MZ Platform

アプリケーションビルダー操作説明書

= Application Builder Operating Manual =

Revision 1.6 [MZ Platform.1.6]

National Institute of
Advanced Industrial Science
and Technology
AIST

独立行政法人
産業技術総合研究所

＝目次＝

1. 概要

設計・製造支援アプリケーションのための共通プラットフォーム研究開発では、ソフトウェアのコンポーネント化（部品化）を行い、システム構築や変更を利用者自身による組み立て作業によって実現することを最終目的としています。共通プラットフォームは様々なシステムで共通に使用される基本コンポーネントと、それら部品を使用してシステムを構築／利用するための環境を提供します。

1.1. プラットフォームの概要

本プラットフォーム上ではソフトウェアをコンポーネント化することで、ソフトウェアの保守性を高めるだけでなく、コンポーネントの接続／構成を容易に、かつ動的に行うことによって、アプリケーションシステム全体をより拡張性のあるものにします。

プラットフォーム上のアプリケーションは機能単位に分割されたコンポーネントによって構成され、コンポーネント間は互いに依存性の無い形で関係付けを行います。プラットフォームの基本アーキテクチャを下図に示します。

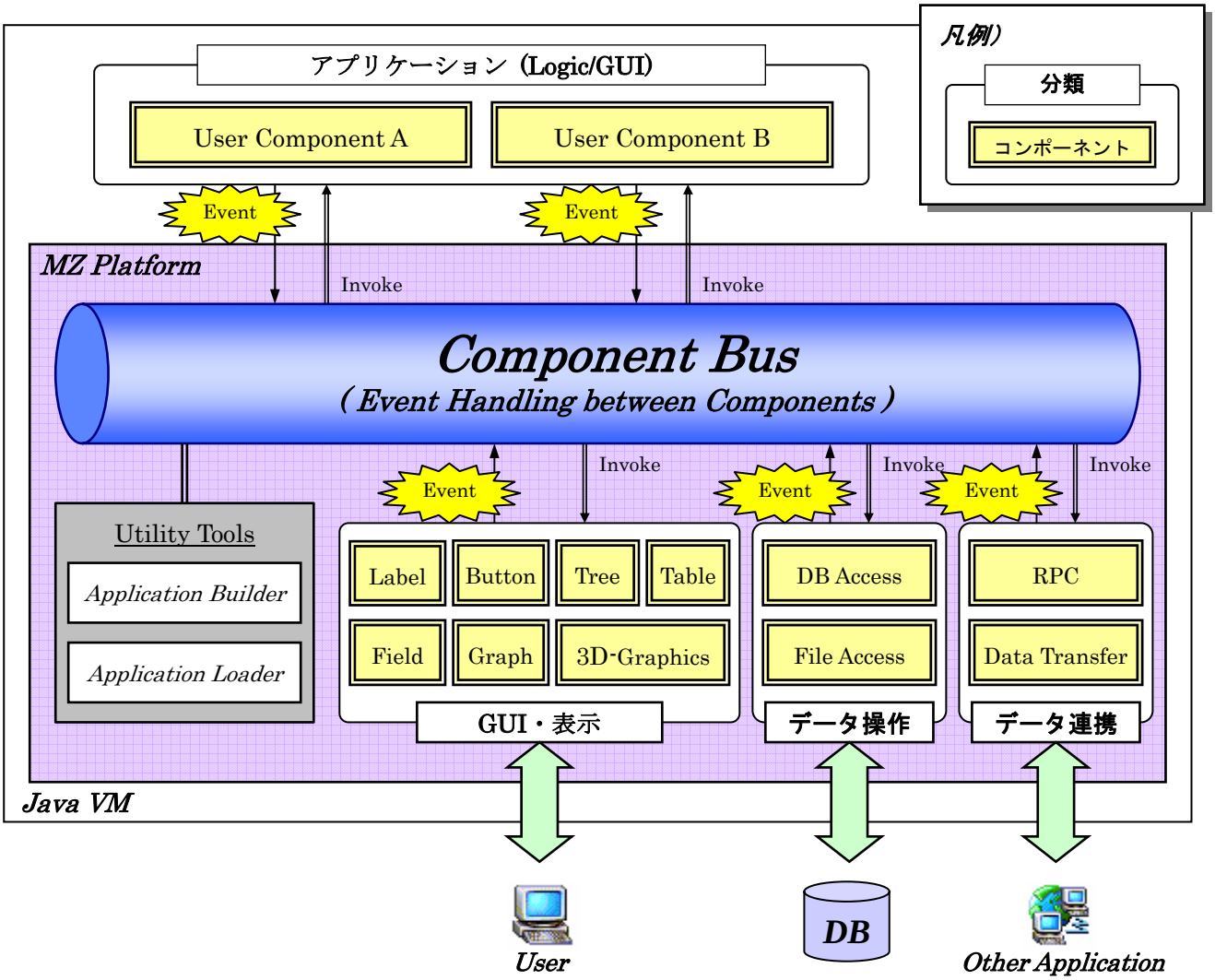


図 1 プラットフォーム基本構造

1.2. コンポーネントの構築

本プラットフォームでは互いに独立したコンポーネント間を接続する仕組みを提供し、各コンポーネント単位での保守性／再利用性を確保します。これを実現させるために、各コンポーネントは以下のルールに従って構築するものとします。

1)Bean として構築

Java のコンポーネント実装である JavaBeans として構築します。[1]

・引数なしのコンストラクタを実装
・直列化が可能（Serializable）
・転送イベントモデル（Delegation Event Model）の使用
・JavaBeans 構成規則に準拠
（プロパティに対する get/set メソッドの提供やイベント操作メソッドの命名規則など）

2)共通インターフェイスの実装

プラットフォームが提供する『コンポーネントインターフェイス』を実装します。

3)データ構造

コンポーネント間で交換するデータの構造は、プラットフォームが提供するものに限定します。

4)イベント

コンポーネントから発生させるイベントは、プラットフォームが処理対象とするイベントのみに限定します。

5)その他

・表示／出力のマルチロケール対応（多国語対応）
・他のコンポーネントの機能／データ構造に非依存

1.3. コンポーネント間の接続

JavaBeans の規定に従い、コンポーネント間の接続は転送イベントモデルを使用し、すべての連携はイベント発生をトリガーにした処理起動によって行われます。プラットフォームはあるコンポーネントからイベントを受け、他のコンポーネントの処理を起動します。コンポーネント接続のための機能として、プラットフォームは以下の機能を提供します。

1)接続関係の定義

コンポーネント間の接続関係を定義するためのツールとして、アプリケーションビルダーを提供します。このツールは GUI や簡易定義言語などを利用して、プログラムを書かずにアプリケーションを構築することができるユーティリティです。

2)動的な処理起動

コンポーネントの接続はアプリケーション実行中でも変更可能とし、アプリケーションの実行を止めることなく仕様変更／動作確認が可能です。そのために、コンポーネント間接続関係はプログラムソース内に埋め込まず、実行時のデータとして管理することによって、動的に処理制御を変更することができるようにします。

[1] JavaBeans の詳細規定については http://java.sun.com/products/javabeans/docs/spec.html を参照

2. MZ Platform

本プラットフォームは、アプリケーション構築のベースとなるコンポーネント開発を支援するための開発環境を提供し、コンポーネント開発におけるフレームワークを定めるものです。（下図太枠が提供範囲）

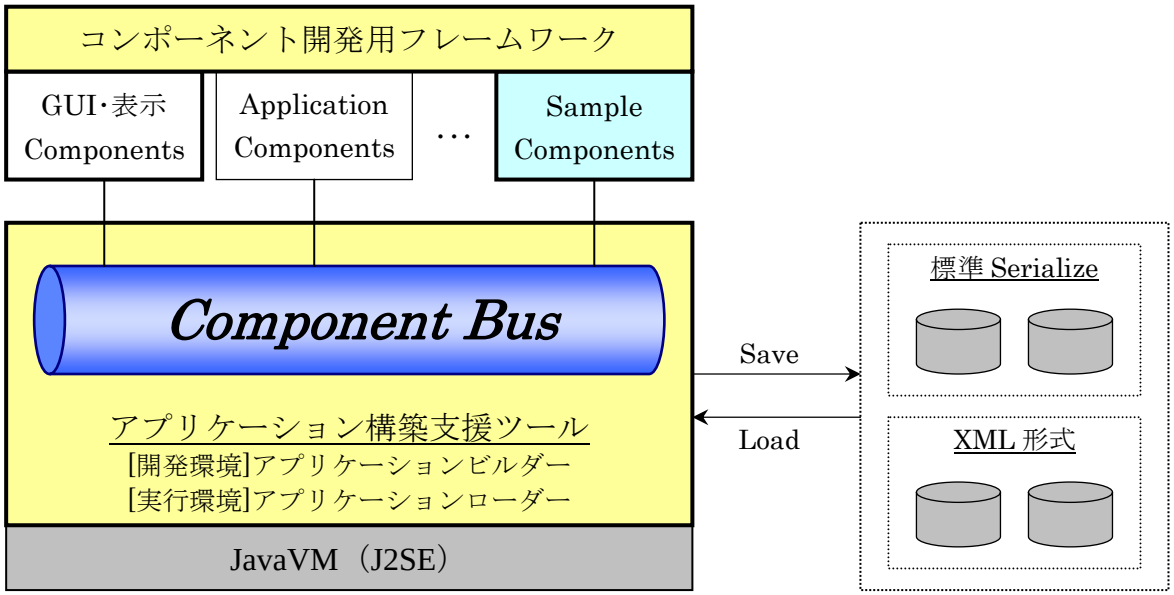


図 2 MZ Platform 提供範囲

本プラットフォームでは以下を提供します。

1)コンポーネント開発用フレームワーク

コンポーネント構築のルールに基づいた、共通クラス／インターフェイス群を提供します。また、コンポーネント開発用に必須メソッドなどを記述した、テンプレートソースを提供します。

2)アプリケーション構築支援ツール

①アプリケーションビルダー

ルールに基づいて構築されたコンポーネントをアプリケーションとして組み立てる機能をもつユーティリティツールを提供します。このツール上ではコンポーネントの貼り付け／属性変更、画面レイアウト設定、コンポーネント間の接続が可能です。また、ここで作成したアプリケーションはローカルファイルに保存し、再利用が可能です。

②アプリケーションローダー

アプリケーションビルダーによって構築／保存された Java-Application を、ファイルからロードし、実行します。

3)Component-Bus

アプリケーションでのコンポーネント管理／コンポーネント接続を行うための、プラットフォーム基幹機能です。

4)サンプル

コンポーネント開発のサンプルとして、サンプルコンポーネントソースを提供します。

3. 用語説明

1)アプリケーション

ある特定の仕事を行うためのソフトウェア。それ自身がある業務の機能をはたす。

■関連用語

プログラミング言語

アプリケーションを作成するために使用する言語。通常のアプリケーションはプログラミング言語によって”プログラム”という単位のソフトウェアを作成し、小さなプログラムの集まりでアプリケーションという 1 つの機能を果たすまとまりとなる。

2)コンポーネント

アプリケーションを構成するソフトウェア部品。コンポーネントはそれだけである機能を提供しており、それぞれの間に依存関係はない。MZ Platform では、アプリケーションを構築するための最小単位の部品をコンポーネントと呼び、すべてのアプリケーションはコンポーネントの組み合わせによって構成されている。なお、コンポーネント自体はプログラムによって作られている。

■関連用語

GUI コンポーネント

グラフィクス・ユーザ・インターフェイスの略。文字での表示／入力だけでなく、表やグラフのようにより見やすく表示し、マウス操作などでの簡単な操作を提供するもの。

コンテナコンポーネント

GUI コンポーネントをまとめてグループとして扱うための部品。例えばウィンドウも複数の GUI コンポーネントをまとめて一つの枠内で表示するためのコンテナコンポーネントである。主なコンテナコンポーネントはウィンドウとパネル。

ユーティリティコンポーネント

数値演算、集計など、それ自身は画面に表示されずに裏で処理をおこなう部品。

コンポーネント属性

コンポーネントがもっている性質。属性はコンポーネント毎にそれぞれ提供されており、例えば GUI コンポーネントであれば、表示する色や大きさなどが属性となる。

3)イベント

コンポーネントの状態の変更を外部に伝える機能。イベントには多くの種類があり、コンポーネントによって発生するイベントが異なる。MZ Platform では、アプリケーションの動作はすべてこの”イベント”の発生をきっかけに行われており、アプリケーションの構築作業は、イベントが発生したときの振る舞いを指定することで行う。イベントの種類についてはチュートリアル付録の『付録 D. イベント』を、各コンポーネントから発生するイベントについてはチュートリアル付録の A～C のコンポーネント紹介を参照。

■関連用語

イベント発生元コンポーネント

イベントを発生させるコンポーネント。アプリケーション構築画面上のコンポーネント接続関係において、左側に位置するコンポーネントであり、イベント処理追加の操作によってイベントに対する処理を追加する。

接続先コンポーネント

イベント発生元コンポーネントから発生したイベントによって起動される処理をおこなうコンポーネント。アプリケーション構築画面上のコンポーネント接続関係において、右側に位置するコンポーネントである。

4)メソッド

コンポーネントに対して外部から操作するための機能。メソッドはコンポーネントに対して外部から処理を呼び出すための唯一の方法であり、コンポーネントの機能によってさまざまなメソッドが提供されている。各コンポーネントが提供しているメソッドについては、チュートリアル付録の A～C のコンポーネント紹介を参照。

■関連用語

メソッド引数

メソッドを起動する際に、外部から与える情報。例えば足し算を行うメソッドに対して、計算対象として与えられる 2 つの数字が“引数”である。

メソッド引数取得方法

メソッド引数の値を指定する形式。引数取得方法には以下の 6 つがあり、引数となる値を持っている対象によってどれかを選択する。

①固定値

メソッド引数に固定の数値や文字列を渡す形式。実行するときの状態に関係なく、常に同じ値が渡される。

②メソッド戻り値

メソッド引数に他のコンポーネントのメソッド戻り値を指定する形式。引数として渡したい情報を他のコンポーネントがもっている場合には、この形式によってコンポーネント内のデータを取得して引数として渡す。

③コンポーネント

メソッド引数にコンポーネント自体を指定する形式。引数として渡したい情報がコンポーネント自身の場合には、この形式によって任意のコンポーネントを引数として渡す。

④イベント内包

メソッド引数にイベントに含まれているデータを指定する形式。例えば“データ変更イベント”には、変更情報がイベントに含まれており、その変更情報をメソッドに渡す必要がある場合には、この形式によってイベントに含まれている情報を引数として渡す。

⑤イベント

メソッド引数にイベントそのものを指定する形式。イベントそのものを引数として受け取ることができるメソッドに対しては、発生したイベントを引数として渡す。

⑥メソッド処理結果

メソッド引数にすでに処理の終わっているメソッドの処理結果（戻り値）を指定する形式。処理結果を次々に引き渡すような場合には、この形式によって処理結果データを引数として渡す。

5)アプリケーションローダー

MZ Platform が提供するアプリケーションを実行するためのツール。アプリケーションビルダーで構築されたアプリケーションを実行するためのもので、組み立てられたアプリケーションを変更することはできない。

6)アプリケーションビルダー

MZ Platform が提供するアプリケーション構築を行うためのツール。コンポーネントを組み合わせてアプリケーションを構築する作業を、マウス操作などの画面操作によって行う。

■関連用語

実行

構築したアプリケーションを実際に動かす操作。アプリケーションビルダーから［実行］ボタンを押下して起動。なお、アプリケーションの実行は、アプリケーションコンポーネントの『アプリケーション開始イベント』に接続されている処理から実施される。

実行（設定可）

実行しながら GUI コンポーネントの属性を編集できる実行形式。アプリケーションビルダーから［実行（設定可）］ボタンを押下して起動。属性設定方法は GUI 部品で準備されており、多くはポップアップメニューによる属性設定を提供している。

画面編集

GUI コンポーネントの配置を行い、画面のレイアウトを設定する機能。アプリケーションビルダーから［画面編集］ボタンを押下して起動。画面配置の方法には以下の 5 つがあり、表示したいレイアウトにあわせて好きな形式を選択する。

①手動配置

GUI 部品を自由な位置に配置できる形式。位置はマウスによって自由に移動することが可能。

②横方向整列

GUI 部品を横方向に一列に並べる形式。表示範囲の横幅が決まっている場合、横いっぱいになったところで折り返す。

③縦方向整列

GUI 部品を縦方向に一列に並べる形式。表示範囲の縦幅が決まっている場合、縦いっぱいになったところで折り返す。

④領域配置

GUI 部品を四方（東／西／南／北）、中央の 5 方向に配置する形式。配置される GUI 部品は表示範囲全体の大きさにあわせて拡大／縮小される。

⑤矩形分割配置

表示範囲全体を N×M の矩形に分割し、その左上から順に配置する形式。配置される GUI 部品の大きさによって、分割される領域は調整される。

帳票編集

アプリケーションから出力する帳票のレイアウトを設定するための機能。アプリケーションビルダーから［帳票編集］ボタンを押下して起動。帳票の要素として指定できるものには以下の 3 つがある。

①ラベル形式

文字列 1 つについて表示するための帳票要素。文字列長に制限はなく、描画領域にあわせて折り返して描画される。また、文字列中に改行コードがある場合、その位置にて改行される。

②テーブル形式

表形式で描画するための帳票要素。テーブル内の各セルの描画は、すべて横幅にあわせて折り返して描画される。また、文字列中に改行コードがある場合、その位置にて改行される。

③GUI 画面要素

GUI 画面のイメージをそのまま描画するための帳票要素。イメージの描画は横幅にあわせて調整され、縦方向のサイズは縦横比率を維持した状態で自動調整される。

ロード 保存されたアプリケーションをアプリケーションビルダー上に読み込む機能。アプリケーションビルダーから［ロード］ボタンを押下して起動。
挿入 保存されたアプリケーションを現在編集中のアプリケーションに追加する機能。アプリケーションビルダーから［挿入］ボタンを押下して起動。
保存 構築したアプリケーションを外部ファイルに保存する機能。アプリケーションビルダーから［保存］ボタンを押下して起動。保存したデータはロード機能により再度アプリケーションビルダー上に読み込むことができる。
上書き保存 保存データをロードした状態、または一度保存した状態で、再度同じファイルにアプリケーションを保存する機能。アプリケーションビルダーから［上書き保存］ボタンを押下して起動。
XML 入力 出力された XML ファイルからアプリケーションをアプリケーションビルダー上に読み込む機能。アプリケーションビルダーから［XML 入力］ボタンを押下して起動。
XML 出力 構築したアプリケーションを XML ファイルに出力する機能。アプリケーションビルダーから［XML 出力］ボタンを押下して起動。
クリア 構築したアプリケーションをすべてクリアし、初期状態に戻す機能。アプリケーションビルダーから［クリア］ボタンを押下して起動。

7)データ構造

アプリケーション上で取り扱うデータには、数値や文字列のように単純に 1 つの値で表現されるものもあれば、複数の値を合わせて構造をもたせた形で表現されるものもある。MZ Platform が提供する最も基本的なデータ構造は、リストデータ構造、表データ構造、ツリーデータ構造の 3 つである。

■関連用語
リスト構造 データが一列に並んだデータ構造。一次元配列。N 個のデータを扱う場合、このリスト構造の長さは N となる。
テーブル構造 データが N×M に並んだデータ構造。2 次元配列。 列　：表データ構造で項目を示すもの。通常の表記では横方向にならぶ単位が列となる。 行　：表データ構造でデータ 1 件を示すもの。通常の表記では縦方向にならぶ単位が行となる セル：表データ構造で 1 つの最小単位のデータ枠を示すもの。セルは○行○列と表現される
ツリー構造 データに階層構造があり、親子関係をもった構造。1 つの親から N 個の子が関係付けられており、最上位階層は 1 つ。

4. プラットフォーム稼動環境構築

4.1. 稼働環境の提供

プラットフォーム稼働環境として、以下のファイルを提供します。

項目	ファイル名	内容
JAR ファイル	jars¥mzplatform.jar	プラットフォーム本体 JAR ファイル ・ComponentBus ・アプリケーショビルダー／ローダー ・イベント関連オブジェクト ・共通データ構造 ・コンポーネントフレームワークオブジェクト ・基本コンポーネント
実行ファイル（Windows 用）	PFBuilder.exe	アプリケーションビルダーの実行モジュール
	PFBuilder_con.exe	コンソール画面表示版アプリケーションビルダー
	PFLoader.exe	アプリケーションローダーの実行モジュール
	PFLoader_con.exe	コンソール画面表示版アプリケーションローダー
設定ファイル	etc¥Platform.ini	実行時設定ファイル
	etc¥PlatformClassPath.ini	Java クラスパス設定ファイル
	etc¥PlatformComponents_ja	コンポーネント一覧ファイル（日本語）
	etc¥PlatformComponents_en	コンポーネント一覧ファイル（英語）
	components	コンポーネント情報ファイル群

4.2. 稼働環境の設定

1)Java 環境変数の設定

MZ Platform は実行時に Java を使用します。MZ Platform とともに Java 実行環境をインストールした場合には環境変数の設定は必要ありませんが、既にインストールしてある Java 実行環境を使用する場合には、Java のインストールフォルダを取得するために、Java のインストールフォルダを取得するために、環境変数 JAVA_HOME を参照しますので、必ず設定してください。

環境変数：JAVA_HOME

設定値　：J2RE 1.4.2_03（または J2SDK 1.4.2_03）が導入されているフォルダ

2)Java クラスパスの設定

Java が実行時に使用するクラスの参照パスを設定します。MZ Platform に対するクラスパスの設定は、MZ Platform インストールフォルダにある、クラスパス設定ファイル "PlatformClassPath.ini" を編集します。このファイルには、1 行につき 1 つのパスを設定し、パスにはローカルマシンの jar/zip ファイル、およびディレクトリを指定することができます。クラスパスの優先順は記述されている順となります。また、行の初めが # である場合、その行はコメントと見なされます。

■PlatformClassPath.ini

```
# MZ Platform ClassPath
jars¥mzplatform.jar

# MZ Checker ClassPath
jars¥MZChecker¥mzchecker.jar
```

↑ 必要に応じて Jar ファイルやフォルダを追加

3)Platform.ini

プラットフォームの動作設定は、初期設定ファイル(導入フォルダ¥etc¥Platform.ini)にて行います。初期設定ファイルのパラメタの内容、設定方法については、“詳細設定説明書”(導入フォルダ¥docs¥manual 内)を参照してください。

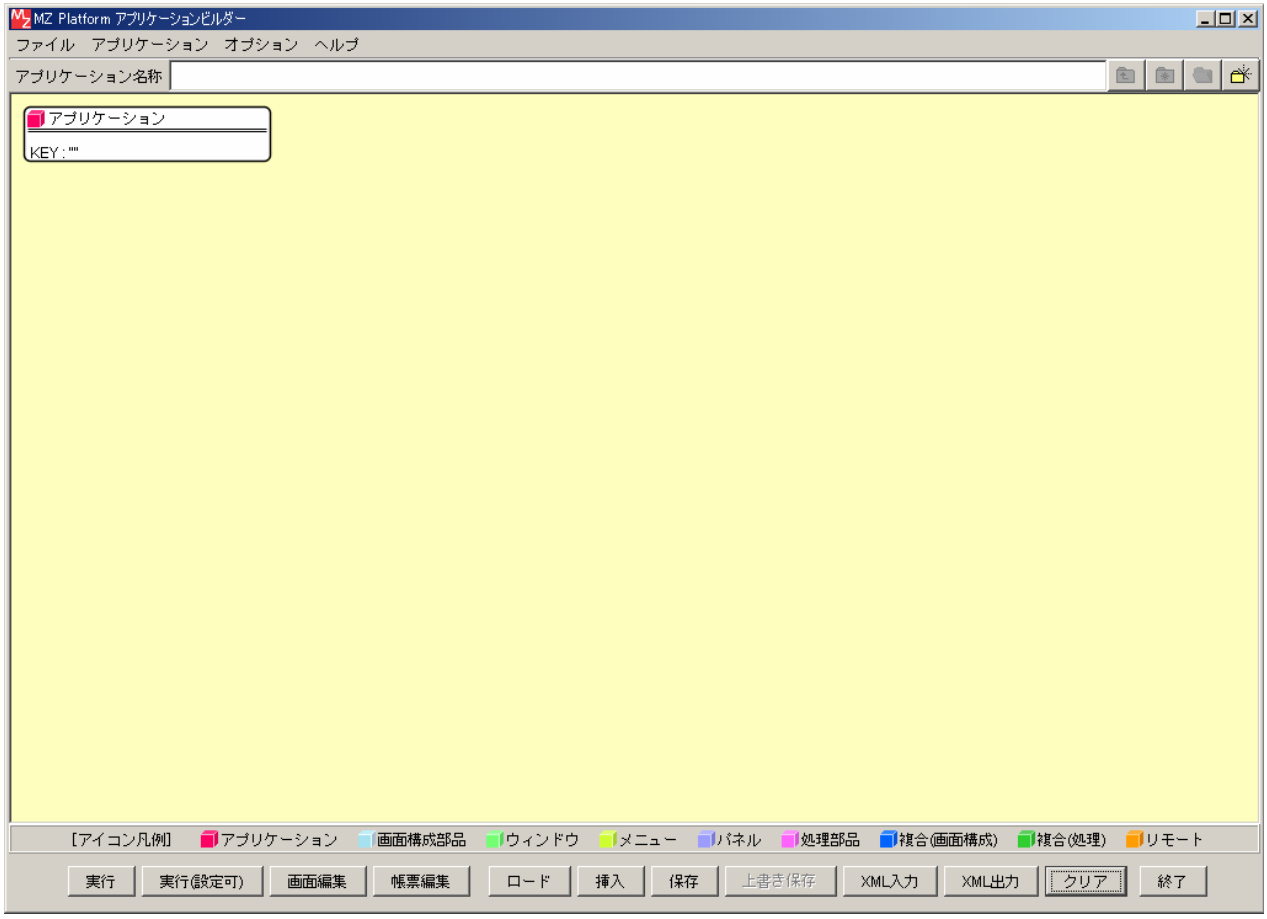
5. アプリケーションの構築

5.1. アプリケーションビルダーの操作方法

5.1.1. アプリケーションビルダーの起動

稼働環境構築を行った上で、スタートメニューからアプリケーションビルダーを起動すると、下のような画面が表示されます。もし、実行中にコンソールを表示させたい場合は、“アプリケーションビルダー（コンソール）”を実行します。

[スタート] －[（すべての）プログラム]－[MZ Platform 1.6]－[アプリケーションビルダー]

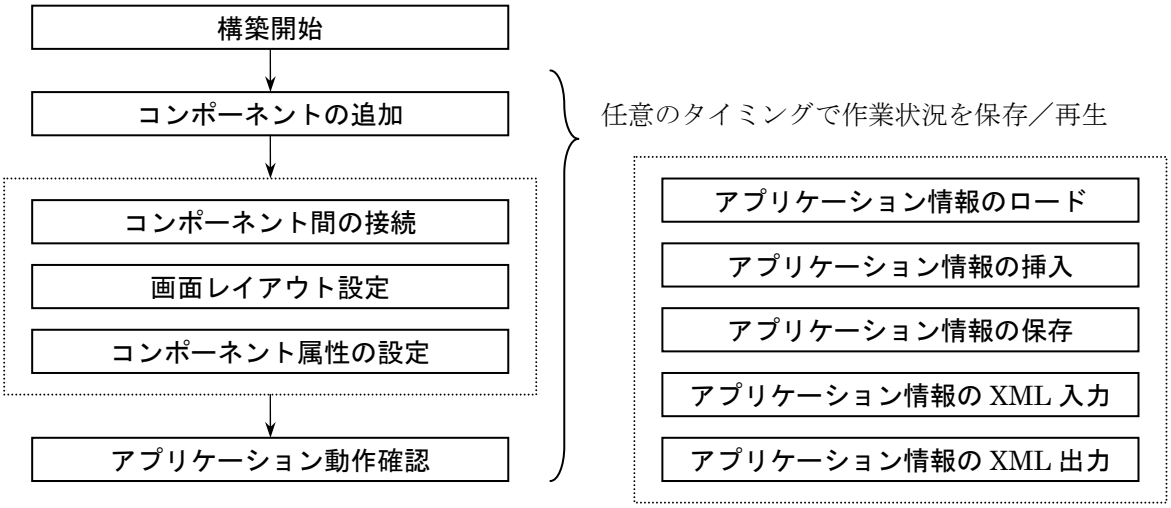


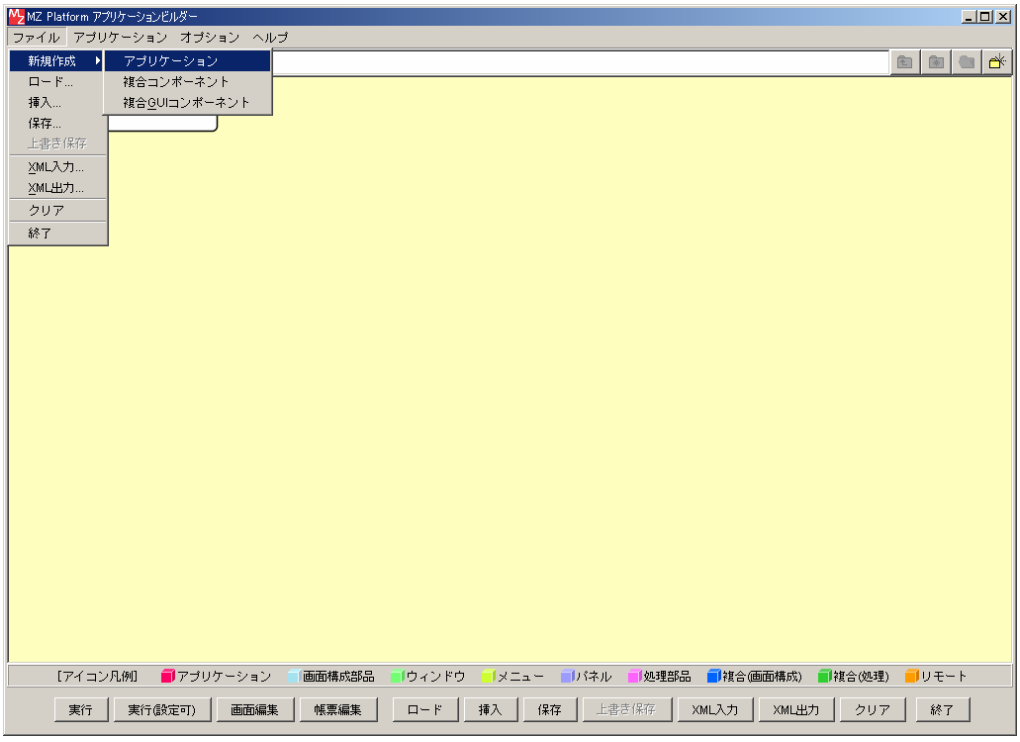
アプリケーションビルダー画面の中央にアプリケーションの情報を表示するための広い領域があります。この表示領域のなかでアプリケーションの組み立てを行います。また、アプリケーション構築作業のための様々な機能は、上部にあるメニューや下部にあるボタンで操作していきます。通常の作業のほとんどは、中央の表示領域と下部のボタンを使用して行われます。

画面のサイズは変更可能ですが、小さくすると画面内に表示がおさまらなくなり、ボタンが表示されなくなってしまいます。しかし、すべてのボタンの機能は画面上部にあるメニューバーからも起動できますので、もし画面上にボタンが表示されない場合には、メニューバーから操作してください。

5.1.2. 構築作業の開始

アプリケーションの構築は、以下の流れで行います。

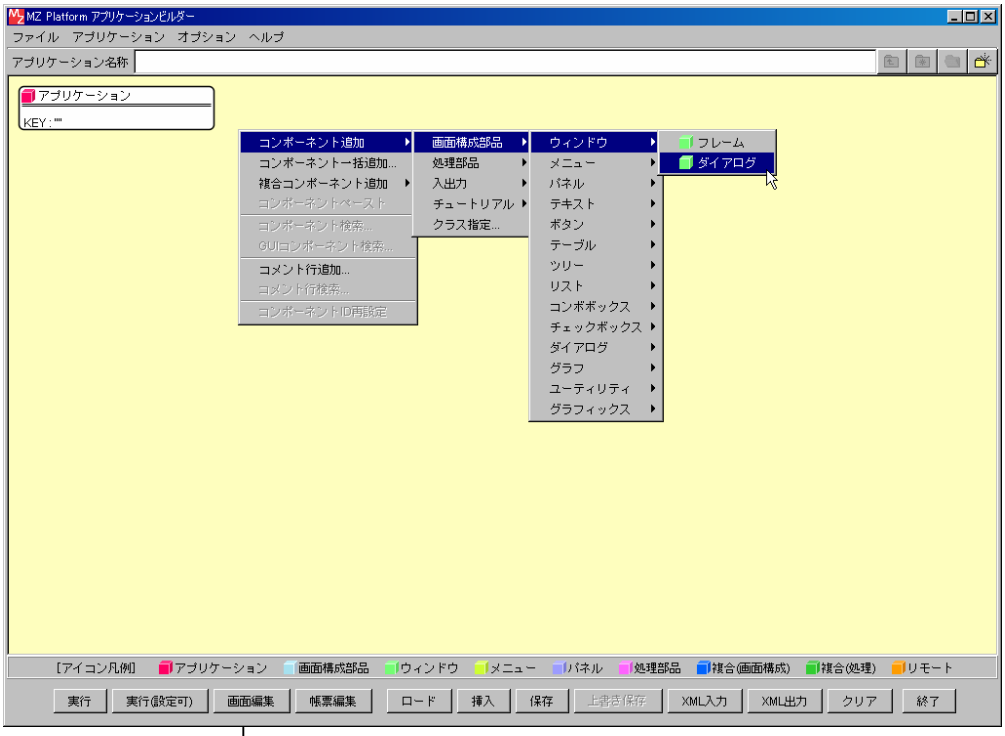


画面	アプリケーションビルダー メイン画面
手順	メニューバーから［ファイル］－［新規作成］－［アプリケーション］を選択  ※特記事項 起動直後はこの操作は不要。 アプリケーション構築作業中に、別のアプリケーションを構築する場合にはこの操作を行うことで構築中のアプリケーション情報がクリアされ、初期状態になります。

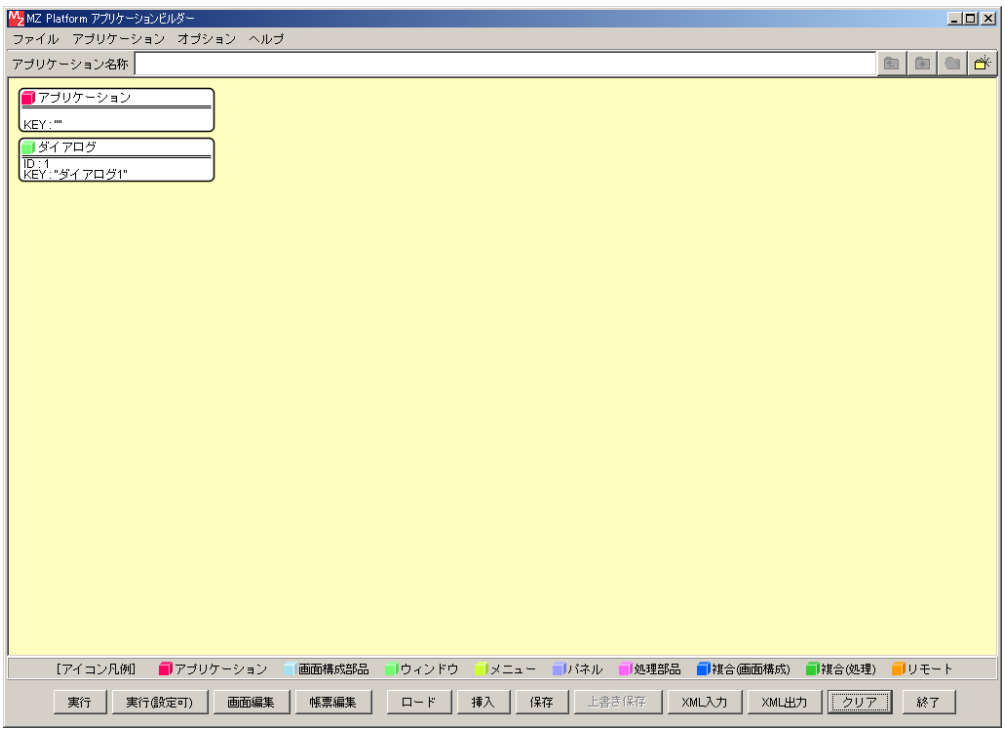
5.1.3. コンポーネントの追加／削除

1)コンポーネントの追加

画面	アプリケーションビルダー メイン画面
手順	①背景にてマウスを右クリックし、コンポーネント追加メニューを表示 ②取り込み対象のコンポーネントを指定

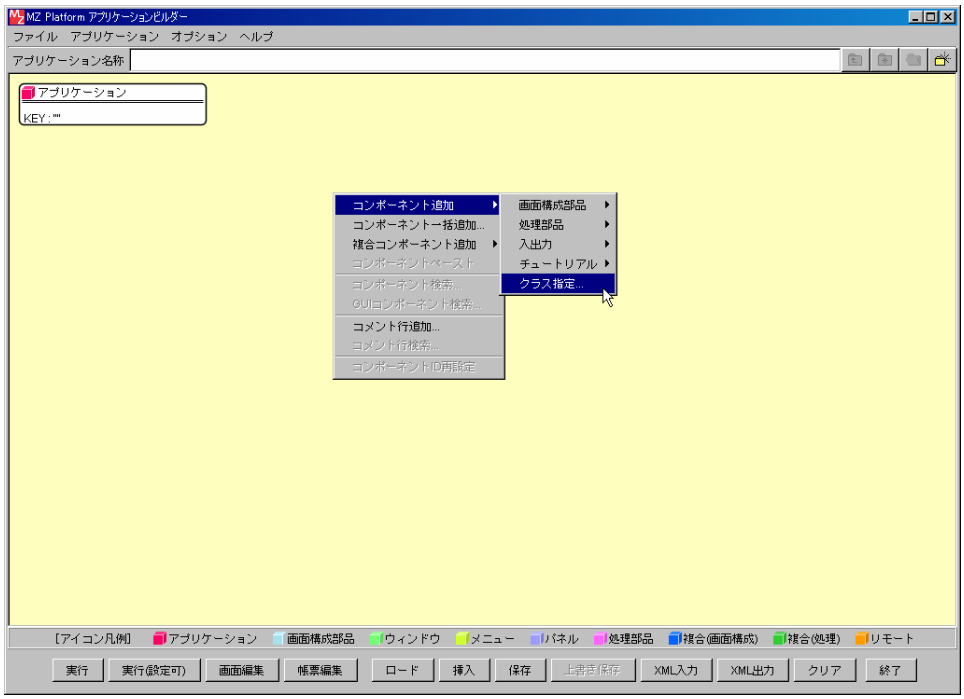


選択したコンポーネントが表示される



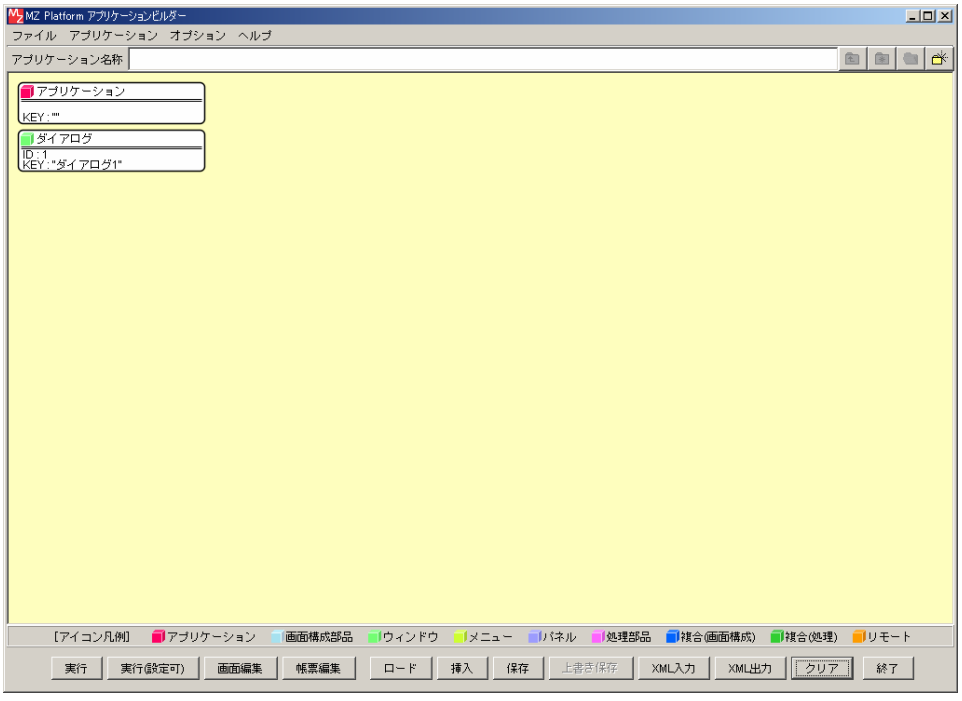
2)クラス指定によるコンポーネント追加

画面	アプリケーションビルダー メイン画面
手順	①背景にてマウス右クリックしコンポーネント追加メニューの［クラス指定...］を選択



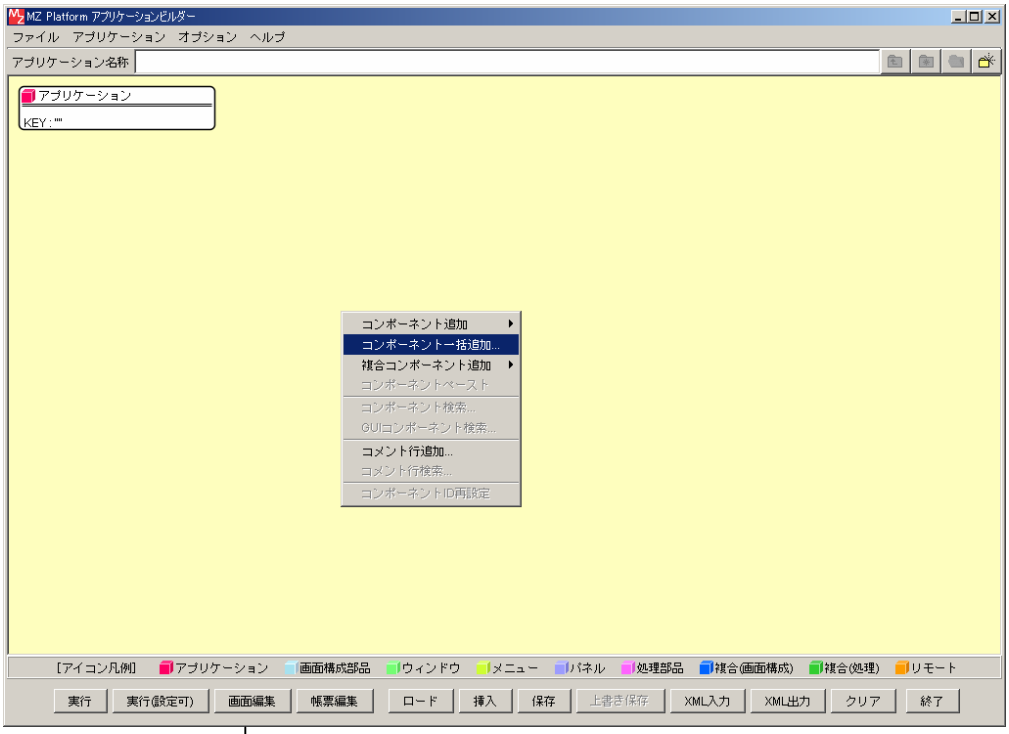
②追加するクラス名を指定する

入力したコンポーネントが表示される

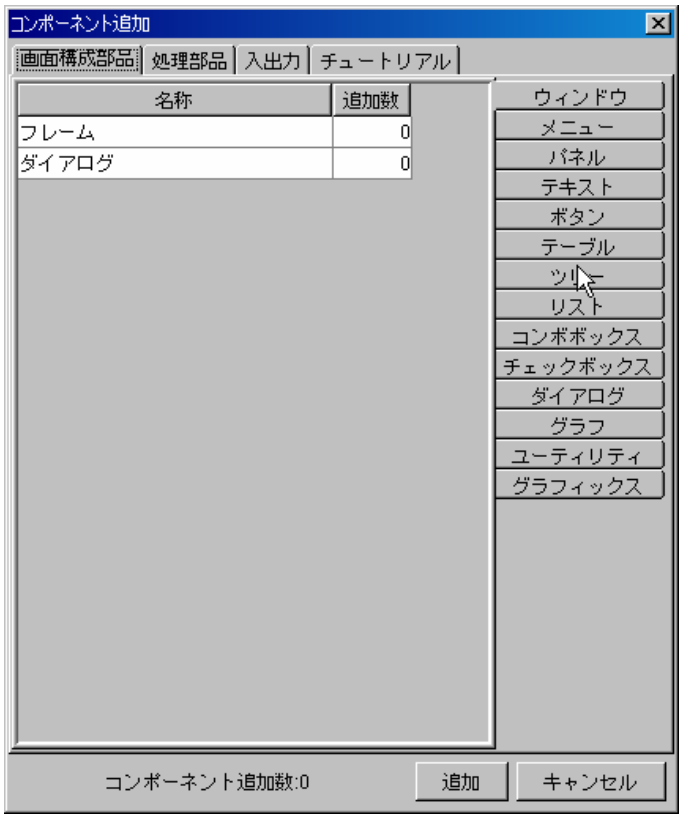


3)コンポーネントの一括追加

画面	アプリケーションビルダー メイン画面
手順	背景にてマウスを右クリックし、[コンポーネント一括追加...] を選択 ※背景でのマウス左ボタンダブルクリック操作でも同様

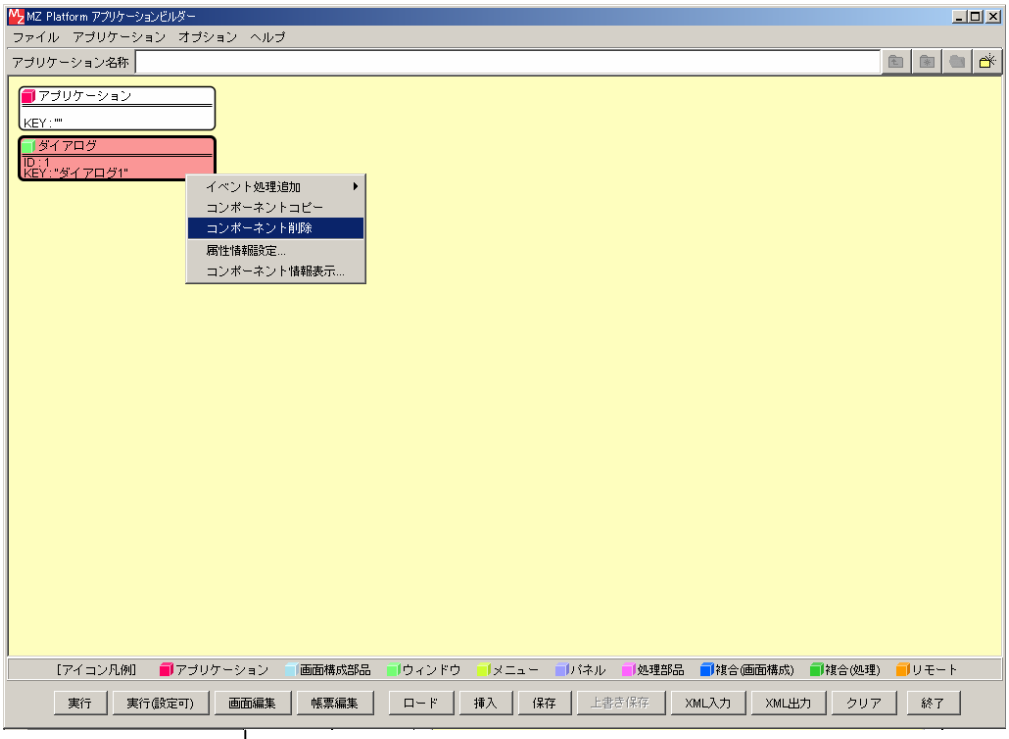


コンポーネント追加画面が表示される

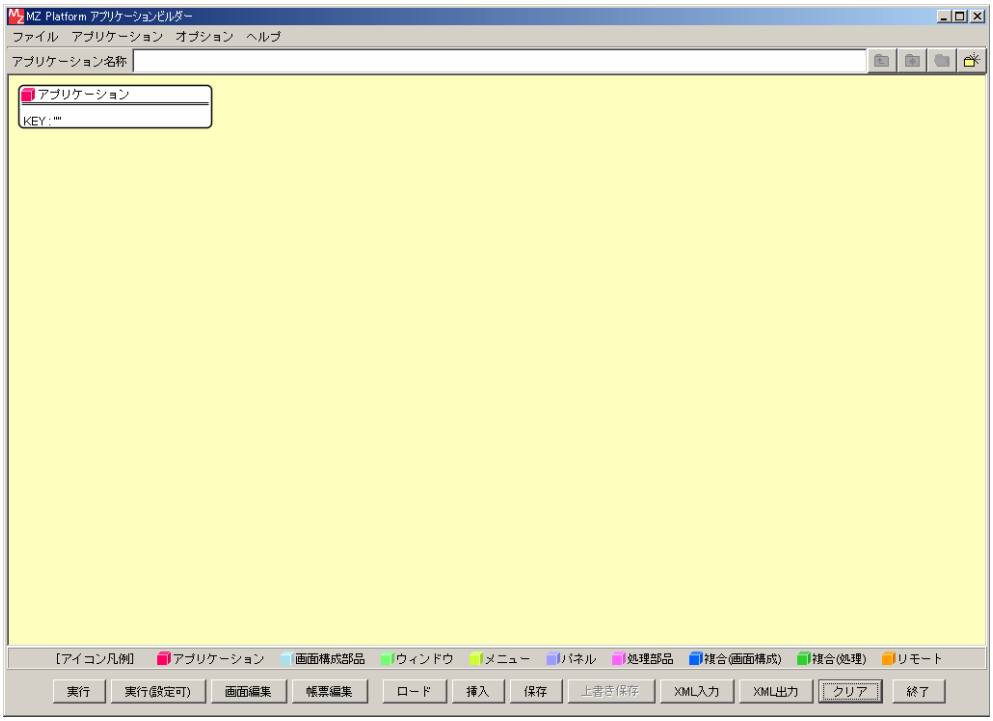


4)コンポーネントの削除

画面	アプリケーションビルダー メイン画面
手順	①削除対象コンポーネント上でマウスを右クリックし、コンポーネント削除を指示 ※注意：他のコンポーネントからの接続先として指定されている場合、 削除はできない（先に接続関係を削除する）



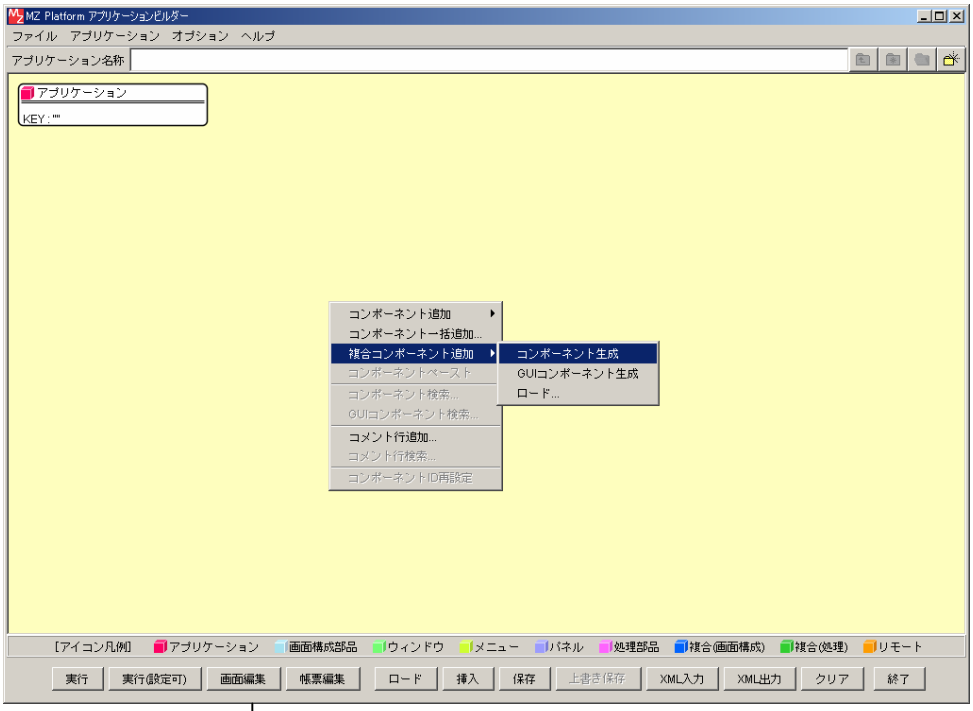
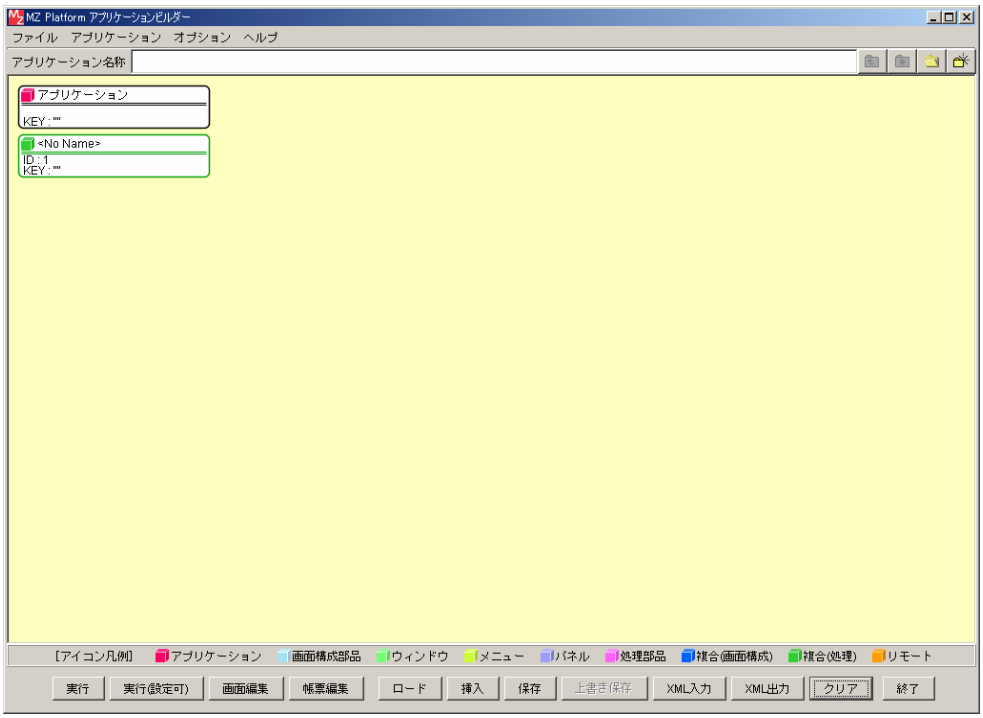
選択したコンポーネントが削除される



