもできます。

例えば、images¥orders¥contacts¥myimage.png のパスは、次のような、さまざまな規則に基づいて assets フォルダの配下に定義することができます。

- **元のファイル名とネストされたサブフォルダ名を使用する：**assets¥images¥orders¥contacts¥myimage.png
- **元のファイル名と 1 つのサブフォルダを使用する：**assets¥images_orders_contacts¥myimage.png
- **変更されたファイル名とネストされないサブフォルダ名を使用する：**
  assets¥images_orders_contacts_myimage.png

ファイルが適切に定義されたことを確認する

上記のように、すべてのファイルが最新であることを確認するために最初のプログラムで ServerFileToClient() 関数を使用して、ファイルを後で使用する場合のパフォーマンスを向上することが推奨されます。

ServerFileToClient() 関数の実行前後に［エラー］処理コマンドを追加することで、パッケージ化が正しく行われたことを確認することができます。タイムスタンプだけがチェックされ、ファイルは再びダウンロードされるため、これらの 2 つのメッセージの間隔は小さくしなければなりません。アクティブティモニターで確認することも可能で、ServerFileToClient() 関数によってサーバに 1 つのリクエストだけが送られることを確認することができます。

内部のファイルのパッケージ化

メインプログラムが開始されるまで、アプリケーションが読み込まれるまでの時間が長くなるケースがあります。

サイズの大きな環境ファイル（例えば、基本色ファイルや、メニュー、メインプログラム、コンポーネント）がある場合、このような現象が発生します。これらのファイルは、全てクライアントにダウンロードされる必要があります。そして、（ネットワーク帯域幅に基づいて）かなりの時間がかかるかもしれません。

モバイルパッケージでは、これらのファイルをパッケージ化することによって、この最初の負荷を改善することができます。この方法は、リソースファイルのパッケージ化の場合とよく似ています。

これらのファイルを見つけて、パッケージ化するには、以下の手順を実行してください。

1. サーバ側の RIA キャッシュフォルダの中身を削除してください。デフォルトでは、これはインストールフォルダ配下の RIACache フォルダになります。動作環境の［リッチクライアントのキャッシュパス］の設定で変更することができます。
2. 実行エンジンを使用してアプリケーションを開くか、クライアントを実行してアプリケーションに接続してください。
3. サーバの RIA キャッシュフォルダを開きます。このフォルダには、アプリケーションで必要なキャッシュファイル（例：myapp_ColorsTable_1.xml）が含まれています。assets フォルダ内のこれらのファイルを（リネームすることなく）取り出したり、配置したりすることができてます。これらはパッケージの一部で、クライアントはサーバに要求する必要はありません。

これらのファイルが変更するたびに、アプリケーションパッケージを更新する必要はありません。ファイルのいずれかが変更されると、クライアントによって自動的にダウンロードされます。

トラブルシューティング

初期起動時にアプリケーションの読み込みにまだ時間かかる場合、サーバ側の RIA キャッシュフォルダの内部ファイルの日付と aseset フォルダのファイルの日付が異なる可能性があります。アーカイブファイルの解凍ソフト（例：7-zip）を使用してモバイルパッケージを開くことで、確認することができます。

そして、パッケージ内の assets フォルダを開き、cachelist.txt ファイルを確認してください。このファイルは、ビルドプロセスの際に作成されます。

ファイルの各行には、リソースファイルの名前とそのタイムスタンプが含まれています。ファイルのタイムスタンプがサーバの RIA キャッシュフォルダ内のファイルの更新日付と同じかどうかを確認してください。

タイムスタンプが異なる場合、更新日付が変わっていないことを確認するまで、サーバの RIA キャッシュから assets フォルダへのリソースの再コピーが行われます。

モバイル用デバイスまたはシミュレータにアプリケーションをインストールする

以下の手順で、モバイルデバイスやシミュレータに RIA アプリケーションをインストールすることができます。

Android

Android にアプリケーションを配布するには、以下のような方法があります。

- APK ファイルを使用するデバイスで実行します（たとえば、E メールの添付ファイルとして APK ファイルを受け取り、それをクリックすることで実行できます）。
- APK ファイルが格納されたサーバにアクセスします。例：http: //192.168.13/PublishedApplications/myapp/my.apk.
  この処理を有効にするには、Web サーバの公開フォルダに APK ファイルを置き、APK ファイルのダウンロードをサポートするために以下のように Web サーバを設定する必要があります。
  IIS マネージャを開き、「Default Web Site」をアクセスします。
  ［MIME の種類］を開き、以下の設定を追加します。
  - **拡張子** …… .apk .
  - MIME **の種類** …… application/vnd.android.package-archive
- アプリケーションを Android マーケットにアップロードします。
- ADB コマンドラインユーティリティを使用する …… シミュレータ、または、ケーブルを経由してデバイスにインストールする場合に利用できます。Andorid SDK のインストールファイルダ（通常は :C:¥Program Files¥Android¥android-sdk）内の platform-tools サブフォルダにコピーされている ADB（Android Debug Bridge）ユーティリティを使用することで、APK ファイルをインストールすることができます。APK をインストールするには、以下のコマンドを実行してください: adb install .r my.apk

iOS

iOS 用のアプリケーションを配布するには、以下のような方法があります。

- App Store からダウンロードします。
- Xcode でアプリケーションを開き、ここから実行します。デバイスに接続してインストールする場合は、予め iOS Dev Center で利用するデバイスを登録する必要があります。
- IPA ファイルをダブルクリックして iTunes に追加します。一旦、iTunes に追加させたら、デバイスを接続して同期させるだけです。
- Web からアクセスします（"iOS Developer Enterprise Program" のアカウントでモジュールを作成する必要があります）。

Web からインストールするには

Web ブラウザ経由でのインストールを行うには、Web サーバ上の公開フォルダに以下のファイルを置く必要があります。

- 以下の行を含んだ Web ページ（.html) ファイル：
  `<a href="itms-services: //?action=download-manifest&url=http://example.com/manifest.plist">Install App</a>`
- アプリケーション（.ipa）ファイル
- マニフェスト（.plist）ファイル …… マニフェストファイル内の情報（例えばアプリ名やバージョン）を、カスタマイズの説明で記述された内容で上書きします。manifest.plist ファイルは、アプリケーションの IPA ファイルを参照しています。

ノート：

- %EngineDir%¥RIAModules¥iOS フォルダ内にあるサンプルファイル（install.html）には、この行が含まれており、このファイルを編集することでマニフェストファイルの位置を定義することができます。
- リッチクライアントインタフェースビルダも、これらのファイルを作成します。
- HTTPS 経由になるようにアプリのマニフェストファイルを定義してください。HTTP を経由してインストールしようとすると、以下のメッセージが表示されます。
  “Cannot install applications because the certificate is not valid”
  この問題を解決するために、” ドロップボックス ” を使用することができます。詳細は、『Magic xpa 逆引きガイド』の「ドロップボックスを使用して iOS のインストールファイルを公開するには」を参照してください。
- IPA ファイル名と PLIST ファイル名には空白を含めないでください。
- IPA と PLIST ファイルのダウンロードが可能になるように Web サーバを設定する必要があります。
  IIS マネージャを開いて、「Default Web Site」を選択し、［MIME の種類］を開き以下の通りに 2 つの項目を追加してください。
  - **拡張子** …… .ipa

- MIME の種類 …… application/octet-stream
- 拡張子 …… .plist
- MIME の種類 …… text/xml

アプリケーションのクライアントを配布したら、アプリケーションが変更する度に再配布する必要はありません。アプリケーションそのものは、クライアントの一部としてパッケージ化されません。他の Magic

RIA アプリケーションと同じように、更新されたアプリケーションがダウンロードされて、自動的に更新されます。つまり、Magic xpa のアップグレードによってクライアント側のバージョンをアップグレードしたいときだけ変更が必要となります。

トラブルシューティングとデバッグ

トラブルシューティング

RIA クライアントに関する最も一般的な問題は、クライアントが RIA アプリケーションを提供する Web サーバに接続することができないということです。

この問題が発生する要因としては以下のことが考えられます。

モバイル用デバイスが、Web サーバと通信できない。

この場合、MRB モニタにもリクエストが表示されず、Web サーバのログにも接続情報が記録されません。

- Wi-Fi を使用して通信している場合は、デバイスが正しい Wi-Fi ルータに接続していることを確認してください。
- デバイスの Web ブラウザで Web サーバに接続できるかどうかを試してください。例 : http : //192.168.137.1/Magic2xScripts