5.1.4. 複合コンポーネントの利用

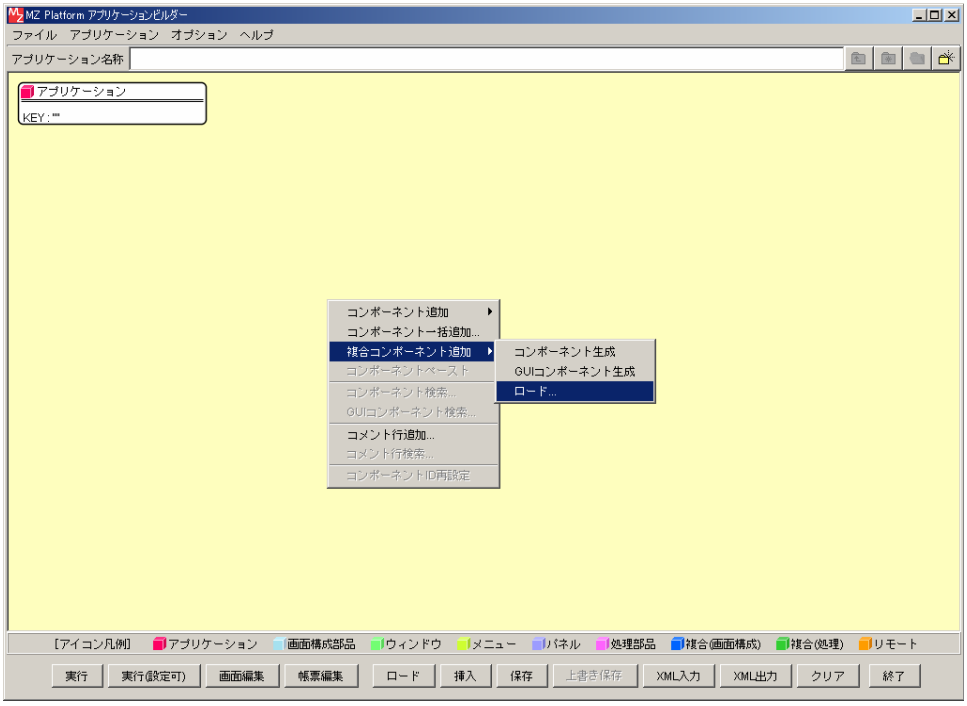
複合コンポーネントの詳細については、次章を参照してください。

1)新規複合コンポーネントの追加

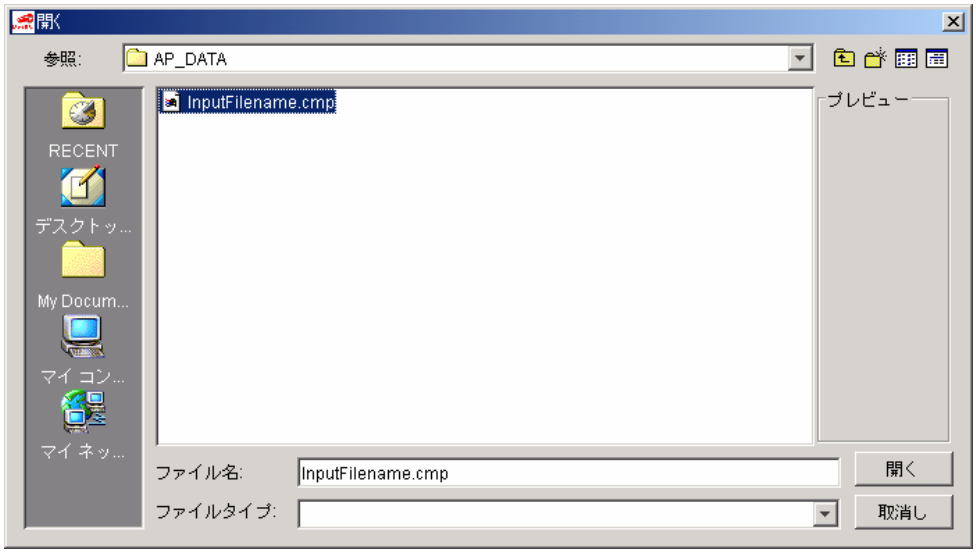
画面	アプリケーションビルダー メイン画面
手順	背景にてマウスを右クリックし、複合コンポーネント追加メニューを表示 （GUI コンポーネントか非 GUI コンポーネントかを選択）  複合コンポーネントが追加される 

2)既存複合コンポーネントの追加

画面	アプリケーションビルダー メイン画面
手順	①背景にてマウスを右クリックし、コンポーネント追加メニューを表示

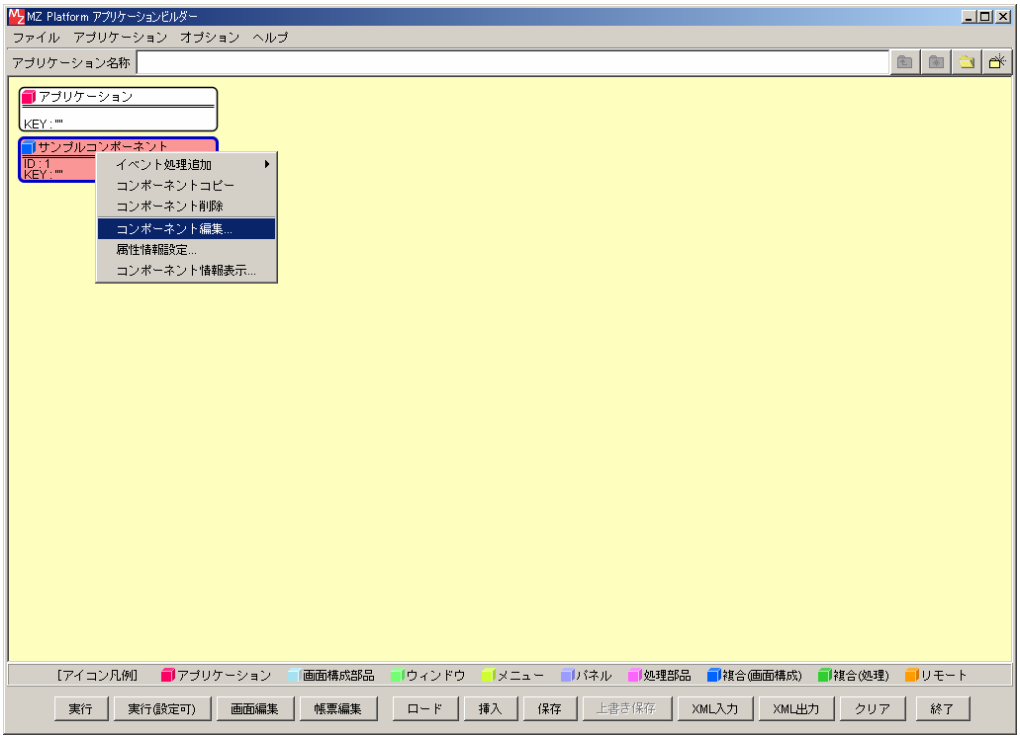


②取り込み対象の複合コンポーネントデータファイルを指定（ファイル選択ダイアログ）

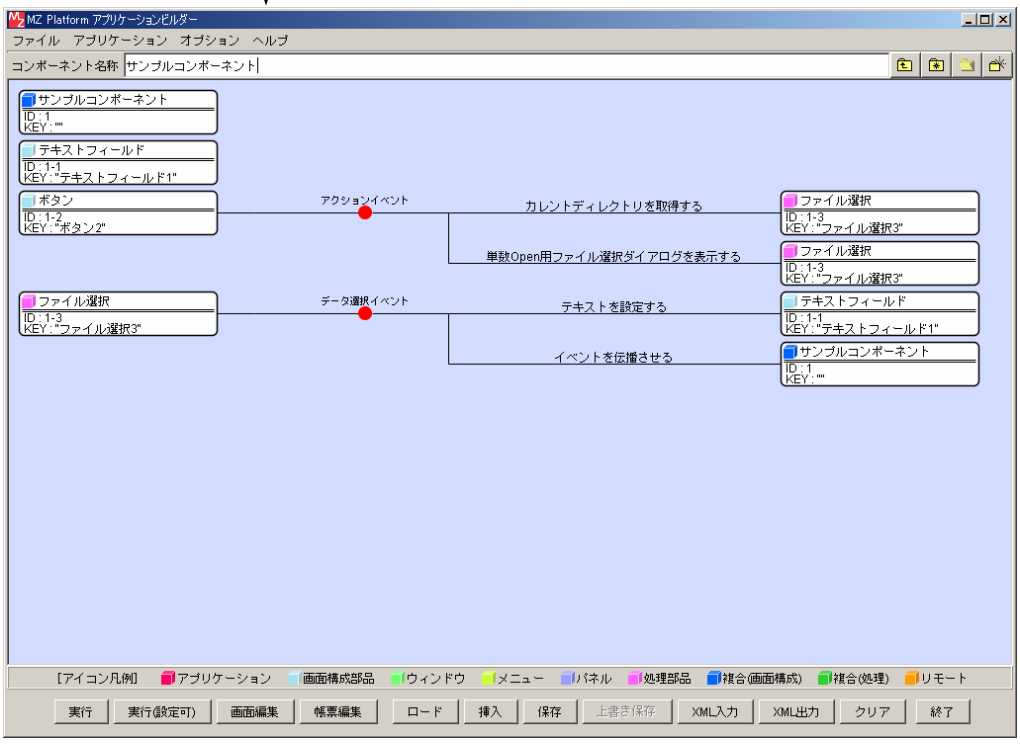


3)複合コンポーネントの編集

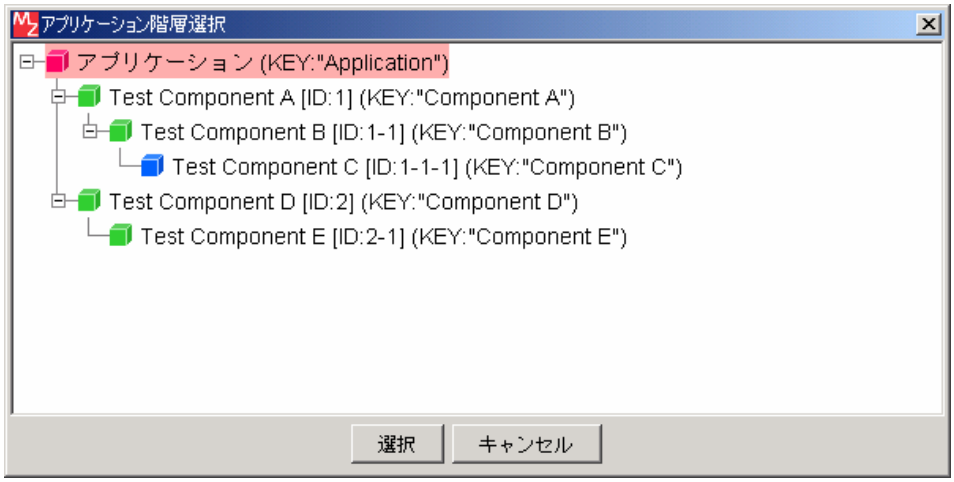
画面	アプリケーションビルダー メイン画面
手順	複合コンポーネント上でマウスを右クリックし、[コンポーネント編集] を選択 ※編集対象のコンポーネント上でマウス左ボタンダブルクリック操作でも同様



複合コンポーネント編集モードになる



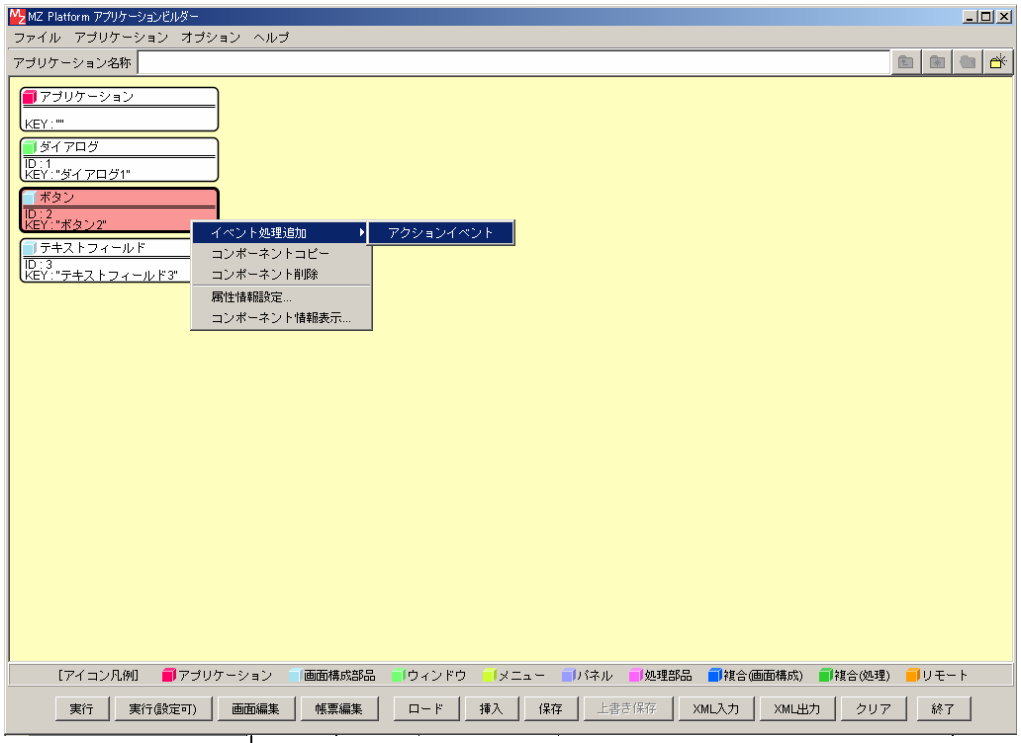
4)編集階層の変更

画面	アプリケーションビルダー メイン画面
手順	画面右上の 3 つのボタンによって階層を選択する。  [上位階層への移動] 表示している状態から順に上の階層に移動  [最上位階層への移動] 編集対象の最上位の階層に移動  [階層選択] 編集対象の階層構造を表示し、選択された階層に移動 ※移動したい階層を選択し、選択ボタン押下により移動。 【アプリケーション階層選択画面】 

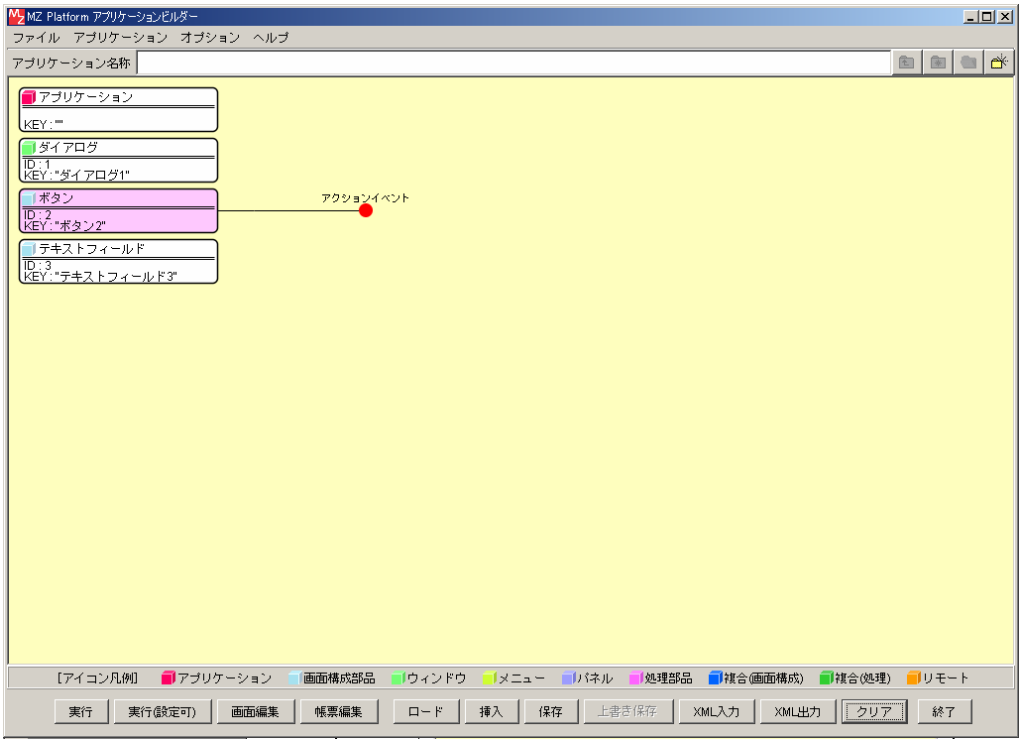
5.1.5. コンポーネント間の接続設定

1)イベント処理の追加

画面	アプリケーションビルダー メイン画面
手順	①追加対象コンポーネント上でマウスを右クリックし、イベント処理追加メニュー表示 ②追加対象のイベントを指定

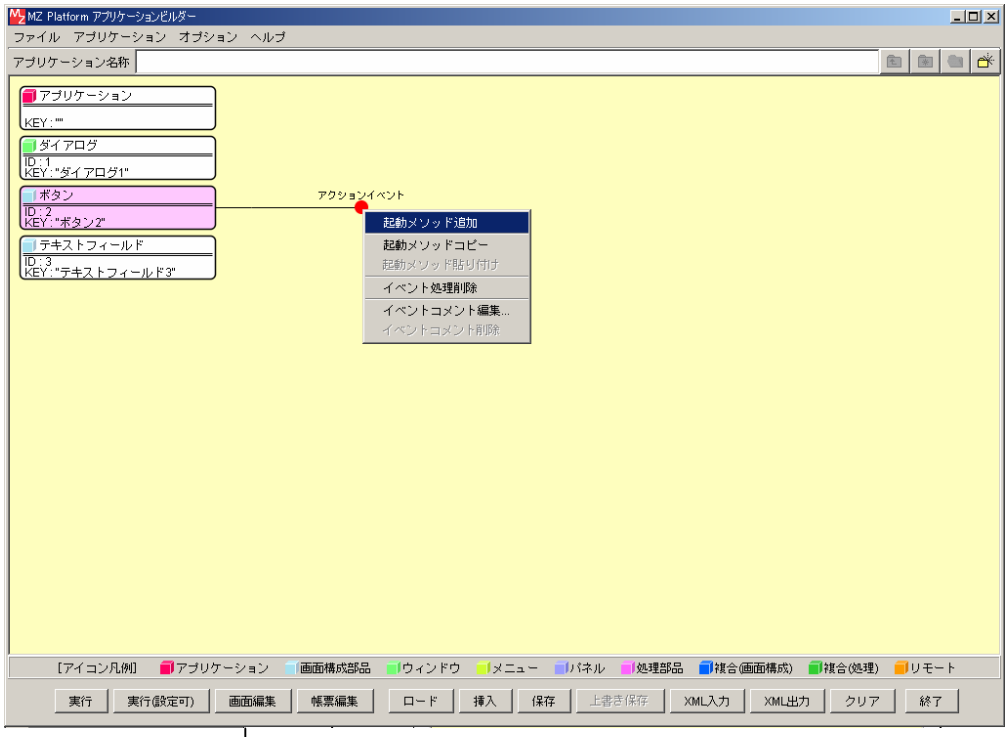


選択したイベント処理が表示される

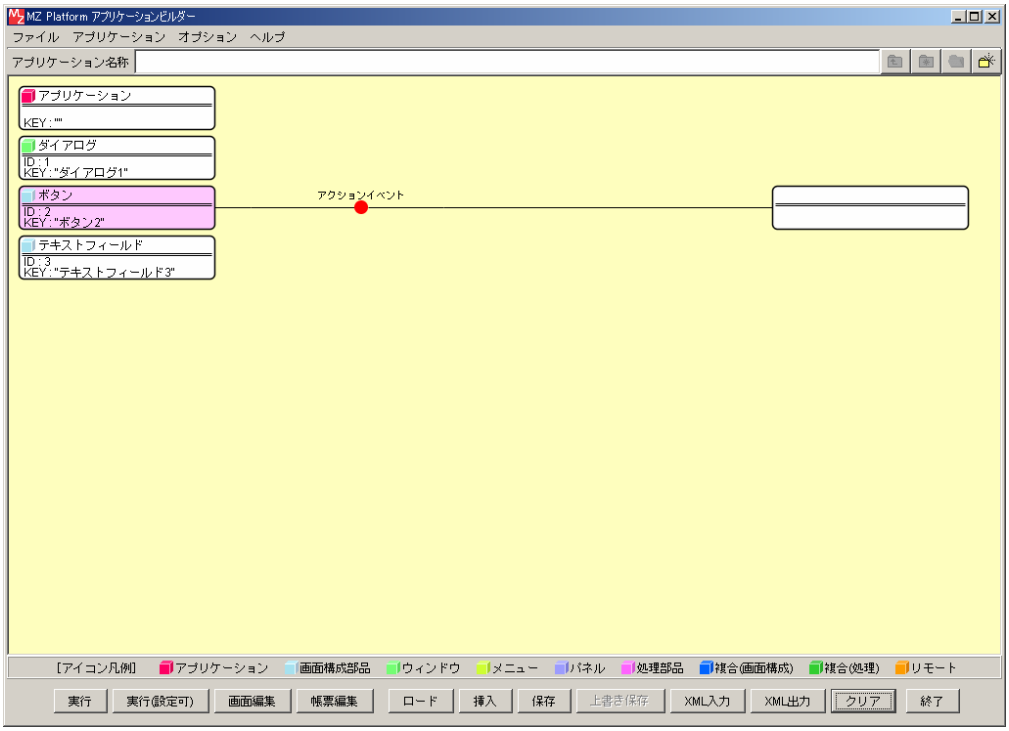


2)イベント接続先の追加

画面	アプリケーションビルダー メイン画面
手順	追加対象のイベント上でマウスを右クリックし、起動メソッド追加を選択 ※追加対象のイベント上でマウス左ボタンダブルクリック操作でも同様

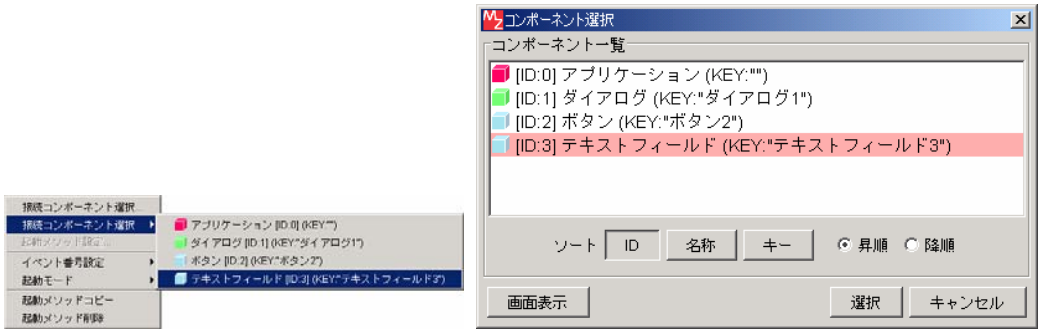


空の接続先が表示される



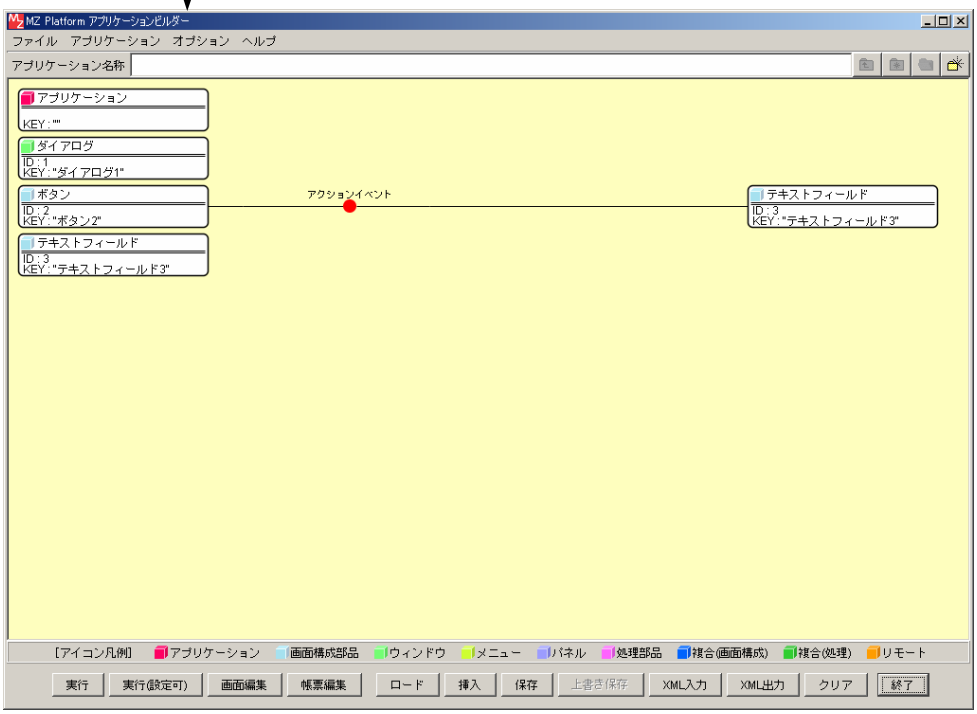
3)イベント接続先コンポーネントの指定

画面	アプリケーションビルダー メイン画面
手順	[方法①] 設定対象の起動メソッド名または接続先コンポーネント上でマウスを 右クリックし、[接続先コンポーネント選択]メニューからコンポーネントを選択 [方法②] 設定対象の起動メソッド名または接続先コンポーネント上でマウスを 右クリックし、[接続先コンポーネント選択...]を選択して コンポーネント一覧を表示し、コンポーネントを選択 ※接続先コンポーネント上でのマウス左ボタンダブルクリック操作でも同様 （設定対象の起動メソッド名をダブルクリックしても一覧は表示されません）



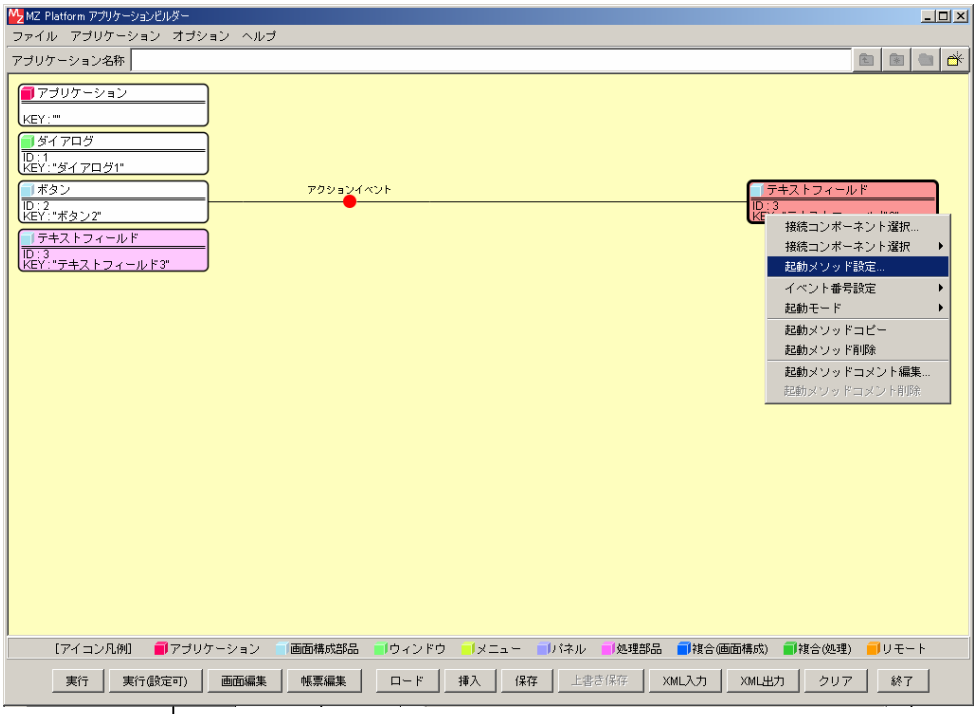
[方法①]　[方法②]

選択したコンポーネントが接続先に表示される

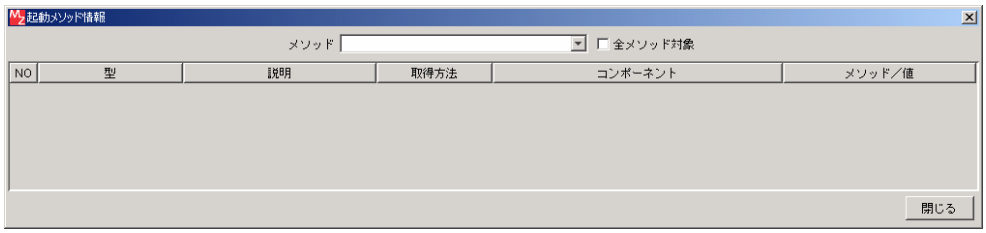


4)起動メソッドの指定

画面	アプリケーションビルダー メイン画面
手順	①設定対象の起動メソッド名または接続先コンポーネント上でマウスを右クリックし、 メソッド設定メニューを表示 ②起動メソッド設定メニューを選択する ※設定対象の起動メソッド名または接続先コンポーネント上での マウス左ボタンダブルクリック操作でも同様



起動メソッド設定画面が表示される

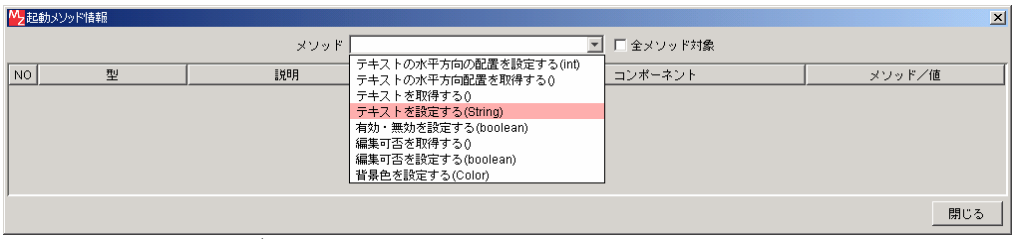


4)起動メソッドの指定 続き

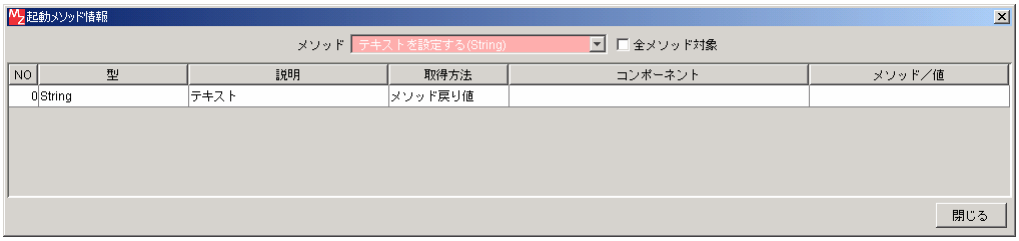
手順

③起動メソッドを選択する

コンポーネントが提供するメソッドのうち、公開設定されているメソッドのみが表示されるため、引数の並びを含めて対象のメソッドを選択する。

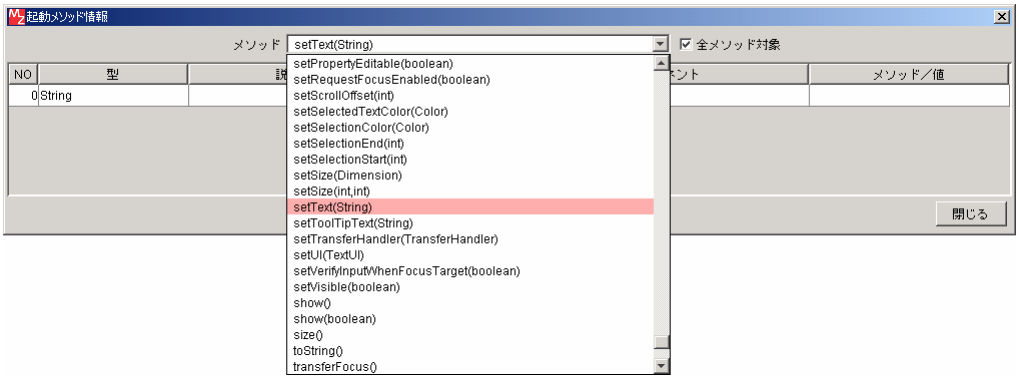


起動メソッドの引数情報が表示される



【補足】

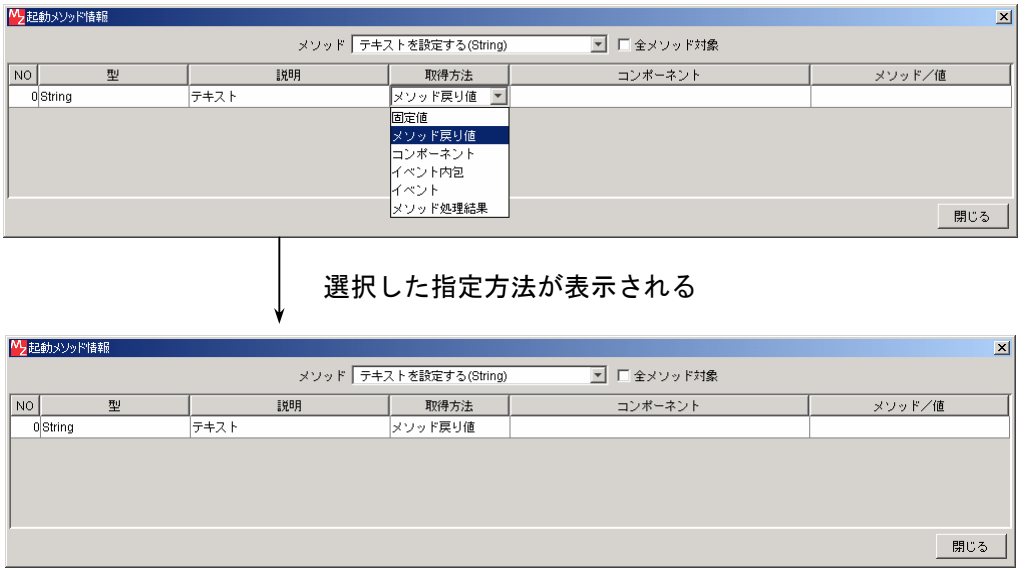
起動するメソッドが表示されない場合、そのメソッドが非公開の設定になっている。
“全メソッド対象”のチェックボックスを ON にすれば、
コンポーネントの全 public メソッドが表示され、選択可能となる。
また、必要に応じてメソッドの情報設定（後述）を行えば、常に表示される。



※注意：メソッドに引数がない場合は以降④～⑦の作業は不要

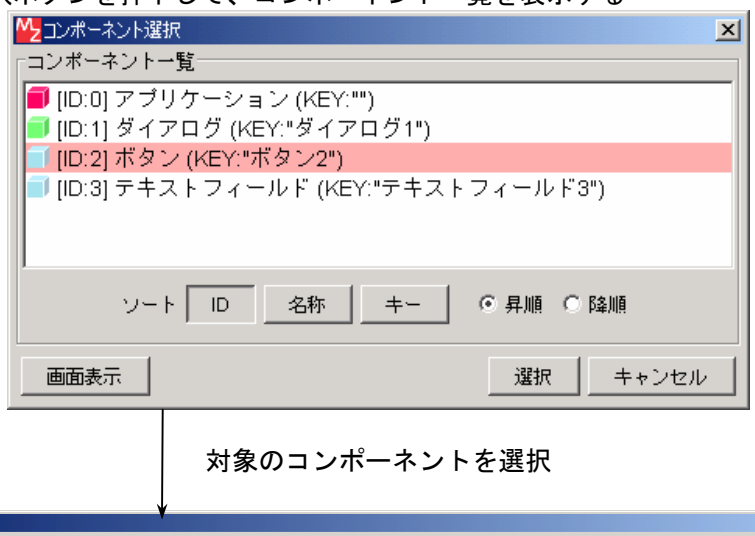
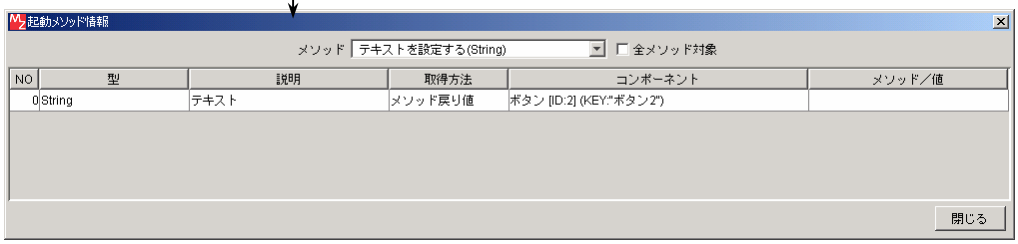
4)起動メソッドの指定 続き

手順	④メソッド引数の指定方法を選択する メソッド引数の指定には以下の 5 種類があり、対象の指定方法を選択する。 ・固定値（リテラル指定） ・メソッド戻り値 ・コンポーネント ・イベント内包データ（イベントオブジェクトのメソッド戻り値指定） ・イベントオブジェクト（イベントオブジェクト自身） ・メソッド処理結果

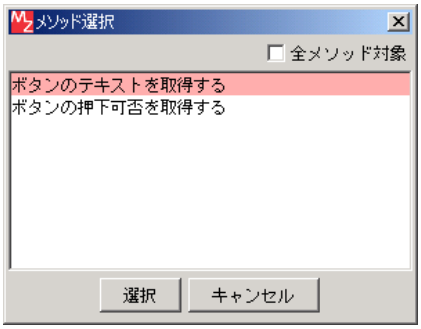
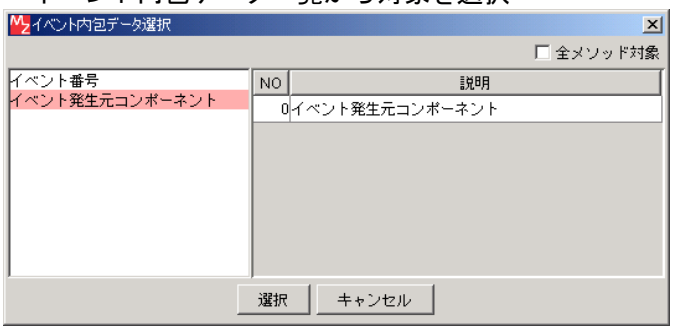


選択した指定方法が表示される

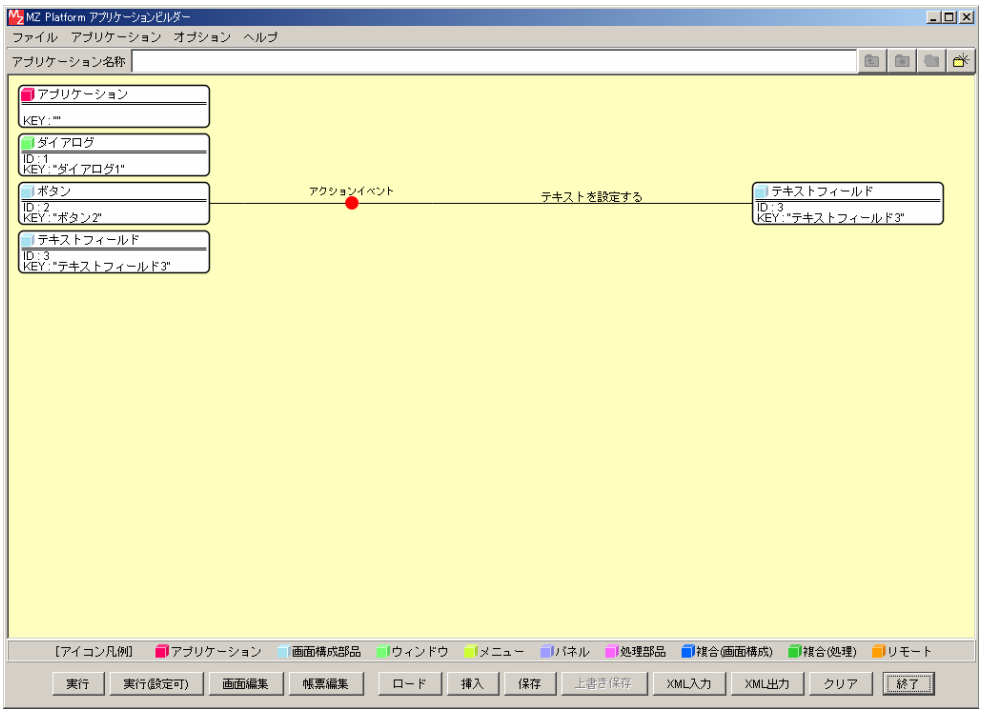
4)起動メソッドの指定 続き

手順	⑤設定するコンポーネントを選択する メソッド引数の指定方法（手順④）にあわせて設定する a)指定方法＝“固定値” 入力不要（入力不可） b)指定方法＝“メソッド戻り値” ◇入力：メソッド起動対象コンポーネントを一覧から選択（操作は下記） ◇入力チェック：なし c)指定方法＝“コンポーネント” ◇入力：引数となるコンポーネントを一覧から選択（操作は下記） ◇入力チェック：コンポーネントの型が引数の型とあっているかチェック d)指定方法＝“イベント内包データ” 入力不要（入力不可） e)指定方法＝“イベント” 入力不要（入力不可） d)指定方法＝“メソッド処理結果” 入力不要（入力不可） ＜コンポーネント選択方法＞ 設定対象の引数情報の“コンポーネント”セルをマウス左ボタンでダブルクリック、もしくは選択ボタンを押下して、コンポーネント一覧を表示する  対象のコンポーネントを選択  例）指定方法＝“メソッド戻り値”の場合

4)起動メソッドの指定 続き

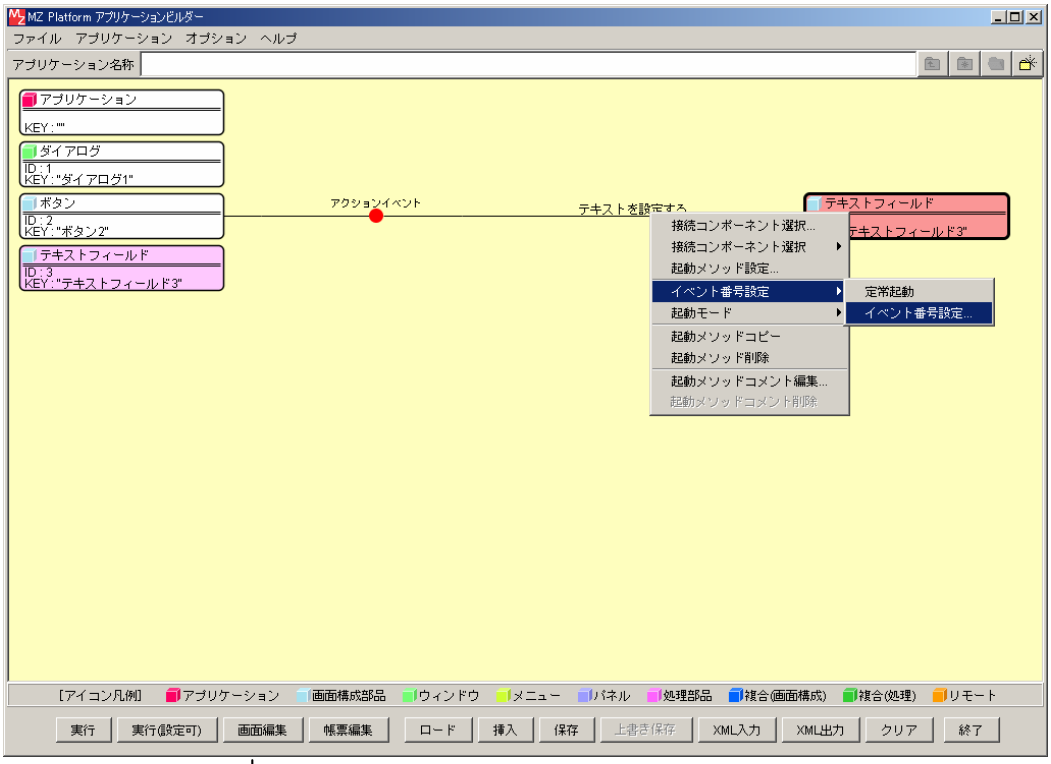
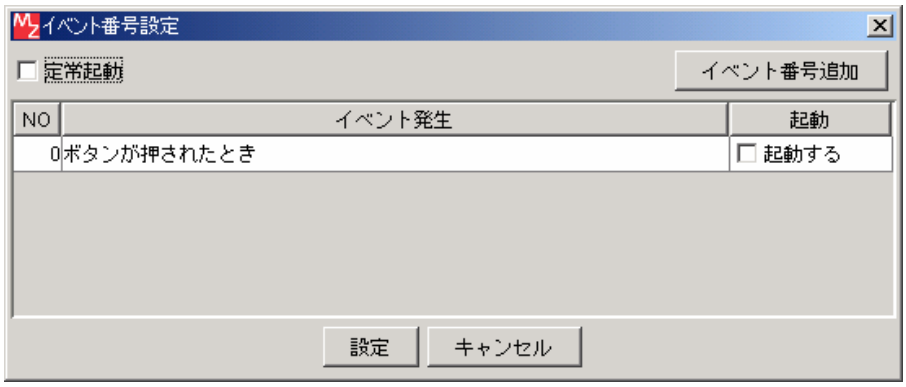
手順	⑥メソッド引数の取得情報を入力する メソッド引数の指定方法（手順④）にあわせて以下の入力を行う。 a)指定方法＝“固定値” ◇入力：“メソッド／値”セルにキーボードからリテラルを入力 String 型：文字列リテラル プリミティブ型（数値／真偽値）：文字列表現（”100”, “1.5”, “true” など） オブジェクト型：null のみ設定可能（未入力 or 文字列“null”） ◇入力チェック：上記入力方法に沿っているかどうかをチェック b)指定方法＝“メソッド戻り値” ◇入力：引数を取得するメソッドを一覧から選択 ・[メソッド／値]セルをマウス左ボタンでクリック ・メソッド一覧から対象を選択  ※補足 対象メソッドが表示されない場合、“全メソッド対象”のチェック ◇入力チェック：なし（設定可能なもののみ選択可能） ※引数のあるメソッドは指定できません c)指定方法＝“コンポーネント” 入力不要（入力不可） d)指定方法＝“イベント内包データ” ◇入力：イベントオブジェクトのデータ取得メソッドを一覧から選択 ・[メソッド／値]セルをマウス左ボタンでクリック ・イベント内包データー覧から対象を選択  ※補足 対象データが表示されない場合、“全メソッド対象”のチェック ◇入力チェック：なし e)指定方法＝“イベント” 入力不要（入力不可） f)指定方法＝“メソッド処理結果” ◇入力：起動メソッドを一覧から選択 ・[メソッド／値]セルをマウス左ボタンでクリック ・イベント内包データー覧から対象を選択 ◇入力チェック：なし

4)起動メソッドの指定 続き

手順	⑦メソッド設定を終了する（[閉じる]ボタンをクリック） ↓ 設定されたメソッド名が表示される 

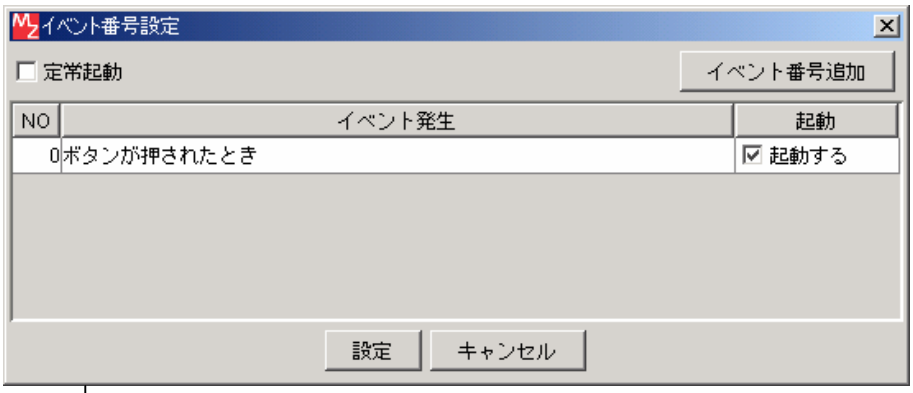
5)起動対象イベント番号の設定

コンポーネントから発生するイベントには、そのイベントの内容をしめす“イベント番号”がついています。アプリケーションの動作は、このイベント番号によって処理を分岐する必要がありますので、起動メソッドそれぞれに対して、対象とするイベントの番号を設定することができます。対象のイベント番号を設定しない場合は、イベント番号に関わらずすべてのイベントに対して処理が行われます。

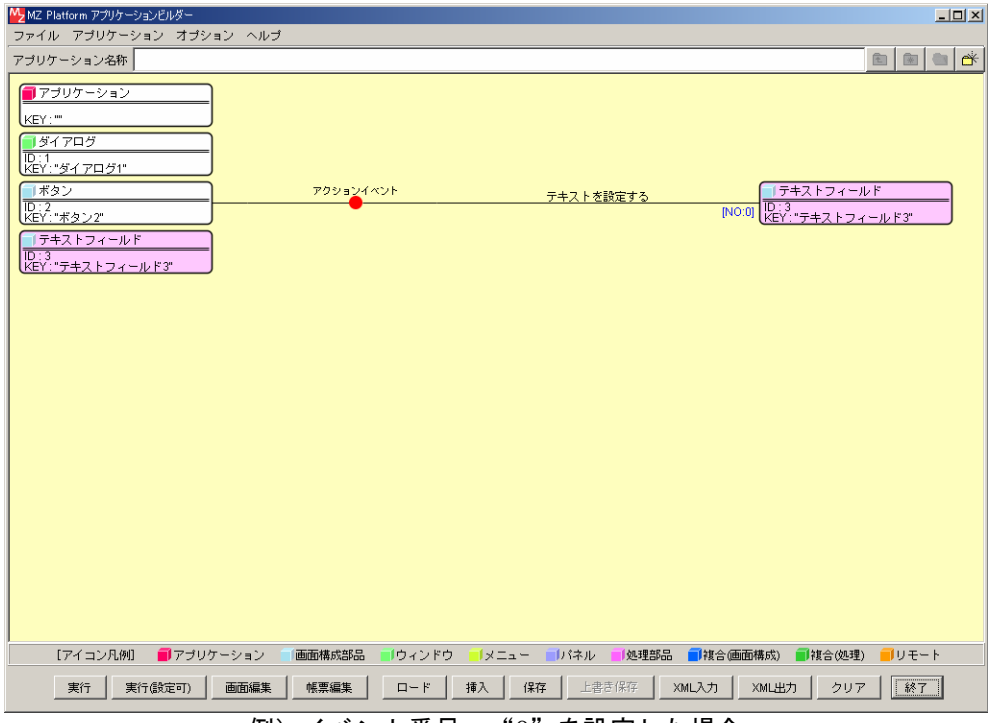
画面	アプリケーションビルダー メイン画面
手順	①設定対象の起動メソッド名または接続先コンポーネント上でマウスを右クリックし、『イベント番号設定』メニューから『イベント番号設定』を選択 （常に起動したい場合は『定常起動』を指示）  発生するイベント番号の一覧が表示される 

5)起動対象イベント番号の設定 続き

手順	②表示されたイベント番号情報を参考に、起動するイベント番号を設定する



設定された番号が表示される（『定常起動』は非表示）



例）イベント番号＝“0”を設定した場合

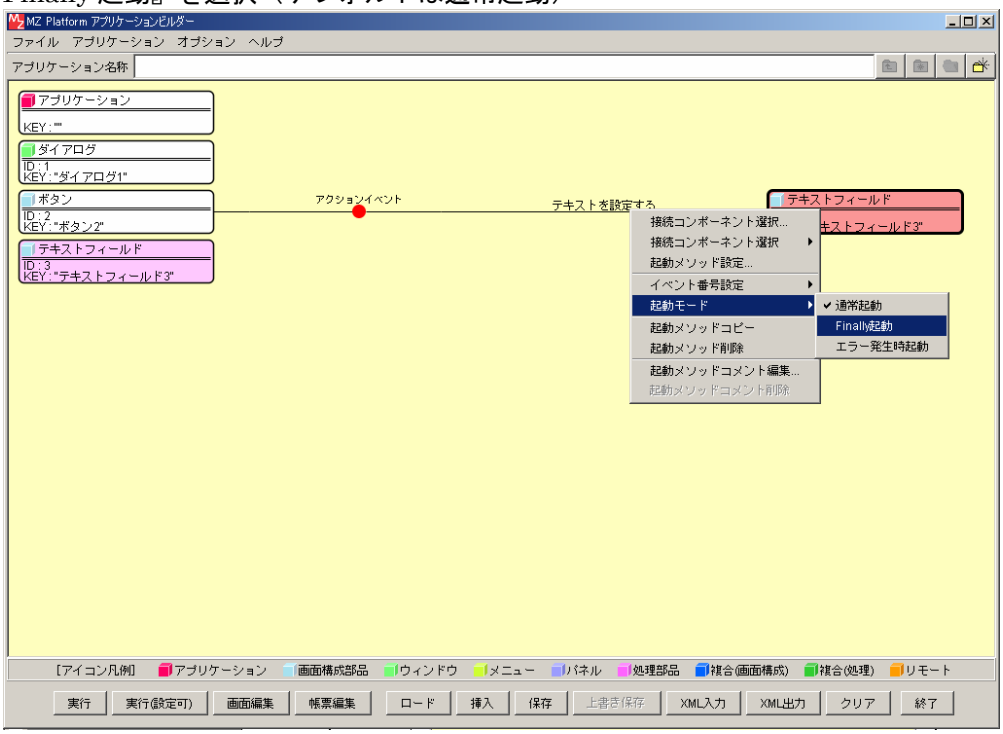
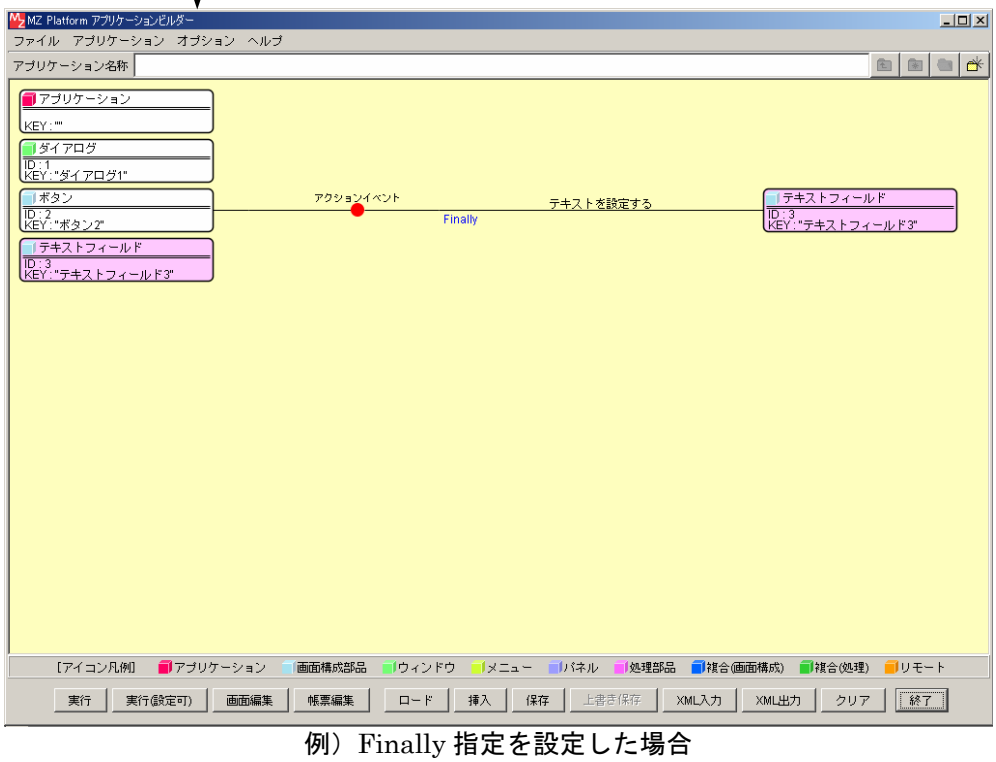
※補足説明

もしイベント番号設定画面上に、指定したいイベント番号が表示されない場合、右上の[イベント番号追加]ボタンを押して、番号を入力します。

また、コンポーネント情報の登録により、表示情報を設定することもできます。

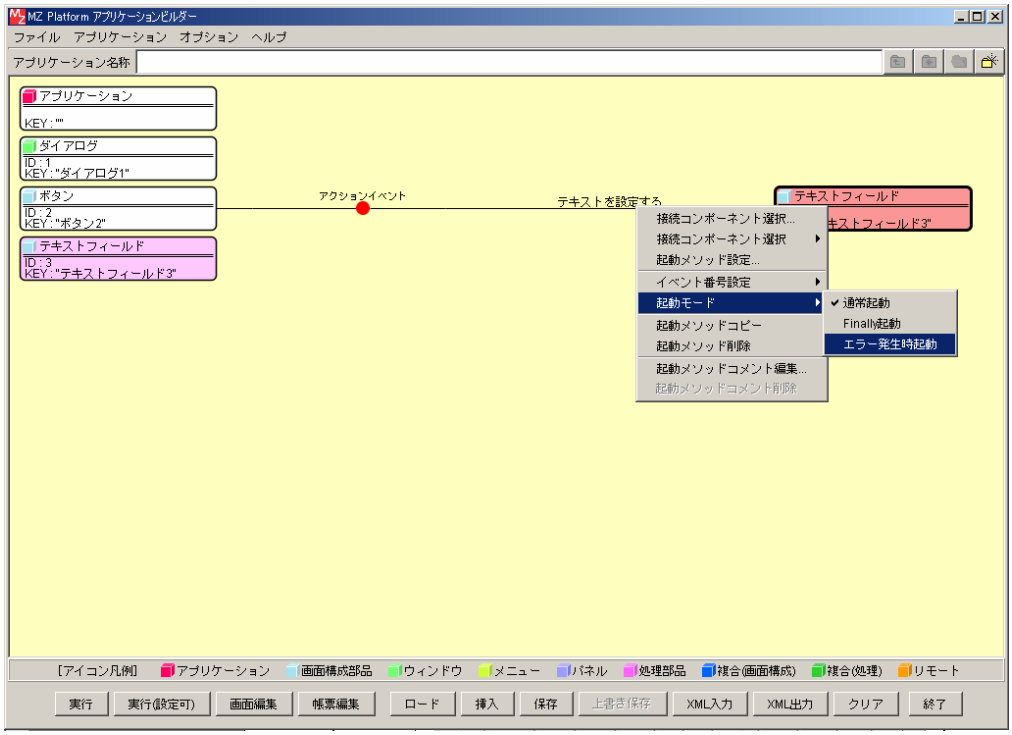
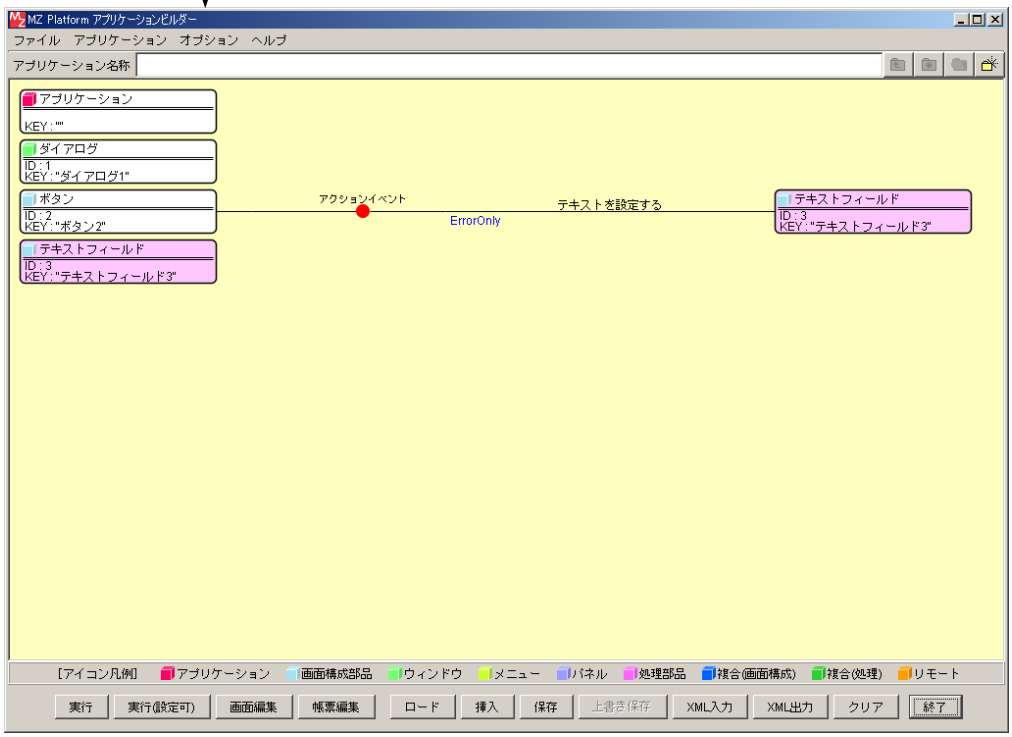
6)起動対象の Finally 化設定

イベントに対する処理は、通常設定された順に逐次実行されますが、途中で例外が起きた場合は続く処理は実行されません。そこで、途中でエラーが発生しても必ず実行する必要のあるメソッドについては“Finally 起動”の設定をすることで、他のメソッド処理結果に関係なく必ず実行されます。

画面	アプリケーションビルダー メイン画面
手順	設定対象の起動メソッド名または接続先コンポーネント上でマウスを右クリックし、『Finally 起動』を選択（デフォルトは通常起動）  Finally 句が表示される（通常起動時は非表示）  例）Finally 指定を設定した場合

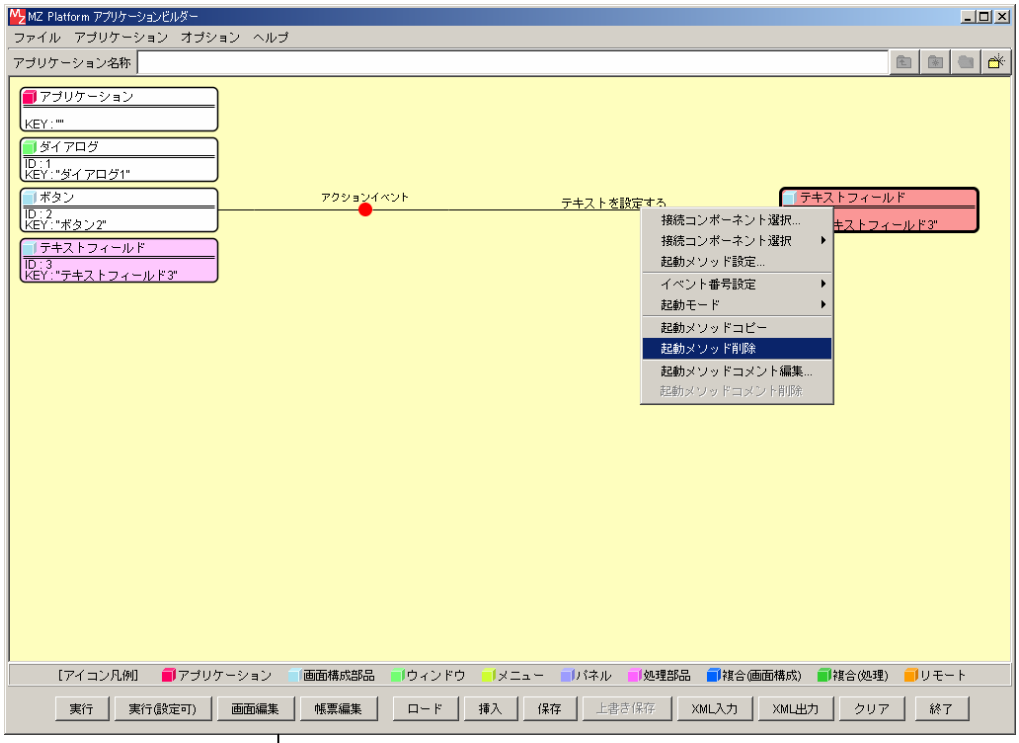
7)起動対象の ErrorOnly 化設定

イベントに対する処理は、通常設定された順に逐次実行されますが、途中で例外が起きた場合にのみ処理を起動したい場合、“ErrorOnly 起動”の設定をします。

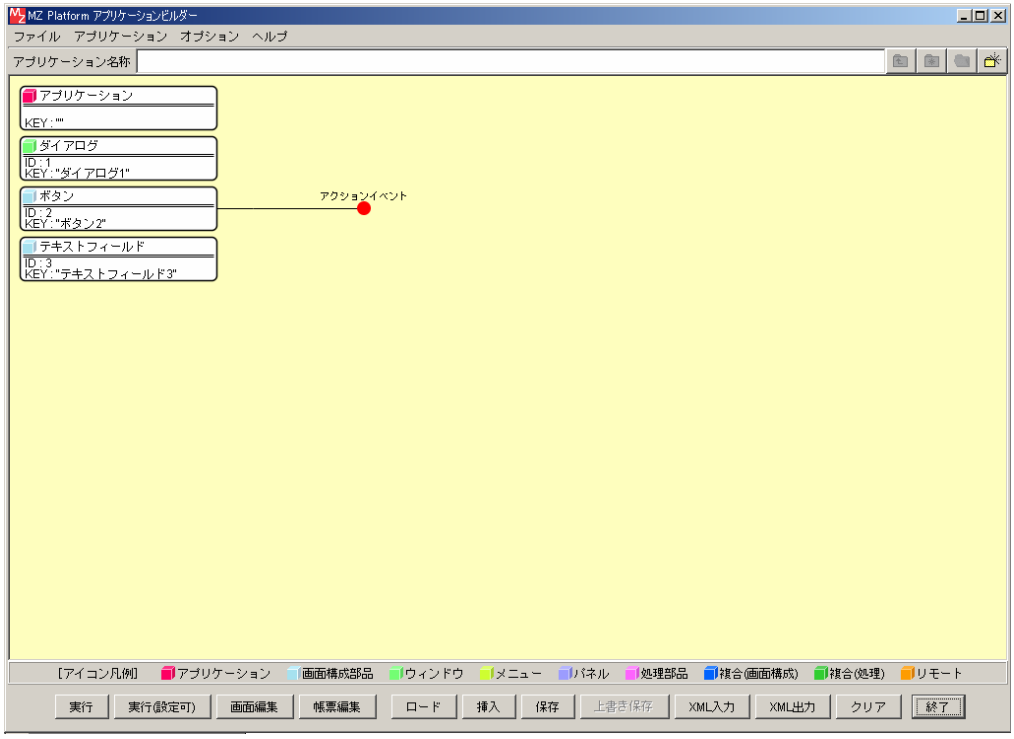
画面	アプリケーションビルダー メイン画面
手順	設定対象の起動メソッド名または接続先コンポーネント上でマウスを右クリックし、『エラー発生時起動』を選択（デフォルトは通常起動）  ErrorOnly が表示される（通常起動時は非表示）  例）ErrorOnly 指定を設定した場合

8)起動メソッドの削除

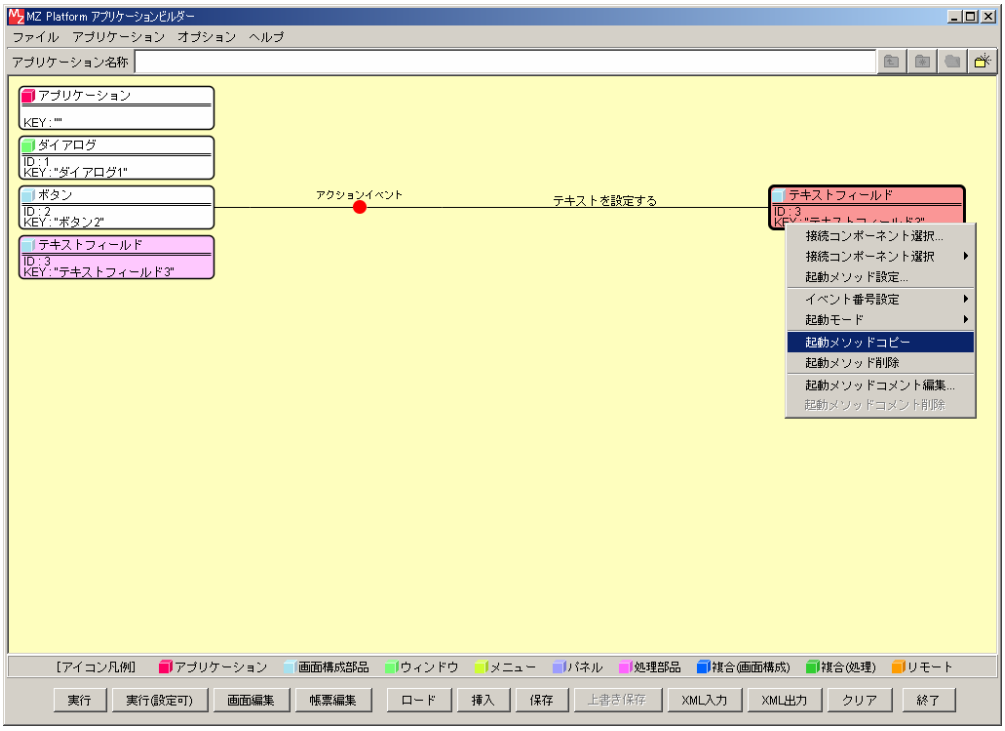
画面	アプリケーションビルダー メイン画面
手順	削除対象の起動メソッド名または接続先コンポーネント上でマウスを右クリックし、『起動メソッド削除』を指示



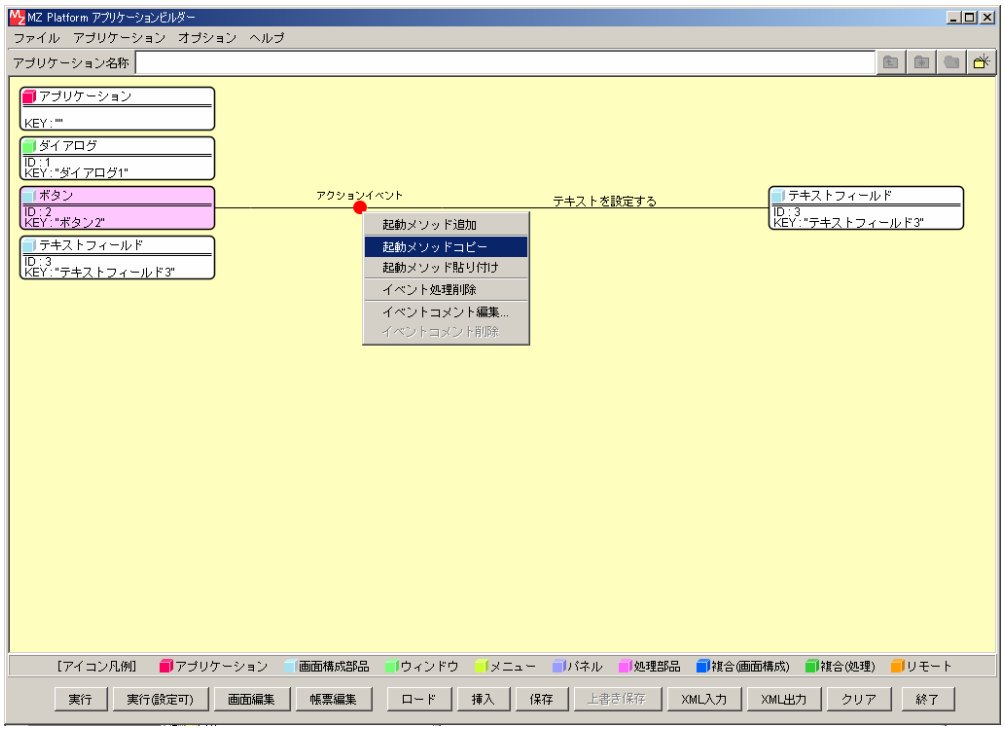
対象の起動メソッドが削除される



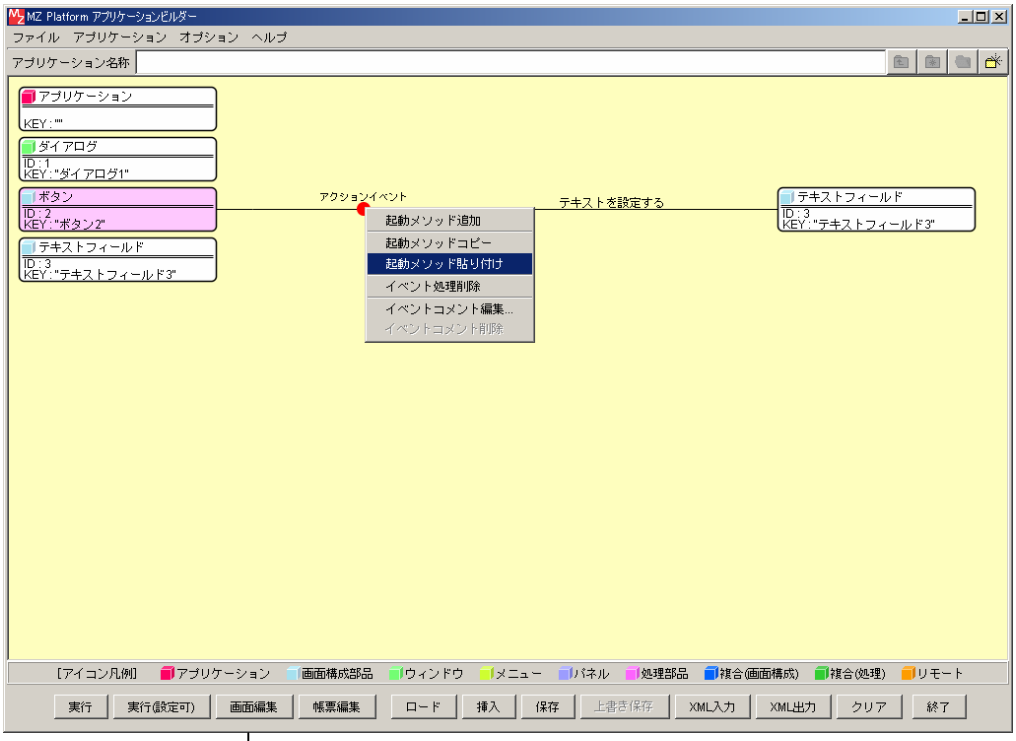
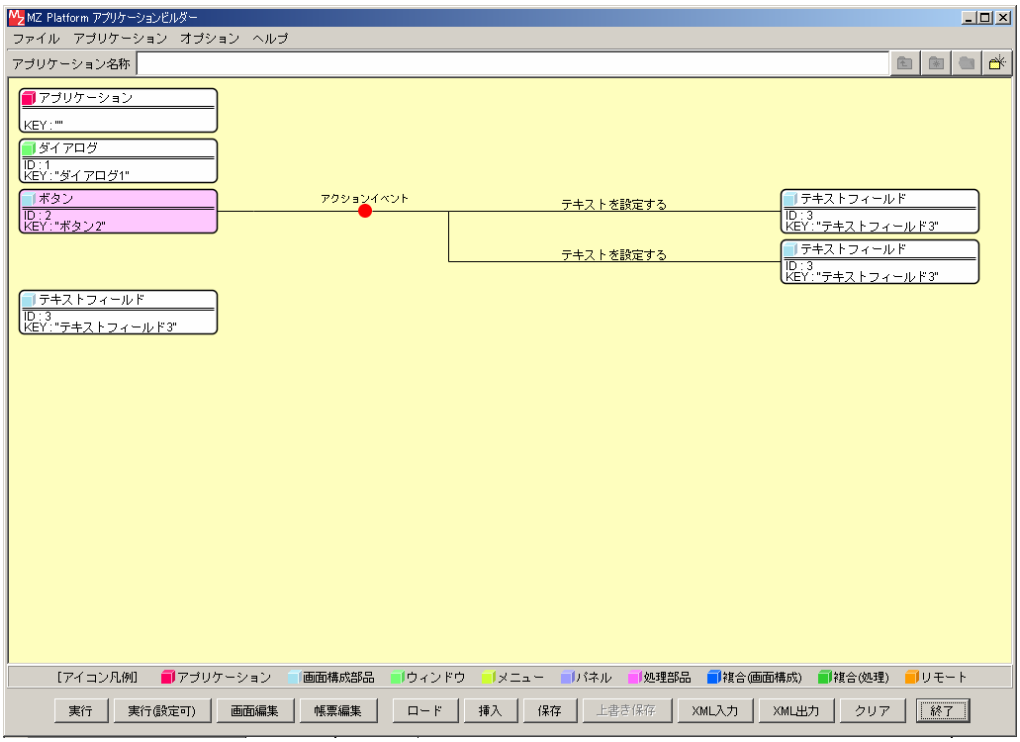
9)起動メソッドのコピー（単一）

画面	アプリケーションビルダー メイン画面
手順	コピー対象の起動メソッド名または接続先コンポーネント上でマウスを右クリックし、『起動メソッドコピー』を指示 → 対象の起動メソッドがペースト対象となる 

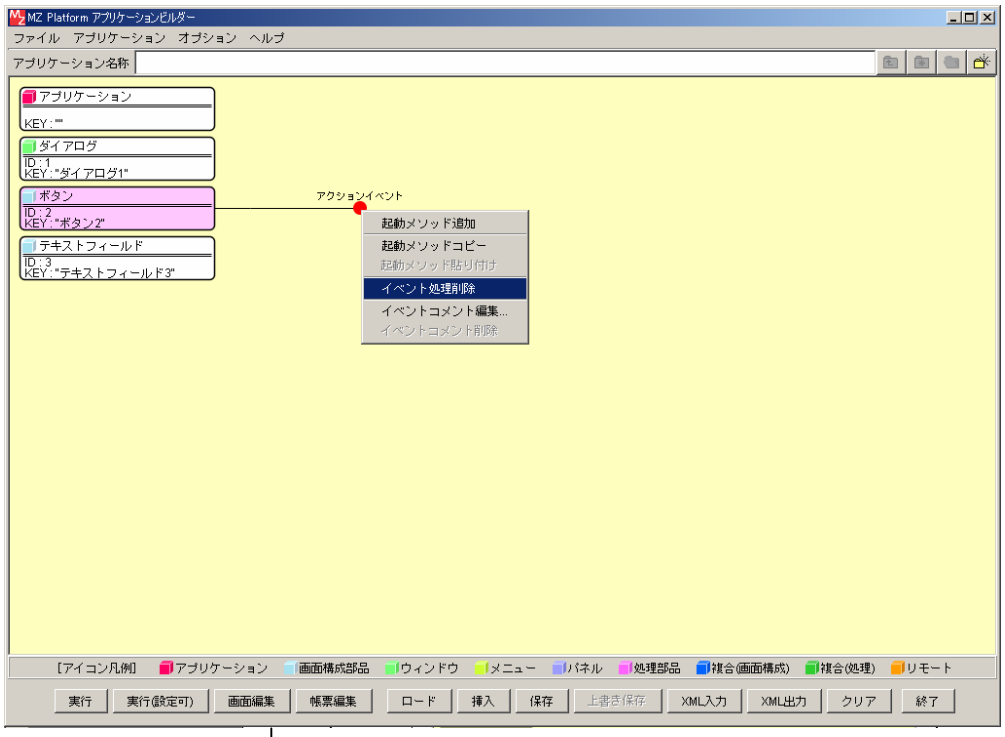
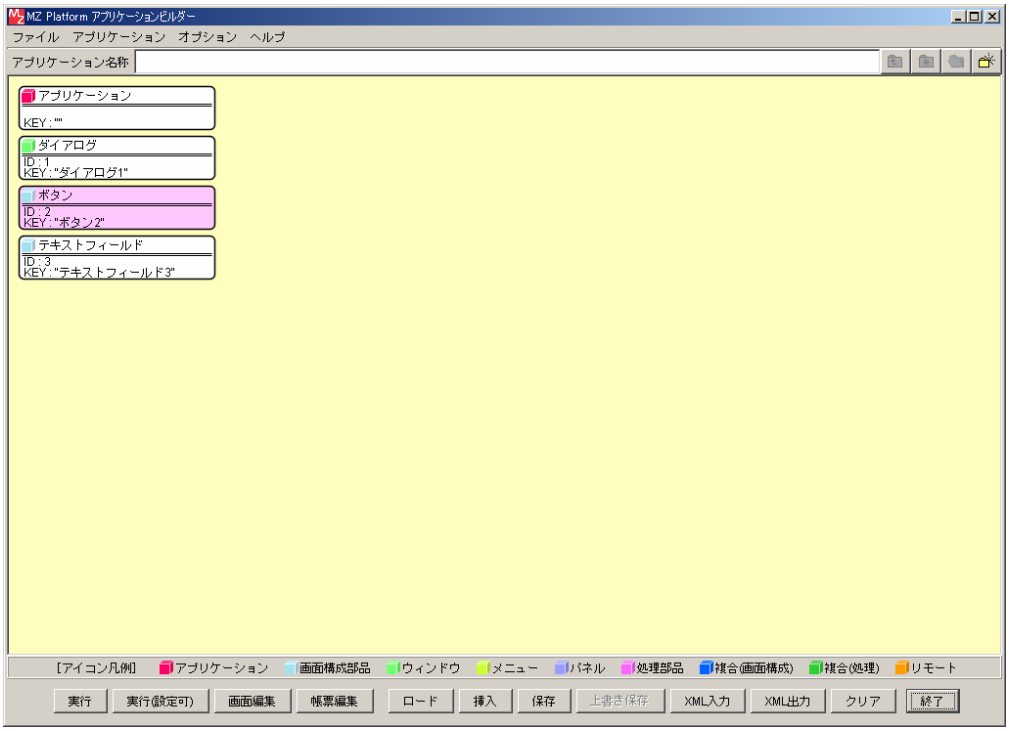
10)起動メソッドのコピー（複数）

画面	アプリケーションビルダー メイン画面
手順	コピー対象のイベント上でマウスを右クリックし、『起動メソッドコピー』を指示 → 対象イベントから起動される一連の処理メソッドがペースト対象となる 

11)起動メソッドのペースト

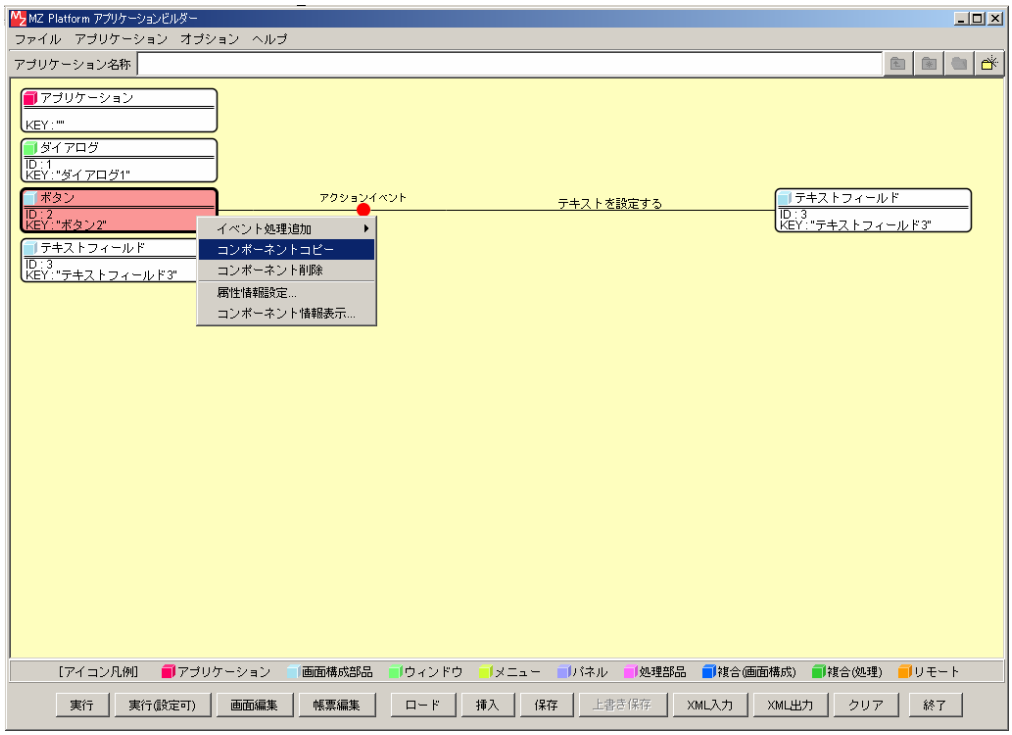
画面	アプリケーションビルダー メイン画面
手順	コピー先のイベント表示上でマウスを右クリックし、『起動メソッド貼り付け』を指示 ※事前にコピー処理を行っておくこと  コピーした起動メソッドが追加される  ※注意事項 ペーストされる内容は、コピー時の設定ではなく、ペースト指示時の設定となります

12)イベント処理の削除

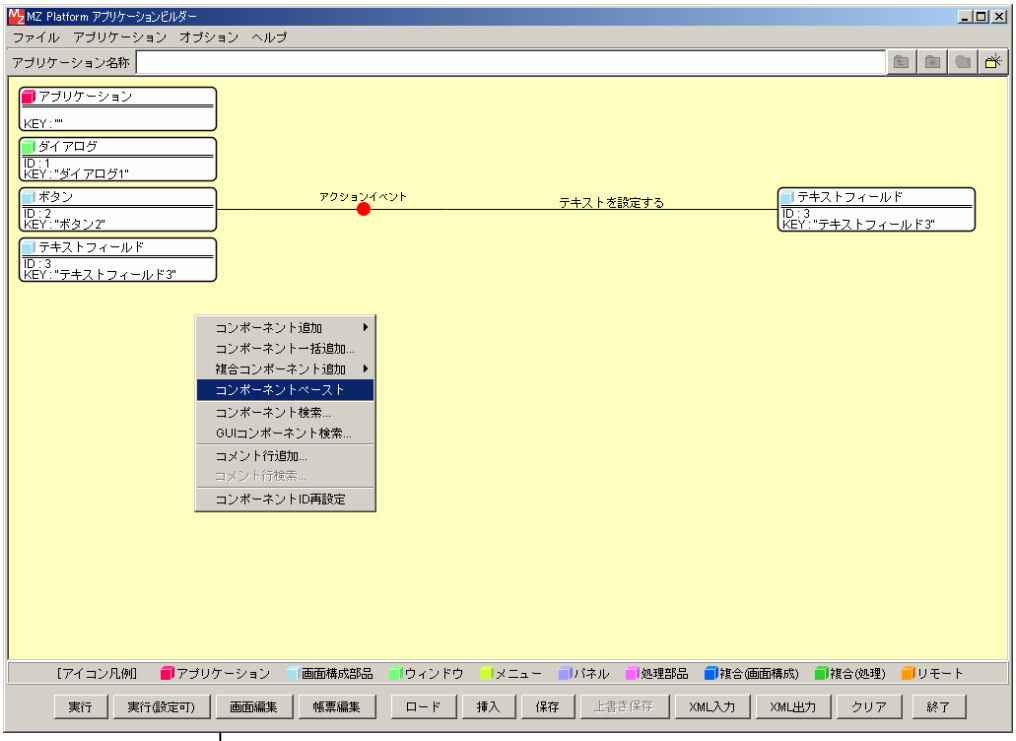
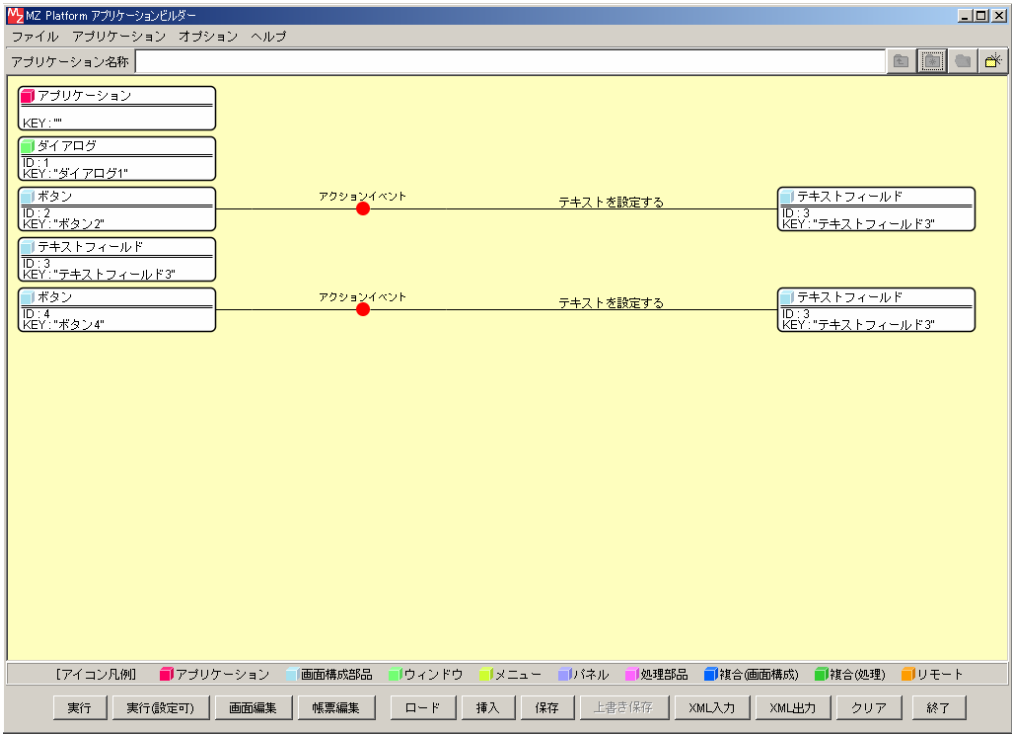
画面	アプリケーションビルダー メイン画面
手順	削除対象のイベント上でマウスを右クリックし、『イベント処理削除』を指示  対象のイベント処理が削除される  ※注意事項 イベントに既に起動メソッドが設定されている場合、それらもまとめて削除されます

5.1.6. コンポーネントのコピー／ペースト

1)コンポーネントのコピー

画面	アプリケーションビルダー メイン画面
手順	コピー対象のコンポーネント上でマウスを右クリックし、『コンポーネントコピー』を指示 → 対象のコンポーネントがペースト対象となる 

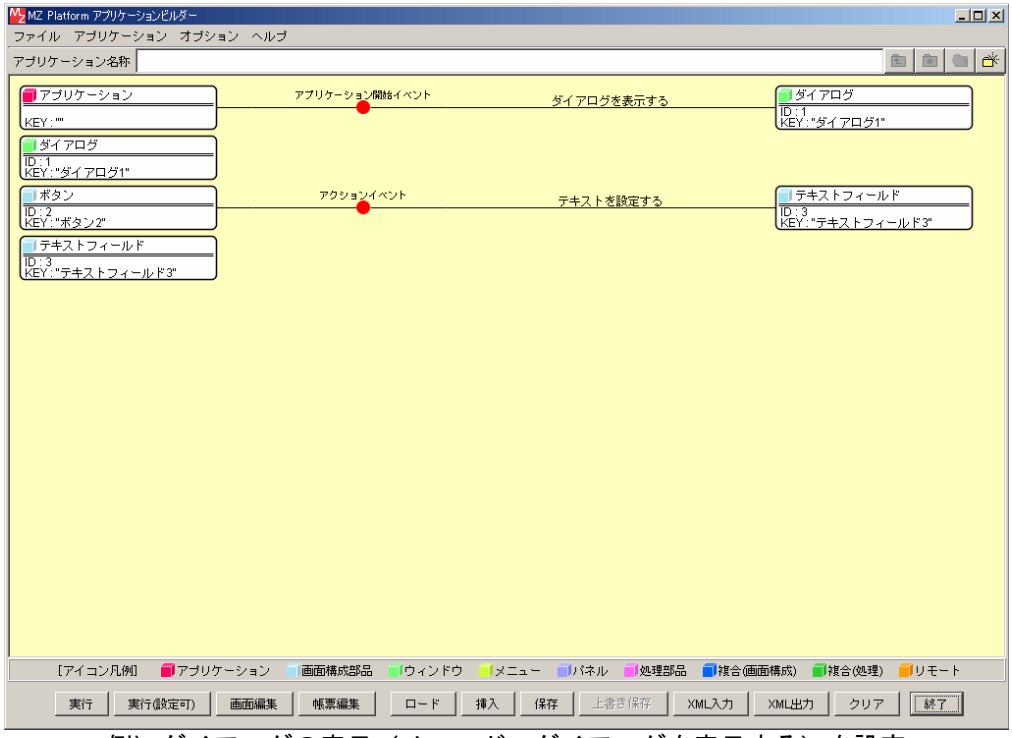
2)コンポーネントのペースト

画面	アプリケーションビルダー メイン画面
手順	背景にてマウスを右クリックし、『コンポーネントペースト』を指示 ※事前にコピー処理を行っておくこと  コピーしたコンポーネントが追加される  **注意事項** コピー元のコンポーネントに設定されているイベント処理もすべてコピーされます

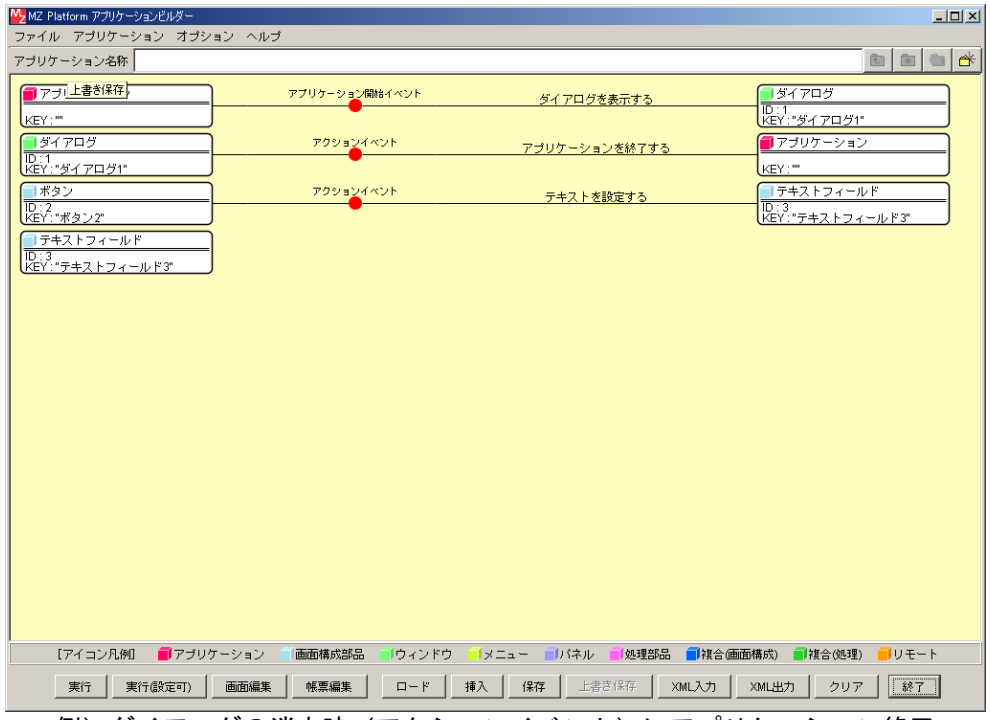
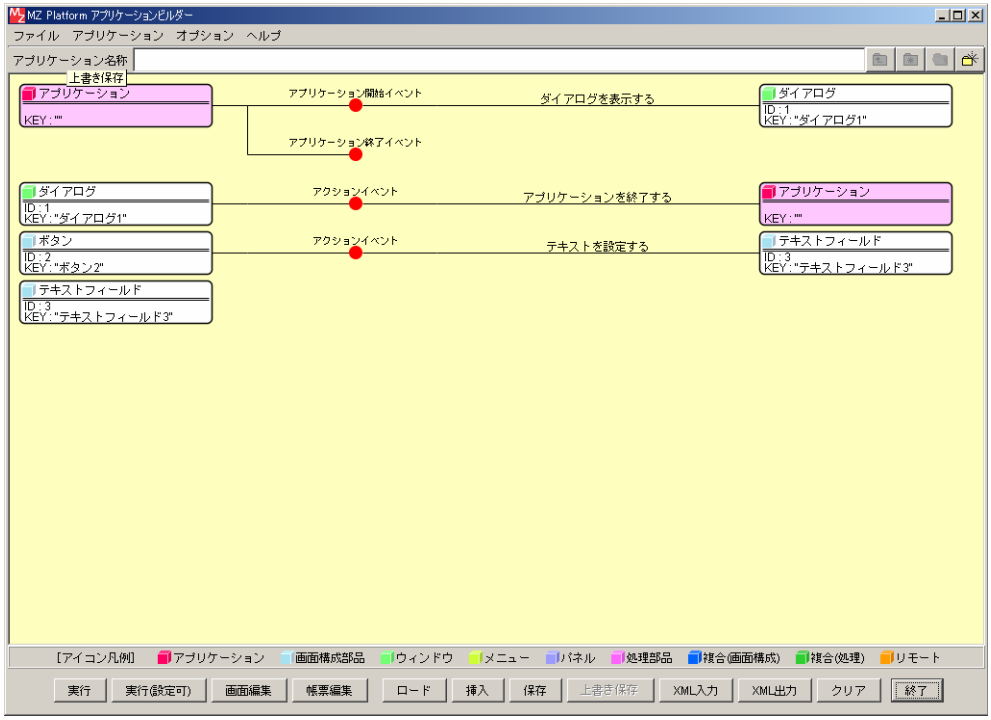
5.1.7. アプリケーション初期処理／終了処理の設定

アプリケーションを実行するには、起動時の処理、および終了時の処理を設定する必要があります。初期処理が何もない場合、アプリケーションは起動されません。アプリケーションの初期処理／終了処理の制御はプラットフォーム基幹機能である Component-Bus が行います。

1)アプリケーション初期処理の設定

画面	アプリケーションビルダー メイン画面
手順	画面最上部に表示されているコンポーネント“アプリケーション”から、 通常のコンポーネント間接続と同様に、アプリケーション開始イベント接続を行う。  例）ダイアログの表示（メソッド：ダイアログを表示する）を設定

2)アプリケーション終了処理の設定

画面	アプリケーションビルダー メイン画面
手順	①アプリケーション終了の設定 任意のコンポーネントから発生するアプリケーションを終了するイベントを、アプリケーションの終了処理（メソッド:アプリケーションを終了する）に接続する。終了処理ではプロセスを終了させるため、全ての画面や処理が強制的に終了される。 ※終了処理を省略するとアプリケーションが終了しないため、必ず実施すること  例）ダイアログの消去時（アクションイベント）にアプリケーション終了 ②アプリケーション終了イベントの設定 アプリケーション終了時に行いたい処理があれば、画面最上部に表示されているアプリケーションコンポーネントから、アプリケーション終了イベントの接続を行う。 

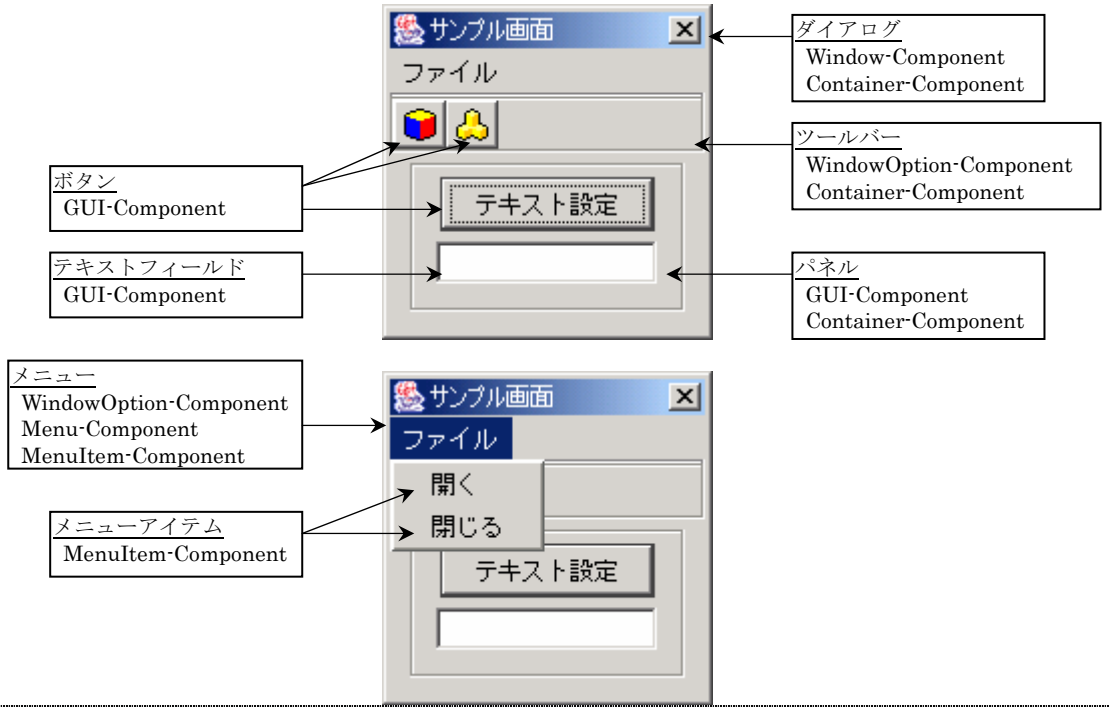
5.1.8. 画面配置の設定

画面を伴うアプリケーションを構築する場合、ダイアログなどの Window コンポーネントを使用して画面のベースとなる Window を作成し、その上にボタンなどの GUI コンポーネントを配置します。

本プラットフォームでは画面の配置を以下のようにコンポーネントを区分し、これらの組み合わせを階層化することによって画面を構成します。

コンポーネント種別	貼付け先	貼付け対象
Window コンポーネント 例：ダイアログ／フレーム	－（貼付け不可）	WindowOption-Component GUI-Component
WindowOption コンポーネント 例：メニュー／ツールバー	Window-Component	任意（コンポーネント依存）
Menu コンポーネント 例：メニュー	任意（コンポーネント依存）	MenuItem-Component
MenuItem コンポーネント 例：メニュー／メニューアイテム	Menu-Component	－（貼付け不可）
GUI コンポーネント 例：ボタン／パネル	Window-Component Container-Component	任意（コンポーネント依存）
画面を伴わないコンポーネント 例：アプリケーションコンポーネント	－（貼付け不可）	－（貼付け不可）
コンテナコンポーネント 例：ダイアログ／パネル／ツールバー	任意（コンポーネント依存）	GUI-Component

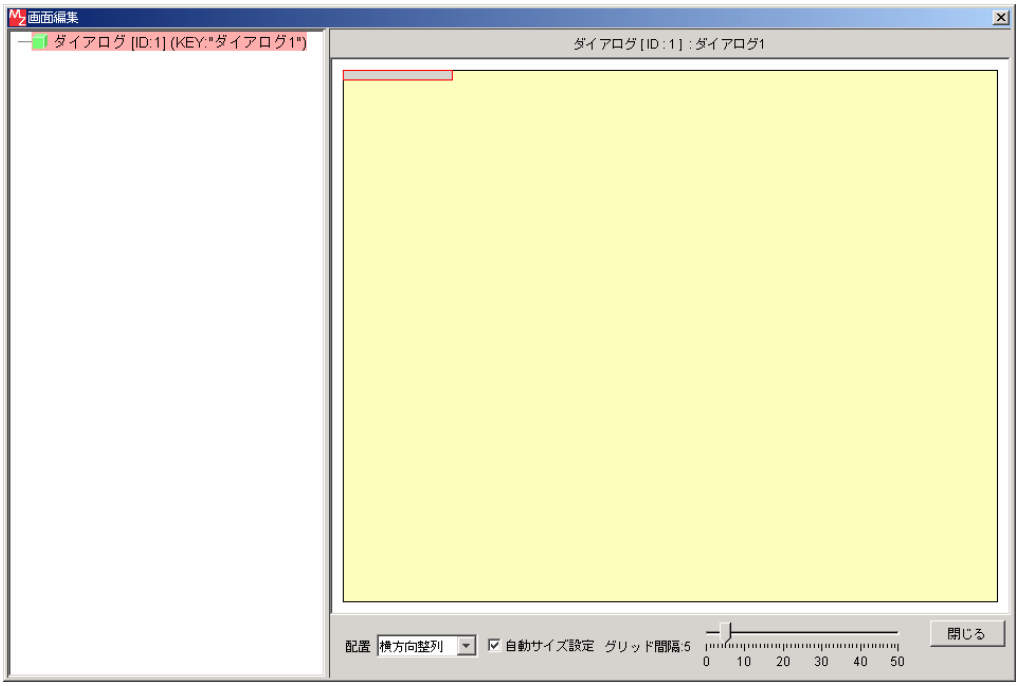
■画面構成例



■画面階層

- ダイアログ：“サンプル画面”
  - メニュー：“ファイル”
    - メニューアイテム：“開く”
    - メニューアイテム：“閉じる”
  - ツールバー
    - ボタン
    - ボタン
  - パネル
    - ボタン：“テキスト設定”
    - テキストフィールド

1)画面の構成

画面	アプリケーションビルダー 画面編集ダイアログ
手順	①画面編集ダイアログの表示 『アプリケーションビルダー メイン画面』の【画面編集】ボタンを押下し、 『画面編集ダイアログ』を表示する 

1)画面の構成 続き

手順

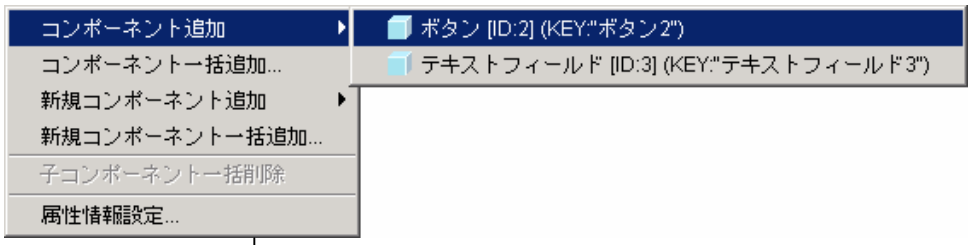
②コンポーネントの貼付け

a)単一コンポーネント追加

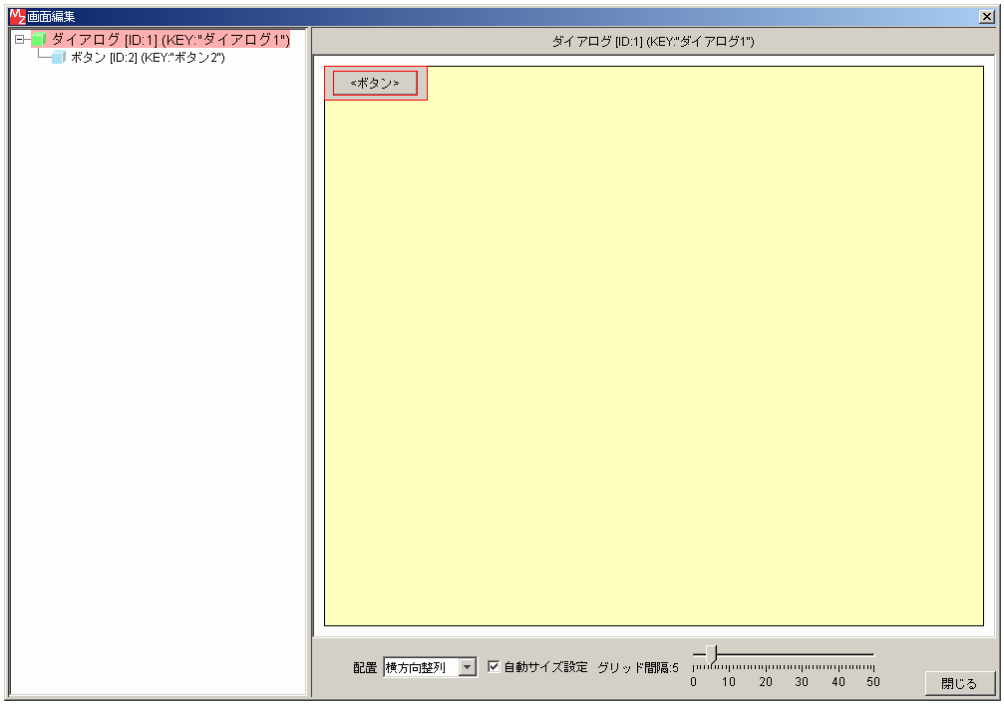
以下のどちらかの操作でメニューを表示し、対象のコンポーネントを選択する

・画面左側の階層ツリーのコンテナコンポーネント
（ダイアログ／フレーム／パネル）を右クリックしてメニューを表示

・画面右側の画面レイアウトの背景を右クリックしてメニューを表示



対象のコンポーネントが追加される



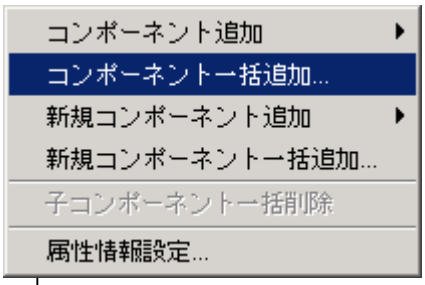
※コンポーネント挿入

以下の操作により、コンポーネントを任意の位置に挿入することも可能

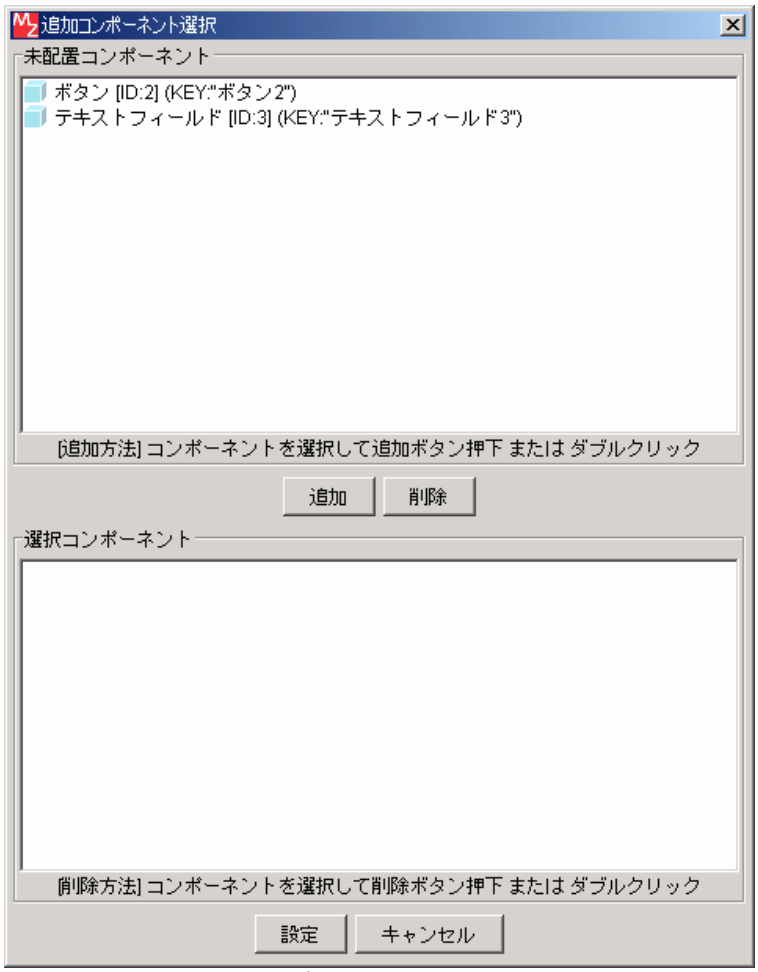
・画面左側の階層ツリーのコンポーネントを右クリックしてメニューを表示

1)画面の構成 続き

手順	②コンポーネントの貼付け b)複数コンポーネント一括追加 以下のどちらかの操作でメニューを表示し、一括追加を選択する ・画面左側の階層ツリーのコンテナコンポーネント （ダイアログ／フレーム／パネル）を右クリックしてメニューを表示 ・画面右側の画面レイアウトの背景を右クリックしてメニューを表示