Web サーバから応答がない場合、Web サーバは正しく設定されていないか、ファイアウォールによって通信がブロックされています。

- ファイアウォールの http/https に対するインバウンド（外部から内部への通信）通信を許可するようにしてください。現象が改善されない場合は、一旦、ファイアウォールを無効にしてみてください。Web ブラウザから Web サーバに接続できたり、RIA アプリケーションを実行させることができるのであれば、ファイアウォールのポート 80/443 のインバウンド接続を有効にして確認してください。
- ポート 80 を有効にします。［コントロールパネル］から Windows ファイアウォールを起動し、［詳細設定］>［受信の規則］を開きます。World Wide Web サービス（HTTP トラフィック）を開き、「接続を許可する」を選択してください。

クライアントアプリケーションが、接続環境を定義したにもかかわらずサーバに接続することができない。

この場合、MRB モニタにはリクエストが表示されませんが、Web サーバのログにはサーバへのアクセスが記録されています。

- execution.properties ファイルが ANSI フォーマットになっていることを確認してください。
- execution.properties ファイルのすべてのパラメータがが正しい値（大文字と小文字を区別しています) になっていることを確認してください。
- settings.properties ファイルに外部実行ファイルを示す URL が定義されている場合は、以下の内容を確認してください。
  - **外部ファイルが、モバイルデバイスからアクセスすることができる。デバイス上の Web ブラウザーで同じ URL を入力することで確認することができます。**
  - **外部ファイルは ANSI フォーマットで、このファイルのすべてのパラメータに正しい値（大文字と小文字を区別しています) が設定されている。**
- Magic RIA サーバが起動されており、環境特性で定義されるプロジェクトが実行されていることを家訓してください。

Windows 上でファイルをチェックする場合は、実行特性ファイルを”execution.properties”という名前で、%EngineDir%\RIAModules\Desktop フォルダにコピーし､そのフォルダ上で MgxpaRIA.exe アプリケーションを実行してください。

設定内容が正しい場合、RIA アプリケーションが Windows 上で実行されます。

デバッグ

Android

Windows の RIA と同じように、Magic.ini ファイルの［MAGIC_RIA］セクションで internalloglevel パラメータに適切な値を定義することによって、モバイルクライアントでのデバッグログを有効にすることができます。

アプリケーションログは、内部のデバイスログに書き込まれます。

Android のデバッグブリッジ（ADB）ユーティリティ（Android SDK フォルダの Platform-tools フォルダ : 通常は ”C:¥Program Files¥Android¥android-sdk¥platform-tools″ にあります）を以下のように使用することでログを参照することができます。

モバイル用デバイス上で ADB ユーティリティを使用するには、モバイル用デバイスが以下に定義される必要があります。

- デバッグモードで実行します。
  これを可能にするには、［設定／アプリケーション／開発］に移動して、［USB デバッグ］をチェックしてください。
- 開発中のアプリケーションのインストールを有効にします。
  これを可能にするには、［設定／アプリケーション］に移動して、［提供不明のアプリ］をチェックしてください。

デバイスのログを参照する場合は、以下のコマンドを実行してください : adb logcat

デバイスログ内で RIA クライアントに関連する内容のみ参照したい場合は、以下のコマンドを実行してください : adb logcat MAGIC_DEBUG:D * : S

このフィルタリングによって異常終了や不正な動作を実行しないことに注意してください。

ログを消去するには、以下のコマンドを実行してください：adb logcat-c。

ログをファイルに保存するには、以下のコマンドを実行してください：adb logcat > file.log。

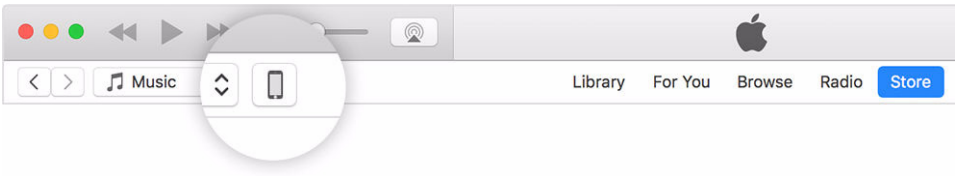
指定したシナリオまたは異常終了のログを提供する必要がある場合、最初にログを消去した後、異常終了の現象を発生させるアクションを実行し、ファイルにログを保存する必要があります。