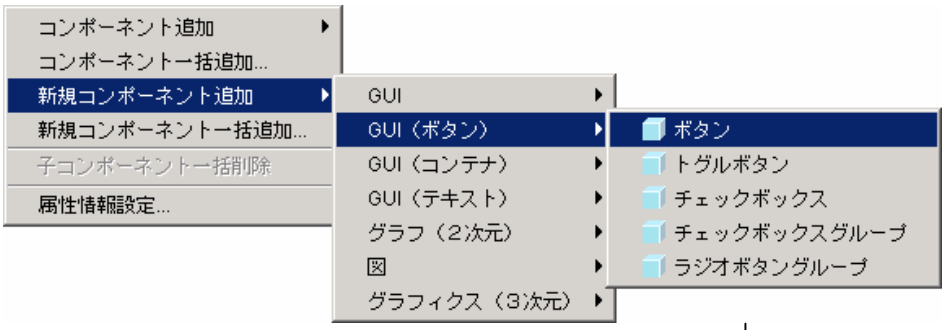


コンポーネント選択画面が表示される

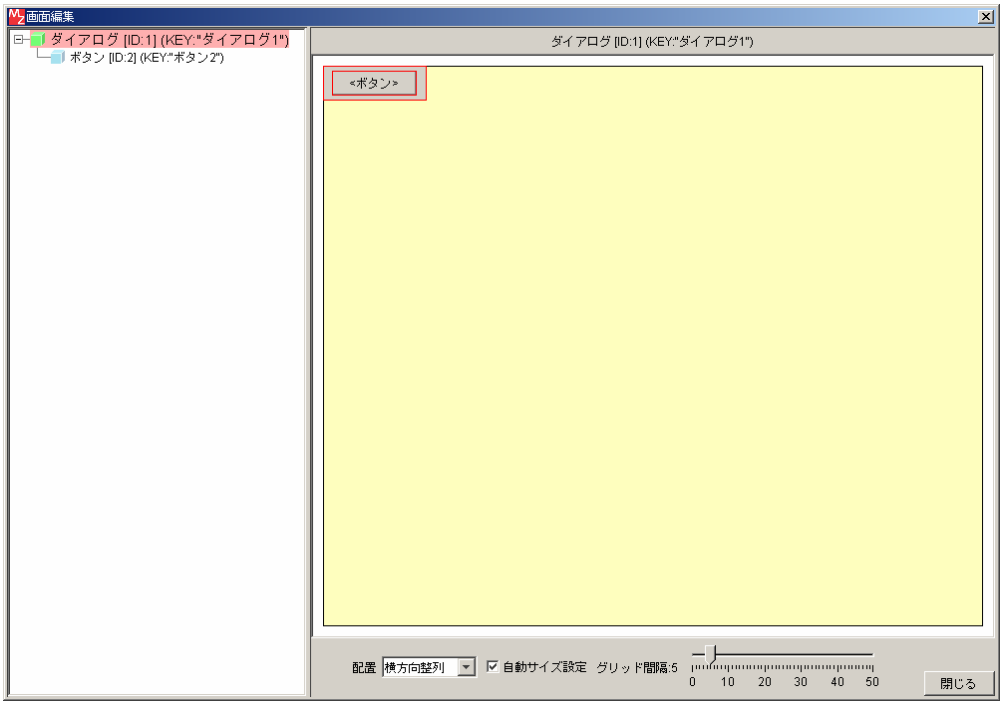


この画面上で配置するコンポーネントを選択する。選択方法は以下の 2 つ。
・上段から対象コンポーネントを選択し（複数可）、[追加]ボタン押下
・上段の対象コンポーネントをダブルクリック

手順	②コンポーネントの貼付け c)コンポーネント新規追加 以下のどちらかの操作でメニューを表示し、一括追加を選択する ・画面左側の階層ツリーのコンテナコンポーネント （ダイアログ／フレーム／パネル）を右クリックしてメニューを表示 ・画面右側の画面レイアウトの背景を右クリックしてメニューを表示

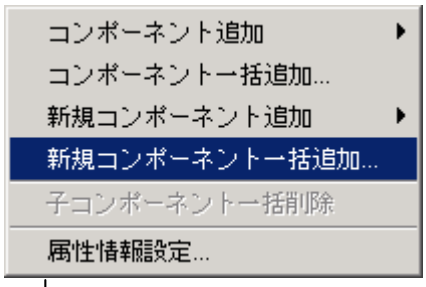


コンポーネントが生成され、追加される

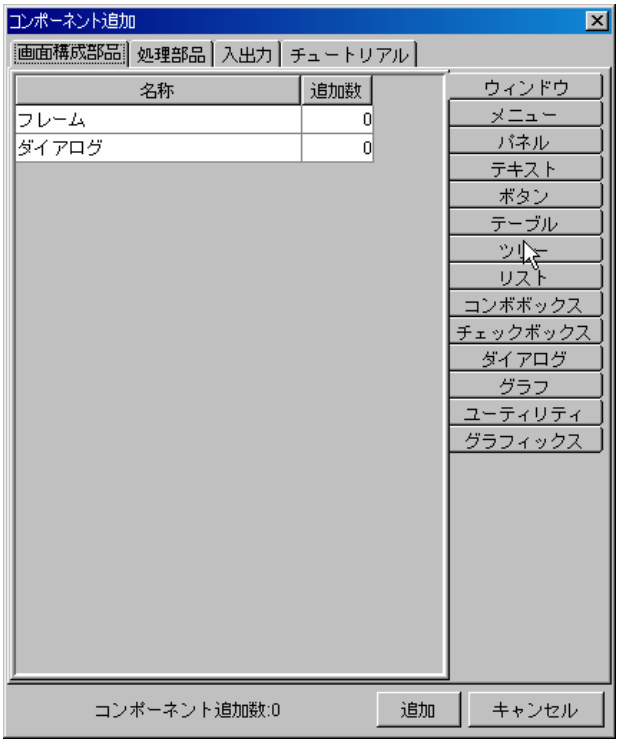


※注意
この機能は新たにコンポーネントを追加するため、ここで指定したものは
アプリケーション構築画面上にも追加される

手順	②コンポーネントの貼付け d)複数コンポーネント新規追加 以下のどちらかの操作でメニューを表示し、一括追加を選択する ・画面左側の階層ツリーのコンテナコンポーネント （ダイアログ／フレーム／パネル）を右クリックしてメニューを表示 ・画面右側の画面レイアウトの背景を右クリックしてメニューを表示



コンポーネント一括追加画面が表示される

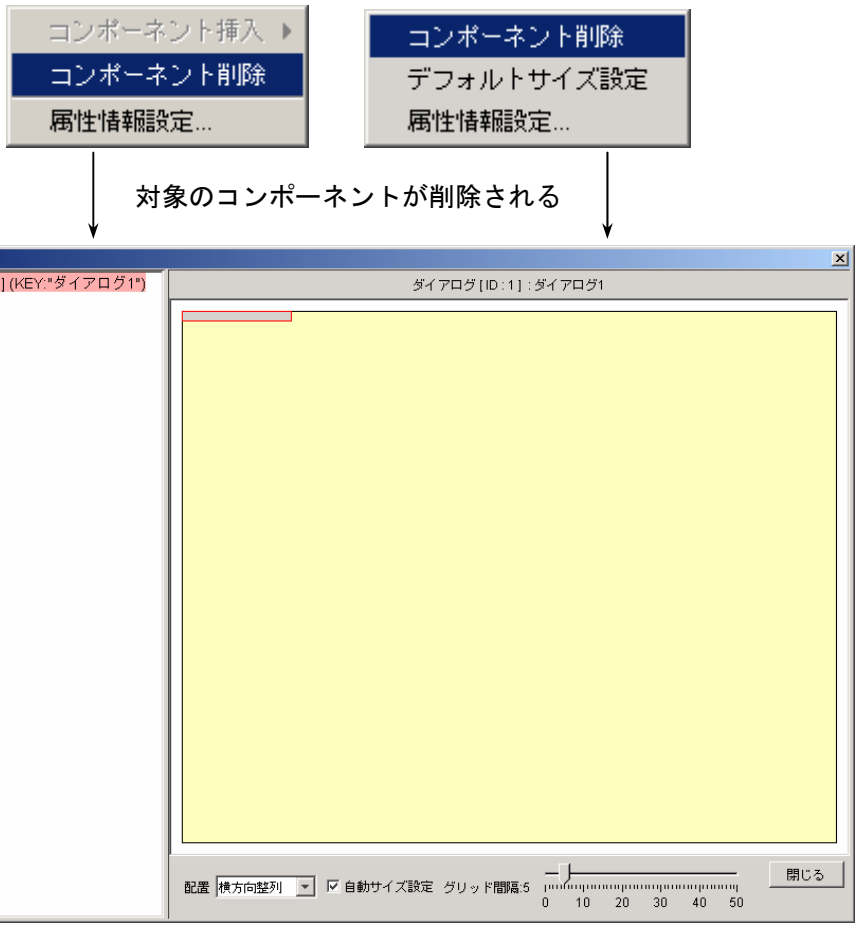


この画面上で追加対象コンポーネントを選択し、[追加]ボタンを押下すると、選択したすべてのコンポーネントが新規に生成され、画面に追加される。

※注意
この機能は新たにコンポーネントを追加するため、ここで指定したものはアプリケーション構築画面上にも追加される

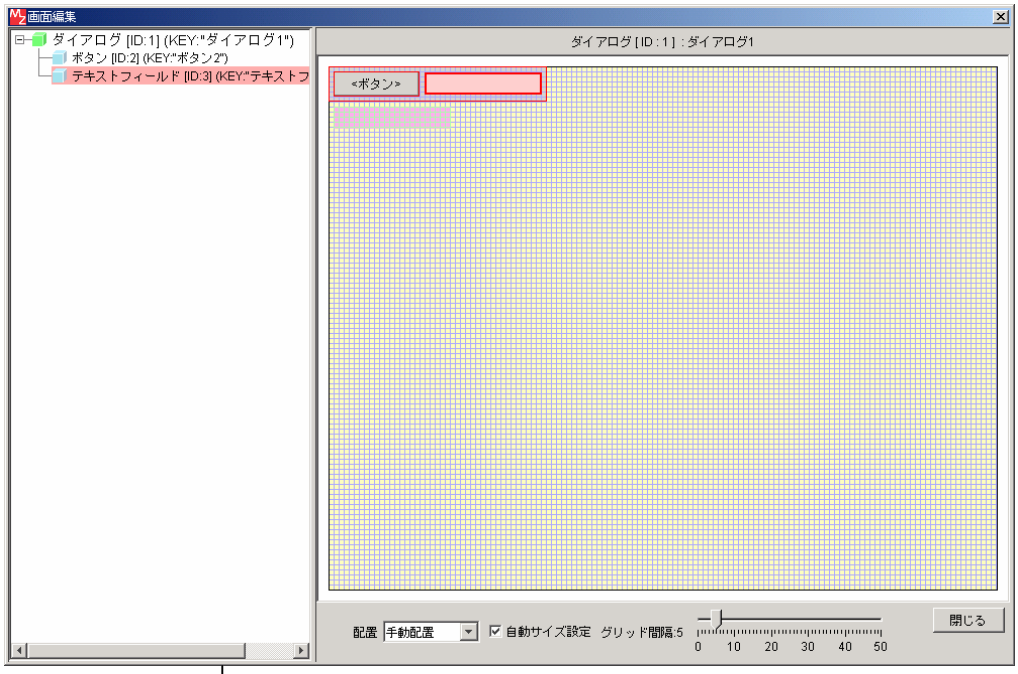
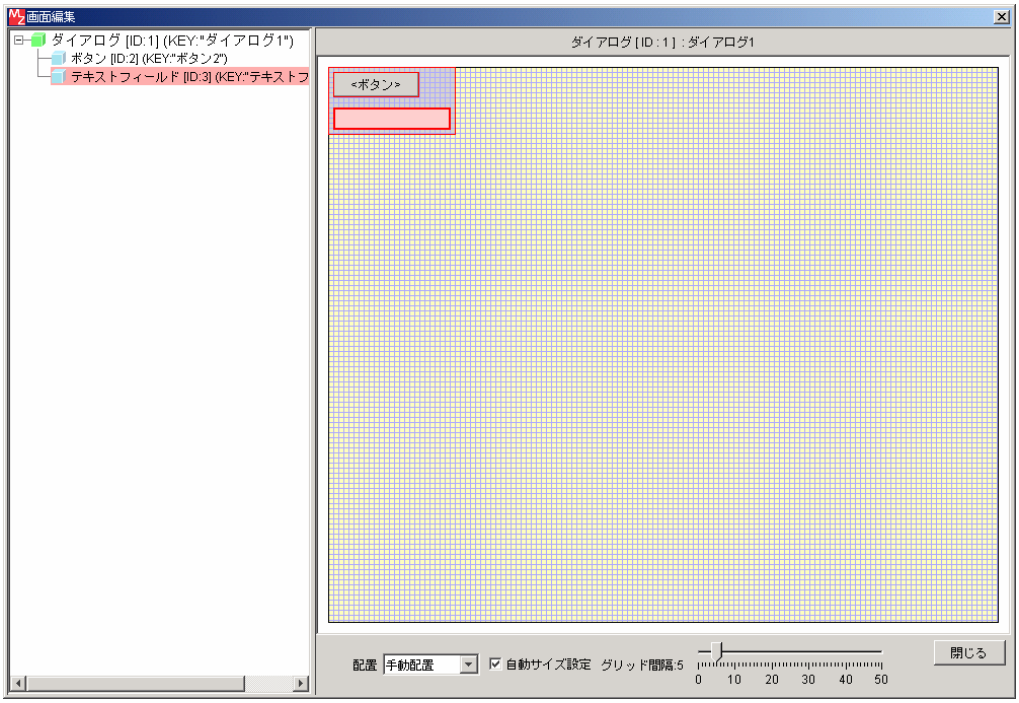
1)画面の構成 続き

手順	③コンポーネントの削除 以下のどちらかの操作でメニューを表示し、コンポーネント削除を選択する ・画面左の階層ツリーの対象コンポーネントを右クリックしてメニュー表示 ・画面右の画面レイアウトの対象コンポーネントを右クリックしてメニュー表示

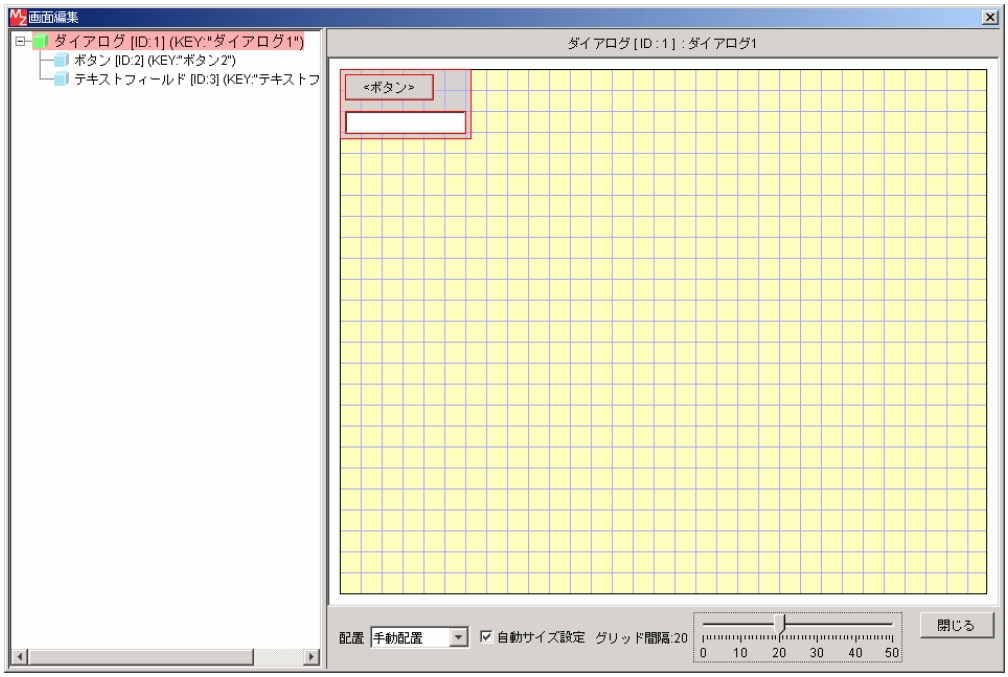


※注意
この機能は画面からコンポーネントを削除するものであり、アプリケーションからは削除されない

2)画面レイアウトの手動設定

画面	アプリケーションビルダー 画面編集ダイアログ
手順	**①手動配置モードに設定** 配置モードを“手動配置”にする **②コンポーネントの移動** **コンポーネントをマウスで Drag によって任意の位置に移動させる**  Drop した位置にコンポーネントが移動する 

2)画面レイアウトの手動設定 続き

手順	③移動量の調整 部品の移動は背景のグリッド線にあわせて行われる。グリッド線の間隔を調整することにより、部品の配置位置を自由に揃えることが可能となる。 グリッド線間隔の調整は、画面下のスライダーによって設定する。 
特記事項	①画面配置モードの切り替え 画面配置モードを『手動配置』以外に切り替えると、画面配置は再設定され以前に設定した配置情報はすべてクリアされる ②配置方法の選択について ◇手動配置 画面レイアウト設定操作は容易に行うことが可能。ただし、ボタンなどの GUI コンポーネントは稼動するプラットフォーム（OS／Window システム）によって表示サイズ／文字サイズなどが変わるため、絶対座標で配置を行う手動配置では注意が必要となる。開発環境／実行環境が同じプラットフォームである場合には、問題なく手動配置を行うことができる。 ◇自動配置（横方向／縦方向／領域／矩形分割） 要件にあわせた配置を行うには、パネルを使用して配置を階層的に構築する必要がある。細かな設定を行うと画面レイアウト設定が複雑になるが、相対的な配置設定を行うため、稼動するプラットフォーム（OS／Window システム）に依存しない汎用的な画面レイアウト設定が可能である。マルチプラットフォームアプリケーションの構築時には、自動配置が望ましい。

3)画面レイアウトの自動設定

画面レイアウトの設定には以下の 4 つの自動配置モードが提供されています。作成する画面設計にあわせて、適切な設定モードを選択してください。

①横方向整列

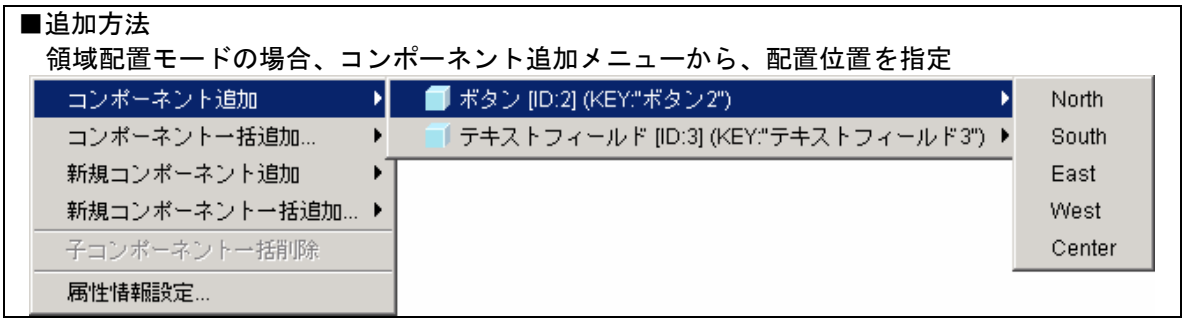
設定された範囲内で、コンポーネントを左から右に並べます。並びきらない場合は次の行（下段）の左端から順に並べます。

②縦方向配置

設定された範囲内で、コンポーネントを上から下に並べます。並びきらない場合は次の列（右側）の上から順に並べます。

③領域配置

配置領域を上側／下側／右側／左側／中央の 5 つに分け、コンポーネントの配置を設定します。コンポーネント配置時には、上側／下側／右側／左側／中央のいずれかを指示します。このとき、全体の配置領域にあわせてコンポーネントの表示サイズも自動的に調整されます。

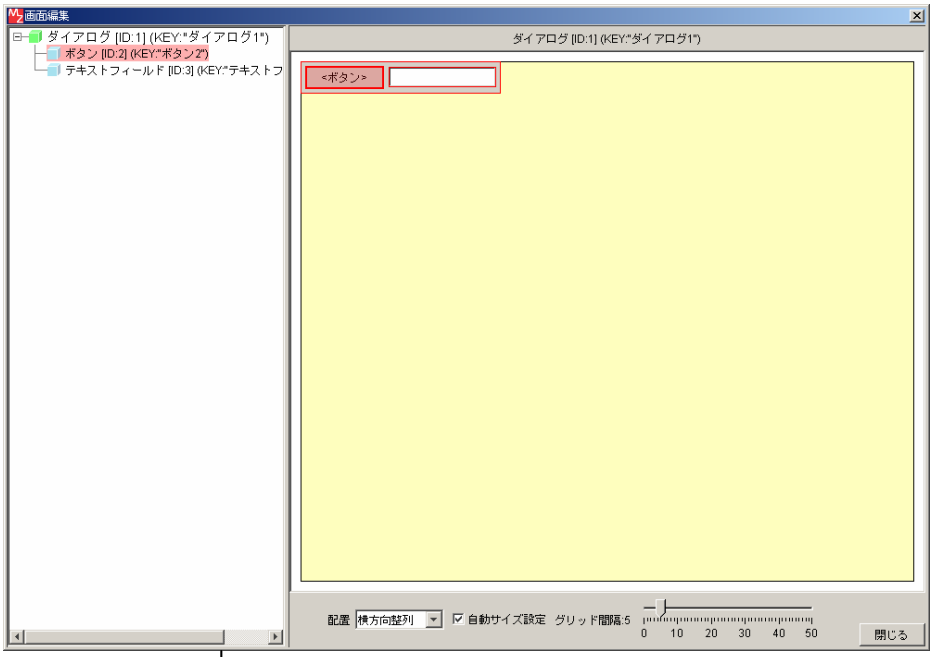
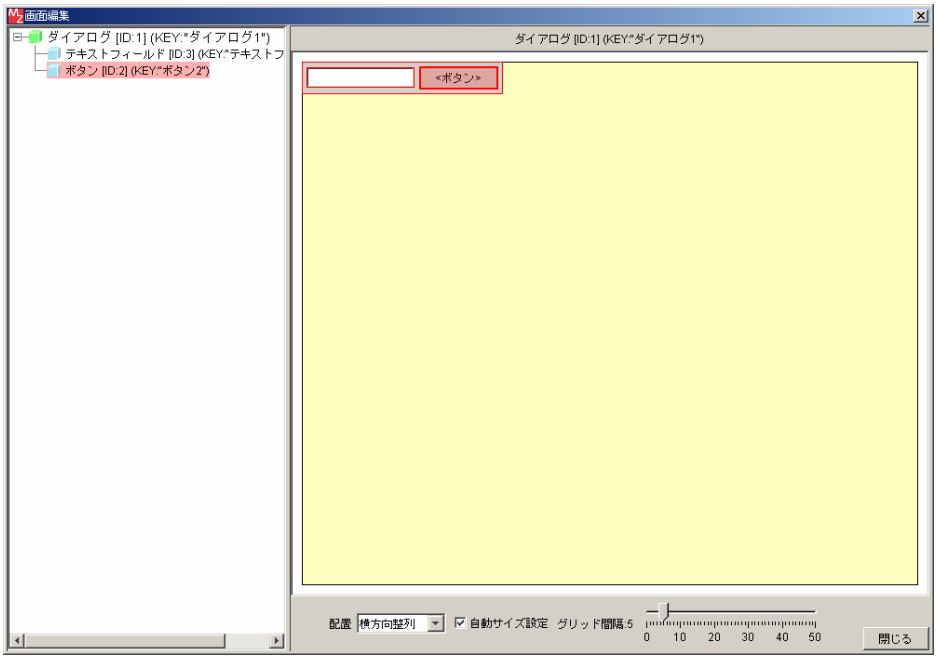


④矩形分割配置

縦横それぞれの表示数を指定することで、表示領域を M×N の領域に分割します。追加された GUI コンポーネントは左上から右に順に配置され、分割された領域の表示サイズは配置される GUI コンポーネントのサイズにあわせて自動調節されます。

4)GUI コンポーネントの順番変更

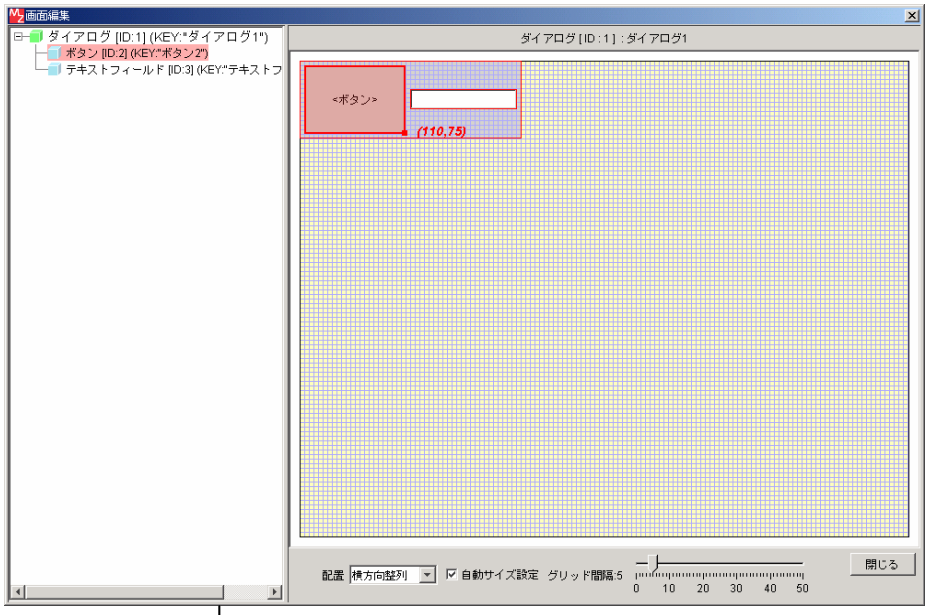
画面レイアウトが横方向配置などの自動設定になっている場合、GUI コンポーネントの表示順は画面左側の画面階層ツリーの表示順（登録順）によって決まります。順序を変更するにはツリー上で、その順序を変更します。

画面	アプリケーションビルダー 画面編集ダイアログ
手順	①GUI コンポーネントの順序変更 ツリー上のコンポーネントをマウスで Drag し、任意の位置に移動する。  Drop した位置で表示順が決定される 

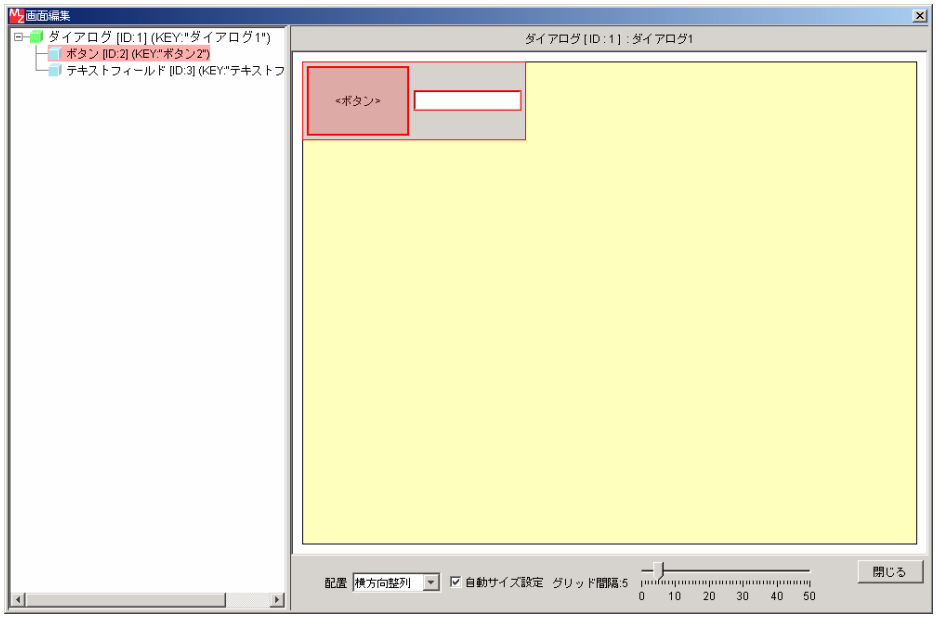
5)GUI コンポーネントのサイズ設定

画面	アプリケーションビルダー 画面編集ダイアログ
手順	①コンポーネントのサイズ変更 コンポーネントの以下の領域でマウス Drag によって、任意のサイズ設定を行う。 ※ドラッグ操作はマウスの位置によって以下のように動きが変わる

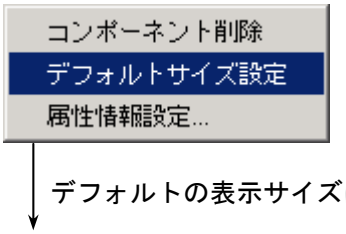
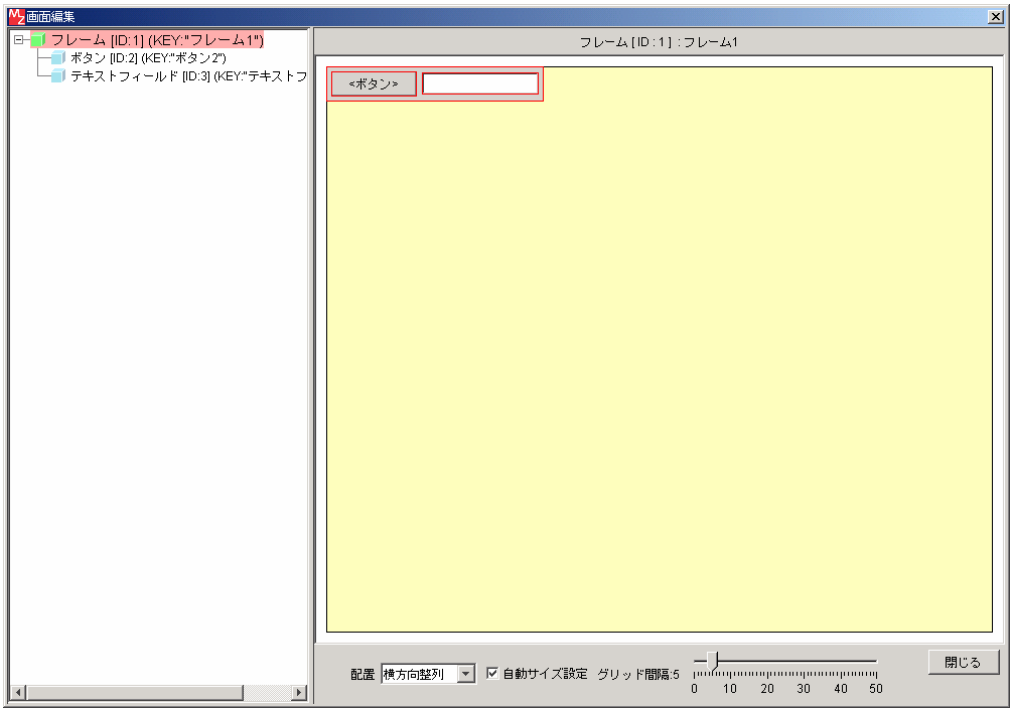
位置	動作
右端	横幅を変更する（縦幅は固定）
下端	縦幅を変更する（横幅は固定）
右下隅	自由にサイズを変更する
上記以外	配置位置を移動する（手動配置時のみ）



Drop した位置でサイズが決定される



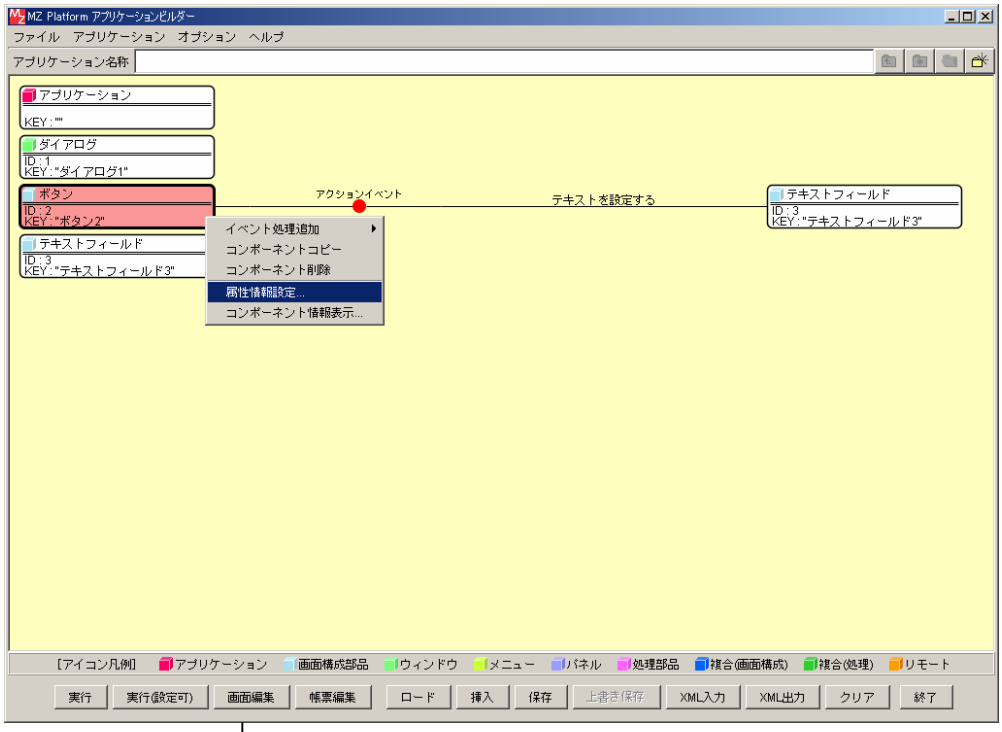
5)GUI コンポーネントのサイズ設定 続き

手順	②コンポーネントのサイズをデフォルトに戻す サイズ設定をデフォルトに戻したい場合、画面右の画面レイアウトの対象コンポーネントを右クリックしてメニュー表示  デフォルトの表示サイズに戻る  ※注意事項 マウスによるサイズ設定については、以下の制限がある ・配置方法が『領域配置』以外の場合に限りサイズ変更が可能 ・GUI 部品によっては、サイズ調整できないものもある

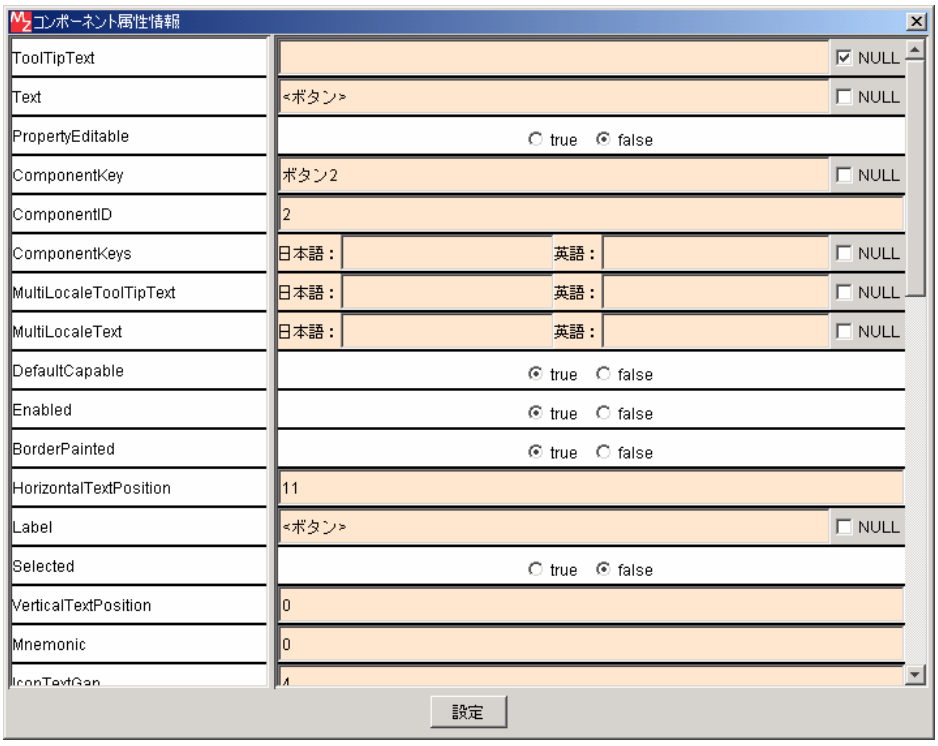
5.1.9. コンポーネント属性の変更

1)アプリケーションビルダー メイン画面からの設定

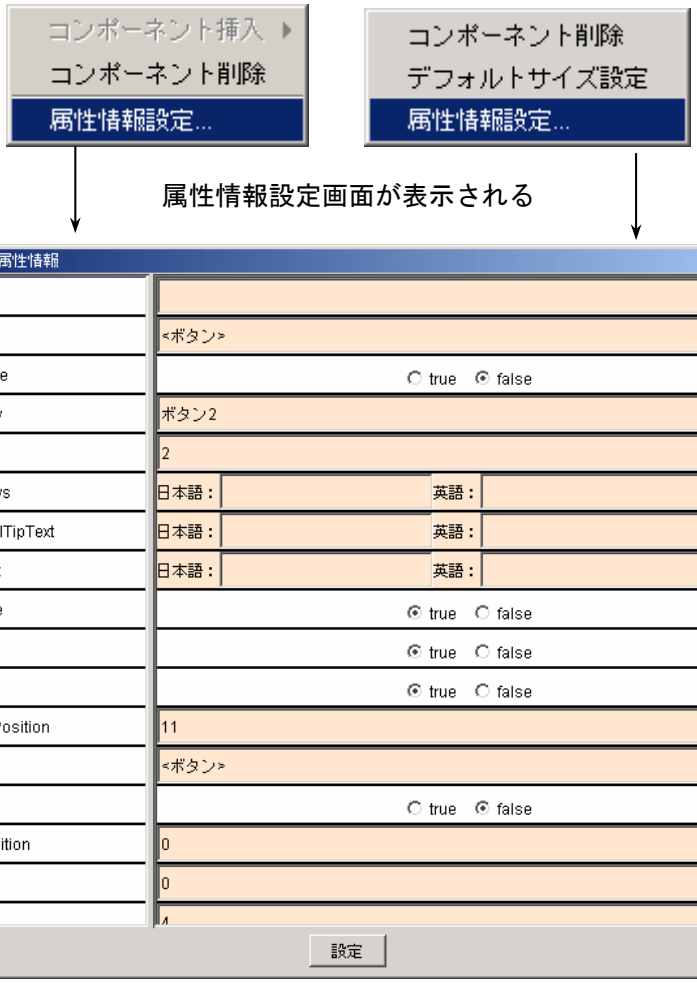
画面	アプリケーションビルダー メイン画面
手順	対象コンポーネントを右クリックしてメニューを表示し、『属性情報設定』を選択 ※設定対象コンポーネントにて左ボタンダブルクリックでも同様



属性情報設定画面が表示される

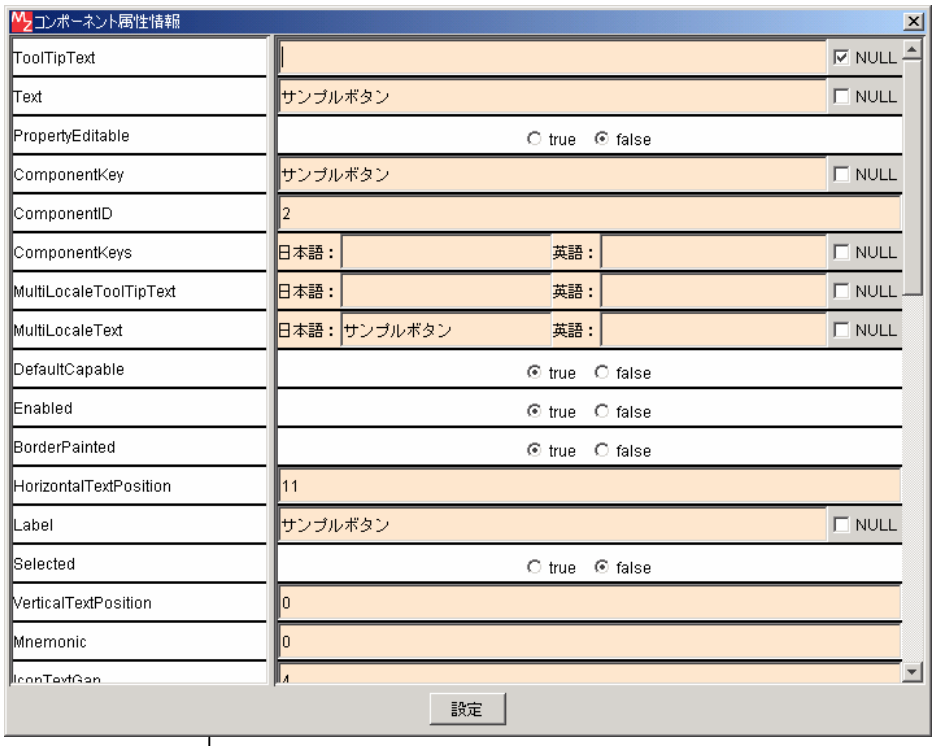


2)画面編集ダイアログからの設定

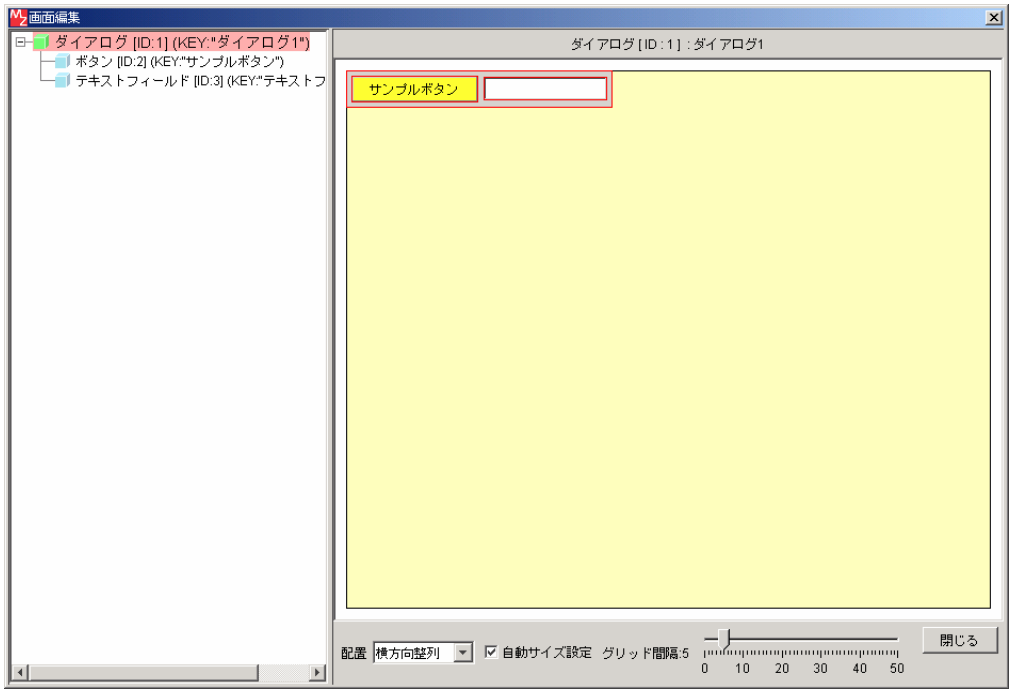
画面	アプリケーションビルダー 画面編集ダイアログ
手順	設定対象のコンポーネントを右クリックしてメニューを表示し、[属性情報設定] を選択。 （ツリーのノードまたはプレビュー内の表示） 

3)属性情報の設定

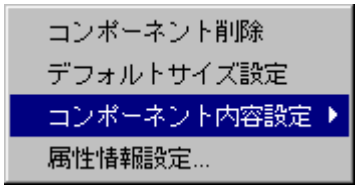
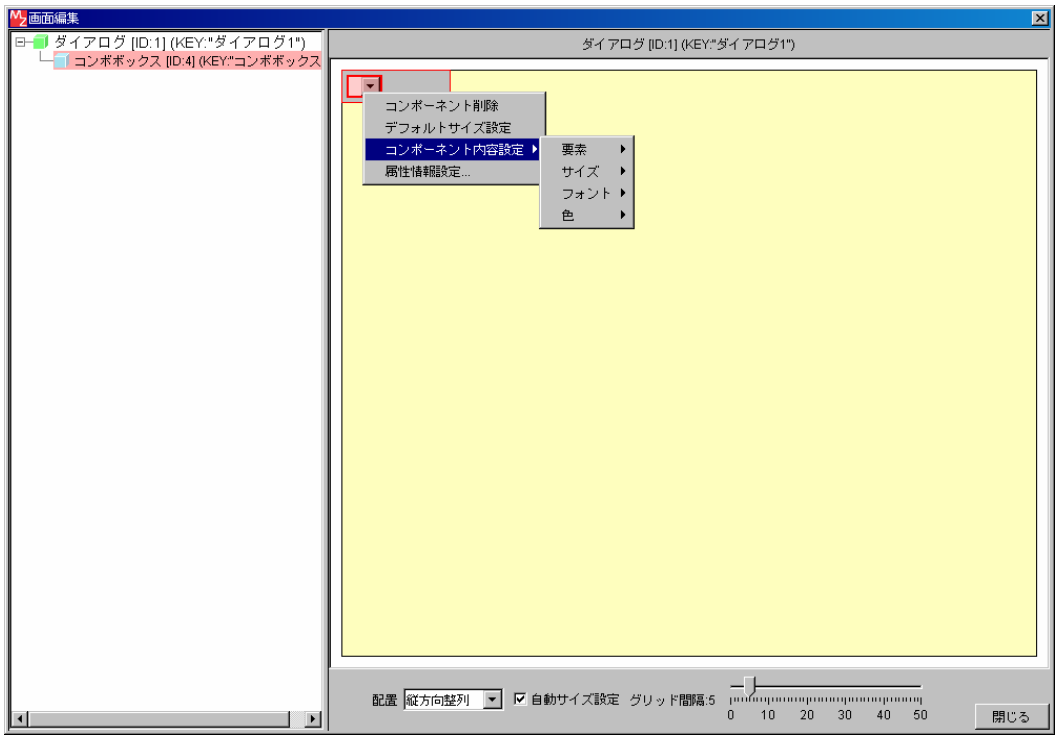
画面	アプリケーションビルダー 属性編集ダイアログ
手順	設定対象の属性をキーボード／マウスから変更する。



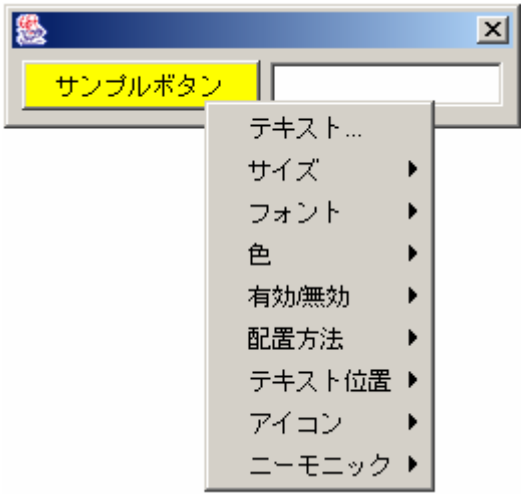
変更した属性情報が反映される



4)ポップアップメニューからの設定

画面	アプリケーションビルダー 画面編集ダイアログ
手順	以下の５つのコンポーネントに限っては、画面編集ダイアログのポップアップメニューから直接属性を設定することもできる。 ・コンボボックス ・リスト ・チェックボックスグループ ・ラジオボタングループ ・テーブル これらのコンポーネントに限り、画面右の画面レイアウトの設定対象コンポーネント上で右クリックした場合に表示されるメニューに、［コンポーネント内容設定］が追加されているので、それを選択。   ［コンポーネント内容設定］のサブメニューは、各コンポーネントが提供する属性設定機能である。これは、次章(5.1.9)の編集可能モードでの実行において表示されるメニューと同じである。 ※注意 この方法では、設定が終わっても画面レイアウトは自動的に再描画されない。設定を反映させるには、一度画面レイアウトを左ボタンクリックする必要がある。

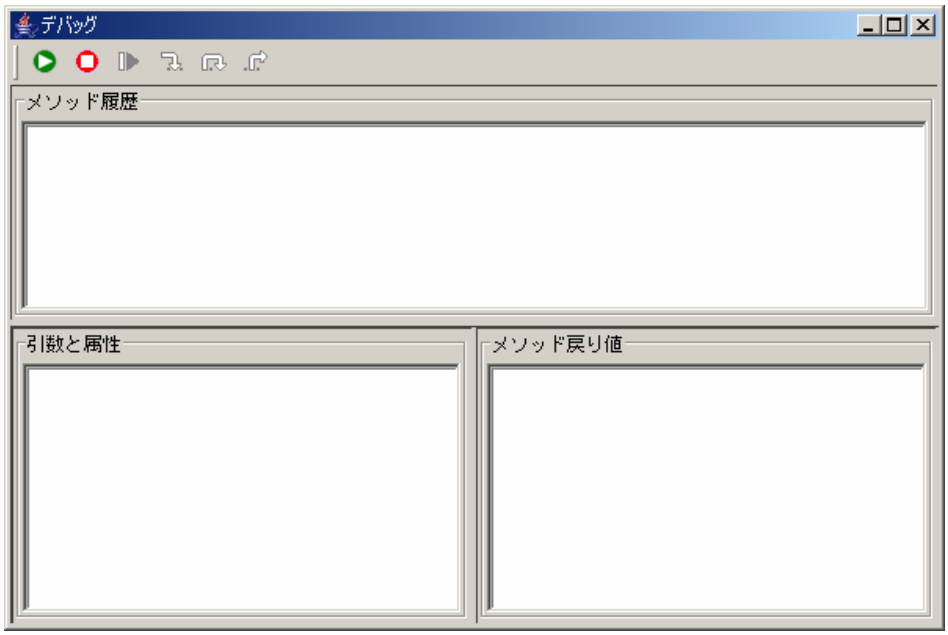
5.1.10. 実行

画面	アプリケーションビルダー メイン画面
手順	①編集可能モードでの実行 [実行(設定可)]ボタンを押下し、構築したアプリケーションを実行する。各コンポーネントが提供する、属性設定機能を使用して属性の設定を行う。 
	②編集不可モードでの実行 [実行]ボタンを押下し、構築したアプリケーションを実行する。 

5.1.11. デバッグ機能

アプリケーションビルダー上でアプリケーション構築を行う過程において、効率よく作業を進めるためにデバッグ機能を提供します。ここで提供するデバッガは、アプリケーションの実行を任意の位置で停止させる「ブレークポイント設定機能」や、動作を確認しながら一つずつ処理を進める「ステップ実行機能」、実行中のデータの状態などを見る「トレース機能」を備えています。

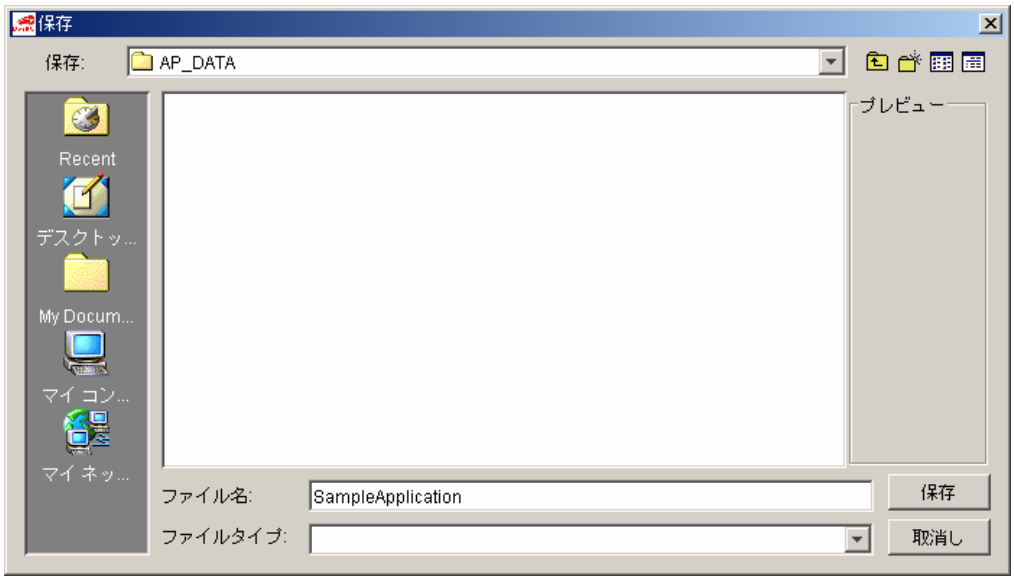
本機能の詳細な使用方法については、「デバッガ操作説明書」をご覧下さい。

画面	アプリケーションビルダー メイン画面
手順	①デバッガの起動 メニューバーから［アプリケーション］−［デバッグ］を選択し、 デバッグ画面を表示する。 

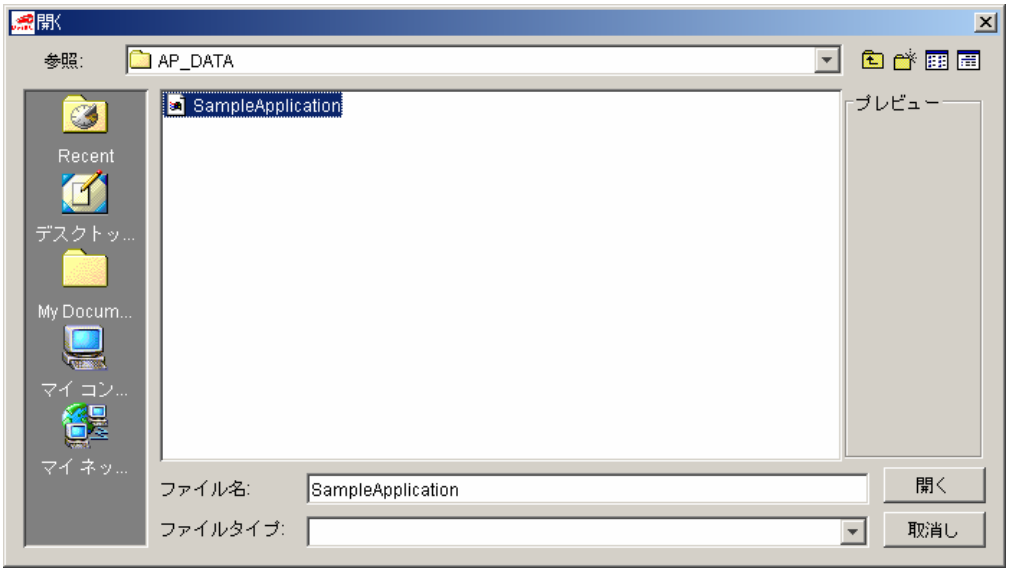
5.1.12. アプリケーションの保存／ロード

作成したアプリケーションをファイルに保存することで、ロード機能を使用して再利用が可能になります。アプリケーションの再編集を行ったり、作成したアプリケーションを別のプラットフォームで使用したりすることがが可能になります。

1)アプリケーションの保存

画面	アプリケーションビルダー メイン画面
手順	①[保存] ボタンをクリック ②保存先ファイル名を指定する 

2)アプリケーションのロード

画面	アプリケーションビルダー メイン画面
手順	①[ロード] ボタンをクリック 現在のアプリケーションがクリアされることが警告される。 必要であれば現状を保存してから、再度[ロード] ボタンをクリックする。 ②ロードファイル名を指定する 

3)アプリケーションの挿入

保存されたアプリケーションを、現在編集中のアプリケーションに挿入する機能です。アプリケーションのファイル情報を読み込み、その中のコンポーネントやイベント接続など、すべての情報を編集中のアプリケーションに追加します。ただし、挿入対象アプリケーションの以下の情報については、挿入時に復元されませんので、注意してください。

＜挿入対象がアプリケーションの場合＞

・アプリケーションコンポーネントからのイベント接続情報

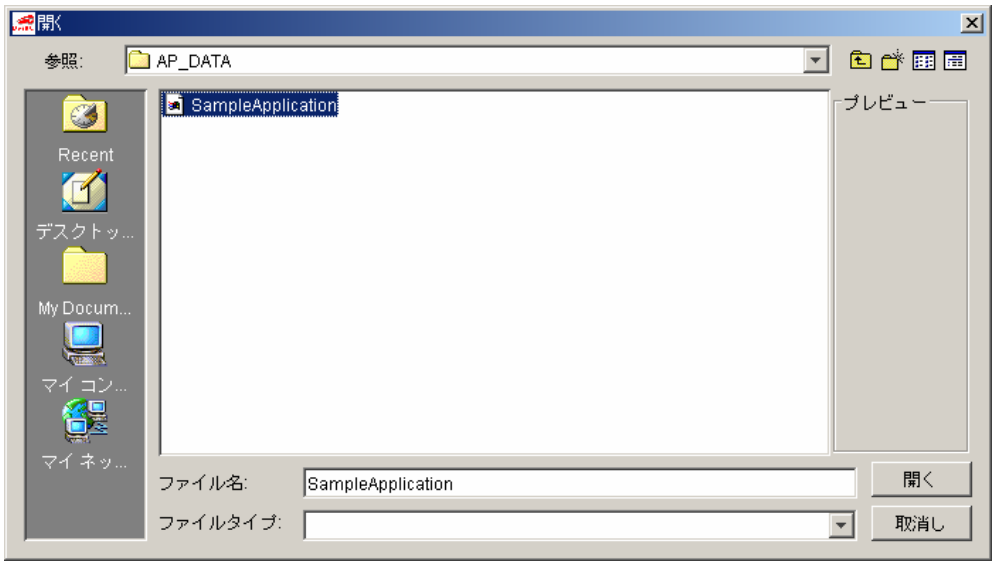
・コンポーネントからアプリケーションコンポーネントへの接続情報

＜挿入対象が複合コンポーネントの場合＞

・最上位複合コンポーネントからのイベント接続情報

・コンポーネントから最上位複合コンポーネントへの接続情報

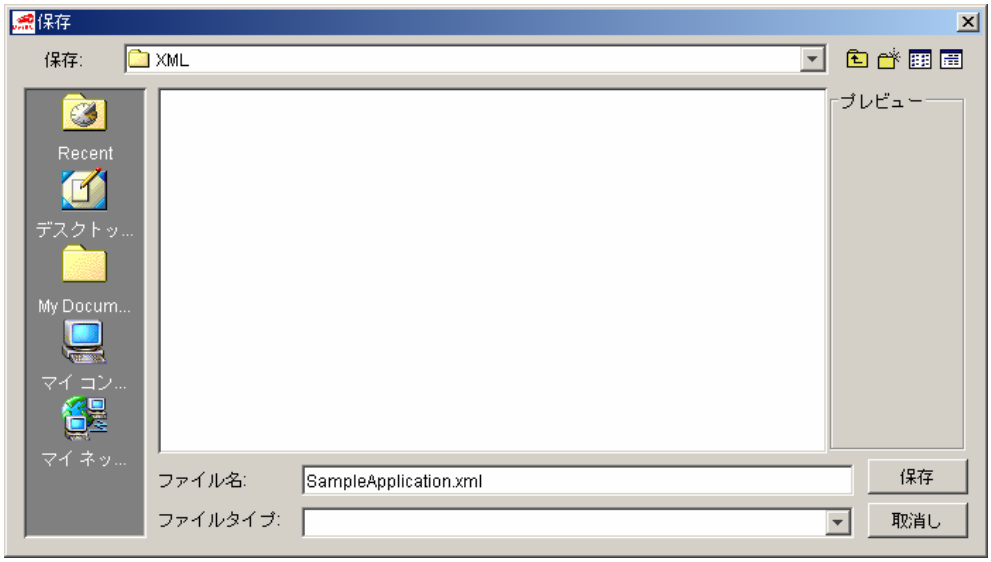
・最上位 GUI 複合コンポーネントの画面レイアウト情報

画面	アプリケーションビルダー メイン画面
手順	［挿入］ボタンをクリックし、挿入ファイル名を指定する 

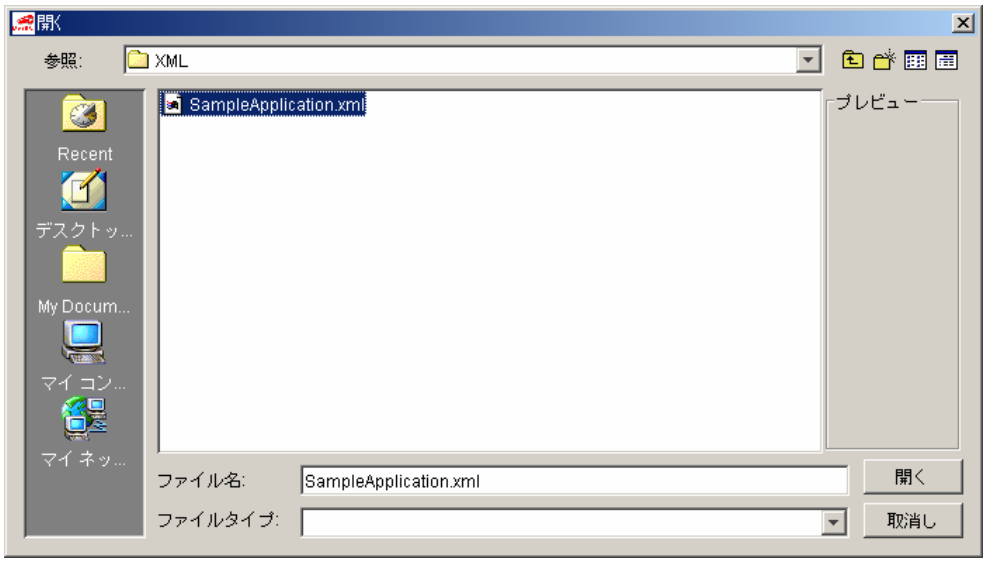
4)アプリケーションの XML 形式ファイル出力

先の保存／ロードは Java の機能を使用したデータの保存のために、保存されたファイルの内容はエディタなどで読むことができず、データ形式は Java の内部仕様に依存しています。

そこで、可視性のある XML テキスト形式でのファイル入出力を行う機能が提供されています。これによって、アプリケーションの再利用だけでなく、アプリケーションの構築や変更などの作業なども可能です。この機能の詳細については、『5.2 XML によるアプリケーション構築』を参照してください。

画面	アプリケーションビルダー メイン画面
手順	①[XML 出力] ボタンをクリック ②保存先ファイル名を指定する 

5)アプリケーションの XML 形式ファイルのロード

画面	アプリケーションビルダー メイン画面
手順	①[XML 入力] ボタンをクリック 現在のアプリケーションがクリアされることが警告される。 必要であれば現状を保存してから、再度[XML 入力] ボタンをクリックする。 ②ロードファイル名を指定する 

6)アプリケーションのクリア

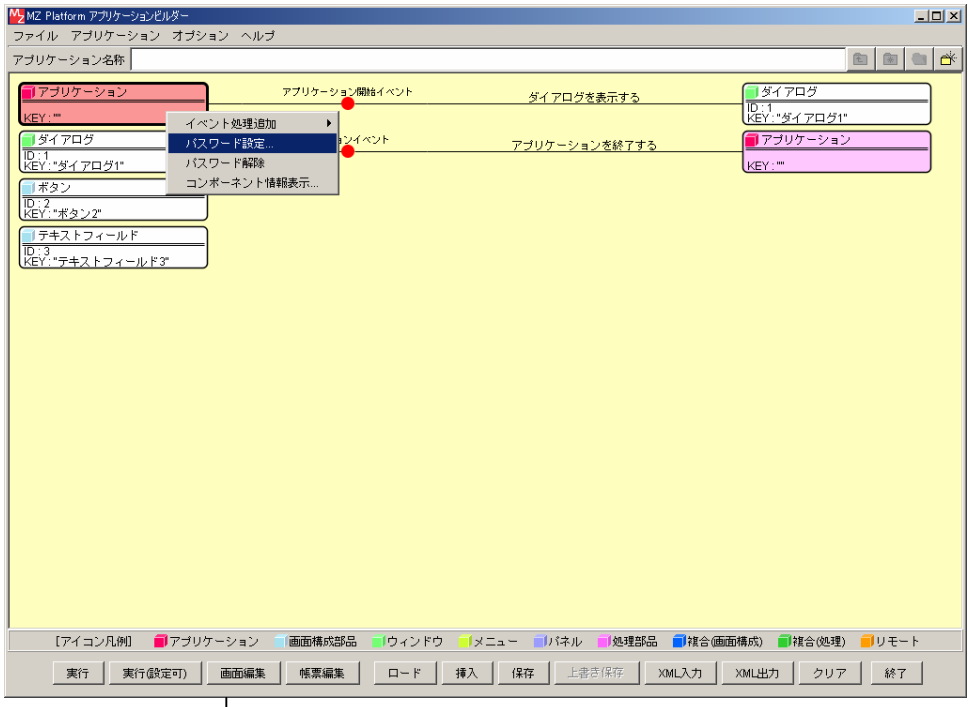
画面	**アプリケーションビルダー メイン画面**
手順	[クリア] ボタンをクリック 現在のアプリケーションがクリアされることが警告される。 必要であれば現状を保存してから、再度[クリア] ボタンをクリックする。 

5.1.13. アプリケーションのパスワードロック機能

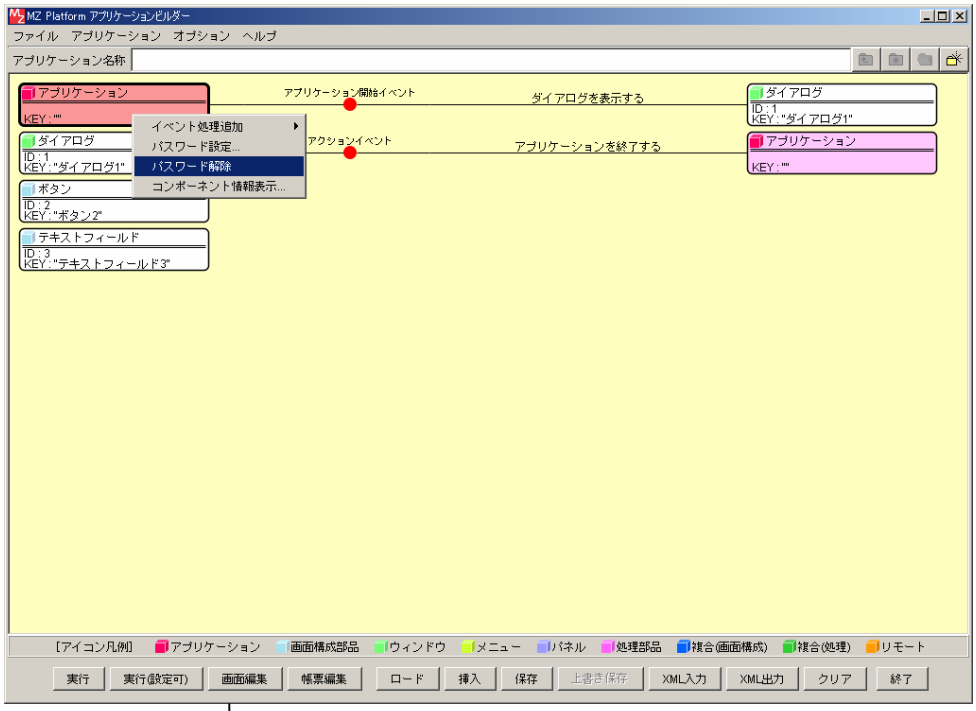
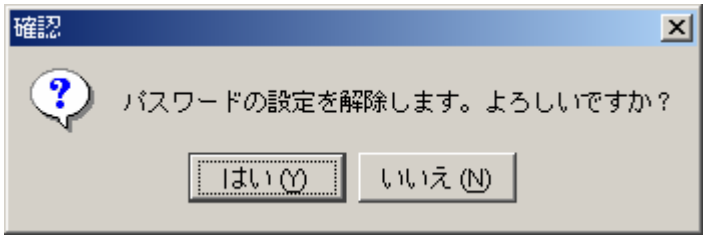
コンポーネントの構成で構築したアプリケーションは、内部構造を見たり、編集したりすることが容易にできます。しかし、アプリケーションを実運用するためには、アプリケーション編集を不可にし、内容を隠したい場合があります。そのため、アプリケーション、および複合コンポーネントにパスワードによってロックをかけることができます。

構築したアプリケーションの内容を公開したくない場合、パスワードロック機能を使用してください。パスワードロックされたアプリケーション、および複合コンポーネントは、実行は通常どおり可能ですが、ビルダー上のロードや複合コンポーネントへの階層移動などについてはパスワードの入力が必要となり、外部への情報漏洩を防ぐことができます。

1)アプリケーション／複合コンポーネントへのパスワード設定

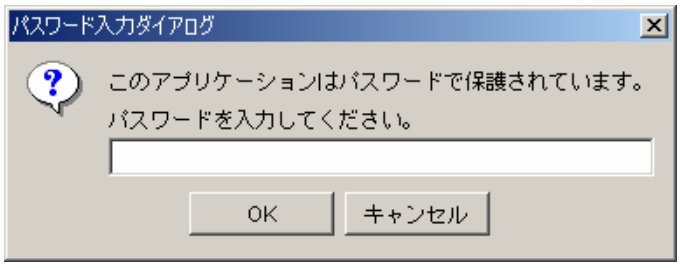
画面	アプリケーションビルダー メイン画面
手順	①最上位コンポーネント上にてマウスを右クリックし、[パスワード設定...]を指定
	
	パスワード入力画面が表示される
	
	②任意のパスワードを入力し、設定ボタンを押下

2)アプリケーション／複合コンポーネントのパスワード解除

画面	アプリケーションビルダー メイン画面
手順	①最上位コンポーネント上にてマウスを右クリックし、[パスワード解除を指定  パスワード解除確認画面が表示される 
	②確認の上で、[はい]ボタンを押下

3)パスワードロックされたアプリケーション／複合コンポーネントの利用

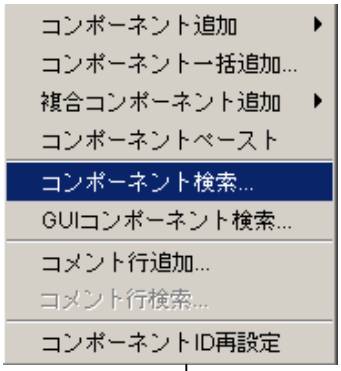
パスワードロックされているアプリケーションや複合コンポーネントのロード時／挿入時、ろっくされた複合コンポーネントへの階層移動などの時に、以下のパスワード入力画面が表示され、正しいパスワード入力時にのみ利用が可能となります。

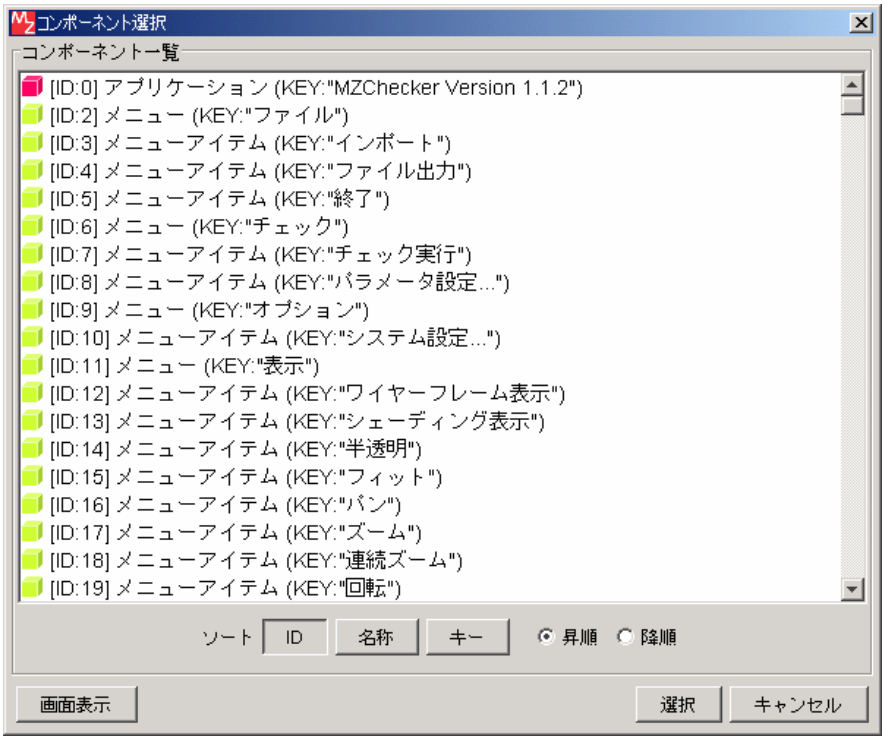


5.1.14. アプリケーション構築時のユーティリティ機能

アプリケーションを構築する際、コンポーネントの数が増えると編集作業が大変になります。そこで、構築作業を効率化するために、いくつかのユーティリティ機能を提供しています。

1)コンポーネントの検索

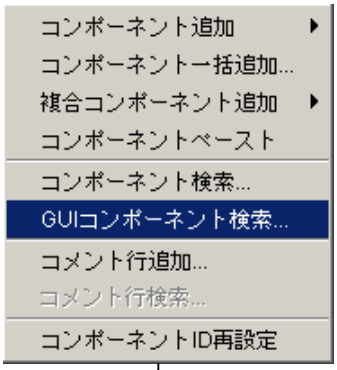
<table>
<tr><td>画面</td><td>アプリケーションビルダー メイン画面</td></tr>
<tr><td>手順</td><td>①背景にてマウスを右クリックし、[コンポーネント検索...]を指定
 

コンポーネント選択画面が表示される
 

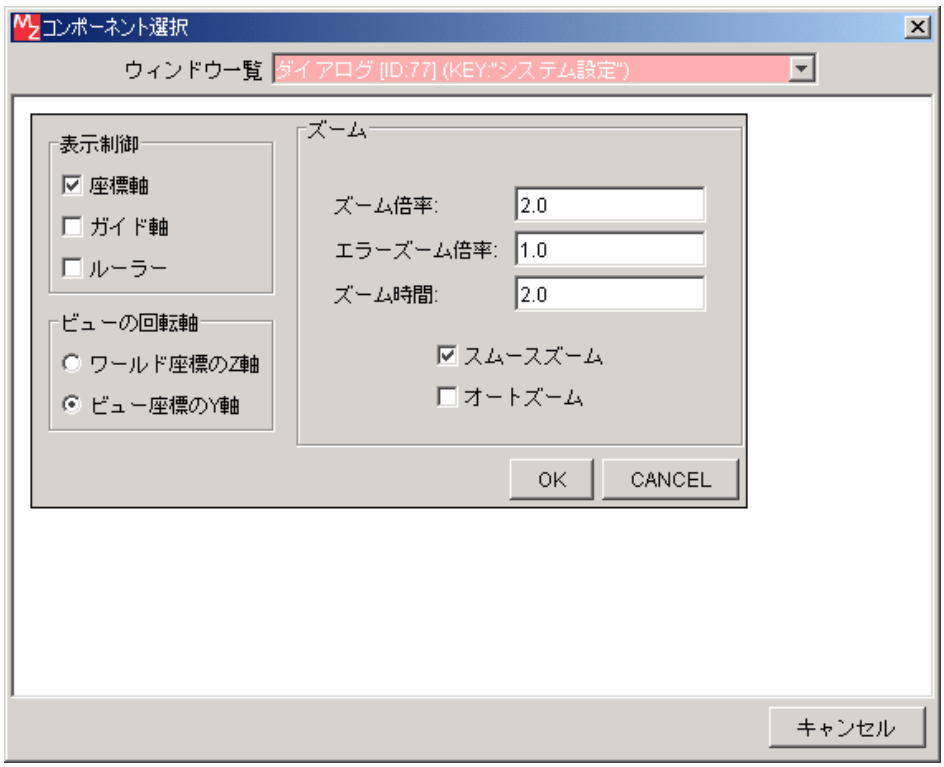
※操作方法
・キーボードから ID を入力すると該当コンポーネントが選択状態になる
・ID／名称／キーについて、それぞれ昇順／降順のソート表示が可能
・[画面表示]ボタンについては GUI コンポーネントの検索を参照</td></tr>
<tr><td></td><td>②検索対象のコンポーネントを選択し、[選択]ボタンを押下すると、
メイン画面の該当コンポーネント表示が選択状態になる
※対象コンポーネントをダブルクリックしても同様</td></tr>
</table>

2)GUI コンポーネントの検索

画面	アプリケーションビルダー メイン画面
手順	①背景にてマウスを右クリックし、[GUI コンポーネント検索...]を指定
	②選択画面上部の“ウィンドウ一覧”のメニューから、 検索対象のコンポーネントが配置されているウィンドウを選択する
	③表示されているプレビュー画面上で、検索対象のコンポーネント表示を ダブルクリックすると、メイン画面の該当コンポーネント表示が選択状態になる



GUI コンポーネント選択画面が表示される

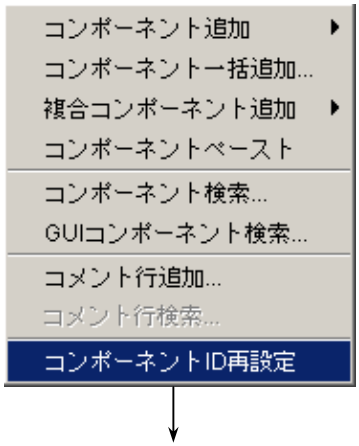


※操作方法

・プルダウンリストから対象の GUI コンポーネント選択することより、検索

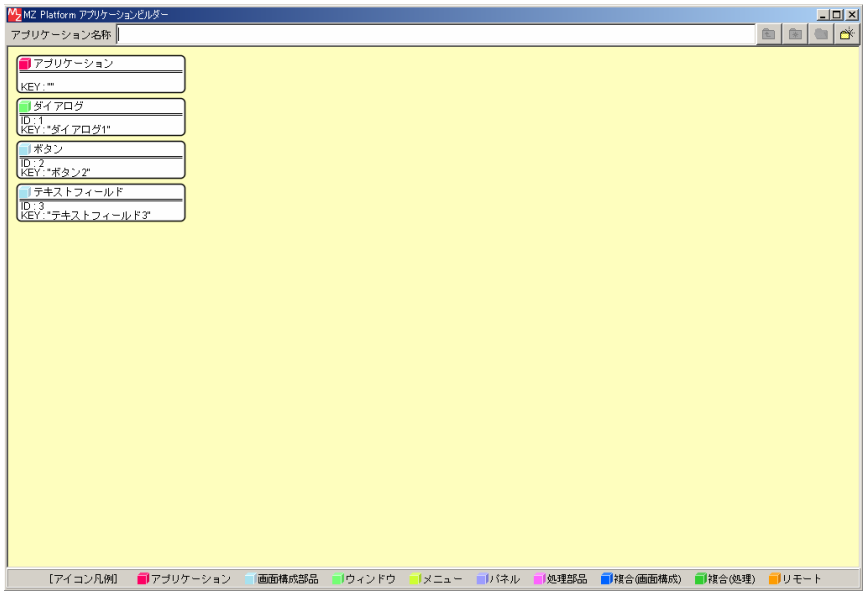
3)コンポーネント ID 再設定

コンポーネントの移動や削除で、コンポーネント ID がバラバラになってしまった場合、表示の並び順に ID の振りなおしを行う機能です。

画面	アプリケーションビルダー メイン画面
手順	背景にてマウスを右クリックし、[コンポーネント ID 順に整列]を指定  コンポーネントの表示順に ID が振りなおされる

4)マルチウインドウ

アプリケーションの編集中に別の箇所や異なる階層を参照したり、複数箇所を見比べながら編集できるように、ビルダー画面を複数開くことのできる機能です。

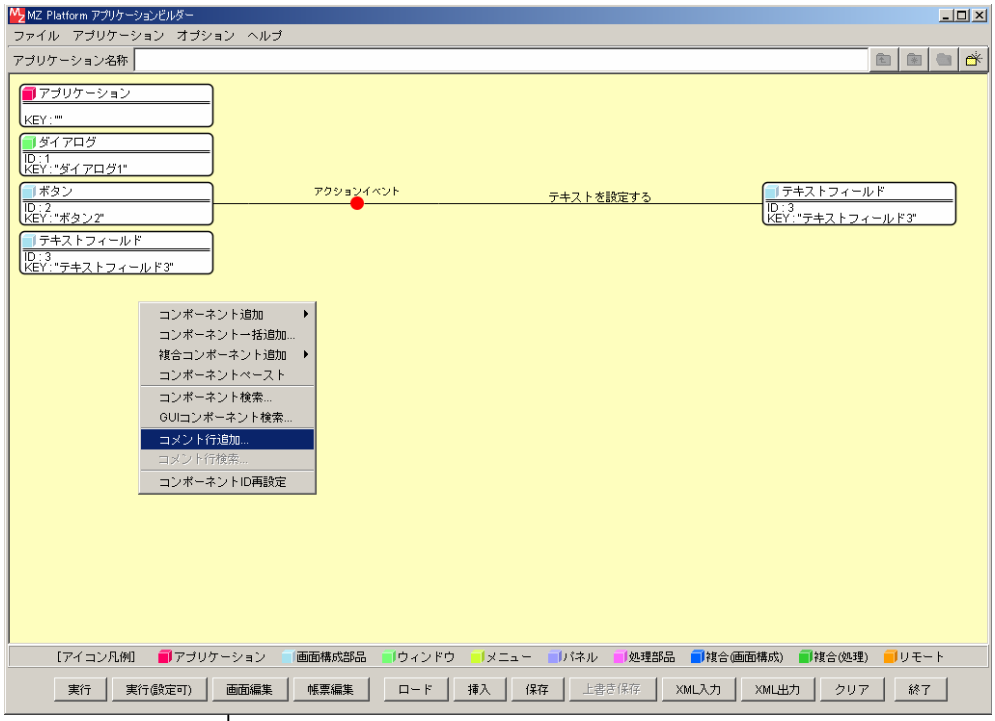
画面	アプリケーションビルダー メイン画面
手順	画面右上のボタンによって新規ウインドウを開く。  [新規ウインドウを開く］現在のビルダー画面と同一の内容が表示されたウインドウが、新規で開く。  ※注意事項 新しく開いたウインドウには実行・保存などの下部ボタン群はついていません

5.1.15. コメント機能

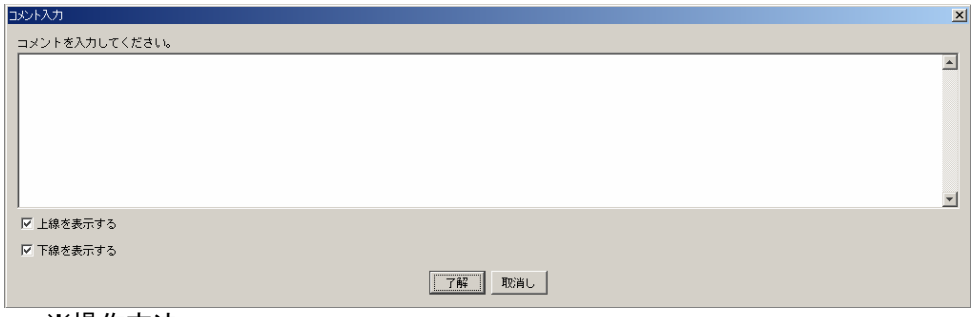
アプリケーションが大規模になってくると、それぞれの部分が何の処理を行っているのかがわかりにくくなります。そこで、アプリケーションビルダー上にコメントを記入する機能を提供しています。

1)全体コメントの記入

画面	アプリケーションビルダー メイン画面
手順	①背景にてマウスを右クリックし、[コメント行追加...]を指定



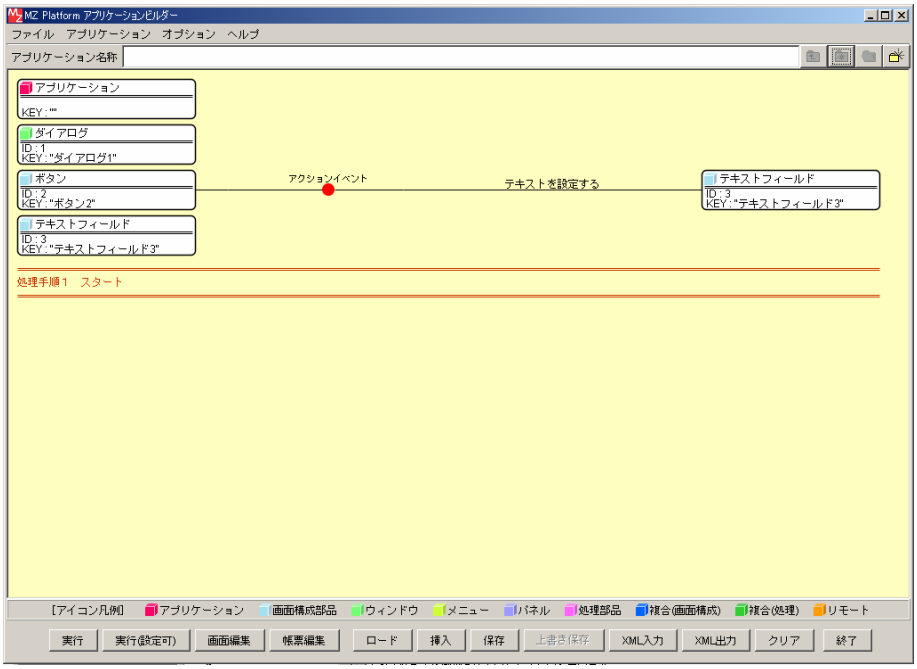
コメント入力画面が表示される



※操作方法

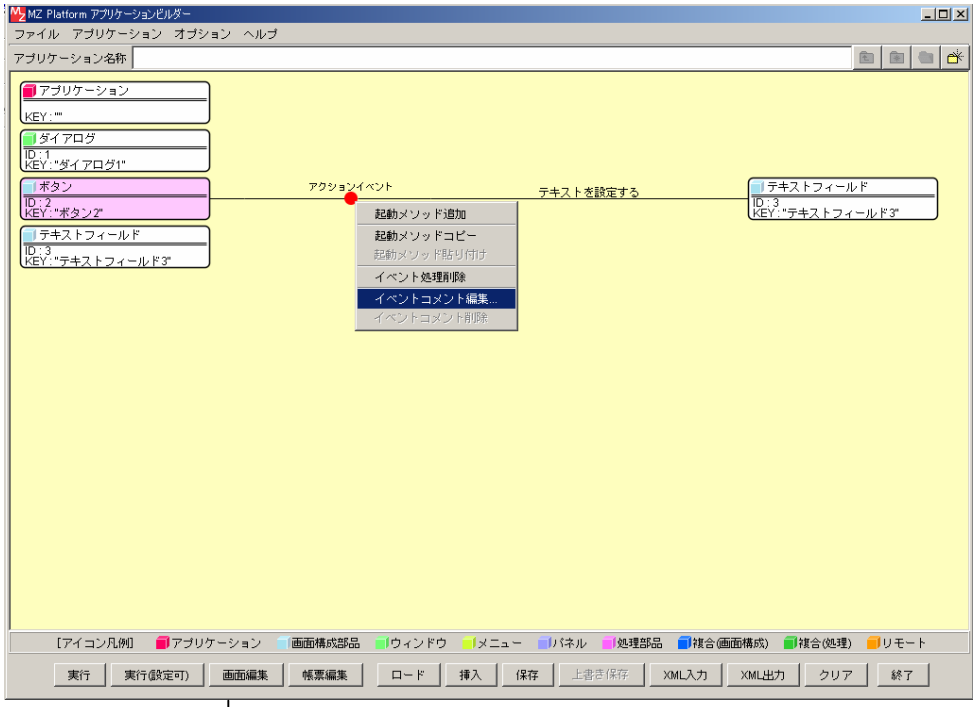
・キーボードよりコメントの内容を入力

・上線および下線の有無をチェック

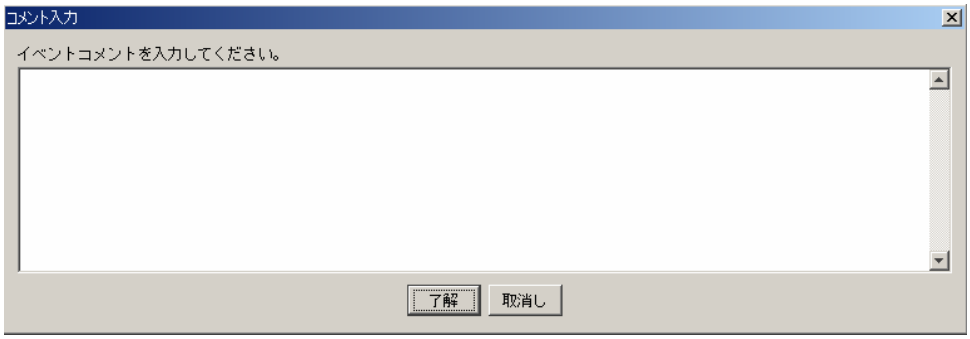
手順	②了解ボタンをクリック ↓ コメントが追加される 

2)イベントコメントの記入

画面	アプリケーションビルダー メイン画面
手順	①設定対象のイベント上でマウスを右クリックし、[イベントコメント編集...]を指定

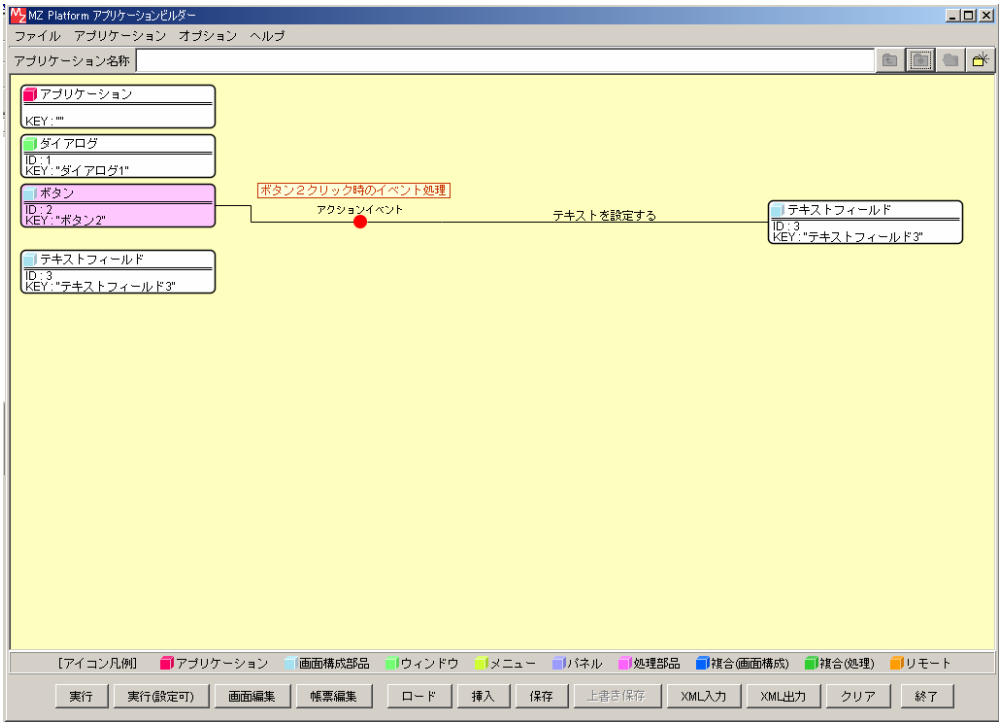


コメント入力画面が表示される



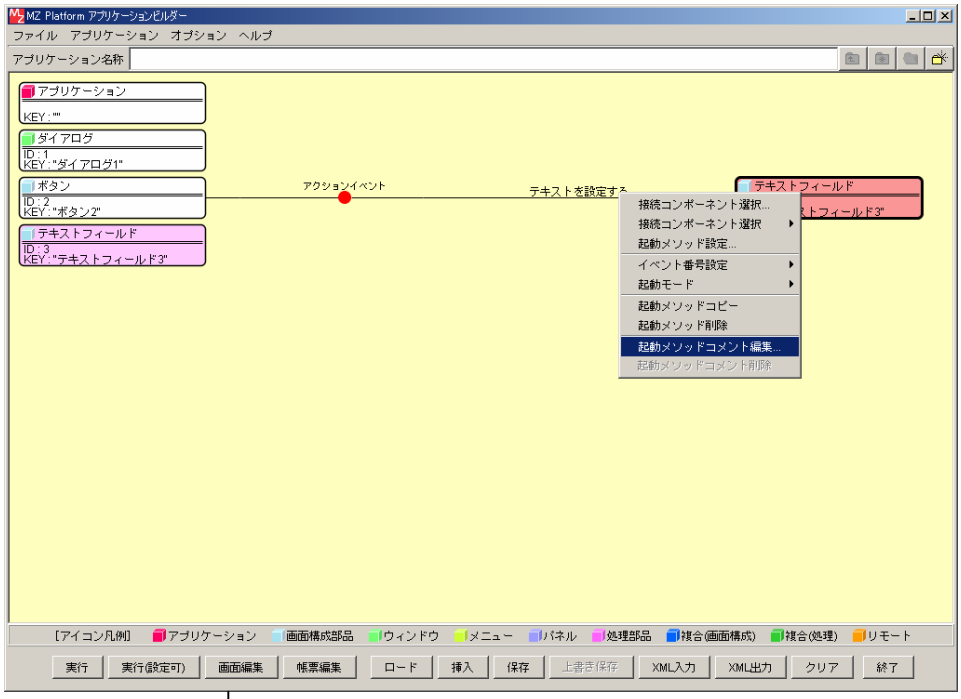
※操作方法

・キーボードよりコメントの内容を入力

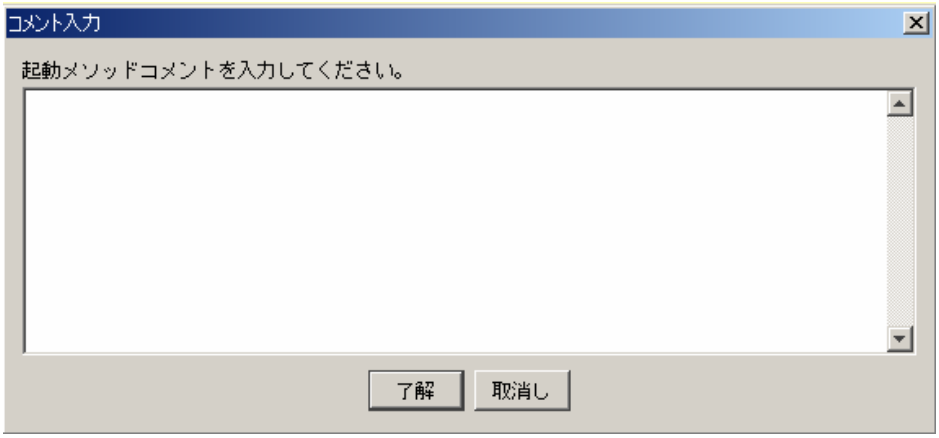
手順	②了解ボタンをクリック ↓ コメントが追加される 

3)起動メソッドコメントの記入

画面	アプリケーションビルダー メイン画面
手順	①設定対象の起動メソッド上でマウスを右クリックし、[起動メソッドコメント編集...]を指定

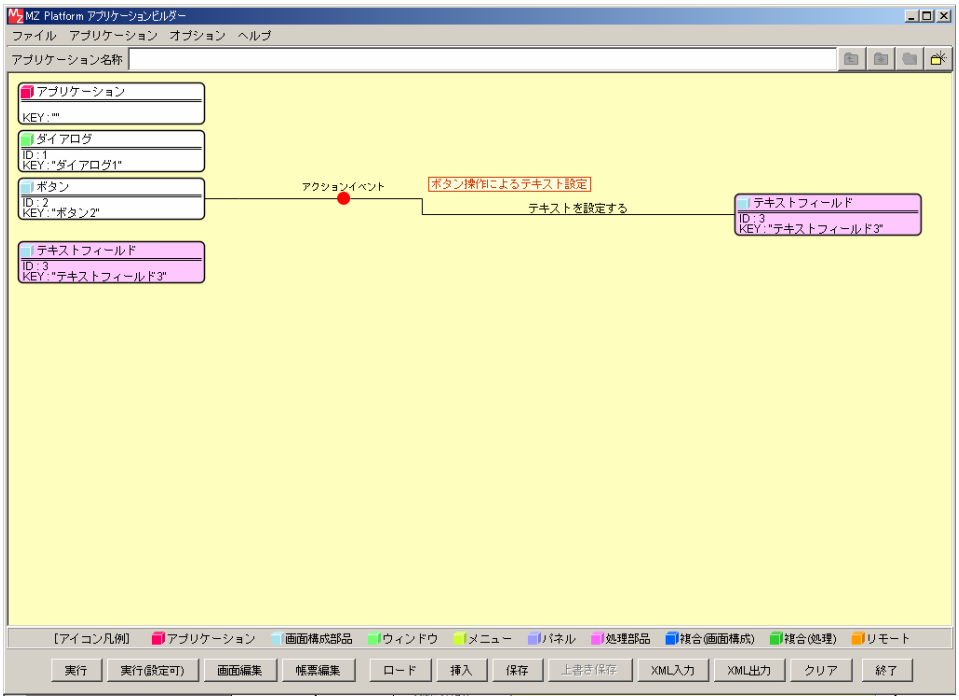


コメント入力画面が表示される



※操作方法

・キーボードよりコメントの内容を入力

手順	②了解ボタンをクリック ↓ コメントが追加される 

5.1.16. その他機能

1)メニューバー

アプリケーションビルダー画面にはメニューバーが提供されており、以下の機能を呼び出すことができます。

メニュー	アイテム	説明
ファイル	新規作成	以下のプログラムの新規作成を行う ①アプリケーション ②複合コンポーネント ③GUI 複合コンポーネント
	ロード	アプリケーションのロードを行う （[ロード]ボタンと同等）
	挿入	アプリケーションの挿入を行う（[挿入]ボタンと同等）
	保存	アプリケーションの保存を行う（[保存]ボタンと同等）
	上書き保存	アプリケーションの上書き保存を行う （[上書き保存]ボタンと同等）
	XML 入力	XML ファイルからアプリケーションのロードを行う
	XML 出力	XML ファイルにアプリケーションの保存を行う
	クリア	アプリケーションのクリアを行う （[クリア]ボタンと同等）
	終了	アプリケーションビルダーを終了する （[終了]ボタンと同等）
アプリケーション	実行	アプリケーションを実行する（[実行]ボタンと同等）
	実行(設定可)	設定可能モードでアプリケーションを実行する （[実行(設定可)]ボタンと同等）
	画面編集	画面レイアウトを編集する（[画面編集]ボタンと同等）
	帳票編集	帳票レイアウトを編集する（[帳票編集]ボタンと同等）
	デバッグ	デバッグ機能を起動する
オプション	コンポーネント情報編集	コンポーネント情報を編集する
	Look&Feel	Look&Feel を以下から設定する ①Windows ②Motif ③Java（Metal）
	ロケール	ロケールを以下から設定する ①日本語 ②英語 ③フランス語（デフォルト設定確認用）
	ログ出力レベル	ログ出力レベルを以下から設定する ①出力なし ②エラーログのみ出力 ③重要ログのみ出力 ④全て出力
	ツールチップ表示	コンポーネント説明および起動メソッド説明等のツールチップ表示有無を設定する
	メモリ使用量表示	以下のメモリサイズを別ダイアログで表示する ①確保しているメモリサイズ ②使用しているメモリサイズ
	メモリ整理	ガベージコレクタ実行
ヘルプ	プラットフォーム ライセンス情報	プラットフォームのライセンス情報を表示する
	アプリケーション ライセンス情報	登録されているアプリケーションライセンス情報の表示、および更新を行う

2)ダブルクリックによるショートカット

アプリケーション構築画面には、マウスの左ダブルクリックによるショートカット操作があります。

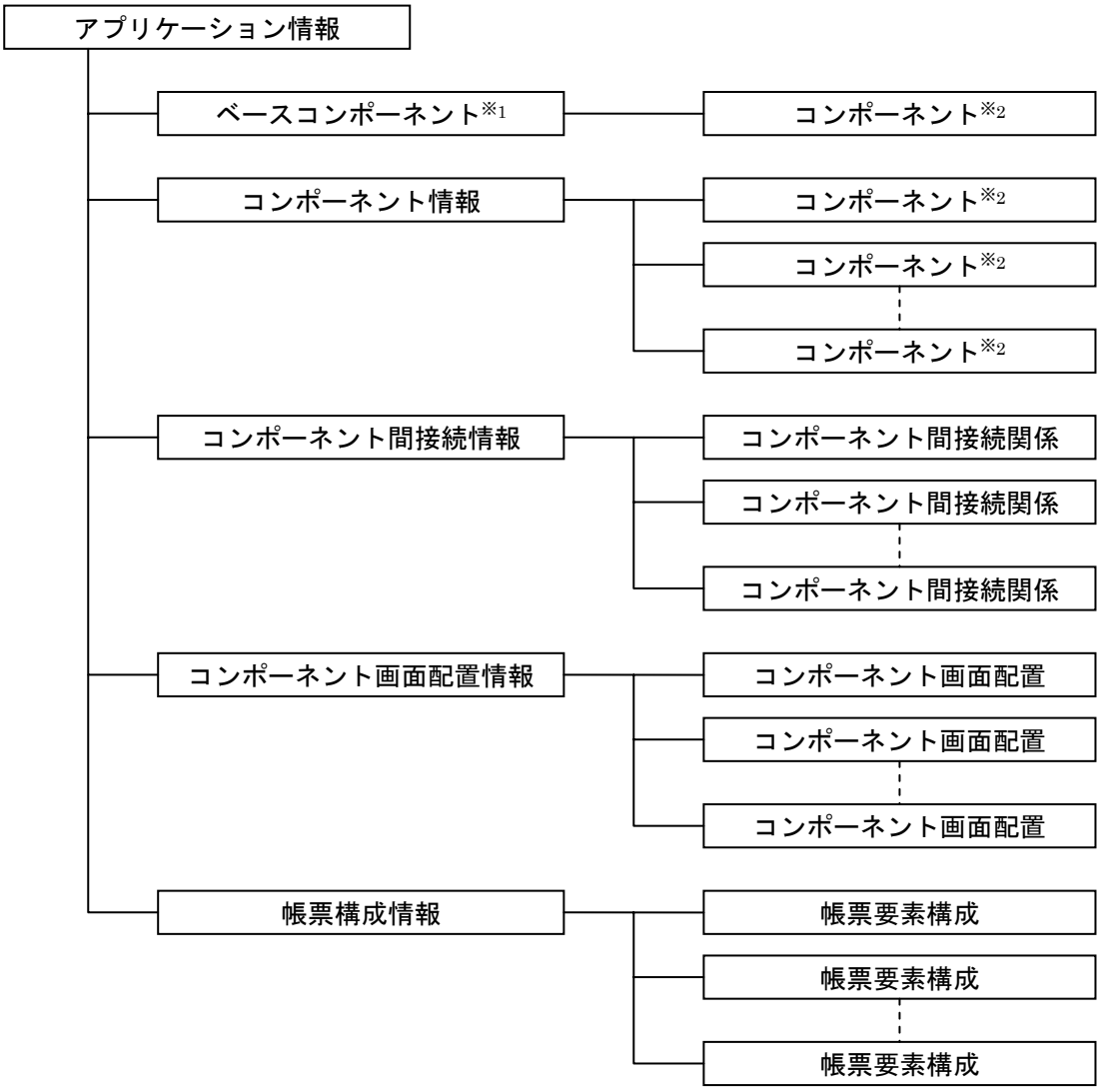
マウス位置	状態	動作
背景	ー	コンポーネント一括追加
コンポーネント（左側矩形）	複合コンポーネント	コンポーネント編集（下位階層編集）
	上記以外	コンポーネント属性編集
イベント（赤円）	ー	起動メソッド追加
起動メソッド名（文字列）	ー	起動メソッド情報編集
接続先コンポーネント（右側矩形）	接続先未設定時	接続先コンポーネント選択
	接続先既設定時	起動メソッド情報編集

5.2. XML によるアプリケーション構築

本プラットフォームが提供する XML 入出力機能を使用し、アプリケーションの構造や動作を XML テキストファイルに記述することでアプリケーションを構築できます。これにより、使い慣れたテキストエディタや、XML エディタなどでアプリケーションの構築が可能です。

テキスト情報であるため、オブジェクトデータの属性情報は指定できませんが、数値や文字列などの属性や、コンポーネント間の接続関係、画面配置情報が設定できます。複合コンポーネントについても対応します。ただし、帳票レイアウトに関する情報は出力対象となっておりません。

5.2.1. XML 形式ドキュメント構造



※1：ベースコンポーネント

ベースコンポーネントは構築対象をしめす特別なコンポーネントで、アプリケーションビルダー画面の最上段に表示されるコンポーネントのこと。具体的には“アプリケーション”か“複合コンポーネント”のどちらか。

※2：コンポーネント

コンポーネントは通常のコンポーネントと複合コンポーネントの両方を示す。

複合コンポーネント情報にはその内部構造や外部インターフェイス情報も含まれる。

5.2.2. XML タグ

アプリケーション情報		
タグ	説明	属性
<information>	情報開始	ー
ー	ベースコンポーネント情報	ー
ー	コンポーネント情報	ー
ー	コンポーネント間接続情報	ー
ー	コンポーネント画面配置情報	ー
</information>	情報終了	ー

ベースコンポーネント情報		
タグ	説明	属性
<base>	ベースコンポーネント情報開始	ー
ー	コンポーネントバス or 複合コンポーネント	ー
</base>	ベースコンポーネント情報終了	ー

コンポーネント情報		
タグ	説明	属性
<components>	コンポーネント情報開始	ー
ー	（コンポーネント or 複合コンポーネント）×N	ー
</components>	コンポーネント情報終了	ー

コンポーネント		
タグ	説明	属性
<component>	コンポーネント開始	class：クラス名
ー	コンポーネント属性情報	ー
</component>	コンポーネント終了	ー

コンポーネント属性情報		
タグ	説明	属性
<properties>	コンポーネント属性情報開始	ー
ー	コンポーネント属性×N	ー
</properties>	コンポーネント属性情報終了	ー

拡張コンポーネント属性情報		
タグ	説明	属性
<additionalproperties>	拡張コンポーネント属性情報開始	ー
ー	コンポーネント属性×N	ー
</additionalproperties>	拡張コンポーネント属性情報終了	ー

コンポーネント属性		
タグ	説明	属性
<property>	コンポーネント属性開始	name：属性名
<type>	属性データ型開始	ー
テキスト要素	属性データ型（データ型名文字列）	ー
</type>	属性データ型終了	ー
<value>	属性値開始	ー
テキスト要素	属性値（データ値文字列表現）	ー
</value>	属性値終了	ー
</property>	コンポーネント属性終了	ー

コンポーネント間接続情報		
タグ	説明	属性
<invocations>	コンポーネント間接続情報開始	ー
ー	コンポーネント間接続×N	ー
</invocations>	コンポーネント間接続情報終了	ー

コンポーネント間接続		
タグ	説明	属性
<invocation>	コンポーネント間接続開始	ー
ー	接続元コンポーネント情報	ー
ー	接続先情報	ー
</invocation>	コンポーネント間接続終了	ー

接続元コンポーネント情報		
タグ	説明	属性
<source>	接続元コンポーネント情報開始	component：コンポーネント ID ※ベースの場合は"base"
<event>	発生イベント開始	ー
テキスト要素	発生イベント（以下のいずれかを指定） PFApplicationStart PFApplicationTerminate PFProcessRequest PFProcessTerminate PFAction PFMouseButton PFMouseMotion PFMouseWheel PFKey PFScroll PFScroll2D PFViewPick PFViewLocate PFViewUpdate PFDataDrop PFDataCreate PFDataSet PFDataUpdate PFDataSelect PFComponentCooperationResult PFPullComponentTransferResult PFPushComponentTransferResult PFPushComponentTransferReceive	ー
</event>	発生イベント終了	ー
</source>	接続元コンポーネント情報終了	ー

接続先情報		
タグ	説明	属性
<targets>	接続先情報開始	ー
ー	接続先×N	ー
</targets>	接続先情報終了	ー

接続先		
タグ	説明	属性
<target>	接続先情報開始	component：コンポーネント ID ※ベースの場合は"base" method：接続メソッド名 event-no：イベント番号 mode：起動モード finally:Finally 起動 error:ErrorOnly 起動
ー	メソッド引数×N	ー
</target>	接続先情報終了	ー

メソッド引数		
タグ	説明	属性
<argument>	メソッド引数開始	kind：引数取得方法 METHOD_RETURN:メソッド値 STATIC_VALUE:固定値 COMPONENT:コンポーネント EVENT_VALUE:イベント内包 EVENT_OBJECT:イベント METHOD_RESULT:メソッド結果
<type>	引数データ型開始　※必須項目	ー
テキスト要素	引数データ型（データ型名文字列）	ー
</type>	引数データ型終了	ー
<value>	引数値開始　※kind=STATIC_VALUE の場合	ー
テキスト要素	引数値（引数値文字列表現）	ー
</value>	引数値終了	ー
<component>	引数コンポーネント開始 ※kind=METHOD_RETURN/COMPONENT の場合	ー
テキスト要素	引数コンポーネント ID	ー
</component>	引数コンポーネント開始終了	ー
<method>	引数取得メソッド開始 ※kind=METHOD_RETURN/EVENT_VALUE の場合	ー
テキスト要素	引数取得メソッド名	ー
</method>	引数取得メソッド開始終了	ー
</argument>	メソッド引数終了	ー

コンポーネント画面配置情報		
タグ	説明	属性
<displays>	コンポーネント画面配置情報開始	ー
ー	コンポーネント画面配置×N	ー
</displays>	コンポーネント画面配置情報終了	ー

コンポーネント画面配置		
タグ	説明	属性
<display>	コンポーネント画面配置開始	container： コンテナコンポーネント ID ※ベースの場合は"base"
ー	配置コンポーネント×N	ー
</display>	コンポーネント画面配置終了	ー

配置コンポーネント		
タグ	説明	属性
<component>	配置コンポーネント開始	position：配置位置 ※領域配置の場合のみ North South East West Center
テキスト要素	コンポーネント ID	ー
</component>	配置コンポーネント終了	ー

帳票構成情報		
タグ	説明	属性
<papers>	帳票構成情報開始	ー
ー	帳票要素構成×N	ー
</papers>	帳票構成情報終了	ー

帳票要素構成		
タグ	説明	属性
<paper>	帳票要素構成開始	id：帳票コンポーネント ID
ー	帳票要素×N	ー
</paper>	帳票要素構成終了	ー

ラベル帳票要素		
タグ	説明	属性
<label>	ラベル帳票要素開始	x, y：配置座標 width, height：サイズ textfont：フォント textcolor：文字色 underline：下線有無 verticalmargin：縦余白 horizontalmargin：横余白 linemargin：行間隔 textposition：文字位置 backgroundcolor：背景色 bordercolor：罫線色 borderwidth：罫線太さ
ー	データ設定（文字列固定 or メソッド取得形式）	ー
</label>	ラベル帳票要素終了	ー

テーブル帳票要素		
タグ	説明	属性
<table>	テーブル帳票要素開始	x, y：配置座標 width, height：サイズ textfont：フォント textcolor：文字色 underline：下線有無 verticalmargin：縦余白 horizontalmargin：横余白 linemargin：行間隔 textposition：文字位置 backgroundcolor：背景色 autoresize：テーブル高さ自動調整 bordercolor：罫線色 linewidth：外枠線太さ
ー	テーブルヘッダ行設定	ー
ー	テーブル列設定情報	ー
ー	テーブル行設定情報	ー
ー	データ設定（メソッド取得形式）	ー
</table>	テーブル帳票要素終了	ー