ADB ユーティリティに関する詳細は、Using the Android Debug Bridge (ADB) Utility. を参照してください。

iOS

ログの設定は、RIA の実行特性ファイルで指定します。

アプリケーションログは、iTunes を使用している PC で読み出すことができます。



デバイスを iTunes に接続し、デバイスアイコンをクリックします。デバイス上の［App］を選択すると、［App］（［ファイル共有の下に表示されています）に実行中の RIA クライアントアプリケーションの名前が表示されています。この名前を選択すると、左側（「XXXX の書類」と表示されます）に実行特性ファイルの "InternalLogFile" で定義されたファイル名が表示されます。

［保存先］ボタンをクリックすることで、Windows の任意のフォルダにファイルをコピーすることができます。

iOS デバイス内に作成されたログファイルは、iTune 上の「XXX の書類」に表示されたファイル名をマウスで選択した状態で Delete キーを押下することで削除できます。

Android で実行

［デバッグ］メニューの［Android で実行］  をオンにすることで、Android デバイスまたはシミュレーターでプログラムを実行させることができます。

このエントリがオンの場合：

- Magic xpa Studio でリッチクライアントプログラムを実行（F7）すると、エンジンが実行モードに切り替わり、開始プログラムとして選択されたプログラムがあるモバイルデバイスまたはシミュレータ上でアプリケーションが起動されます。
- Magic xpa Studio でプロジェクトを実行（Ctrl+F7）すると、エンジンが実行モードに切り替わり、execution properties ファイルで定義された開始プログラムを使用しているモバイルデバイスまたはシミュレータ上でアプリケーションが起動されます。

このように動作させるには、最初に以下のように設定する必要があります。

1. モバイルデバイスまたはシミュレータにアプリケーションをインストールします。製品に添付している Myapp.apk ファイルでインストールすることができます。このファイルは RIAModules¥Android フォルダにインストールされています。アプリケーションをカスタマイズして、Android デバイスにインストールする方法については、「アプリケーションのカスタマイズ」（ページ：27）や「モバイル用デバイスまたはシミュレータにアプリケーションをインストールする」（ページ：37）を参照してください。
2. アプリケーションがプロジェクトを実行している Magic xpa サーバに接続できるようにするために、デバイスをネットワークに接続します。
3. Android の［設定］メニューの［開発者向けオプション］から、［USB デバッグ］オプションを有効にしてください。
4. デバイスを使用する場合、デバイスを USB ケーブルを使用して PC に接続します。場合によっては、汎用的な Google USB ドライバ（Android SDK マネージャーから、Extrs フォルダ内の Google USB ドライバーパッケージをインストールします）またはデバイス用のドライバをインストールする必要があります。
5. Windows 側の［デバイスマネージャー］で無効なデバイスとして表示されている場合は、ドライバをインストールする必要があります。その際、Android SDK のインストールフォルダ内の extras¥google¥usb_driver¥android_winusb. inf を指定してください。
6. インストレーションフォルダの RIAModules¥Android フォルダにコピーされている Run.bat ファイルのアプリケーションのパッケージ名を定義します。これを行うには、テキストエディタでファイルを開き、PackageName の値をビルドしたアプリケーションのパッケージ名に変更します。パッケージ名は、以下の値を指定します。
   - **build.cmd を使用してビルドする場合 …… settings.properties ファイルに定義されている値**
   - **インタフェースビルダを使用してビルドした場合 …… ［Andorid の設定］ダイアログの［パッケージ名］に設定された値になります。**

   たとえば、RIAModules¥Android フォルダ内の Myapp.apk を使用する場合、パッケージ名は ”my.application.full” に設定されています。

パッケージ名は、全て小文字で指定してください。大文字が含まれていると、この機能は動作しません。

デバイスを PC に接続すると、デバイス側で「USB デバッグを許可しますか？」という確認ダイアログが表示されます。ここで、［OK］をクリックするとデバイスにアクセスできるようになります。接続しても、このダイアログが表示されない場合は、デバイスドライバのインストールなどが行われていない可能性があります。