テーブルヘッダ行設定		
タグ	説明	属性
<header>	テーブルヘッダ行設定開始	visible：表示有無 linewidth：ヘッダ線太さ usetableattribute：属性継承 textfont：フォント textcolor：文字色 underline：下線有無 verticalmargin：縦余白 horizontalmargin：横余白 linemargin：行間隔 textposition：文字位置 backgroundcolor：背景色

テーブル列設定情報		
タグ	説明	属性
<columns>	テーブル列設定情報開始	linewidth：縦線太さ
ー	テーブル列設定×N	ー
</columns>	テーブル列設定情報終了	ー

テーブル列設定		
タグ	説明	属性
<column>	テーブル列設定開始	index：列インデックス width：列幅 usetableattribute：属性継承 textfont：フォント textcolor：文字色 underline：下線有無 verticalmargin：縦余白 horizontalmargin：横余白 linemargin：行間隔 textposition：文字位置 backgroundcolor：背景色
ー	表示パターン設定情報	ー
</column>	テーブル列設定終了	ー

表示パターン設定情報		
タグ	説明	属性
<pattern>	表示パターン設定情報開始	ー
ー	表示パターン設定（数値 or 日付 or 論理）	ー
</pattern>	表示パターン設定情報終了	ー

数値型表示パターン設定		
タグ	説明	属性
<number>	数値型表示パターン設定開始	ー
テキスト要素	フォーマット文字列	ー
</number>	数値型表示パターン設定終了	ー

日付型表示パターン設定		
タグ	説明	属性
<date>	日付型表示パターン設定開始	ー
テキスト要素	フォーマット文字列	ー
</date>	日付型表示パターン設定終了	ー

論理型表示パターン設定		
タグ	説明	属性
<boolean>	論理型表示パターン設定開始	ー
ー	真値表示パターン設定	ー
ー	偽値表示パターン設定	ー
</boolean>	論理型表示パターン設定終了	ー

真値表示パターン設定		
タグ	説明	属性
<t>	真値表示パターン設定開始	ー
テキスト要素	表示文字列	ー
</t>	真値表示パターン設定終了	ー

偽値表示パターン設定		
タグ	説明	属性
<f>	偽値表示パターン設定開始	ー
テキスト要素	表示文字列	ー
</f>	偽値表示パターン設定終了	ー

テーブル行設定情報		
タグ	説明	属性
<rows>	テーブル行設定情報開始	linewidth：横線太さ
ー	テーブル行設定×N	ー
</rows>	テーブル行設定情報終了	ー

テーブル行設定		
タグ	説明	属性
<row>	テーブル行設定	index：行インデックス height：行高さ

バーコード帳票要素		
タグ	説明	属性
<barcode>	バーコード帳票要素開始	x, y：配置座標 width, height：サイズ codetype：コード体系 displaystringflag：データ表示有無 addcheckdigitflag： チェックディジット付加 originalsize：原寸表示 bordercolor：罫線色 borderwidht：罫線太さ
ー	データ設定（文字列固定 or メソッド取得形式）	ー
</barcode>	バーコード帳票要素終了	ー

イメージ帳票要素		
タグ	説明	属性
<image>	イメージ帳票要素開始	x, y：配置座標 width, height：サイズ originalsize：原寸表示 bordercolor：罫線色 borderwidth：罫線太さ
ー	データ設定（イメージ情報 or メソッド取得形式）	ー
</image>	イメージ帳票要素終了	ー

画面イメージ帳票要素		
タグ	説明	属性
<display>	画面イメージ帳票要素開始	x, y：配置座標 width, height：サイズ originalsize：原寸表示 bordercolor：罫線色 borderwidth：罫線太さ
ー	データ設定（コンポーネント指定形式）	ー
</display>	画面イメージ帳票要素終了	ー

データ設定		
タグ	説明	属性
<data>	データ設定開始	type：データ形式 string:文字列指定 method:メソッド取得 image:イメージデータ component:コンポーネント指定
テキスト要素	出力文字列（文字列固定の場合のみ）	ー
テキスト要素	イメージのバイト列（イメージ情報の場合のみ）	ー
ー	データ設定（メソッド取得 or コンポーネント）	ー
</data>	データ設定終了	ー

メソッド取得設定		
タグ	説明	属性
<component>	取得元コンポーネント情報	id：コンポーネント ID
<method>	取得元メソッド情報	name：メソッド名
<combinative>	下位階層取得元コンポーネント情報 （取得元コンポーネントが下位階層の場合）	id：コンポーネント ID ※階層情報含む（例："1-2-2"）

コンポーネント指定設定		
タグ	説明	属性
<component>	取得元コンポーネント情報	id：コンポーネント ID
<combinative>	下位階層取得元コンポーネント情報 （取得元コンポーネントが下位階層の場合）	id：コンポーネント ID ※階層情報含む（例："1-2-2"）

複合コンポーネント		
タグ	説明	属性
<component>	複合コンポーネント開始	reference：外部参照ファイル名
ー	コンポーネント属性情報	ー
ー	コンポーネント情報	ー
ー	コンポーネント間接続情報	ー
ー	コンポーネント画面配置情報	ー
ー	外部公開メソッド情報	ー
</component>	複合コンポーネント終了	ー

外部公開メソッド情報		
タグ	説明	属性
<methods>	外部公開メソッド情報開始	ー
ー	外部公開メソッド×N	ー
</methods>	外部公開メソッド情報終了	ー

外部公開メソッド		
タグ	説明	属性
<method>	外部公開メソッド	name：メソッド名 component:元コンポーネント ID alias：公開メソッド名（別名）

5.2.3. データ表現形式

XML 形式テキストベースでの表現には制限がありますので、あらゆる情報を出力できるわけではありません。例えば数値や文字列のようなテキスト表現可能なものは XML 出力可能ですが、独自オブジェクトなどについては出力できません。

1)出力対象

①コンポーネント属性情報

属性情報として XML 形式に表現できるデータは以下に限定されます。

- プリミティブ型（boolean, byte, char, short, int, short, long, float, double）
- 文字列型（java.lang.String）
- サイズ情報（java.awt.Dimension）
- 点情報（java.awt.Point）
- 枠情報（javax.swing.border.Border）
  ※EmptyBorder, EtchedBorder, LineBorder, BevelBorder のみ
- アイコン情報（javax.swing.ImageIcon） ※イメージ情報のみ
- イメージ情報（java.awt.Image）
- 色情報（java.awt.Color）
- フォント情報（java.awt.Font）
- マルチロケール文字列（jp.go.aist.dmrc.platform.util.PFMultiLocaleString）
- オブジェクトリスト（jp.go.aist.dmrc.platform.util.PFObjectList）
- オブジェクトテーブル（jp.go.aist.dmrc.platform.util.PFObjectTable）
- オブジェクトツリー（jp.go.aist.dmrc.platform.util.PFObjectTree）
- シリアライズ可能なオブジェクト（java.io.Serializable）

なお、オブジェクト型属性（上記データ型も含める）に null 値を設定する場合には、“<value>”要素を記述しないようにします。また、階層をもったオブジェクト（リスト／テーブル／ツリー）については、下位階層までたどって出力します。

②メソッド引数情報

メッド引数の固定値として XML 形式に表現できるデータは以下に限定されます。

- プリミティブ型（boolean, byte, char, short, int, long, float, double）
- プリミティブラッパークラス型（Boolean, Byte, Char, Short, Integer, Long, Float, Double）
- 数値クラス型（java.math.BigInteger, java.math.BigDecimal）
- 文字列型（java.lang.String）
- クラス型（java.lang.Class）

なお、オブジェクト型属性（上記データ型も含める）に null 値を設定する場合には、“<value>”要素を記述しないようにします。

2)テキスト表現形式

データ型	表現形式	表現例
プリミティブ型 （ラッパークラス含む）	toString()の表現形式	true　false　100　1.5　A
サイズ（Dimension）	width,height	100,100
点（Point）	x,y	100,100
枠（Border）	種別番号（右参照）	0：空枠（EmptyBorder） 1：エッジング（EtchedBorder） 2：ライン（LineBorder） 3：浮き出し斜影（BevelBorder:RAISED） 4：くぼみ斜影（BevelBorder:LOWERED）
アイコン（ImageIcon）	アイコンイメージ情報 width,height,[imege-data]	10,10,[00000000……00000000]
イメージ情報（Image）	width,height,[imege-data]	10,10,[00000000……00000000]
色情報（Color）	red,green,blue,alpha	255,255,255,255
フォント（Font）	name,style,size ※style 0:PLAIN 1:BOLD 2:ITALIC 3:BOLD&ITALIC	Dialog,3,20 Arial,0,30
マルチロケール文字列 （PFMultiLocaleString）	ロケール毎に対応する文字列を記述	<local> <lang>ja</lang> <string>日本語</string> </local> <local> <lang>en</lang> <string>English</string> </local>
オブジェクトリスト （PFObjectList）	リスト要素の文字列表現を順に記述	<element> <type>java.lang.String</type> <value>AAAAA</value> </element> <element> <type>java.lang.String</type> <value>BBBBB</value> </element>

データ型	表現形式	表現例
オブジェクトテーブル （PFObjectTable）	テーブル要素の文字列表現を一行単位に記述	<row> <element> <type>java.lang.String</type> <value>AAAAA</value> </element> <element> <type>java.lang.Integer</type> <value>10000</value> </element> <row> <row> <element> <type>java.lang.String</type> <value>BBBBB</value> </element> <element> <type>java.lang.Integer</type> <value>20000</value> </element> <row>
オブジェクトツリー （PFObjectTree）	ツリー要素の文字列表現をツリー構造で記述	<element> <type>java.lang.String</type> <value>ROOT</value> <child> <element> <type>java.lang.String</type> <value>LEAF-1</value> <child/> </element> <element> <type>java.lang.String</type> <value>LEAF-2</value> <child/> </element> </child> <element>
シリアライズ可能 オブジェクト	シリアライズデータのバイト列	00000000……00000000

5.2.4. XML 形式表現サンプル（参考）

```
<?xml version="1.0" encoding="Shift_JIS"?>
<information>
    <!-- Base Component -->
    <base>
        <!—サンプルアプリケーション -->
        <component class="jp.go.aist.dmrc.platform.base.PFComponentBus">
            <properties>
                <property name="ComponentID">
                    <type>int</type>
                    <value>0</value>
                </property>
                <property name="ExecuteMode">
                    <type>boolean</type>
                    <value>false</value>
                </property>
                <property name="ApplicationName">
                    <type>java.lang.String</type>
                    <value></value>
                </property>
            </properties>
        </component>
    </base>

    <!-- Component Declarations -->
    <components>
        <!-- 月別平均気温 -->
        <component class="jp.go.aist.dmrc.platform.beans.gui.container.PFFrame">
            <properties>
                <property name="ComponentID">
                    <type>int</type>
                    <value>1</value>
                </property>
                <property name="ComponentKey">
                    <type>java.lang.String</type>
                    <value>月別平均気温</value>
                </property>
                <property name="Title">
                    <type>java.lang.String</type>
                    <value>月別平均気温</value>
                </property>
                <property name="MultiLocaleTitle">
                    <type>jp.go.aist.dmrc.platform.util.PFMultiLocaleString</type>
                    <value>
                        <locale>
                            <lang>ja</lang>
                            <string>月別平均気温</string>
                        </locale>
                    </value>
                </property>
                <property name="Size">
                    <type>java.awt.Dimension</type>
                    <value>480,510</value>
                </property>
            </properties>
        </component>
        <!-- テーブル 2 -->
        <component class="jp.go.aist.dmrc.platform.beans.gui.table.PFTable">
            <properties>
                <property name="ComponentID">
                    <type>int</type>
                    <value>2</value>
                </property>
                <property name="ComponentKey">
                    <type>java.lang.String</type>
```

```
        <value>テーブル 2</value>
    </property>
    <property name="ObjectTable">
        <type>jp.go.aist.dmrc.platform.util.PFObjectTable</type>
        <value>
            <row>
                <element>
                    <type>java.lang.String</type>
                    <value>1 月</value>
                </element>
                <element>
                    <type>java.lang.Double</type>
                    <value>-4.3</value>
                </element>
                <element>
                    <type>java.lang.Double</type>
                    <value>5.4</value>
                </element>
                <element>
                    <type>java.lang.Double</type>
                    <value>16.3</value>
                </element>
            </row>
            <row>
                <element>
                    <type>java.lang.String</type>
                    <value>2 月</value>
                </element>
                <element>
                    <type>java.lang.Double</type>
                    <value>-3.7</value>
                </element>
                <element>
                    <type>java.lang.Double</type>
                    <value>5.8</value>
                </element>
                <element>
                    <type>java.lang.Double</type>
                    <value>16.4</value>
                </element>
            </row>
            <row>
                <element>
                    <type>java.lang.String</type>
                    <value>3 月</value>
                </element>
                <element>
                    <type>java.lang.Double</type>
                    <value>0.0</value>
                </element>
                <element>
                    <type>java.lang.Double</type>
                    <value>8.7</value>
                </element>
                <element>
                    <type>java.lang.Double</type>
                    <value>18.3</value>
                </element>
            </row>
            <row>
                <element>
                    <type>java.lang.String</type>
                    <value>4 月</value>
                </element>
                <element>
                    <type>java.lang.Double</type>
```

```
                <value>6.6</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>14.2</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>21.2</value>
            </element>
        </row>
        <row>
            <element>
                <type>java.lang.String</type>
                <value>5 月</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>12.1</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>18.7</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>23.8</value>
            </element>
        </row>
        <row>
            <element>
                <type>java.lang.String</type>
                <value>6 月</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>16.2</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>21.7</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>26.4</value>
            </element>
        </row>
        <row>
            <element>
                <type>java.lang.String</type>
                <value>7 月</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>20.3</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>25.3</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>28.4</value>
            </element>
        </row>
        <row>
```

```
            <element>
                <type>java.lang.String</type>
                <value>8 月</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>21.7</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>27.1</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>28.2</value>
            </element>
        </row>
        <row>
            <element>
                <type>java.lang.String</type>
                <value>9 月</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>17.4</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>23.2</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>27.3</value>
            </element>
        </row>
        <row>
            <element>
                <type>java.lang.String</type>
                <value>10 月</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>11.0</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>17.8</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>24.6</value>
            </element>
        </row>
        <row>
            <element>
                <type>java.lang.String</type>
                <value>11 月</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>4.5</value>
            </element>
            <element>
                <type>java.lang.Double</type>
                <value>12.8</value>
            </element>
```

```
                        <element>
                            <type>java.lang.Double</type>
                            <value>21.6</value>
                        </element>
                    </row>
                    <row>
                        <element>
                            <type>java.lang.String</type>
                            <value>12 月</value>
                        </element>
                        <element>
                            <type>java.lang.Double</type>
                            <value>-1.2</value>
                        </element>
                        <element>
                            <type>java.lang.Double</type>
                            <value>8.1</value>
                        </element>
                        <element>
                            <type>java.lang.Double</type>
                            <value>18.3</value>
                        </element>
                    </row>
                </value>
            </property>
            <property name="ColumnWidthList">
                <type>jp.go.aist.dmrc.platform.util.PFObjectList</type>
                <value>
                    <element>
                        <type>java.lang.Integer</type>
                        <value>65</value>
                    </element>
                    <element>
                        <type>java.lang.Integer</type>
                        <value>139</value>
                    </element>
                    <element>
                        <type>java.lang.Integer</type>
                        <value>148</value>
                    </element>
                    <element>
                        <type>java.lang.Integer</type>
                        <value>142</value>
                    </element>
                </value>
            </property>
            <property name="Font">
                <type>java.awt.Font</type>
                <value>Dialog,0,12</value>
            </property>
            <property name="Size">
                <type>java.awt.Dimension</type>
                <value>470,220</value>
            </property>
        </properties>
        <additionalproperties>
            <property name="ColumnNames">
                <type>jp.go.aist.dmrc.platform.util.PFObjectList</type>
                <value>
                    <element>
                        <type>java.lang.String</type>
                        <value>月</value>
                    </element>
                    <element>
                        <type>java.lang.String</type>
                        <value>気温(札幌)</value>
```

```
                    </element>
                    <element>
                        <type>java.lang.String</type>
                        <value>気温(東京)</value>
                    </element>
                    <element>
                        <type>java.lang.String</type>
                        <value>気温(那覇)</value>
                    </element>
                </value>
            </property>
            <property name="ColumnTypes">
                <type>jp.go.aist.dmrc.platform.util.PFObjectList</type>
                <value>
                    <element>
                        <type>java.lang.Class</type>
                        <value>class java.lang.String</value>
                    </element>
                    <element>
                        <type>java.lang.Class</type>
                        <value>class java.math.BigDecimal</value>
                    </element>
                    <element>
                        <type>java.lang.Class</type>
                        <value>class java.math.BigDecimal</value>
                    </element>
                    <element>
                        <type>java.lang.Class</type>
                        <value>class java.math.BigDecimal</value>
                    </element>
                </value>
            </property>
        </additionalproperties>
    </component>
    <!-- 気象情報 (Lesson.8) -->
    <component class="jp.go.aist.dmrc.platform.beans.tutorial.PFWeatherInformation">
        <properties>
            <property name="ComponentID">
                <type>int</type>
                <value>3</value>
            </property>
            <property name="ComponentKey">
                <type>java.lang.String</type>
                <value>気象情報 (Lesson.8)</value>
            </property>
        </properties>
    </component>
    <!-- 平均気温 -->
    <component class="jp.go.aist.dmrc.platform.beans.gui.chart.PFLineChart">
        <properties>
            <property name="ComponentID">
                <type>int</type>
                <value>4</value>
            </property>
            <property name="ComponentKey">
                <type>java.lang.String</type>
                <value>平均気温</value>
            </property>
            <property name="Background">
                <type>java.awt.Color</type>
                <value>255,255,255,255</value>
            </property>
            <property name="Font">
                <type>java.awt.Font</type>
                <value>Dialog,0,12</value>
            </property>
```

```
                <property name="HeaderTitle">
                    <type>java.lang.String</type>
                    <value>平均気温</value>
                </property>
                <property name="Size">
                    <type>java.awt.Dimension</type>
                    <value>470,245</value>
                </property>
            </properties>
        </component>
        <!-- 印刷 -->
        <component class="jp.go.aist.dmrc.platform.beans.gui.button.PFButton">
            <properties>
                <property name="ComponentID">
                    <type>int</type>
                    <value>5</value>
                </property>
                <property name="ComponentKey">
                    <type>java.lang.String</type>
                    <value>印刷</value>
                </property>
                <property name="Text">
                    <type>java.lang.String</type>
                    <value>印刷</value>
                </property>
                <property name="PropertyEditable">
                    <type>boolean</type>
                    <value>true</value>
                </property>
            </properties>
        </component>
        <!-- 帳票 -->
        <component class="jp.go.aist.dmrc.platform.beans.system.print.PFPaper">
            <properties>
                <property name="ComponentID">
                    <type>int</type>
                    <value>6</value>
                </property>
                <property name="ComponentKey">
                    <type>java.lang.String</type>
                    <value>帳票</value>
                </property>
                <property name="PaperWidth">
                    <type>float</type>
                    <value>210.0</value>
                </property>
                <property name="PaperHeight">
                    <type>float</type>
                    <value>297.0</value>
                </property>
                <property name="PrintTopMargin">
                    <type>float</type>
                    <value>20.0</value>
                </property>
                <property name="PrintBottomMargin">
                    <type>float</type>
                    <value>20.0</value>
                </property>
                <property name="PrintLeftMargin">
                    <type>float</type>
                    <value>20.0</value>
                </property>
                <property name="PrintRightMargin">
                    <type>float</type>
                    <value>20.0</value>
                </property>
```

```
            </properties>
            <additionalproperties>
                <property name="PaperSize">
                    <type>int</type>
                    <value>14</value>
                </property>
            </additionalproperties>
        </component>
    </components>

    <!-- Invocation Declarations -->
    <invocations>
        <invocation>
            <!--  -->
            <source component="base">
                <event>PFApplicationStart</event>
            </source>
            <targets>
                <!-- テーブル 2 -->
                <target component="2" mode="NORMAL" method="setObjectTable">
                    <argument kind="METHOD_RETURN">
                        <type>jp.go.aist.dmrc.platform.util.PFObjectTable</type>
                        <component>3</component>
                        <method>getTemperatureData</method>
                    </argument>
                </target>
                <!-- 平均気温 -->
                <target component="4" mode="NORMAL" method="setObjectTable">
                    <argument kind="METHOD_RETURN">
                        <type>jp.go.aist.dmrc.platform.util.PFObjectTable</type>
                        <component>3</component>
                        <method>getTemperatureData</method>
                    </argument>
                </target>
                <!-- 月別平均気温 -->
                <target component="1" mode="NORMAL" method="show"/>
            </targets>
        </invocation>
        <invocation>
            <!-- 月別平均気温 -->
            <source component="1">
                <event>PFAction</event>
            </source>
            <targets>
                <!--  -->
                <target component="base" mode="NORMAL" method="terminateApplication"/>
            </targets>
        </invocation>
        <invocation>
            <!-- テーブル 2 -->
            <source component="2">
                <event>PFDataUpdate</event>
            </source>
            <targets>
                <!-- 平均気温 -->
                <target component="4" event-no="0" mode="NORMAL" method="setObjectTable">
                    <argument kind="EVENT_VALUE">
                        <type>jp.go.aist.dmrc.platform.util.PFObjectTable</type>
                        <method>getSourceData</method>
                    </argument>
                </target>
            </targets>
        </invocation>
        <invocation>
            <!-- 平均気温 -->
            <source component="4">
```

```
            <event>PFDataUpdate</event>
        </source>
        <targets>
            <!-- テーブル 2 -->
            <target component="2" mode="NORMAL" method="setObjectTable">
                <argument kind="EVENT_VALUE">
                    <type>jp.go.aist.dmrc.platform.util.PFObjectTable</type>
                    <method>getSourceData</method>
                </argument>
            </target>
            <!-- テーブル 2 -->
            <target component="2" mode="NORMAL" method="setColumnName">
                <argument kind="STATIC_VALUE">
                    <type>java.lang.String</type>
                    <value>月</value>
                </argument>
                <argument kind="STATIC_VALUE">
                    <type>int</type>
                    <value>0</value>
                </argument>
            </target>
        </targets>
    </invocation>
    <invocation>
        <!-- 印刷 -->
        <source component="5">
            <event>PFAction</event>
        </source>
        <targets>
            <!-- 帳票 -->
            <target component="6" mode="NORMAL" method="previewPaper">
                <argument kind="COMPONENT">
                    <type>java.awt.Component</type>
                    <component>1</component>
                </argument>
            </target>
        </targets>
    </invocation>
</invocations>

<!-- Display Declarations -->
<displays>
    <!-- 月別平均気温 -->
    <display container="1">
        <!-- テーブル 2 -->
        <component position="Center">2</component>
        <!-- 平均気温 -->
        <component position="Center">4</component>
        <!-- 印刷 -->
        <component position="Center">5</component>
    </display>
</displays>
```

```
<!-- Paper Declarations -->
<papers>
    <paper id="6">
        <label backgroundcolor="255,255,255,255" textcolor="0,0,0,255"
         width="130.0" height="15.0" underline="false" textposition="22" linemargin="0.0"
         horizontalmargin="5.0" bordercolor="0,0,0,255" bordermargin="0.5"
         x="20.0" y="10.0" textfont="Dialog,0,24" verticalmargin="5.0">
            <data type="string">月別平均気温</data>
        </label>
        <table backgroundcolor="255,255,255,255" textcolor="0,0,0,255" autoresize="true"
         width="150.0" underline="false" textposition="11" linemargin="0.0"
         horizontalmargin="5.0" bordercolor="0,0,0,255" linewidth="2.0"
         x="10.0" y="30.0" textfont="Dialog,0,14" height="120.29724" verticalmargin="5.0">
            <header backgroundcolor="255,255,153,255" textcolor="0,0,0,255" underline="false"
             textposition="22" linemargin="0.0" visible="true" horizontalmargin="5.0" linewidth="2.0"
             usetableattribute="false" textfont="Dialog,0,14" verticalmargin="5.0" />
            <columns linewidth="0.5">
                <column backgroundcolor="255,255,255,255" textcolor="0,0,0,255" width="37.5"
                 underline="false" textposition="22" linemargin="0.0" horizontalmargin="5.0"
                 index="0" usetableattribute="false" textfont="Dialog,0,12" verticalmargin="5.0">
                </column>
                <column backgroundcolor="255,255,255,255" textcolor="0,0,0,255" width="37.5"
                 underline="false" textposition="32" linemargin="0.0" horizontalmargin="5.0"
                 index="1" usetableattribute="false" textfont="Dialog,0,12" verticalmargin="5.0">
                    <pattern>
                        <number>#,##0.### ℃</number>
                        <date>yyyy/MM/dd HH:mm:ss</date>
                        <boolean>
                            <t>TRUE</t>
                            <f>FALSE</f>
                        </boolean>
                    </pattern>
                </column>
                <column backgroundcolor="255,255,255,255" textcolor="0,0,0,255" width="37.5"
                 underline="false" textposition="32" linemargin="0.0" horizontalmargin="5.0"
                 index="2" usetableattribute="false" textfont="Dialog,0,12" verticalmargin="5.0">
                    <pattern>
                        <number>#,##0.### ℃</number>
                        <date>yyyy/MM/dd HH:mm:ss</date>
                        <boolean>
                            <t>TRUE</t>
                            <f>FALSE</f>
                        </boolean>
                    </pattern>
                </column>
                <column backgroundcolor="255,255,255,255" textcolor="0,0,0,255" width="37.5"
                 underline="false" textposition="32" linemargin="0.0" horizontalmargin="5.0"
                 index="3" usetableattribute="false" textfont="Dialog,0,12" verticalmargin="5.0">
                    <pattern>
                        <number>#,##0.### ℃</number>
                        <date>yyyy/MM/dd HH:mm:ss</date>
                        <boolean>
                            <t>TRUE</t>
                            <f>FALSE</f>
                        </boolean>
                    </pattern>
                </column>
            </columns>
            <rows linewidth="0.5">
                <row height="10.230556" index="0" />
                <row height="9.172222" index="1" />
                <row height="9.172222" index="2" />
                <row height="9.172222" index="3" />
                <row height="9.172222" index="4" />
                <row height="9.172222" index="5" />
                <row height="9.172222" index="6" />
```

```
                        <row height="9.172222" index="7"/>
                        <row height="9.172222" index="8"/>
                        <row height="9.172222" index="9"/>
                        <row height="9.172222" index="10"/>
                        <row height="9.172222" index="11"/>
                        <row height="9.172222" index="12"/>
                    </rows>
                    <data type="method">
                        <component id="2"/>
                        <method name="getObjectTable"/>
                    </data>
                </table>
                <display height="86.43055" x="3.0" bordercolor="0,0,0,255" width="165.80556"
                 originalsize="true" bordermargin="0.0" y="159.0">
                    <data type="component">
                        <component id="4"/>
                    </data>
                </display>
            </paper>
        </papers>

</information>
```

※本サンプルは一部省略している部分があり、このままでは使用できませんのでご注意ください

6. 帳票の作成

プラットフォーム上で扱われているデータや画面を帳票出力するために、以下の機能を提供します。

1)帳票レイアウト設計機能

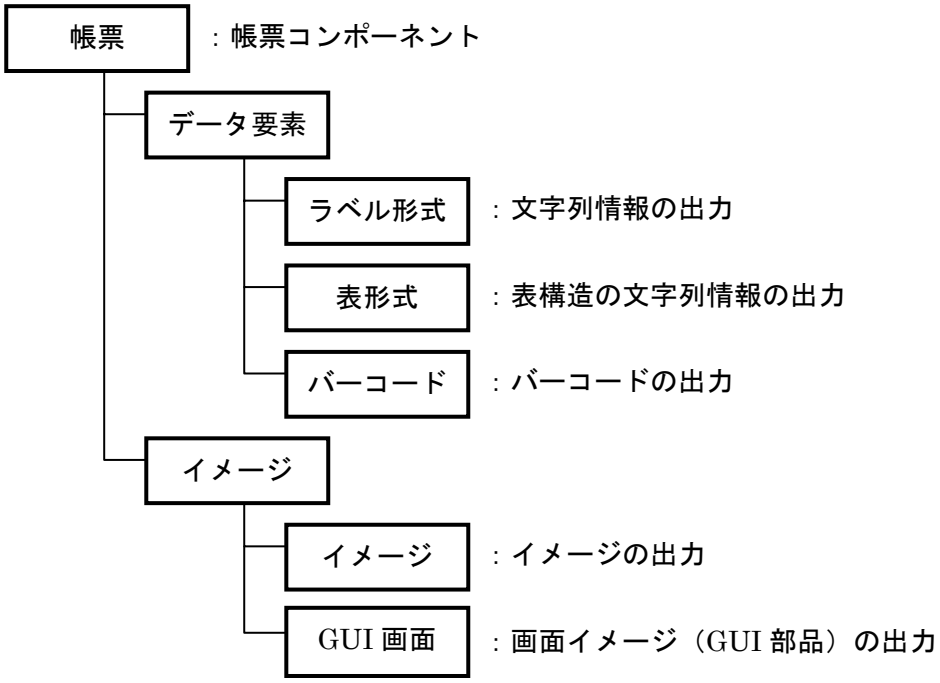
アプリケーション構築機能として、帳票レイアウト設計機能を提供します。帳票レイアウト設計機能は、アプリケーションビルダーの機能として開発者向けに提供します。

2)帳票印刷機能

帳票の印刷、プレビュー表示、印刷設定の機能を、帳票コンポーネントとして提供します。

6.1. 帳票のデータ構造

一枚の帳票は、以下のような構造で構成されています。帳票を作成する作業は、以下の構造を構築することになります。



6.1.1. 帳票コンポーネント

◆機能

・印刷機能

・印刷プロパティ（プリンタ選択／用紙設定／枚数設定など）設定

・印刷プレビュー機能（印刷イメージ表示／帳票レイアウト設定）

◆属性

・帳票サイズ

・帳票方向

・帳票余白（上下左右）

◆ビルダー上での操作

・属性の設定

6.1.2. 帳票構成要素：ラベル形式要素

◆概要

文字列 1 つについて表示するための帳票要素。Java の String クラスに対応し、文字列長に制限はなく、サイズにあわせて折り返して描画される。また、文字列中に改行コードがある場合、その位置にて改行される。

[イメージサンプル]

◆属性

・文字フォント
・文字色
・下線有無
・文字表示位置
・テキスト余白（縦方向／横方向）
・行間
・背景色
・罫線色
・罫線幅

◆データ設定方法

①テキスト入力

固定値として文字列を入力する。

②メソッド戻り値

コンポーネントのメソッドの戻り値で文字列を設定する。ただし、指定できるメソッドは引数の無いものに制限する。

◆画面操作

・属性の変更
・描画位置の設定
・描画サイズの設定
・テキストの入力
・データ取得メソッドの設定

6.1.3. 帳票構成要素：テーブル形式要素

◆概要

表形式で描画するための帳票要素。プラットフォームが提供するデータ構造 PFObjectTable クラスに対応する。各セルの描画は、すべて横幅にあわせて折り返して描画される。また、文字列中に改行コードがある場合、その位置にて改行される。縦幅については、デフォルトではテキスト描画に必要な領域にあわせて自動調整され、それより小さな値には設定できない（大きな値に設定することは可能）。なお、描画属性については、表全体の設定、ヘッダ行に対する設定、カラム毎にデータ行に対する設定ができる。

[イメージサンプル]

列 A	列 B	列 C
データ A1	データ B1	データ C1
データ A2	データ B2	データ C2

◆属性

①テーブル全体属性

・文字フォント／文字色／下線有無／文字表示位置
・テキスト余白（縦方向／横方向）／行間
・背景色／罫線色／罫線幅（外枠／ヘッダ区切／縦方向／横方向）
・テーブル縦幅の自動調整モード

②ヘッダ行属性

・ヘッダ行描画有無
・テーブル全体属性の引継ぎ
・文字フォント／文字色／下線有無／文字表示位置
・テキスト余白（縦方向／横方向）／行間／背景色

③カラム別データ行属性　※各カラム単位で設定

・テーブル全体属性の引継ぎ
・文字フォント／文字色／下線有無／文字表示位置
・テキスト余白（縦方向／横方向）／行間／背景色
・テキスト表示パターン（対象データ型：数値／論理値／日付）

◆データ設定方法

コンポーネントのメソッドの戻り値で PFObjectTable を設定する。ただし、指定できるメソッドは引数の無いものに制限する。

◆画面操作

・属性の変更（テーブル全体／ヘッダ行／カラム別データ行）
・描画位置の設定
・描画サイズの設定（横方向のみ）
・カラム幅の設定
・行高さの設定
・縦方向行高さの自動調整
・データ取得メソッドの設定

6.1.4. 帳票構成要素：バーコード要素

◆概要

バーコード表示するための帳票要素。文字列情報を入力とし、指定されたコード体系に変換して出力する。バーコードイメージは拡大／縮小が可能。なお、バーコードの描画はイメージに落とすことはしないため、拡大や縮小によって文字情報や線情報が崩れてしまうことはない。

[イメージサンプル]

◆属性

・コード体系
・データ文字列表示有無
・チェックディジット有無
・罫線色
・罫線幅

◆データ設定方法

①テキスト入力

固定値として文字列を入力する。

②メソッド戻り値

コンポーネントのメソッドの戻り値で文字列を設定する。ただし、指定できるメソッドは引数の無いものに制限する。

◆画面操作

・属性の変更
・描画位置の設定
・描画サイズの設定（横幅のみ／縦幅は自動調整）
・テキストの入力
・データ取得メソッドの設定

6.1.5. 帳票構成要素：イメージ要素

◆概要

イメージを表示するための帳票要素。Java の Image データを入力とし、出力する。イメージの描画は横幅にあわせて調整され、縦方向のサイズは縦横比率を維持した状態で自動調整される。

◆属性

・罫線色
・罫線幅

◆データ設定方法

①ファイルからロード

イメージファイルを指定し、イメージ情報をロードする。

②メソッド戻り値

コンポーネントのメソッドの戻り値でイメージ（Image）を設定する。ただし、指定できるメソッドは引数の無いものに制限する。

◆画面操作

・属性の変更
・描画位置の設定
・描画サイズの設定（横幅のみ／縦幅は自動調整）
・イメージファイルからのロード
・データ取得メソッドの設定

6.1.6. 帳票構成要素：GUI 画面要素

◆概要

GUI 画面のイメージをそのまま描画するための帳票要素。プラットフォームが提供する GUI コンポーネントである PFGUIComponent を指定する。イメージの描画は横幅にあわせて調整され、縦方向のサイズは縦横比率を維持した状態で自動調整される。なお、GUI コンポーネントの描画はイメージに落とすことはせず、GUI コンポーネントの描画ロジックを使用するため、拡大や縮小によって文字情報や線情報が崩れてしまうことはない。

◆属性

・罫線色
・罫線幅

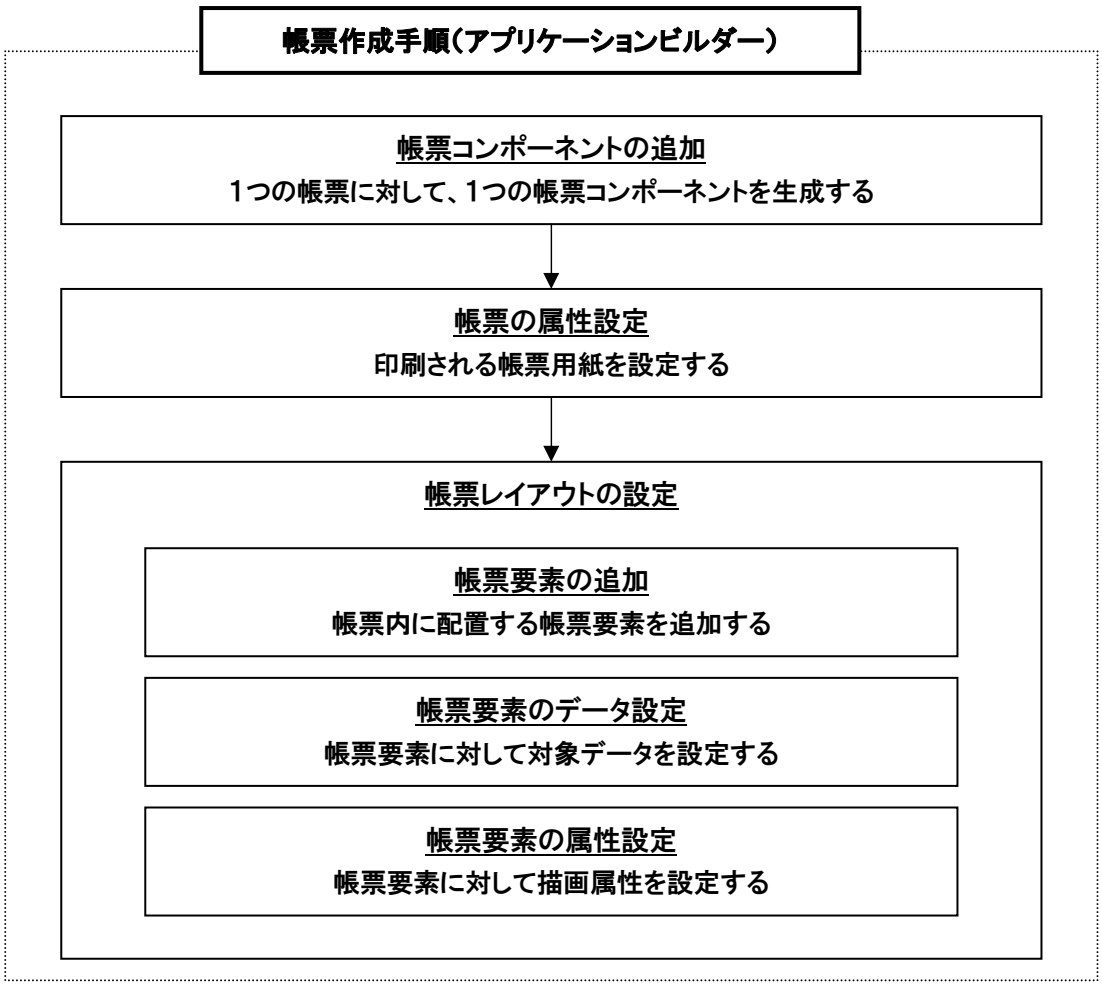
◆データ設定方法

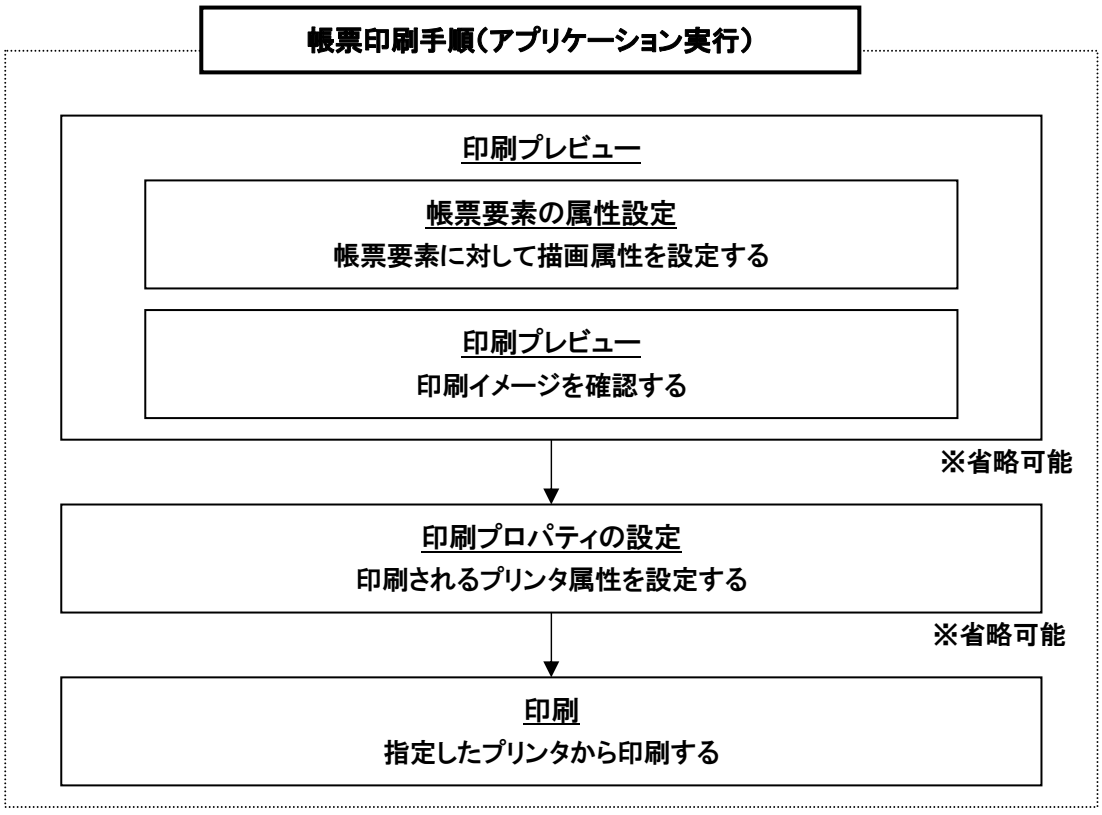
アプリケーション上に存在する GUI コンポーネントを設定する。

◆画面操作

・属性の変更
・描画位置の設定
・描画サイズの設定（横幅のみ／縦幅は自動調整）
・対象 GUI コンポーネントの設定

6.2. 帳票作成／印刷の流れ





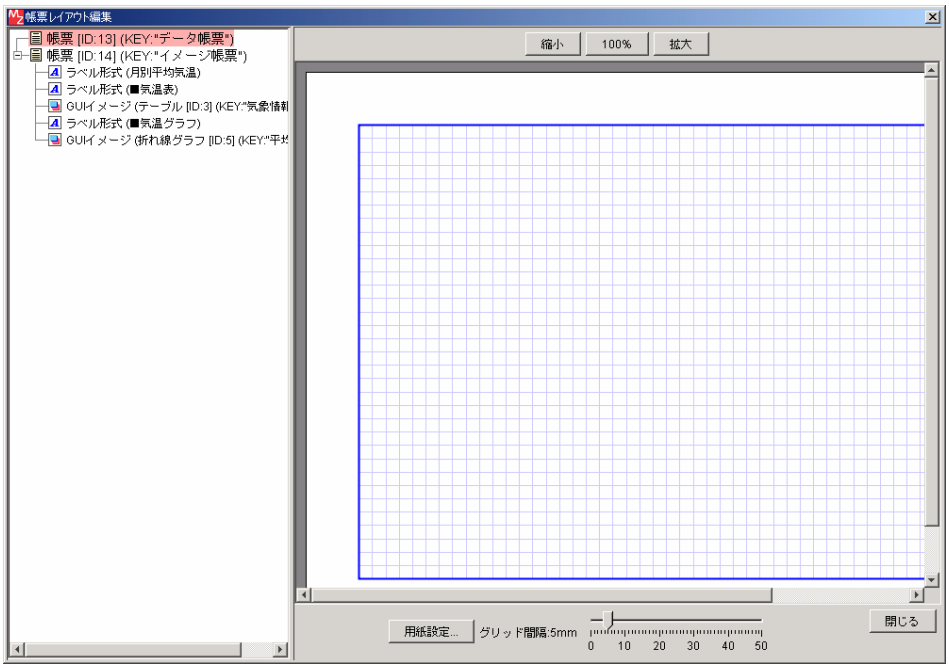
6.3. 帳票作成の操作手順

1)帳票コンポーネントの追加

帳票印刷機能を提供する、『帳票コンポーネント』をアプリケーションに追加します。帳票コンポーネントは、GUI や汎用ユーティリティと同じように標準コンポーネントとして提供され、追加操作は通常のコンポーネント追加操作によって行います。

2)帳票の属性設定

帳票が持つ属性を設定します。設定できる属性は先に列挙したもので、これらを設定するための設定画面を提供します。

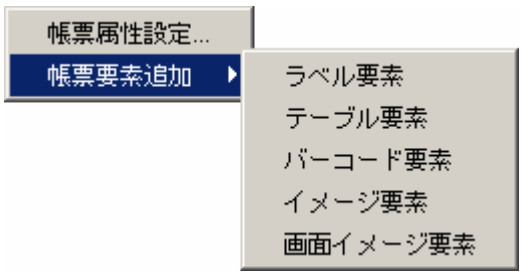
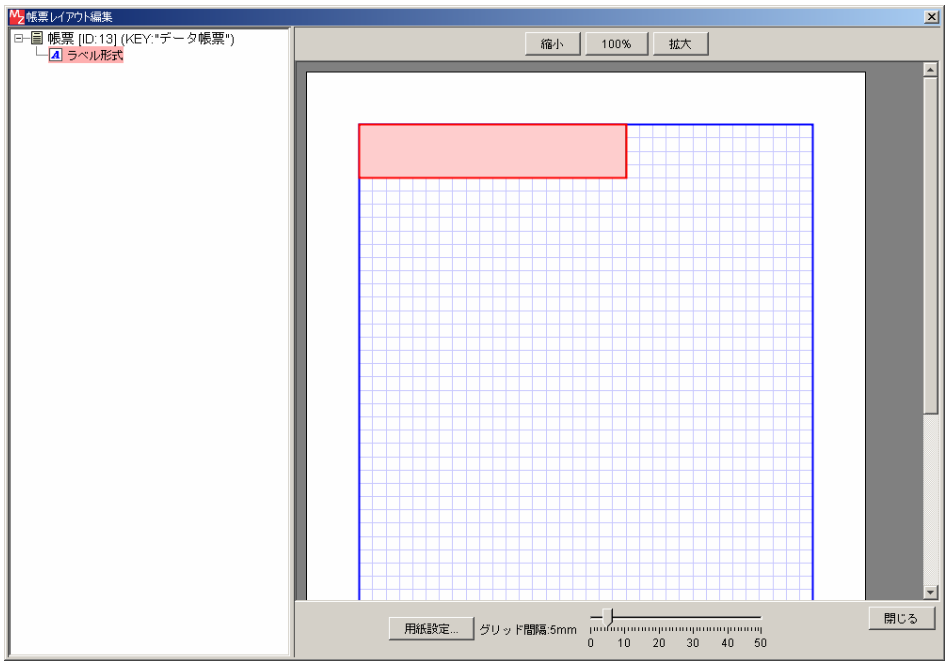
画面	アプリケーションビルダー メイン画面
手順	①アプリケーションビルダーの［帳票編集］ボタンを押下 → 帳票レイアウト設定画面が表示される  ②帳票レイアウト設定画面の帳票階層ツリーから編集対象の帳票を選択 ③帳票レイアウト設定画面の［用紙設定…］ボタンを押下 ※帳票の余白をマウス左ボタンでダブルクリックしても同様 → 用紙設定画面が表示される  ④用紙設定画面上で属性を設定

3)帳票レイアウトの設定

作成する帳票にあわせて帳票要素を追加し、それぞれについて設定を行うことで、対象となる帳票のレイアウトを設定します。

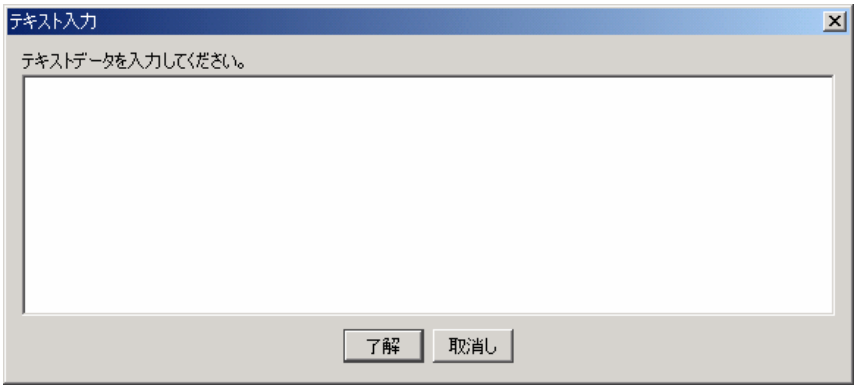
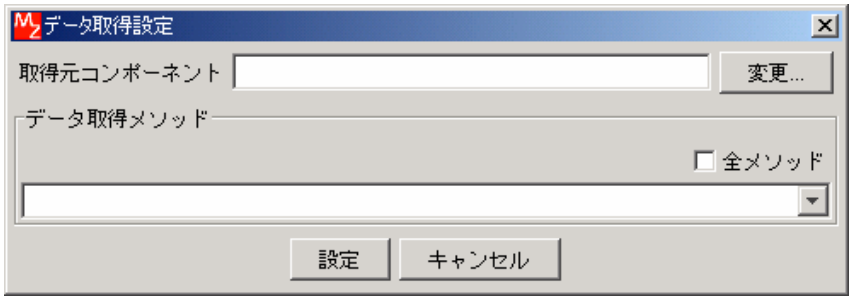
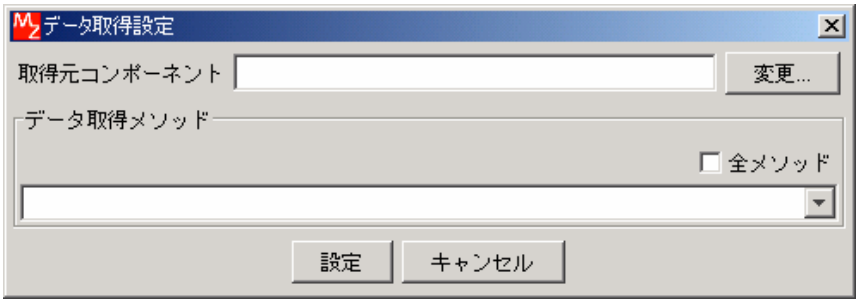
a)帳票要素の追加

帳票要素を追加します。プラットフォームから提供されている帳票要素から、目的に適した要素を選択します。

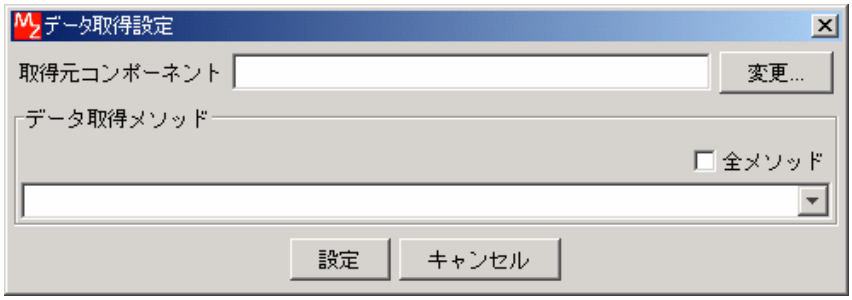
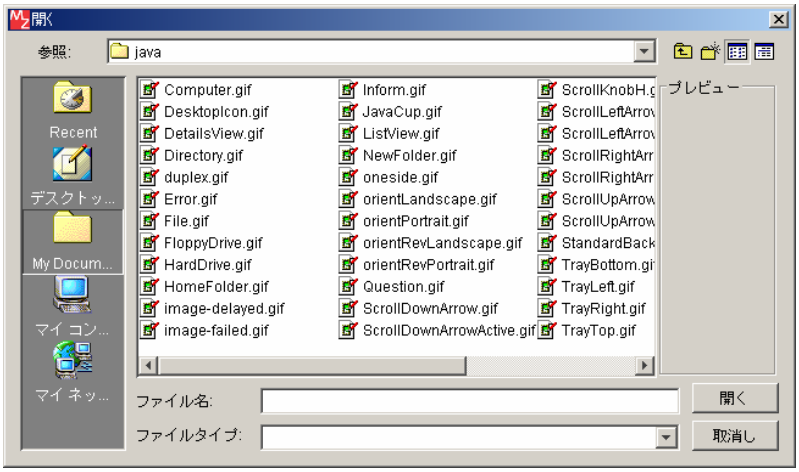
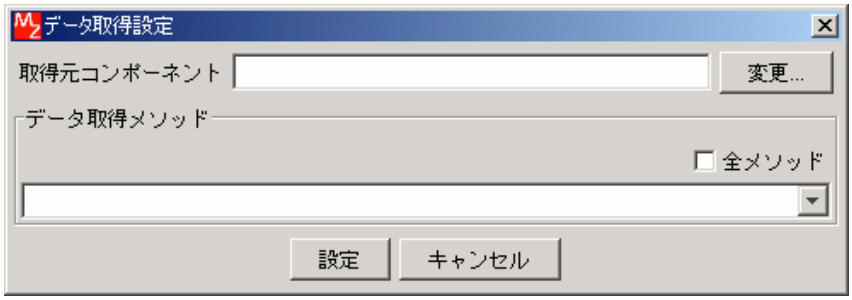
画面	アプリケーションビルダー 帳票レイアウト設定画面
手順	①帳票階層ツリーの帳票ノードからマウス右ボタンクリックでメニューを表示 ※帳票イメージの余白部分でのマウス右ボタンクリックでも同様  ②追加したい帳票要素種類を選択 → 指定した帳票要素が帳票階層ツリーとレイアウト画面に追加される 

b)帳票要素のデータ設定

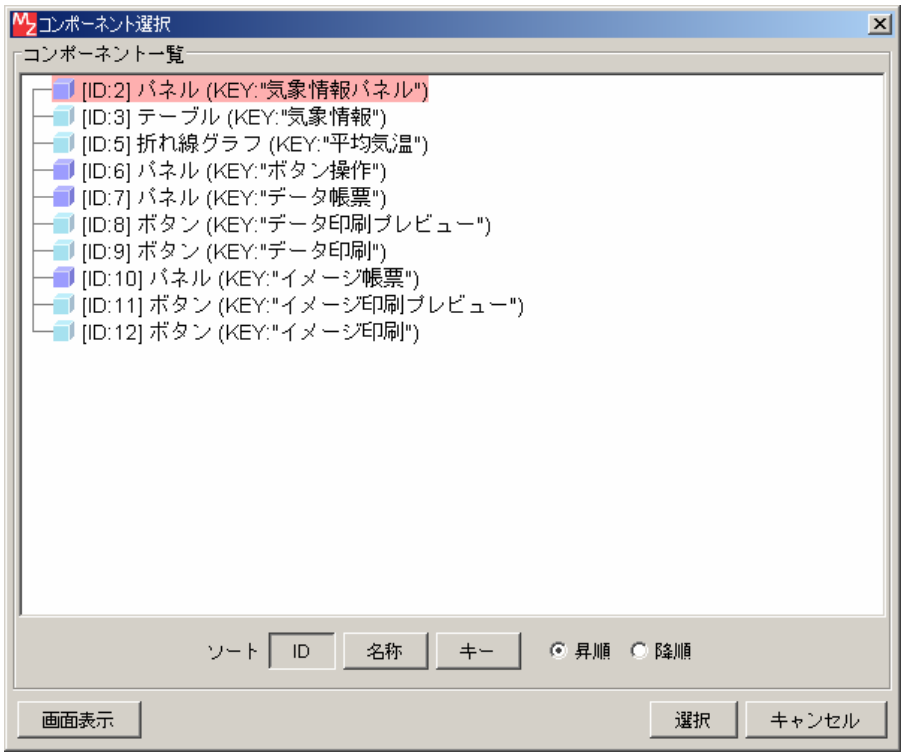
追加した帳票要素について、データの設定を行います。データ設定方法は帳票要素によって異なります。

画面	アプリケーションビルダー 帳票レイアウト設定画面
手順	①帳票階層ツリーの要素ノードからマウス右ボタンクリックでメニューを表示 ※帳票イメージの各要素でのマウス右ボタンクリックでも同様 ②データ設定を行う ◇ラベル形式の場合 ［指定方法Ａ］テキスト入力 描画したいテキストを入力する。改行を含んだ文字列も設定可能。  ［指定方法Ｂ］コンポーネントのメソッド戻り値設定 コンポーネントとメソッドを選択することにより、データの取得方法を設定。 指定できるメソッドは、戻り値が void でなく、引数がないものに限定。  ◇テーブル形式の場合 コンポーネントとメソッドを選択することにより、データの取得方法を設定。 指定できるメソッドは、戻り値が PFObjectTable に変換可能で、 引数がないものに限定。 

b)帳票要素のデータ設定 続き

手順	◇バーコード形式の場合 ［指定方法Ａ］テキスト入力 バーコード化したいデータを入力する。  ［指定方法Ｂ］コンポーネントのメソッド戻り値設定 コンポーネントとメソッドを選択することにより、データの取得方法を設定。 指定できるメソッドは、戻り値が void でなく、引数がないものに限定。  ◇イメージの場合 ［指定方法Ａ］イメージファイルからのロード イメージファイルを選択する。  ［指定方法Ｂ］コンポーネントのメソッド戻り値設定 コンポーネントとメソッドを選択することにより、データの取得方法を設定。 指定できるメソッドは、戻り値が Image に変換可能で、引数がないものに限定。 

b)帳票要素のデータ設定 続き

手順	◇画面イメージの場合 GUI コンポーネントを選択。選択画面は通常のコンポーネント選択画面。 

c)帳票要素の配置／サイズの設定

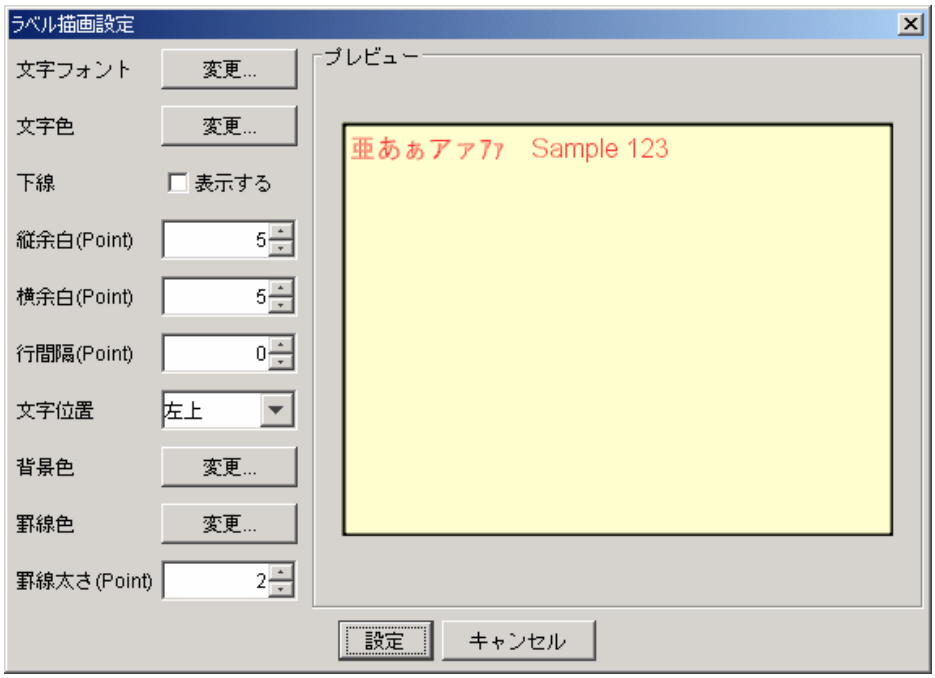
追加した帳票要素について、配置の位置やサイズの設定を行います。

画面	アプリケーションビルダー 帳票レイアウト設定画面
手順	①帳票イメージの各要素をドラッグすることで位置設定が可能（全要素共通） ②帳票イメージの各要素の縁をドラッグすることで描画サイズの設定が可能 ラベル　：横サイズ[右端]／縦サイズ[下端]／全体サイズ[右下隅] テーブル　：カラム幅[カラム右端]／行高さ[行下端] バーコード　：横方向[右端]　※縦方向は自動リサイズ イメージ　：横方向[右端]　※縦方向は自動リサイズ 画面イメージ　：横方向[右端]　※縦方向は自動リサイズ

d)帳票要素の属性設定

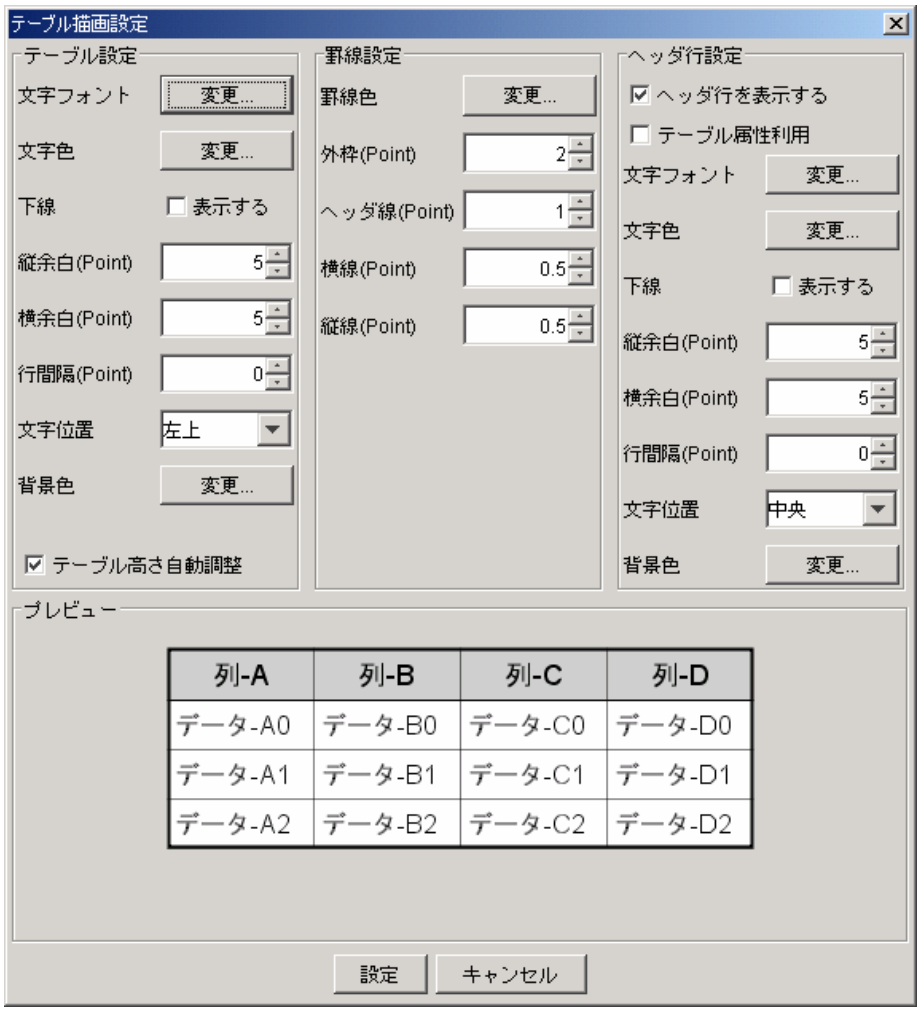
追加した帳票要素について、属性設定を行います。属性設定方法は帳票要素によって異なります。

画面	アプリケーションビルダー 帳票レイアウト設定画面
手順	①帳票階層ツリーの要素ノードからマウス右ボタンクリックでメニュー表示 ※帳票イメージの各要素でのマウス右ボタンクリックでも同様 ②属性設定を選択 ③属性設定画面上で入力 ◇ラベル形式の場合



d)帳票要素の属性設定 続き

手順	◇テーブル形式の場合

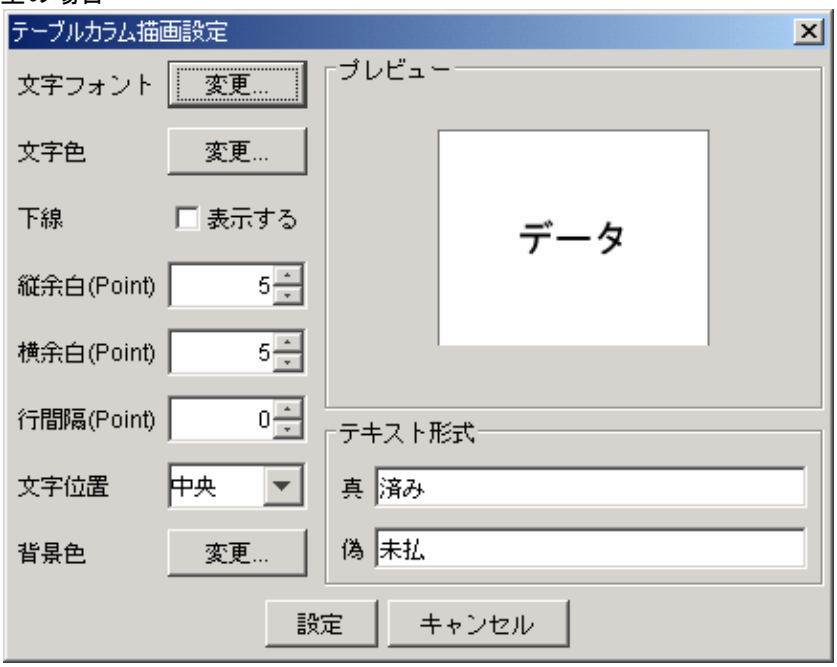


d)帳票要素の属性設定 続き

手順	◇バーコードの場合  ◇イメージ／画面イメージの場合 

e)テーブルカラム描画設定

テーブル形式要素の場合、各テーブルカラム毎に属性を設定できます。

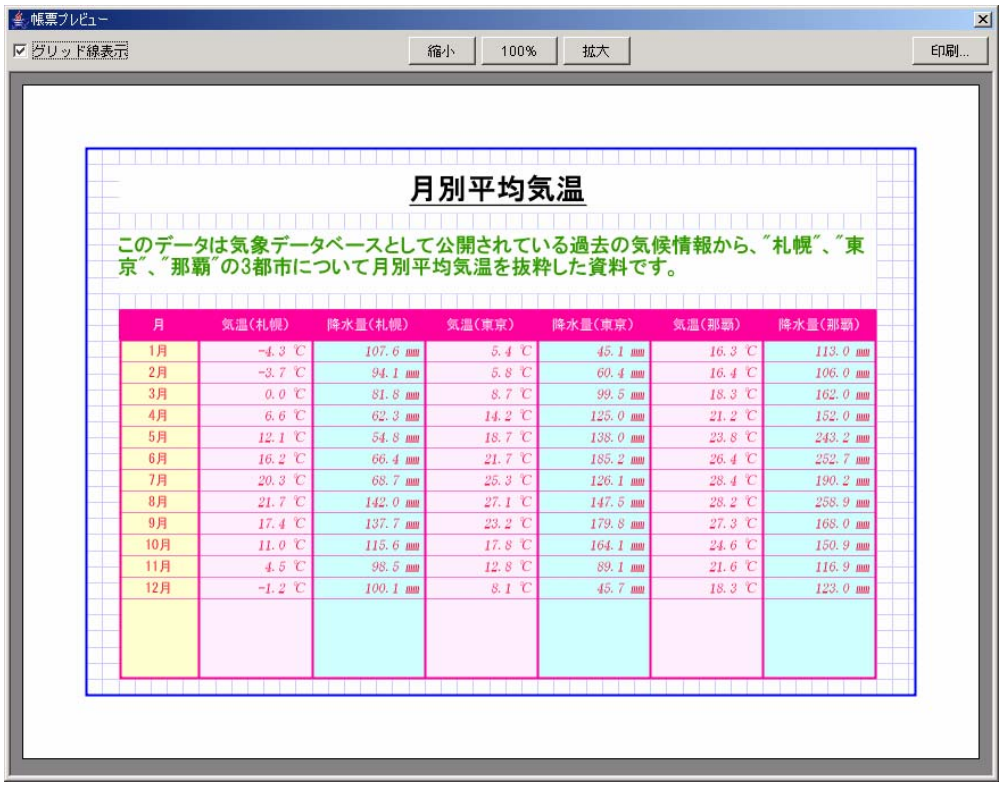
画面	アプリケーションビルダー 帳票レイアウト設定画面
手順	①帳票階層ツリーのテーブル帳票要素ノードからマウス右ボタンクリックで メニュー表示 ※帳票イメージのテーブル要素でのマウス右ボタンクリックでも同様 ②カラム描画設定メニューから対象のカラムを選択 ③テーブルカラム描画設定画面上で入力 ◇数値型の場合  ［表示パターン例：#,##0 円］ ３桁ごとにカンマ表示／１の位は必ず表示／小数点以下なし／単位は“円” ◇論理型の場合  ※表示パターン 数値型：java.text.DecimalFormat のパターン文字列を使用 日付型：java.text.SimpleDateFormat のパターン文字列を使用

6.4. 帳票印刷手順

6.4.1. 帳票印刷プレビュー

帳票コンポーネントは印刷イメージを確認するために、プレビュー画面表示メソッドを提供します。このメソッドの引数は、パラメタ設定ダイアログの親コンポーネントを指定するためのものです。プレビュー画面は 50～250%の倍率で表示ができ、アプリケーション構築時と同様に、帳票および帳票要素の描画属性を編集することも可能です。

画面	アプリケーション 帳票印刷プレビュー画面
手順	①表示の拡大／縮小 拡大：[拡大]ボタンの押下 縮小：[縮小]ボタンの押下 等倍表示：[100%]ボタンの押下 ②配置／サイズなどの設定 ・帳票イメージの各要素をドラッグすることで位置設定が可能（全要素共通） ・帳票イメージの各要素の縁をドラッグすることで描画サイズの設定が可能 ③属性設定 帳票イメージの各要素でのマウス右ボタンクリックでメニューから、 属性設定画面を表示して設定する。 ④テーブルカラム描画設定　※テーブル帳票要素の場合 帳票イメージのテーブル要素でのマウス右ボタンクリックでメニューを表示から、 対象のカラムの描画設定画面を表示して設定する。



月	気温(札幌)	降水量(札幌)	気温(東京)	降水量(東京)	気温(那覇)	降水量(那覇)
1月	-4.3 ℃	107.6 mm	5.4 ℃	45.1 mm	16.3 ℃	113.0 mm
2月	-3.7 ℃	94.1 mm	5.8 ℃	60.4 mm	16.4 ℃	106.0 mm
3月	0.0 ℃	81.8 mm	8.7 ℃	99.5 mm	18.3 ℃	162.0 mm
4月	6.6 ℃	62.3 mm	14.2 ℃	125.0 mm	21.2 ℃	152.0 mm
5月	12.1 ℃	54.8 mm	18.7 ℃	138.0 mm	23.8 ℃	243.2 mm
6月	16.2 ℃	66.4 mm	21.7 ℃	185.2 mm	26.4 ℃	252.7 mm
7月	20.3 ℃	68.7 mm	25.3 ℃	126.1 mm	28.4 ℃	190.2 mm
8月	21.7 ℃	142.0 mm	27.1 ℃	147.5 mm	28.2 ℃	258.9 mm
9月	17.4 ℃	137.7 mm	23.2 ℃	179.8 mm	27.3 ℃	168.0 mm
10月	11.0 ℃	115.6 mm	17.8 ℃	164.1 mm	24.6 ℃	150.9 mm
11月	4.5 ℃	98.5 mm	12.8 ℃	89.1 mm	21.6 ℃	116.9 mm
12月	-1.2 ℃	100.1 mm	8.1 ℃	45.7 mm	18.3 ℃	123.0 mm

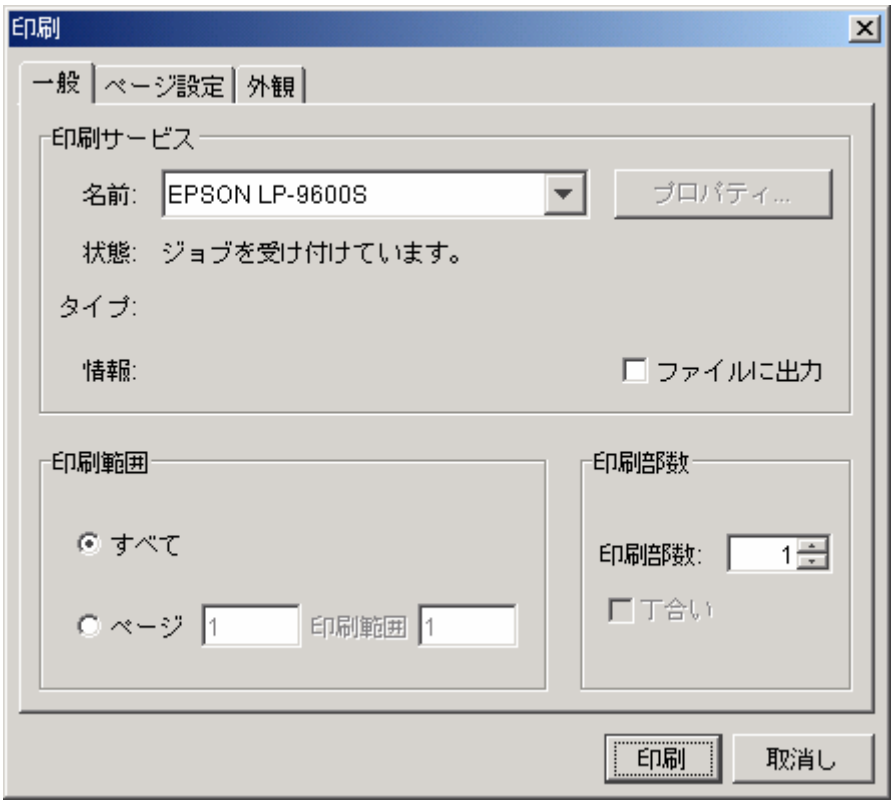
6.4.2. 帳票印刷

帳票コンポーネントは印刷を行うために、以下の 2 つのメソッドを提供します。

①印刷（printPaper() ※引数なし）

②印刷（printPaper(boolean) ※引数あり）

メソッドの引数は、印刷時のパラメタ設定を行うためのダイアログを表示するかどうかを指定するもので、“printPaper()”は“printPaper(false)”と同じです。印刷パラメタ設定ダイアログでは、プリンタの選択や拡大／縮小、印刷部数などの設定が可能であり、これを表示しない場合はデフォルトのプリンタから、デフォルトの設定で印刷されます。



7. 複合コンポーネントの構築

7.1. 複合コンポーネント

複合コンポーネントとは、いくつかのコンポーネントをグループ化して 1 つのコンポーネントのように扱うもので、複数のコンポーネントから新たにコンポーネントを構築する仕組みです。具体的にはイベント伝播によって関係付けられた複数のコンポーネント群を新たに 1 つのコンポーネントとして切り出し、その複合コンポーネントについて、外部に公開するメソッドや、発生させるイベントなどを定義します。通常のプログラミングのモジュール分割や共通ライブラリ作成のような考え方を、コンポーネント指向的に実装したものです。

下図で示すとおり、複合コンポーネントでは内部構造と外部のインターフェイスを定義します。

1)コンポーネント構造

複合コンポーネント内のコンポーネント構造を管理しまう。通常のアプリケーション同様、コンポーネント間はイベントによるメソッド起動によって接続されます。

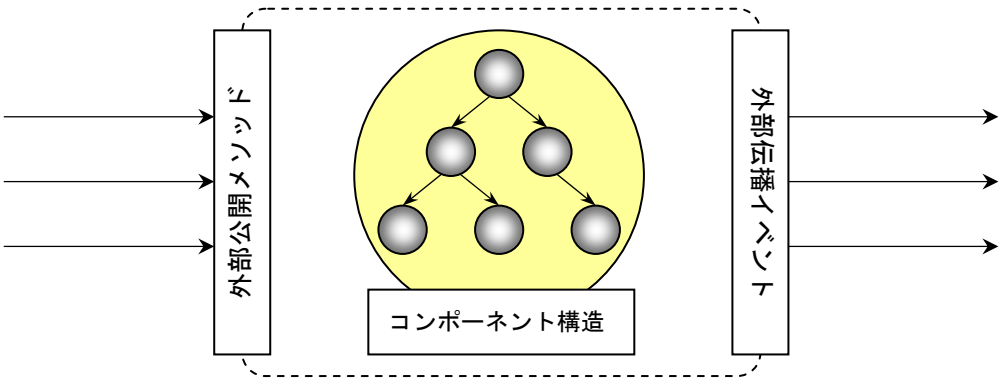
2)外部インターフェイス

①外部公開メソッド

複合コンポーネントが外部に公開するメソッドを設定します。複合コンポーネント内にあるすべてのコンポーネントのメソッドを公開するとアプリケーション構築作業が非効率的になるため、外部に公開するメソッドを選択する機能を提供します。また、メソッド名の重複を回避するために、メソッド名の別名機能を提供します。

②発生イベント

複合コンポーネントから外部に伝播させるイベントを設定します。複合コンポーネント内にあるコンポーネントから発生するイベントのうち、外部に伝播させるイベントのみを選択します。このとき、複数の内部コンポーネントから発生するイベントを識別させるために、イベント番号を設定する機能を提供します。



複合コンポーネントはその用途によって、以下の 2 種類があります。

1)GUI 複合コンポーネント

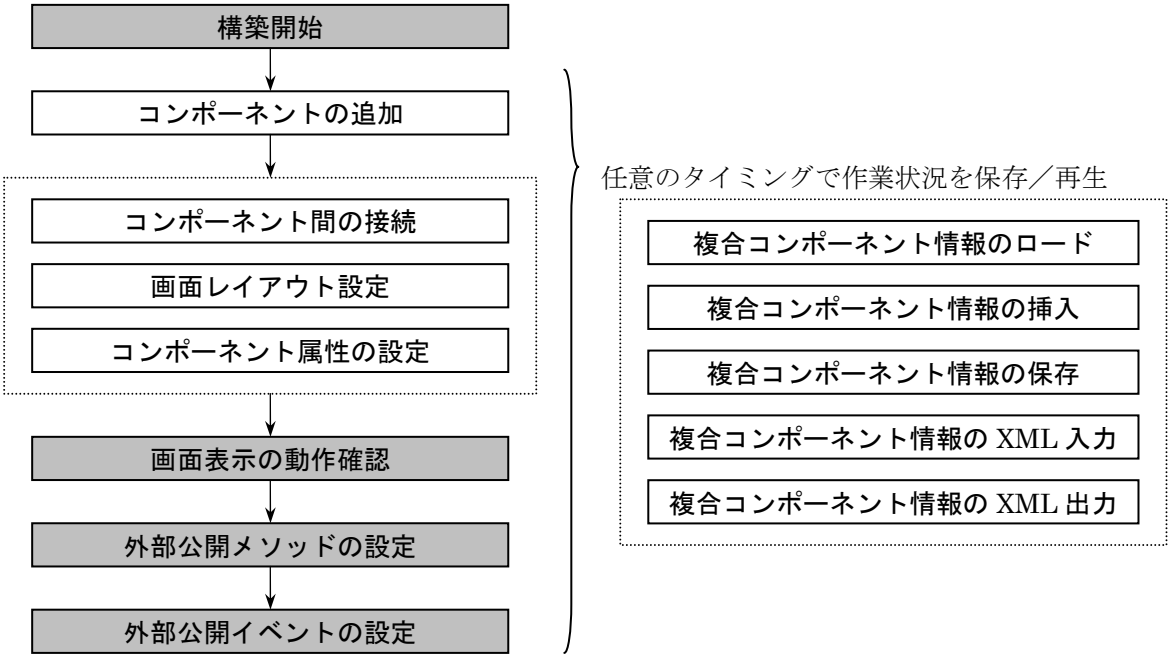
複合コンポーネント自身が GUI コンポーネントとして、他のウィンドウやパネルに貼り付け可能なコンポーネントです。構築時に画面構成も設定します。

2)非 GUI 複合コンポーネント

複合コンポーネント自身は GUI コンポーネントではなく、他のウィンドウやパネルに貼り付けられないコンポーネントです。ただし、非 GUI 複合コンポーネントから別ウィンドウを表示することは可能です。

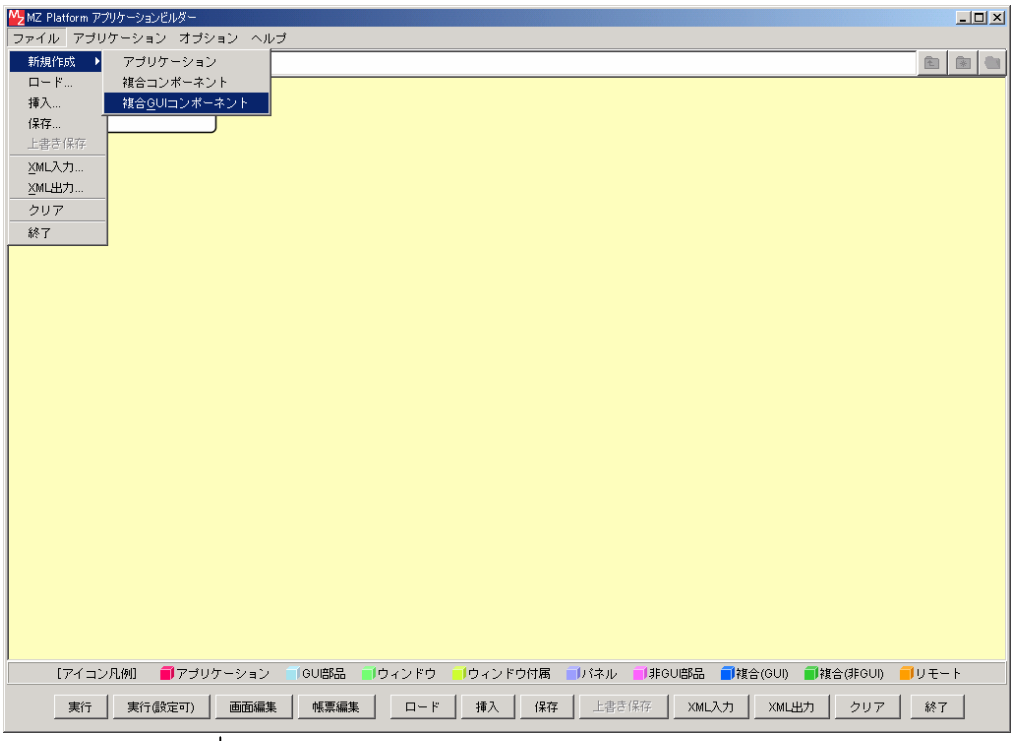
7.2. GUI 複合コンポーネントの構築

GUI 複合コンポーネントの構築は、以下の流れで行います。基本的な操作はアプリケーション構築とほぼ同じですので、以降、操作の異なる操作（下図の網掛け部分）のみ説明します。それ以外の操作については、前述の操作手順を参照してください。

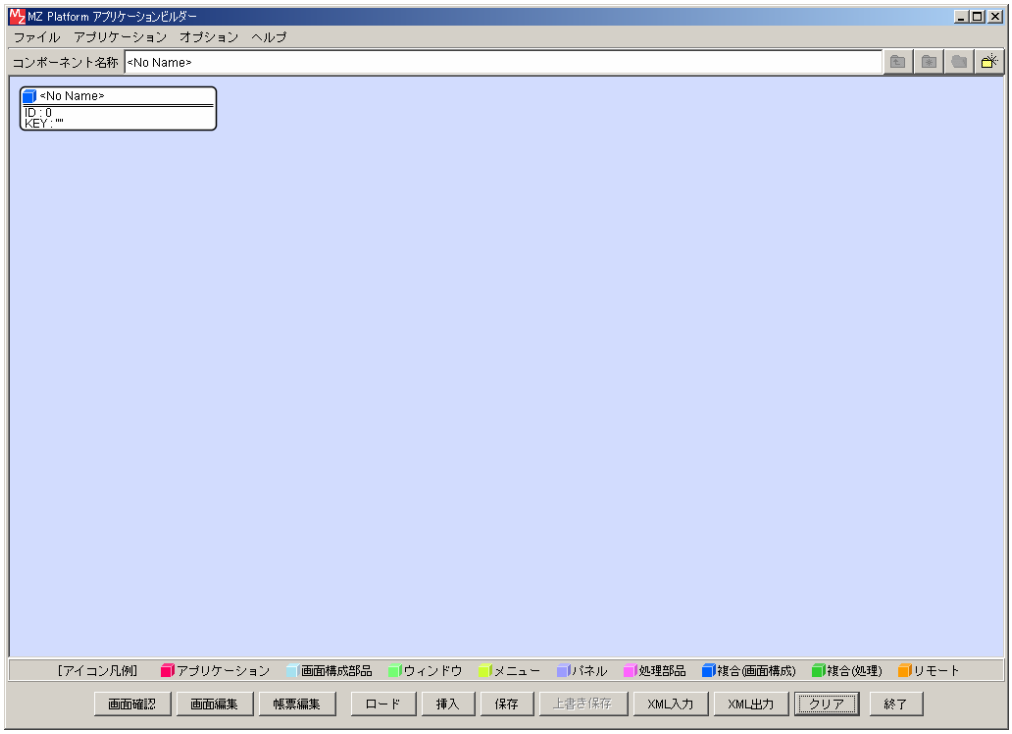


7.2.1. 構築作業の開始

画面	アプリケーションビルダー メイン画面
手順	メニューバーから［ファイル］－［新規作成］－［複合 GUI コンポーネント］を選択

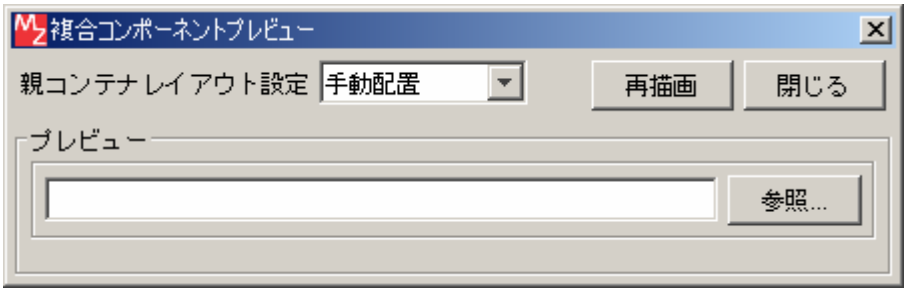


GUI 複合コンポーネント構築モードになる



7.2.2. 画面表示の動作確認

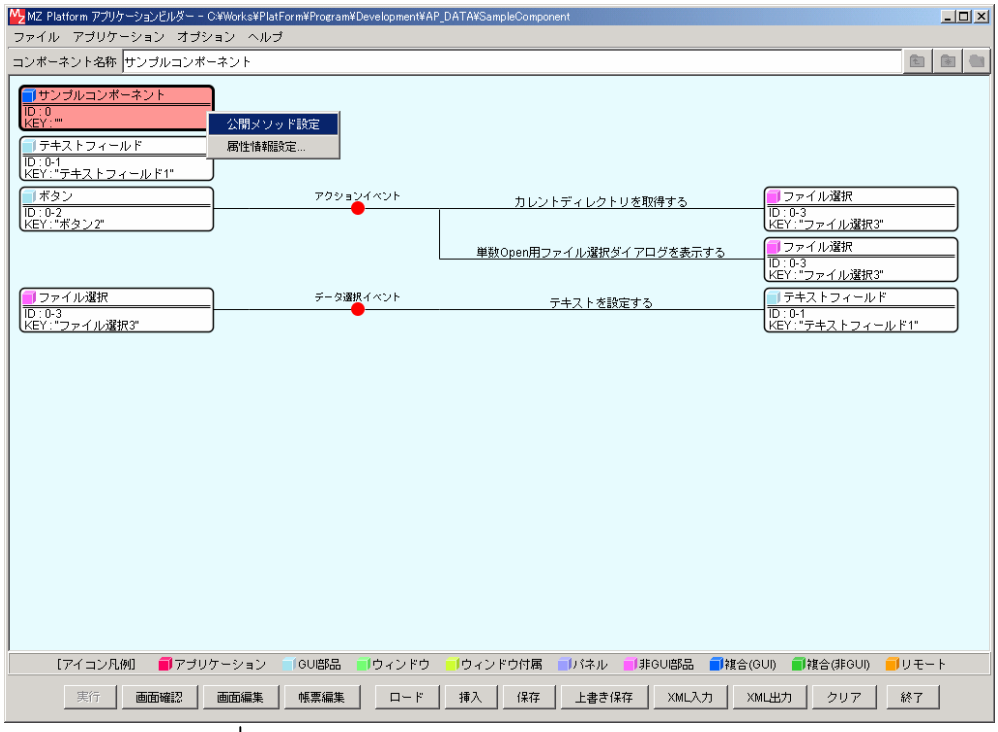
アプリケーションと同様に、画面レイアウトの設定については画面編集のウィンドウ上で行います。しかし複合コンポーネントは、実際に実行して画面を確認することができませんので、画面の確認を行うための機能として、画面確認機能を提供します。

画面	アプリケーションビルダー メイン画面
手順	①［画面確認］ボタンをクリックし、画面プレビューを表示する  ※画面イメージ この画面はテキストフィールドとボタンを組み合わせた、 GUI 複合コンポーネントの画面確認を行った例
	②表示されたプレビューで以下を確認／設定する 【表示イメージの確認】 必要に応じて親コンテナレイアウト設定を変更し、このコンポーネントが画面に配置されたときの表示イメージを確認する。 【コンポーネントの属性設定】 各 GUI コンポーネントが提供する属性設定機能により、表示方法などの属性を設定する。（アプリケーション構築時の“実行(設定可)”と同等機能）

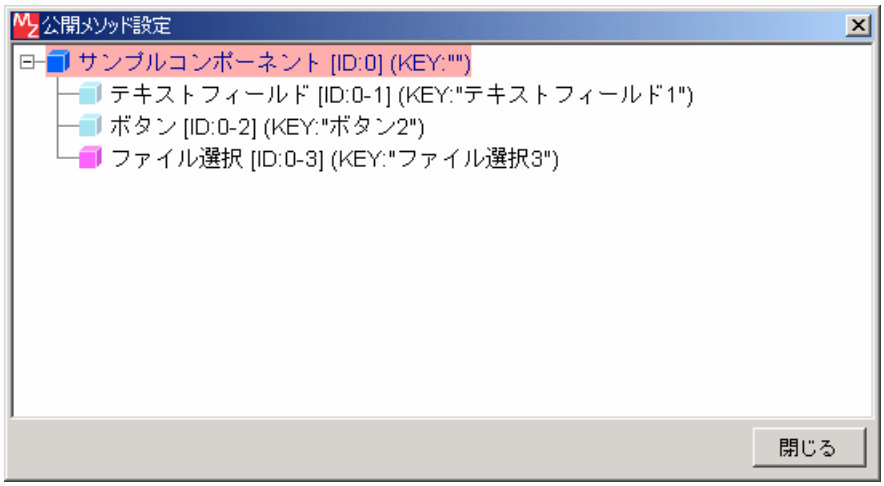
7.2.3. 外部公開メソッドの設定

1)公開メソッドの選択

画面	アプリケーションビルダー メイン画面
手順	①画面最上段の複合コンポーネント上でマウス右クリックしてメニューを表示し、 ［公開メソッド設定］を選択

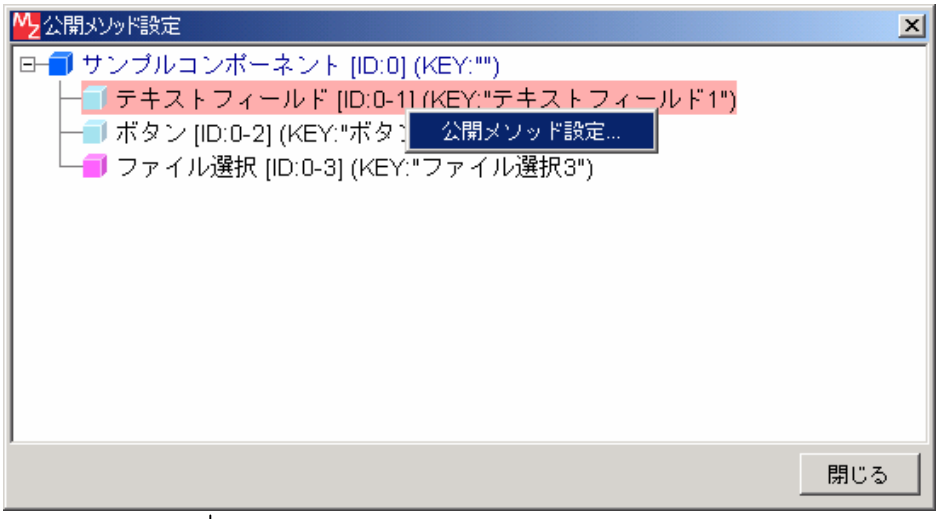


公開メソッド選択画面が表示される

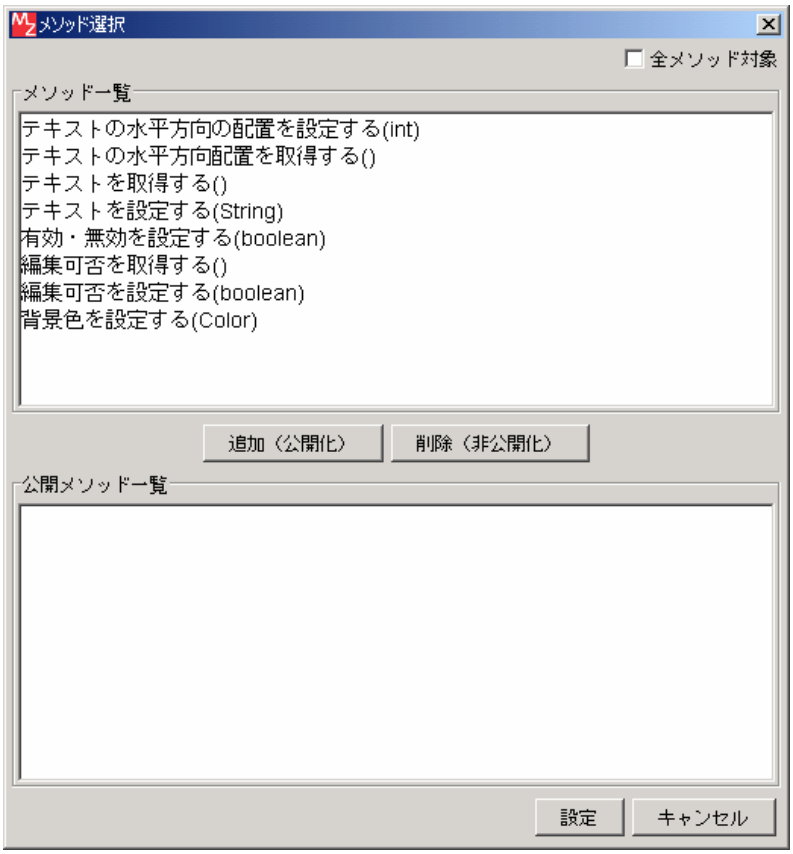


1)公開メソッドの選択 続き

手順	②公開メソッド設定画面上で、公開するメソッドをもつコンポーネントからマウス右クリックでメニューを表示し、［公開メソッド設定］を選択



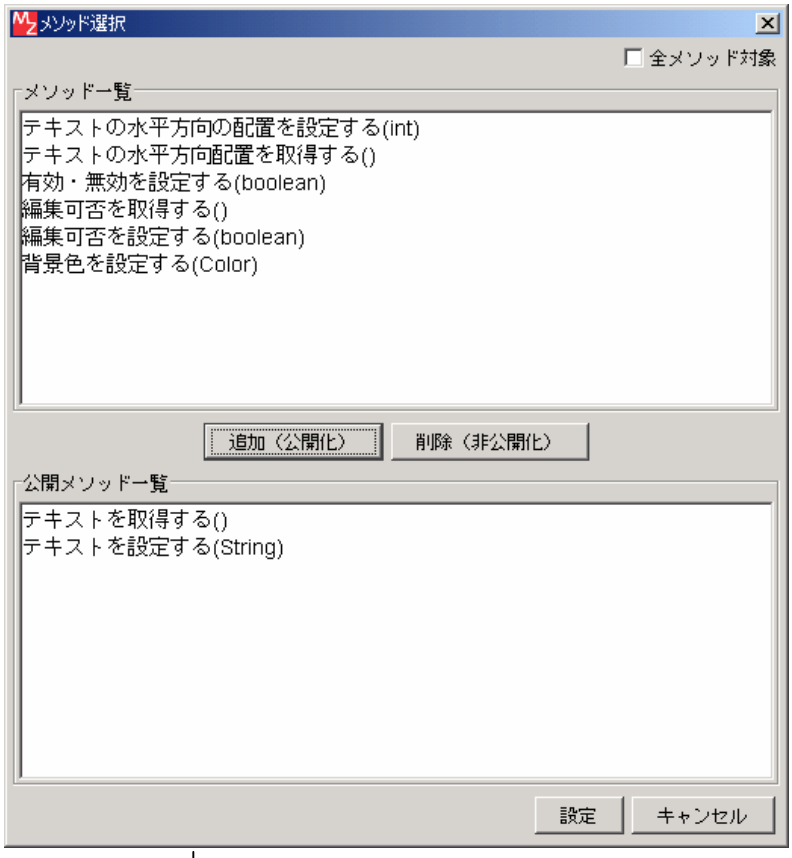
メソッド選択画面が表示される



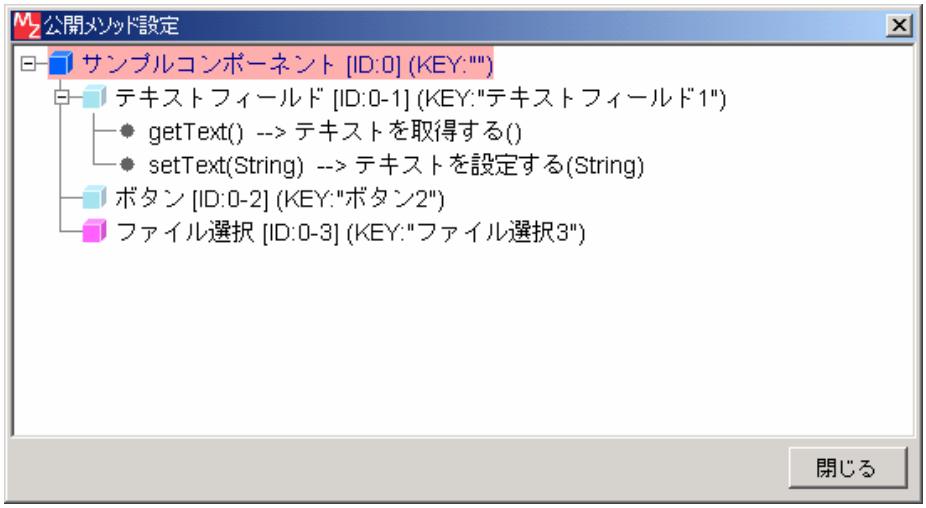
【補足】
起動するメソッドが表示されない場合、そのメソッドが非公開の設定になっている。
“全メソッド対象”のチェックボックスを ON にすれば、
コンポーネントの全 public メソッドが表示され、選択可能となる。
また、必要に応じてメソッドの情報設定(後述)を行えば、常に表示されるようになる。

1)公開メソッドの選択 続き

手順	③メソッド選択画面上で、公開するメソッドを“公開メソッド一覧”に追加する （“メソッド一覧”でメソッドを選択し、[追加] ボタン押下）

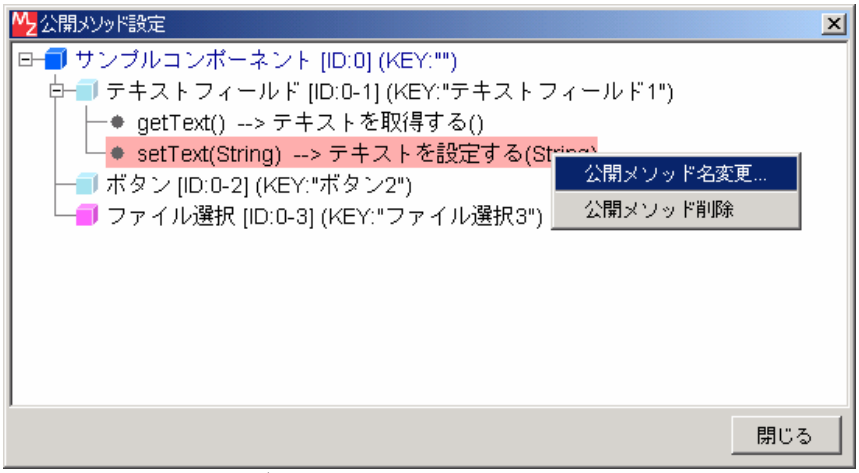


[設定] ボタンで反映される



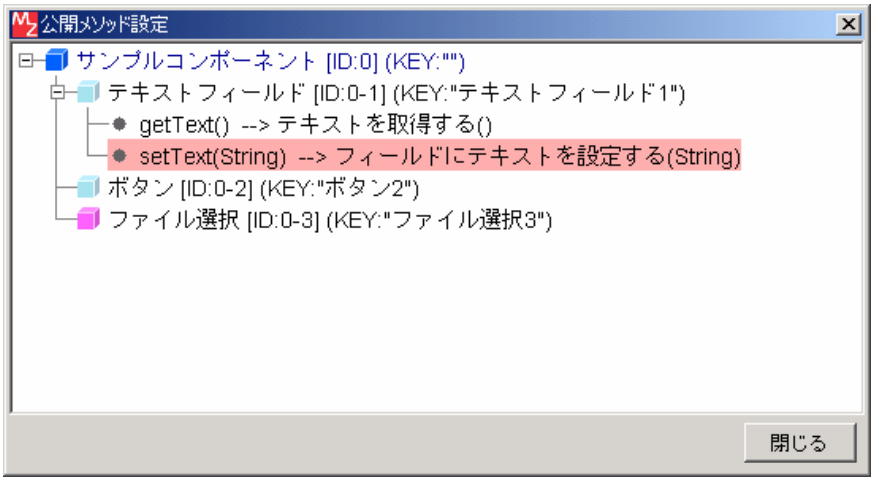
2)公開メソッドの別名設定

画面	公開メソッド設定画面
手順	別名設定したいメソッドからマウス右ボタンでメニューを表示し、 ［公開メソッド名変更...］を選択する

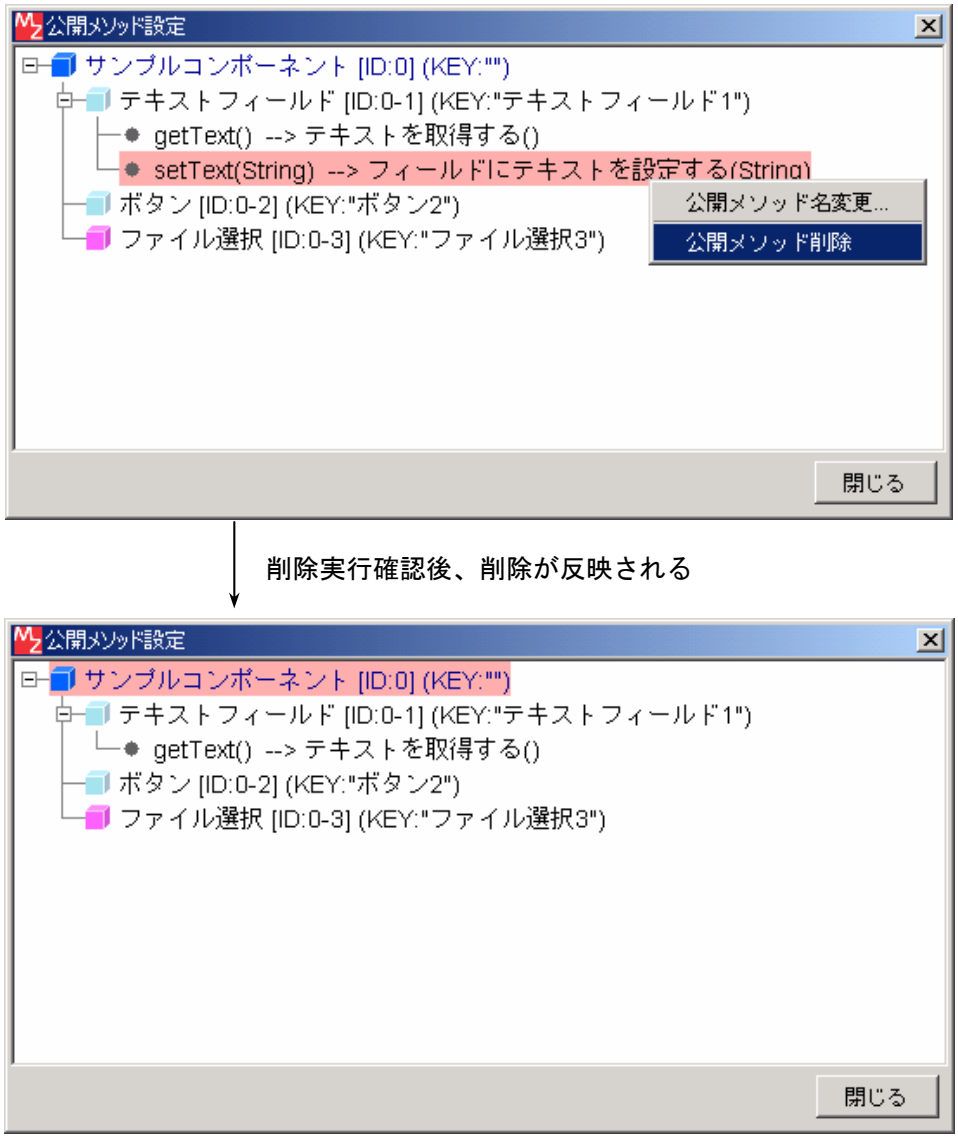


入力画面が表示される

別名が反映される



3)公開メソッドの削除

画面	公開メソッド設定画面
手順	公開設定したメソッドからマウス右ボタンでメニューを表示し、 ［公開メソッド削除］を選択する  削除実行確認後、削除が反映される

7.2.4. 外部公開イベントの設定

外部公開イベントは、内部で発生したイベントを複合コンポーネントのイベント外部伝播メソッドを呼び出すことで設定します。設定方法は通常のコンポーネント間接続と同様です。

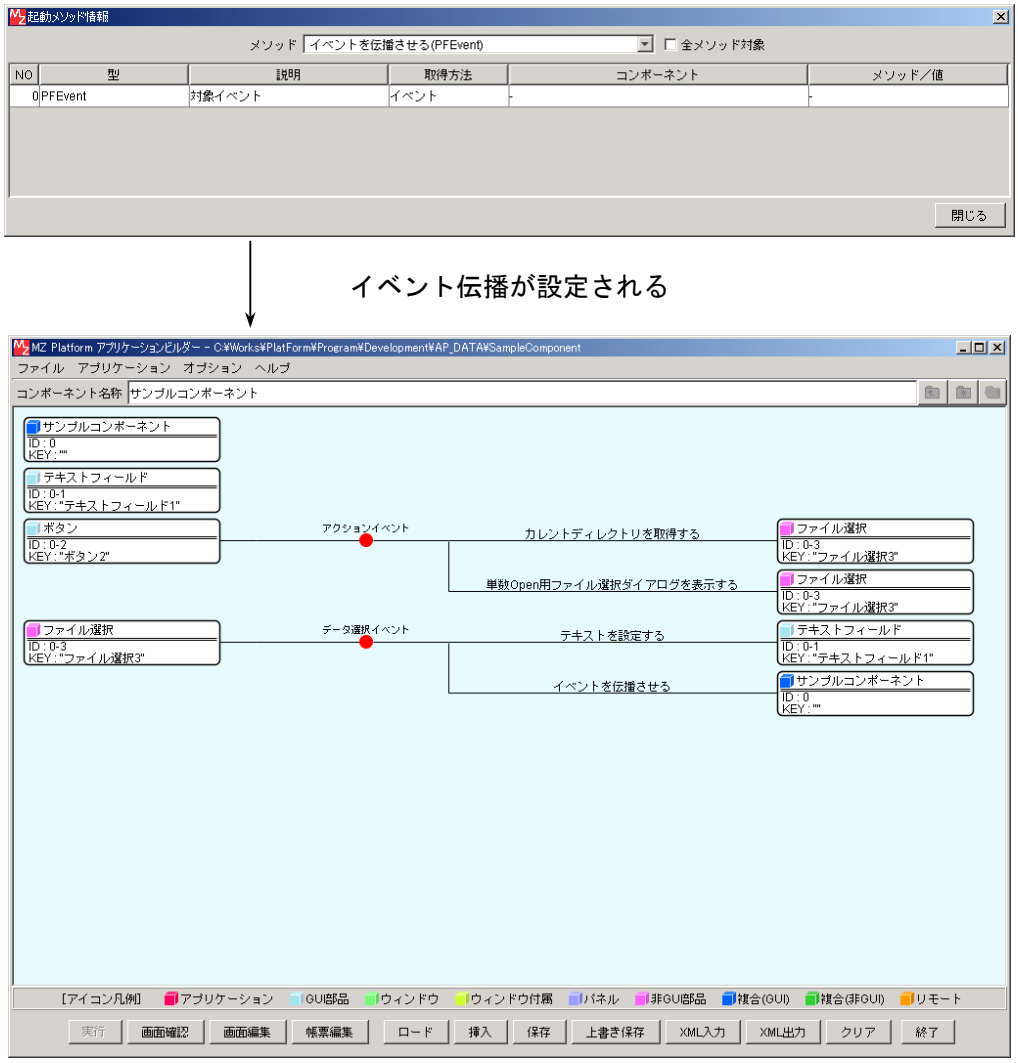
イベント外部伝播メソッドには、2 つの引数形式があります。

①イベントを伝播させる（PFEvent）　［notifyEvent(PFEvent)］

受け取ったイベントをそのまま外部に伝えます。

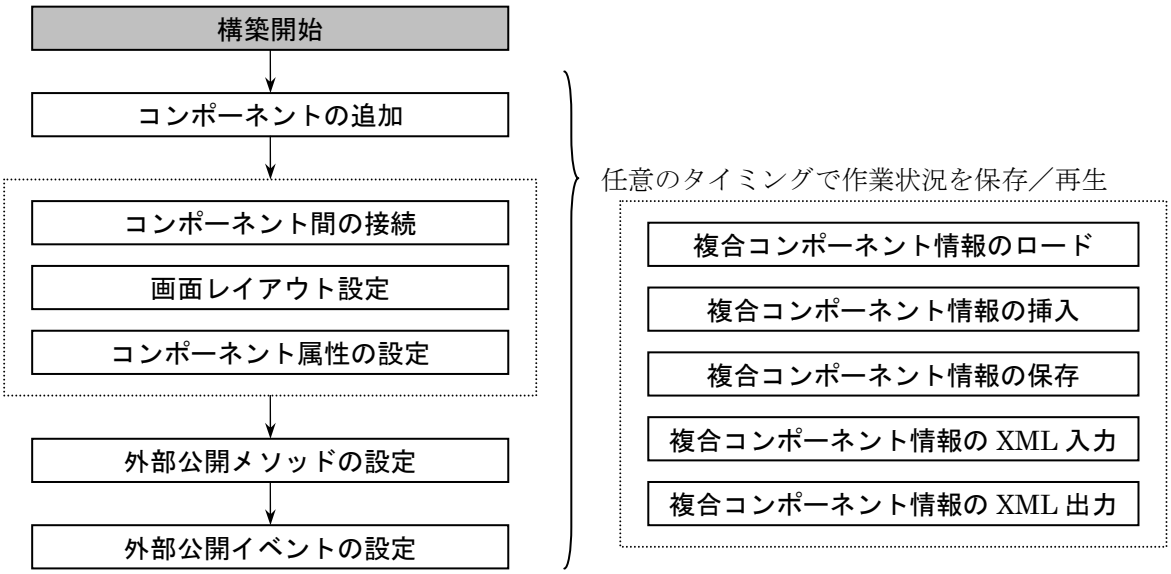
②イベント番号を指定してイベントを伝播させる（PFEvent, int）　［notifyEvent(PFEvent,int)］