PC がデバイスにアクセスすることができるかどうかをチェックするには、コマンドプロンプトを開き、インストレーションフォルダ内の RIAModules¥Utils¥ADB フォルダにアクセスして、以下のコマンドを実行してみてください。

```
adb devices
```

正常に接続されていれば、以下のように表示されます。

```
List of devices attached
XXXXXXXXXXX        device
```

Android デバイスまたはシミュレーターでのアプリケーションの実行は、Android Debug Bridge (ADB) ユーティリティを使用して行われます。

インストレーションフォルダ内の RIAModules¥Android フォルダにコピーされている Run.bat ファイルを修正することで、ADB コマンドを変更することができます。

Magic xpa が実行しているサーバが DNS に定義されていない場合、ホスト名を IP アドレスに変換できないため、Android クライアントは Magic xpa サーバへのアクセスが失敗しエラーを返します。

このような場合、［動作環境］ダイアログの［HTTP リクエスタ］で IP アドレスを設定する必要があります。

例：http：//192.168.0.111/Magic3xScripts/MGrqispi.dll

Android におけるストレージの強化

Magic xpa の Android 11 (API-30) をターゲットとするモバイルアプリケーションには、スコープドアクセスが与えられ、外部ストレージにアクセスすることができます。

つまり、これらのアプリケーションは、アプリケーション固有のディレクトリや外部ストレージに作成した特定のメディアのみにアクセスすることができます。

スコープドストレージとは？

スコープドストレージとは、最も関連性の高いディレクトリやメディアファイルへのアクセスを許可することです。外部ストレージにあるアプリケーションとユーザデータの保護を強化することを目的としています。スコープ付きストレージは、Android のユーザとして、ファイルを制御し、整理することができるより良い方法です。

詳しくは、こちらをご覧ください。

https://developer.android.com/about/versions/11/privacy/storage

スコープ付きストレージは、以下の 2 箇所で有効です。

- ClientFile 関数
- ［コール OS］処理コマンド

ClientFile 関数によるスコープ付きストレージの変更

関数	スコープ付きストレージの実装前 (すべての Android バージョン)		スコープ付きストレージの実装後 (すべての Android バージョン)	
	アプリケーションキャッシュ (内部 / 外部)	共有 / 外部ストレージ	アプリケーションキャッシュ (内部 / 外部)	共有 / 外部ストレージ
ClientFileCopy	✔	✔	✔	✘
ClientFileDelete	✔	✔	✔	✘
ClientFileRename	✔	✔	✔	✘
ClientFile2Blb	✔	✔	✔	✘
ClientFileExists	✔	✔	✔	✘
ClientFile2Server	✔	✔	✔	✘
ClientFileListGet	✔	✔	✔	✘
ClientBlb2File	✔	✔	✔	✘

[コール OS コマンド] によるスコープ付きストレージの変更

処理コマンド	スコープ付きストレージの実装前 (すべての Android バージョン)		スコープ付きストレージの実装後 (すべての Android バージョン)	
	アプリケーションキャッシュ (内部 / 外部)	共有 / 外部ストレージ	アプリケーションキャッシュ (内部 / 外部)	共有 / 外部ストレージ
[コール OS コマンド] による PDF	✔	✔	✔	✖

凡例 :

サポート	✔
未サポート	✖

表に記載されているすべてのファイル関数のファイルパスは、関数のソースおよび宛先パラメーターと見なされます。

サポートバージョン : 4.8.1

Android Debug Bridge (ADB) ユーティリティを使用する

Android Debug Bridge (ADB) のコマンドラインユーティリティを使用することで、接続されたデバイスを Android エミュレータまたは Android のインスタンスにアクセスすることができます。

ADB ユーティリティは、SDK フォルダの platform-tools フォルダにコピーされており、通常は以下のフォルダ内になります。

```
C:¥Users¥<UserId>¥AppData¥Local¥Android¥Sdk¥platform-tools
```

そして、インストールフォルダ内の RIAModules¥Utils¥ADB サブフォルダにもあります。

ADB ユーティリティは、以下の場合に役立ちます。

- モバイルデバイスまたはシミュレータにアプリケーションをインストールする。……「モバイル用デバイスまたはシミュレータにアプリケーションをインストールする」（ページ：37）を参照してください。
- クライアントのログを参照する。……トラブルシューティングとデバッグを参照してください。
- シミュレータまたはモバイル用デバイス上で直接プログラムまたはプロジェクトを実行する。……「Android　で実行」（ページ：42）を参照してください。

モバイル用デバイス上で ADB ユーティリティを使用するには、モバイルデバイスが以下のように定義されていることを確認してください。

- デバッグモードで実行する。…… これを有効にするには、Android の［設定］メニューから［開発者向けオプション］を選択し、［USB デバッグ］をチェックします。adb shell ip -f inet addr
- 非マーケット対応のアプリケーションのインストールを有効にします。…… これを有効にするには、［設定］メニューから［セキュリティ］を選択し、［提供元不明のアプリ］をチェックします。

ADB ユーティリティは、デバイスを USB ケーブルや Wi-Fi 経由で PC に接続して使用することができます。Wi-Fi 上で使用するには、後述する定義内容にもとづいていくつかの設定を行う必要があります。

ADB ユーティリティがデバイスに接続することができるかどうかを確認するには、単にデバイスを USB ケーブルを使用して PC に接続して、以下のコマンドを実行してください。

```
adb devices
```

リストにデバイスの一覧が表示されます。

以下も参照してください：http://developer.android.com/tools/help/adb.html

Android デバイスが Windows で認識されない場合は、デバイス一覧は表示されません。Windows のデバイスマネージャーで確認した上で、Android SDK の USB ドライバ（Android¥sdk¥extras¥google¥usb_driver）を Windows にインストールしてください。

Wi-Fi 経由で ADB を使用する

以下の手順に従うことで、Android 4.x 上で Wi-Fi 経由で ADB ユーティリティを使用することができます。

1. デバイス上で USB のデバッグを有効にします。
2. USB ケーブルを使用して Android デバイスを PC に接続します。
3. Wi-Fi 経由でデバイスと PC をネットワークに接続します。
4. デバイスの［設定］メニューの［Wi-Fi］の接続ステーション名か、以下のコマンドを使用してデバイスの IP アドレスを確認します。

   ```
   adb shell ip -f inet addr show wlan0
   ```

5. 以下のコマンドを実行することで、TCP/IP 経由で ADB を有効にします。

   ```
   adb tcpip 555
   ```

   このコマンドを実行すると以下のメッセージが表示されます。

   ```
   restarting in TCP mode port: 5555
   ```

   そしてデバイス側で USB でのデバッグを確認するダイアログが表示されます。
6. デバイスを PC から切り離します。
7. 以下のコマンドを実行して、ADB サーバを停止します。

   ```
   adb kill-server
   ```

8. 以下のコマンドを実行して、Wi-Fi 上の PC をデバイスに接続します。

   ```
   adb connect <your device's IP address>
   ```

このコマンドを実行すると以下のメッセージが表示されます。

```
connected to <your device's IP address>
```

これで Wi-Fi 経由で ADB を使用することができます。動作を確認するには、以下のコマンドを実行してください。

```
adb devices
```

9. USB モードに戻るには、以下のコマンドを実行してください。

```
adb usb
```

モバイルアプリのパフォーマンス改善

モバイルアプリケーションの性能を向上させるには、以下のような対応が必要です。

冗長なメニューエントリを削除する

新規に作成するプロジェクトには、" デフォルトプルダウンメニュー " が作成されます。これは、MDI アプリケーション用に設計されたものです。モバイルデバイスには MDI が存在しないため、モバイルアプリケーションを開発する場合は、このデフォルトエントリを削除することを推奨します。

冗長なコンポーネントを削除する

新規に作成するプロジェクトには、エンドユーザー機能コンポーネントが作成されます。このコンポーネントは、通常モバイル用アプリケーションでは必要としません。したがって、モバイルアプリケーションを開発する際、このデフォルトエントリを削除することを推奨します。

[テーブル] コントロールやテーブルに配置されたコントロールで透過色の使用を避ける

テーブルをスクロールする際のパフォーマンス低下を引き起こす可能性があるため、透過色の使用は推奨されません。

[テーブル] コントロール上で代替色の使用を避ける

テーブルをスクロールする際のパフォーマンス低下を引き起こす可能性があるため、代替色の使用は推奨されません。

Xcode でビルドする場合は、この修正だけで十分ですが、コマンドラインでビルドする場合は、以下を参照してください。