受け取ったイベントのイベント番号を第 2 引数に置き換えて外部に伝えます。このとき、イベントオブジェクトは複製してから伝播しますので、このメソッド後に続くメソッド処理においては、もとのイベント番号のままで処理が行われます。

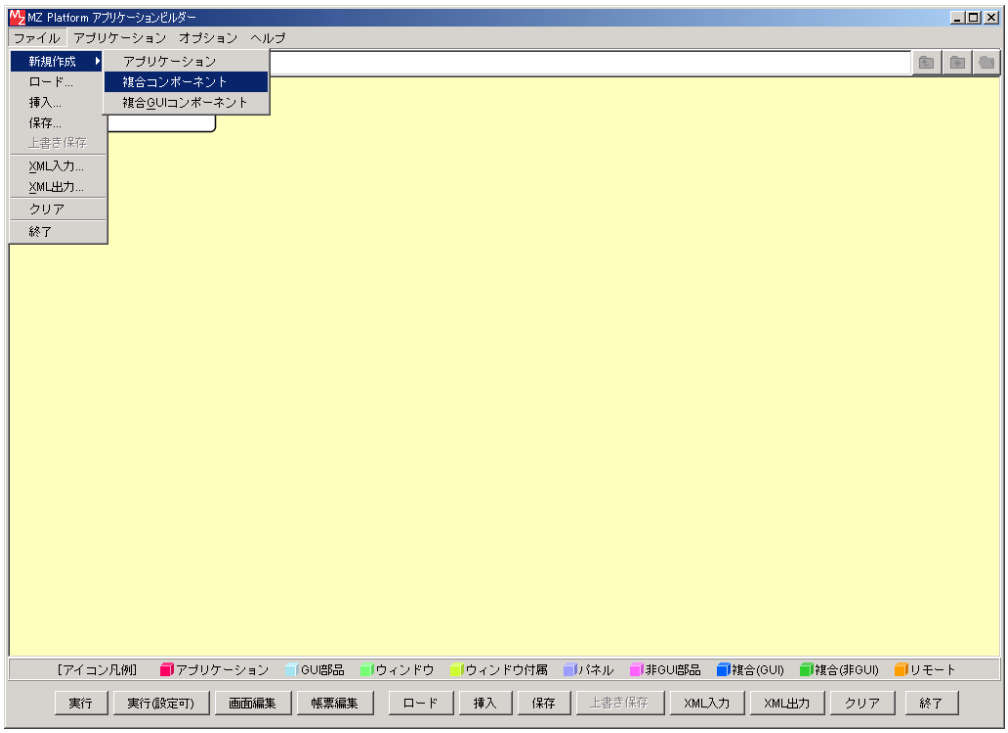
画面	アプリケーションビルダー メイン画面
手順	通常の手順で、伝播させたいイベントから複合コンポーネントのメソッド “イベントを伝播させる(PFEvent)”に接続する ※引数の取得方法には“イベント”を設定する 

7.3. 非 GUI 複合コンポーネントの構築

非 GUI 複合コンポーネントの構築は、以下の流れで行います。操作は GUI 複合コンポーネント構築と同じです。



画面	アプリケーションビルダー メイン画面
手順	メニューバーから［ファイル］－［新規作成］－［複合コンポーネント］を選択



7.4. 複合コンポーネントの利用

複合コンポーネントは、アプリケーションからコンポーネントとして使用可能です。以下の点で通常のコンポーネントと異なります。

1)コンポーネント追加方法

複合コンポーネントの追加は、メイン画面の背景メニューから［複合コンポーネント追加］を選択し、三つの追加操作が可能です。

①新規に複合コンポーネント（非 GUI）を追加する

［コンポーネント生成］を選択すると、複合コンポーネント編集画面に変わります。

②新規に複合コンポーネント（GUI）を追加する

［GUI コンポーネント生成］を選択すると、複合コンポーネント編集画面に変わります。

③既存の複合コンポーネントを追加する

［ロード...］を選択すると、ファイル選択画面が表示され、ファイルを選択します。

2)複合コンポーネントの編集

アプリケーションに貼り付けた複合コンポーネントは、再編集が可能です。

3)複合コンポーネントのメソッド

他のコンポーネントからの接続を受けるメソッドは、各複合コンポーネントで設定されている外部公開メソッドのみです。それ以外のメソッドは接続時に表示されません。また、別名が設定されている場合、別名が表示されます。

4)複合コンポーネントから発生するイベント

複合コンポーネントから発生するイベントは、各複合コンポーネントで設定されている外部公開イベントのみです。具体的には、複合コンポーネント内でイベント伝播メソッドによって外部伝播設定されているイベントのみ発生します。

5)複合コンポーネントの属性

複合コンポーネントの属性編集画面に表示される属性は、コンポーネント ID／コンポーネント Key の基本属性と、外部公開メソッドによって設定／取得可能なもの（setXxxx()/getXxxx()が公開されているもの）のみです。外部で属性として扱いたい場合、外部公開メソッドとして設定しておく必要があります。

7.5. 複合コンポーネントの外部参照化

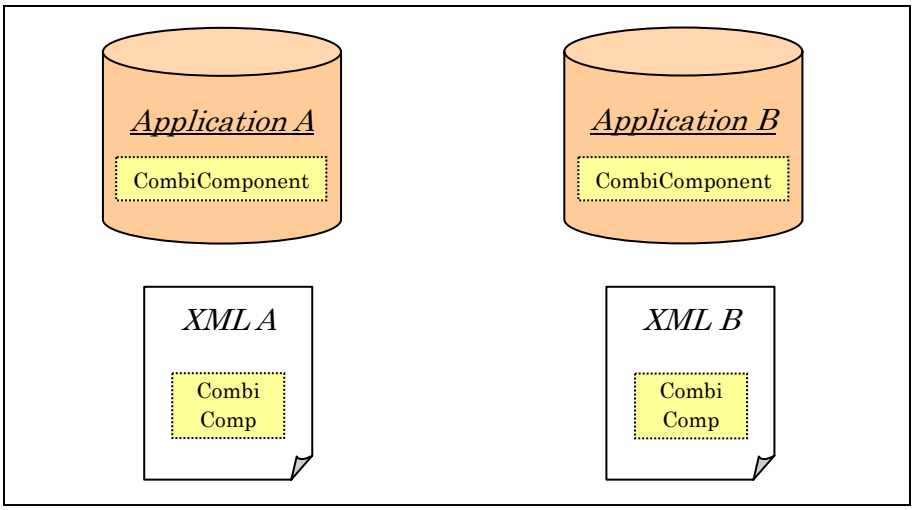
通常、複合コンポーネントはアプリケーション内に含まれてしまうため、複合コンポーネントをアプリケーションと独立した一つのコンポーネントとして扱うことが困難です。それを回避するために、複合コンポーネントの外部参照化、という機能を提供します。

7.5.1. 複合コンポーネント外部参照の考え方

アプリケーション内における複合コンポーネントのシリアライズの形態として以下の2形態を提供し、その使い分けについては複合コンポーネントの属性を操作することによって利用者が設定可能です。

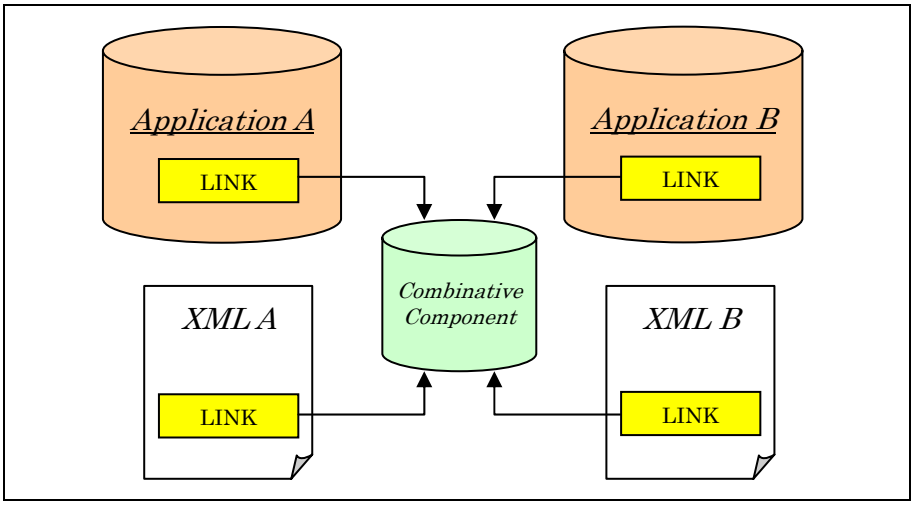
1)アプリケーションの一部として扱う

従来のプラットフォームの保存形式。複合コンポーネントを含むアプリケーションを保存した場合、アプリケーションデータに内包される形で保存される。この場合、複合コンポーネントがアプリケーション内に取り込まれてしまうため、アプリケーションと複合コンポーネントを並行して開発するといった独立性がなくなってしまう。ただし、複合コンポーネントがそのアプリケーション固有のものであれば、この形式でも構わない。



2)複合コンポーネントを独立したファイルで扱う

複合コンポーネントを含むアプリケーションを保存した場合、アプリケーションデータ内にはその複合コンポーネントが保存されているデータへの参照情報（ファイル名）が保存され、その実体は保存されない。これによって、ある複合コンポーネントを複数のアプリケーションで使用する場合に複合コンポーネントのみを変更すればその変更がすべてに反映される、といった共有のための仕組みが実現できる。



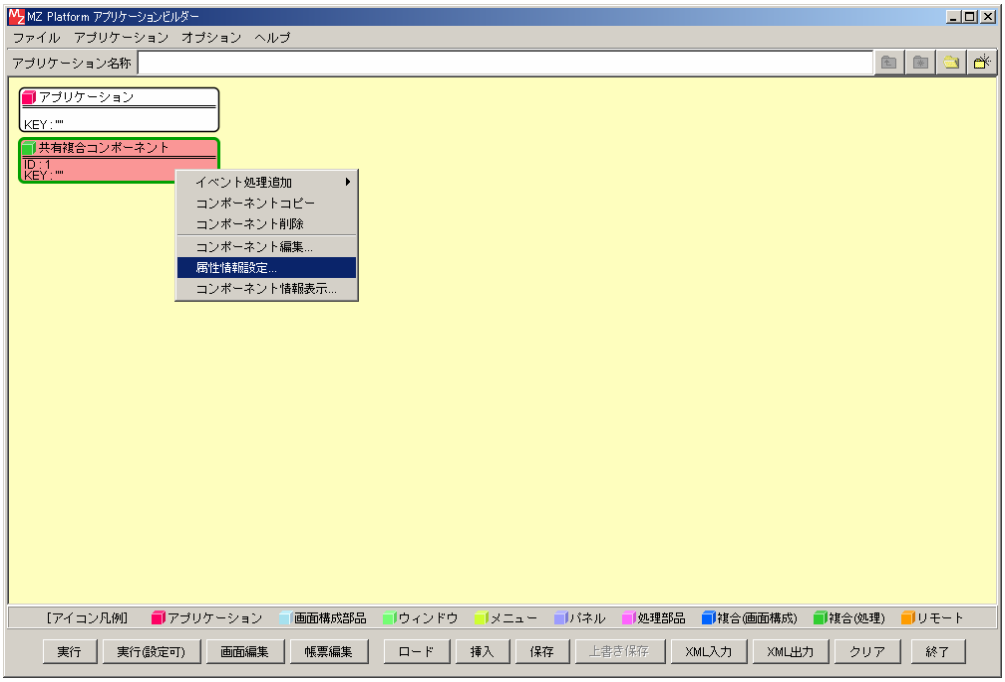
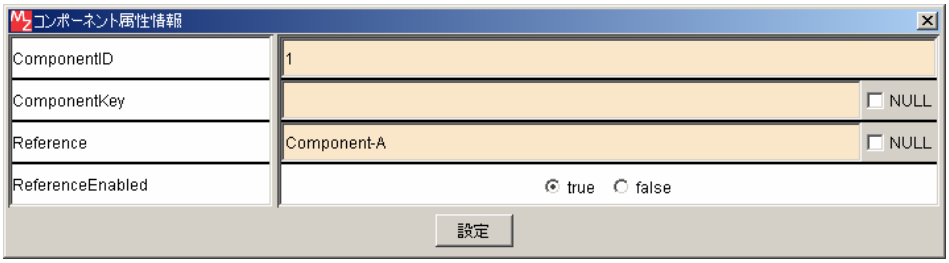
7.5.2. 複合コンポーネントの外部参照ファイル

外部参照化された複合コンポーネントの保管場所は初期設定ファイル（Platform.ini）に設定するものとし、ロード時にはそのフォルダから複合コンポーネントをロードします。ただし、このフォルダへのデータファイル保管はアプリケーション保存時に自動で行われるわけではないため、対象となる複合コンポーネントを単独でこのフォルダに保存する操作が必要となります。

```
◇Platform.ini
  ・・・・・・
  CombinativeComponentsFolder=<外部参照複合コンポーネント保存先フォルダ名>  ← 新規追加設定
```

7.5.3. 外部参照設定方法

複合コンポーネントの外部参照化設定は、以下の手順で行います。

画面	アプリケーションビルダー メイン画面
手順	①外部参照化対象の複合コンポーネントの属性編集を指示 
	②以下の属性に値を設定する ReferenceEnabled：外部参照フラグ（true：外部参照／false：非外部参照） Reference：外部参照ファイル名（先述の保管場所からの相対パス） 

7.5.4. XML 出力機能におけるのパスワードロックと外部参照設定

パスワードロックされた複合コンポーネントは、XML 出力することで内部情報が容易に見えてしまうため、パスワードロックされた複合コンポーネントを含むアプリケーションについては、XML 出力できないようにガードされています。

ただし、パスワードロックされている複合コンポーネントが外部参照設定されている場合は、内部情報が見られないため、XML 出力が可能となります。内部情報を隠すためにパスワードロックしたアプリケーションの XML 出力には、外部参照設定機能を利用してください。

7.5.5. 外部参照化されたデータファイル名

外部参照化された複合コンポーネントのデータファイル保管場所は一つであるため、データファイル名はできるだけ他と重複しない識別しやすいものとしてください。

また、複合コンポーネントの外部参照名を設定する際、存在しないファイルや間違ったものを指定するといった不注意があると、アプリケーションデータが正しく復元できなくなってしまいます。特に、外部参照先が自分自身になっているなどといった参照のループが設定されてしまった場合、ロード処理が終わらなくなってしまいます。設定時にファイル名をよく確認するとともに、間違えにくいファイル名にするようにご注意ください。

8. リモートアプリケーションとの連携

8.1. データ連携機能

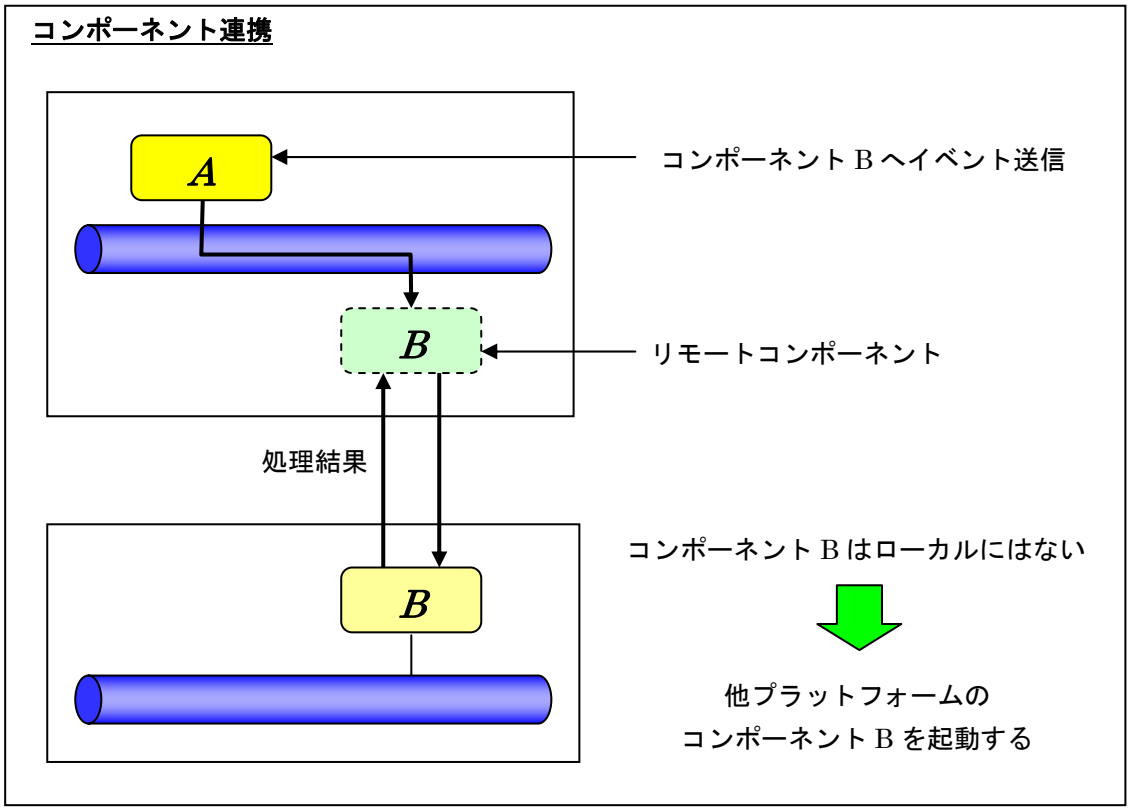
データ連携とは、複数のプラットフォームが互いに連携して処理を行う機能であり、具体的には以下の 2 つの機能があります。

[コンポーネント連携] 他のプラットフォーム上のコンポーネントを遠隔利用する

[コンポーネント転送] 他のプラットフォーム上のコンポーネントを転送

コンポーネント連携は、他のプラットフォーム上にあるコンポーネントを、あたかもローカルのプラットフォーム上にあるかのように扱える機能です。アプリケーション上は、他のプラットフォーム上に存在するコンポーネントをリモートコンポーネントとして扱います。

連携先のコンポーネントの処理結果はリモートコンポーネントに返され、その情報（処理の終了ステイタスやデータなど）はリモートコンポーネントから発生するコンポーネント連携処理結果通知イベントによって取得可能です。



コンポーネント転送は、通常のコンポーネントと同様にコンポーネントを追加し、メソッドの呼び出しによって処理を設定します。

＜注意事項＞

データ連携機能を利用するには、そのための設定が必要です。詳細は、インストール CD に収められている"データ連携導入手順書.pdf"をご覧ください。

8.2. アプリケーション構築方法

コンポーネント転送は、通常のアプリケーション構築と全く同じであるため、ここではコンポーネント連携機能について説明します。アプリケーションビルダーでは、他プラットフォーム上のものを含めて全てのコンポーネントを、その存在場所を気にせずに呼び出すことができます。

8.2.1. コンポーネント連携機能の準備

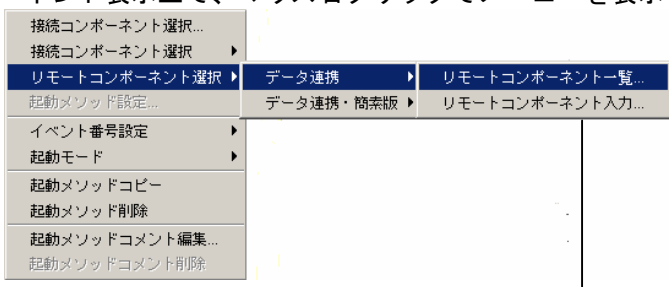
コンポーネント連携を使用するには、初期設定ファイル上でこの機能を使用することを宣言する必要があります。Platform.ini で以下のように設定されている場合に限り、コンポーネント連携を行うアプリケーションを構築することができます。

UseDataCooperation=true	True の場合のみ連携機能が使用可能
BrokerAddress=Broker-A	接続先ブローカ名を設定（簡素版では不要）
PlatformName=Platform-A	プラットフォーム名を設定
LocalhostAddress=xxx.yyy.com	プラットフォーム起動マシンのアドレス（サーバ版では不要）

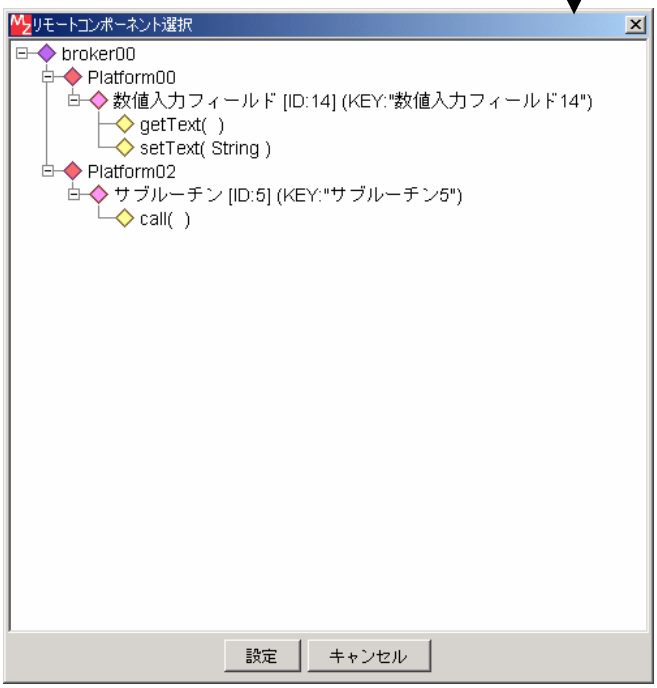
8.2.2. コンポーネント連携の設定

1)一覧からの選択（サーバ版）

画面	アプリケーションビルダー メイン画面
手順	①接続先コンポーネント表示上で、マウス右クリックでメニューを表示

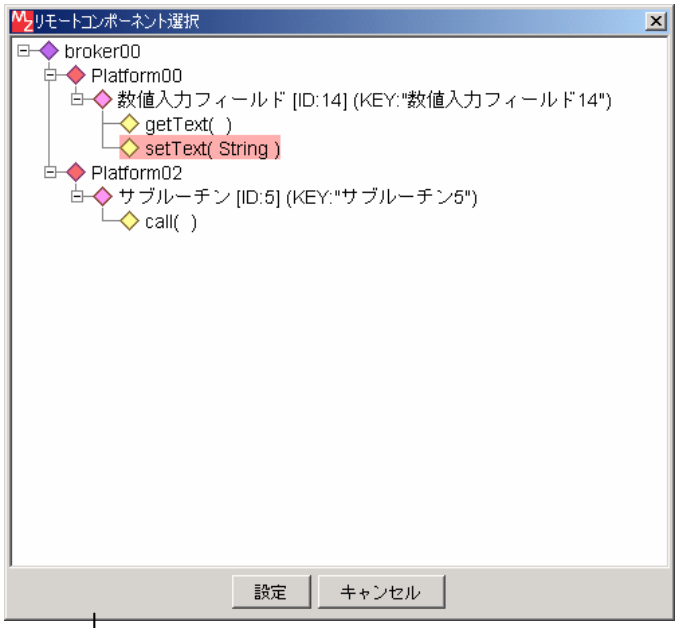


レジストリサーバに登録済みの
コンポーネント一覧が表示される

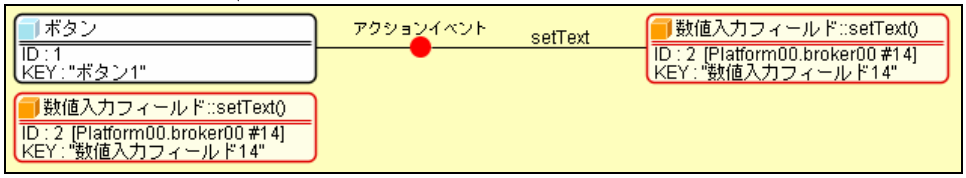


1)一覧からの選択（サーバ版） 続き

手順	②表示された一覧から連携先のメソッドを選択

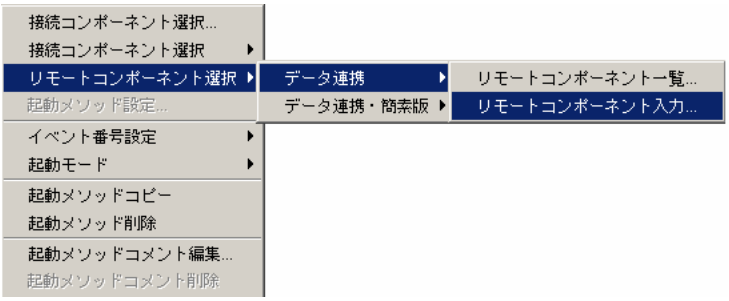
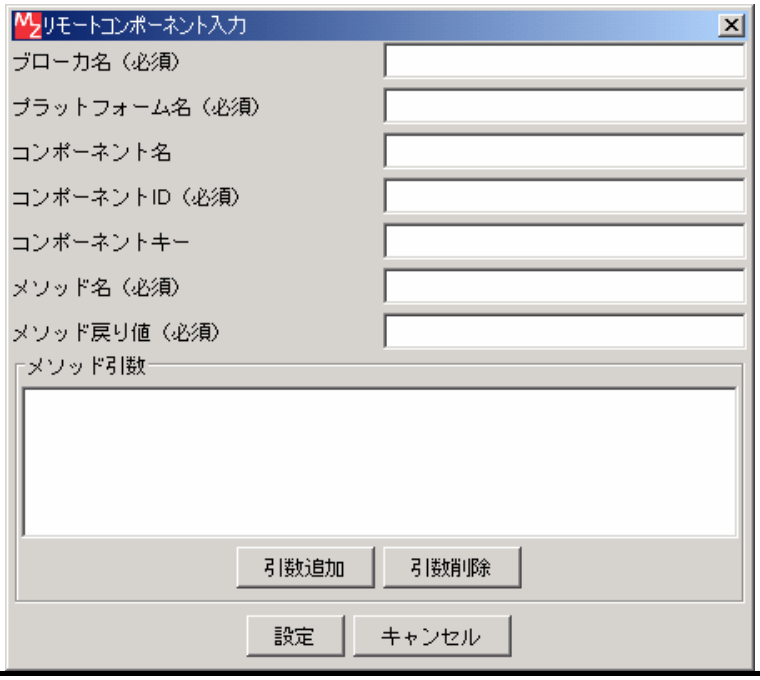


リモートコンポーネントが追加される



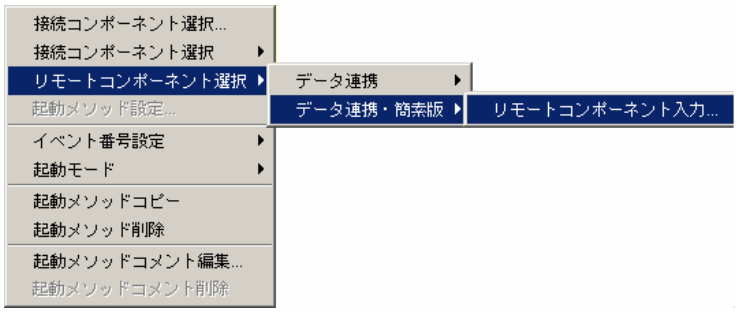
	③通常の手順でメソッド起動設定を行う

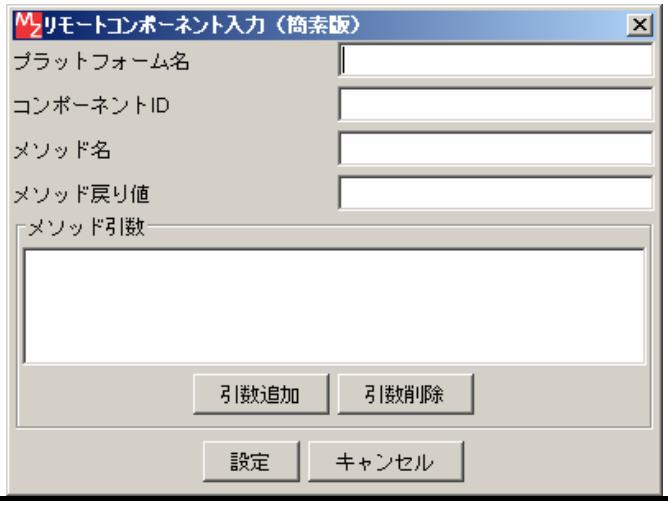
2)連携先情報の入力（サーバ版）

画面	アプリケーションビルダー メイン画面
手順	①接続先コンポーネント表示上で、マウス右クリックでメニューを表示 
	②表示された入力画面に連携先のメソッド情報を入力 

3) 連携先情報の入力（簡素版）

画面	アプリケーションビルダー メイン画面
手順	①接続先コンポーネント表示上で、マウス右クリックでメニューを表示
	②表示された入力画面に連携先のメソッド情報を入力

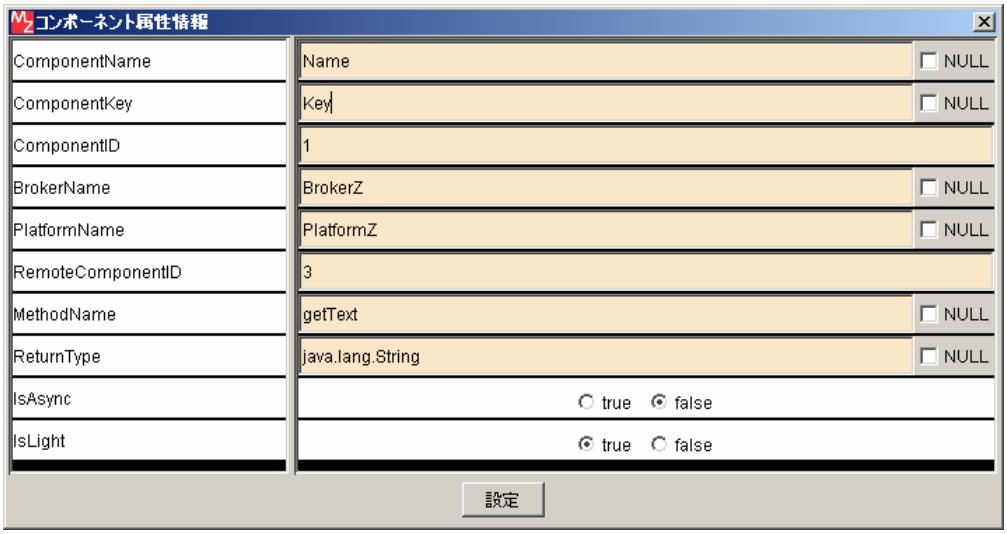




8.2.3. コンポーネント連携の属性設定

左側に表示されるリモートコンポーネントの属性を編集することで、連携先のコンポーネントの ID とそれが存在するブローカ名・プラットフォーム名などを変更することができます。また、コンポーネント連携を同期的に実行するか非同期的に実行するか（属性：IsAsync）、サーバ版を利用するか簡素版を利用するか（属性：IsLight）を設定することもできます。

これらの属性は通常の属性編集画面上で実行可能です。



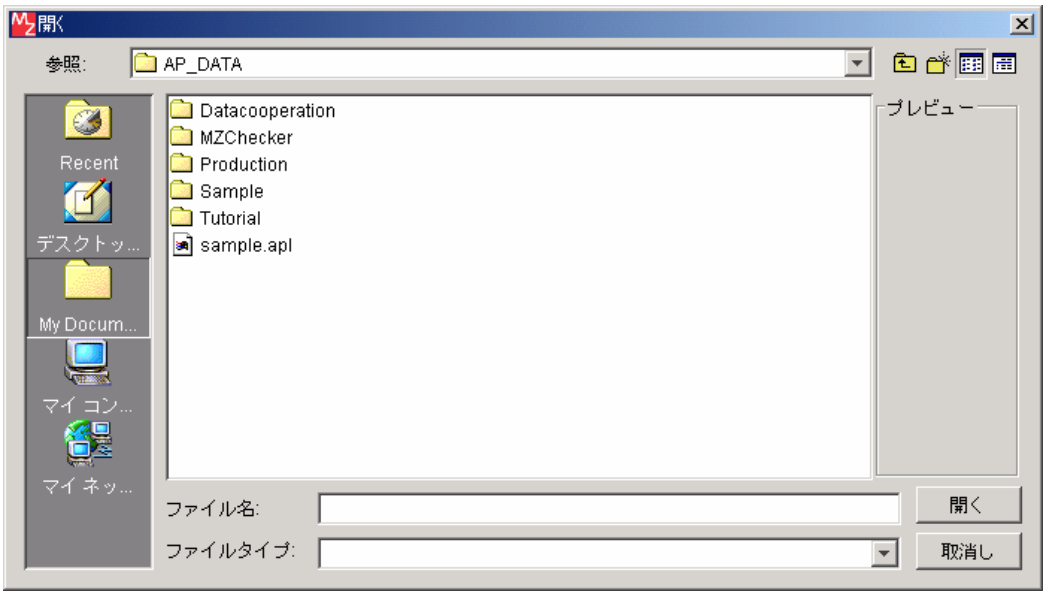
9. アプリケーションの実行（アプリケーションローダー）

構築されたアプリケーションは、アプリケーションローダーから実行します。アプリケーションの実行はアプリケーションビルダーからもできますが、アプリケーションを変更する必要がない場合、または変更させたくない場合、アプリケーションローダーを使用して実行します。

稼働環境構築を行った上で、スタートメニューからアプリケーションローダーを起動します。

[スタート] －[（すべての）プログラム]－[MZ Platform 1.6]－[アプリケーションローダー]

また、実行中にコンソールを表示させたい場合は、“アプリケーションローダー（コンソール)”を実行します。下のようなファイル選択画面が表示されますので、アプリケーションを選択します。



アプリケーションローダーの実体は、“導入フォルダ¥1.6 ¥PFLoader.exe”です。このプログラムには実行時引数が指定でき、アプリケーションファイル名を指定することで、ファイル選択の操作を省略することができます。

◇引数なし

ファイル選択ダイアログが表示され、そこで指定されたファイルをロードして起動します。

◇引数あり

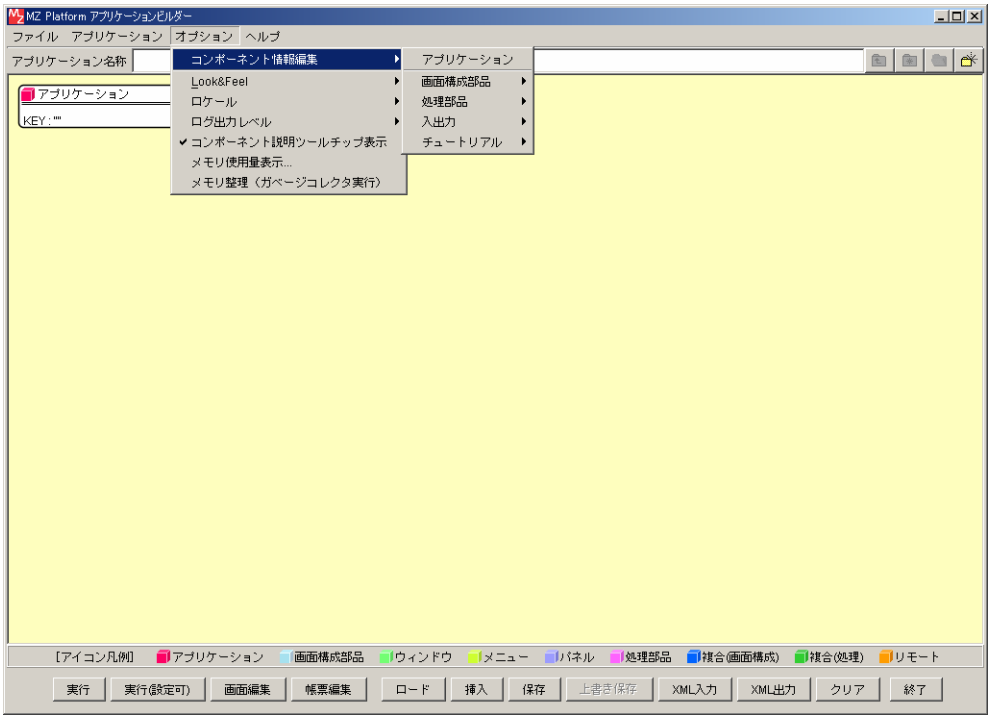
引数で指定したファイルからアプリケーション情報をロードして、起動します。

実行形式： PFLoader <アプリケーションファイル名>

10. コンポーネント情報の編集

アプリケーション構築時の操作支援として、アプリケーションビルダー上にコンポーネントのメソッド情報やイベント情報が表示されます。コンポーネント情報は、XML テキストファイル形式で実行環境内に提供されます。この情報は利用環境で変更が可能ですので、必要に応じて編集してください。

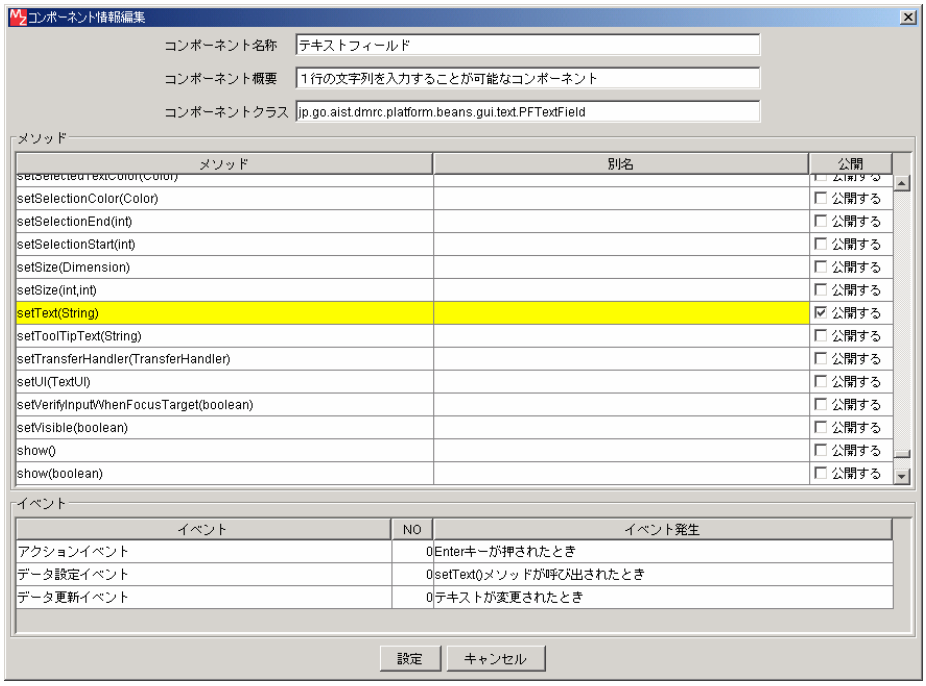
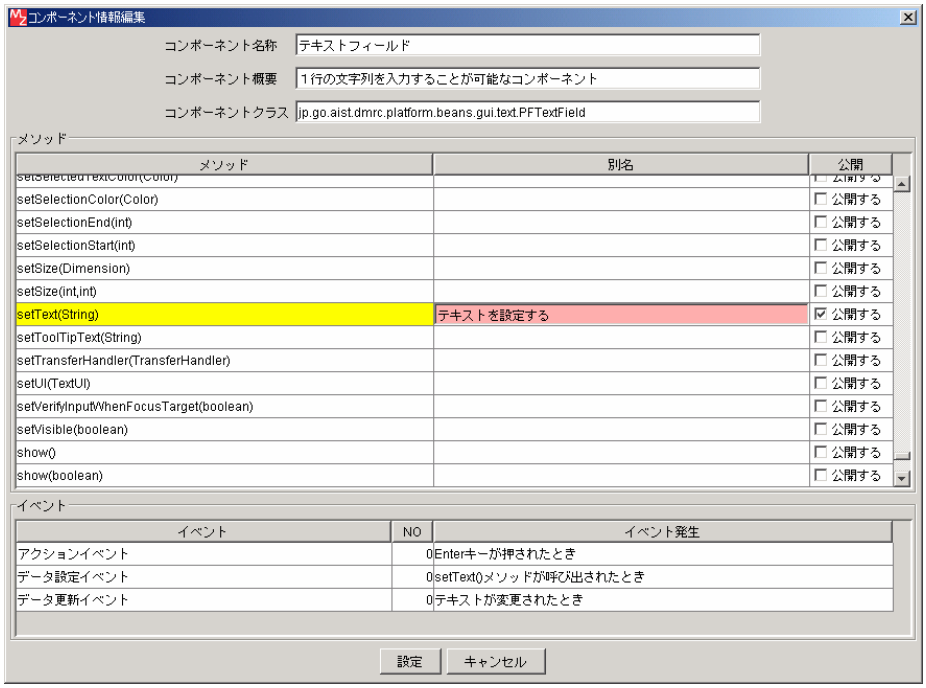
コンポーネント情報の編集操作は、アプリケーションビルダーから起動されるコンポーネント情報編集ツールによって行います。

画面	アプリケーションビルダー メイン画面
手順	①メニューバーから［オプション］－［コンポーネント情報編集］を選択 
	②編集対象コンポーネントの選択により、編集画面が表示される

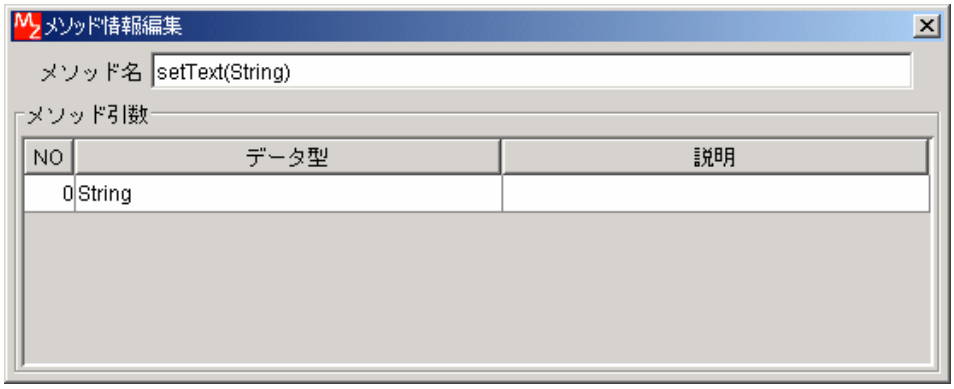
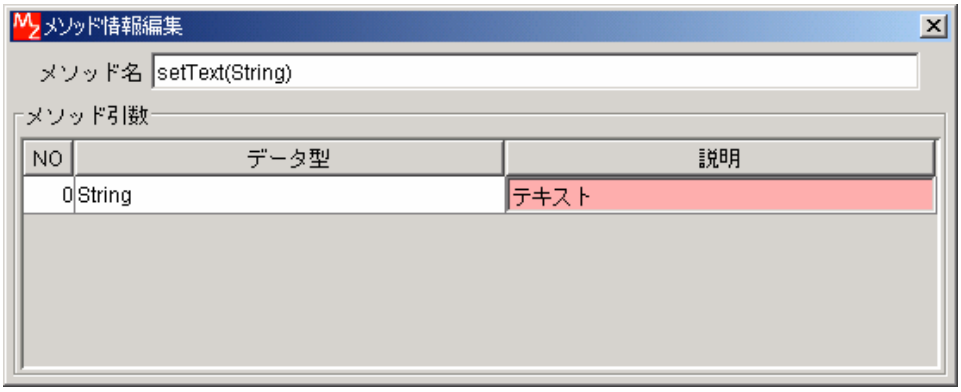
10.1. メソッド情報の設定

アプリケーション構築時に表示されるコンポーネントのメソッド情報は、すべてこのメソッド情報設定にて登録された情報です。不要なメソッドを表示させないようにしたり、メソッドの処理内容を設定したりすることで、利用者独自の実行環境構築が可能です。

10.1.1. メソッドの公開設定

画面	コンポーネント情報編集画面
手順	①公開したいメソッドの“公開する”チェックボックスにチェックを入れる → 該当メソッド行が黄色で表示される 
	②メソッドの表示文字列を“別名”セルに設定する （セルをダブルクリックすることで入力可能） 

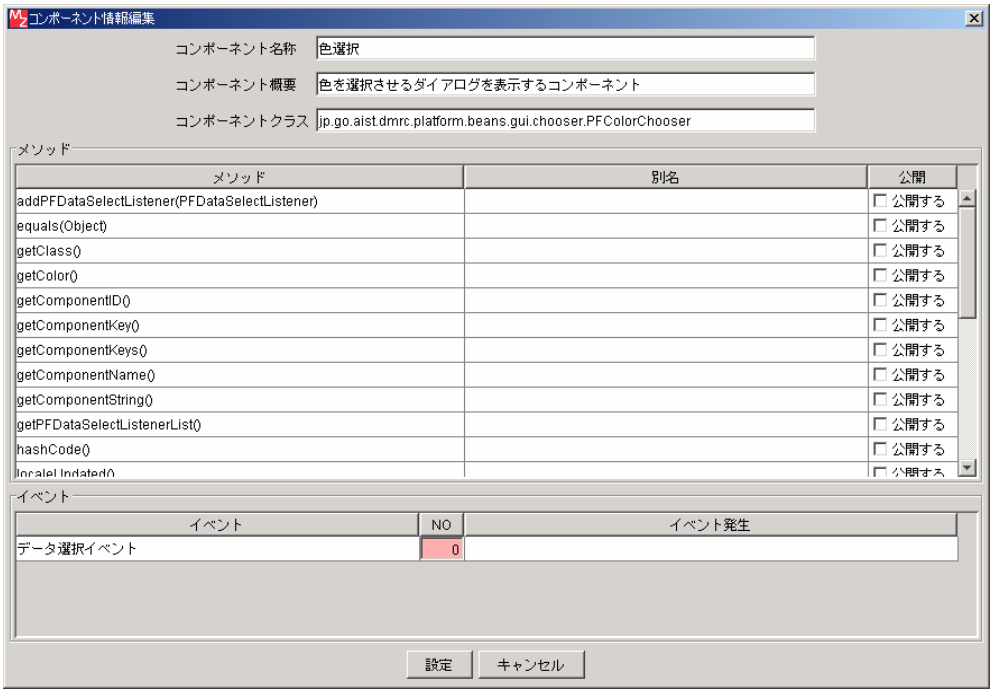
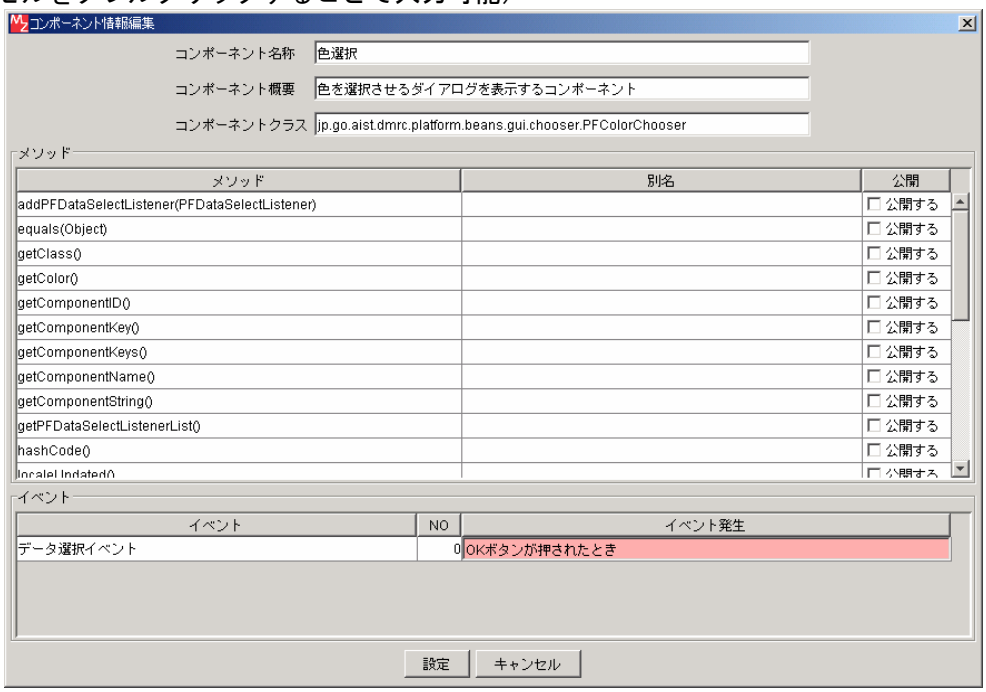
10.1.2. メソッド引数の設定

画面	コンポーネント情報編集画面
手順	①該当メソッドのメソッド名をマウスで左クリック → メソッド情報編集画面が表示される 
	②各引数に対する説明文字列を“説明”セルに設定する （セルをダブルクリックすることで入力可能） 
	③右上の[×]ボタンで終了（確定）

10.2. イベント情報の設定

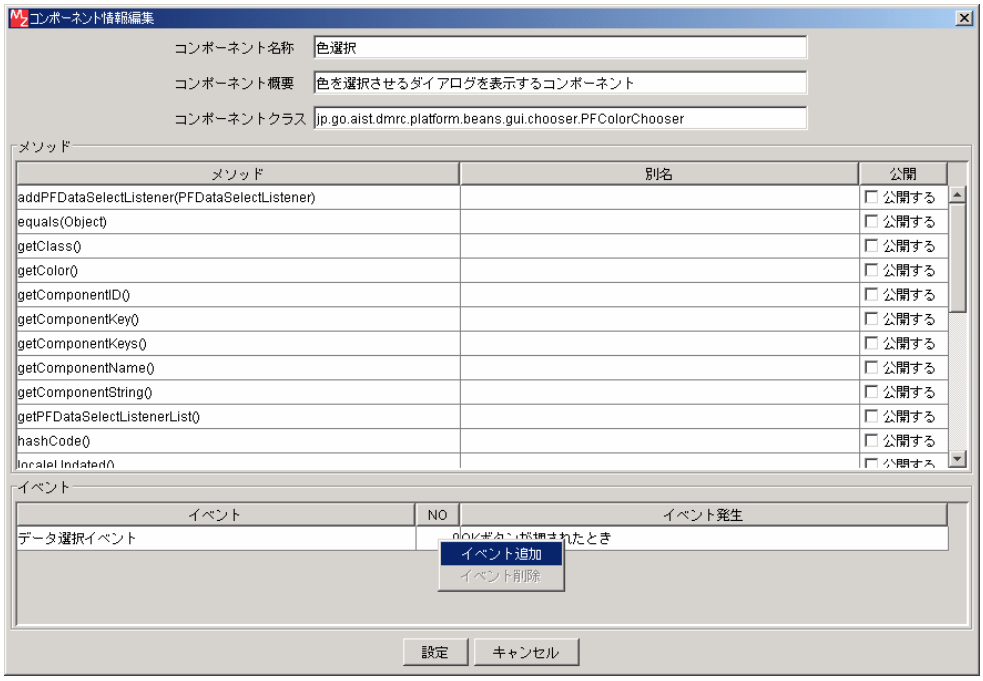
アプリケーション構築時に表示されるイベント情報は、すべてこのイベント情報設定にて登録された情報です。イベント番号やイベント内包データの説明が設定可能です。

10.2.1. イベント番号の設定

画面	コンポーネント情報編集画面
手順	①設定するイベント番号を、“NO”セルに入力する （セルをダブルクリックすることで入力可能） 
	②イベントの発生トリガーの説明を、“イベント発生”セルに入力する （セルをダブルクリックすることで入力可能） 

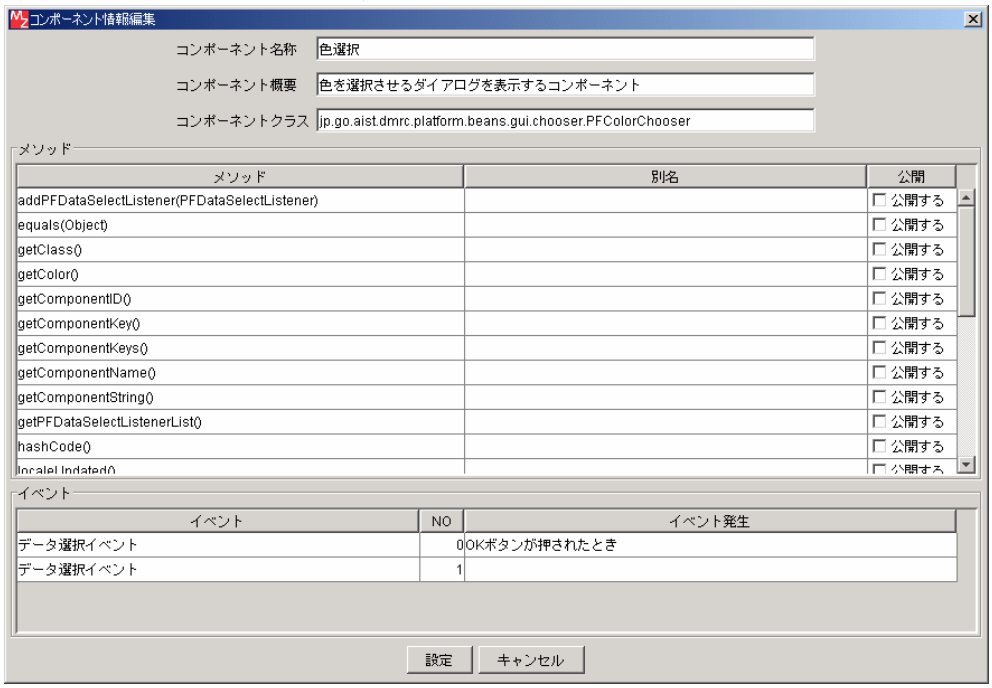
10.2.2. イベント番号の追加

画面	コンポーネント情報編集画面
手順	①追加するイベントの“NO”セル上でマウス右クリックし、[イベント追加]を選択



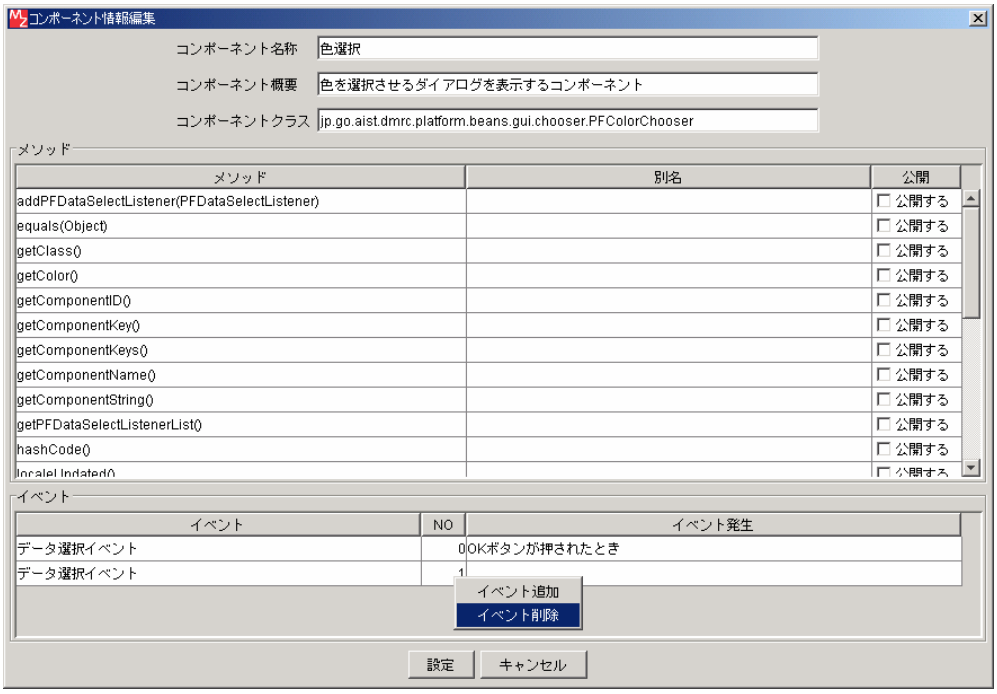
②追加したいイベント番号を入力
（入力しない場合は未入力のまま、または取消しボタン押下）

イベント番号が追加される

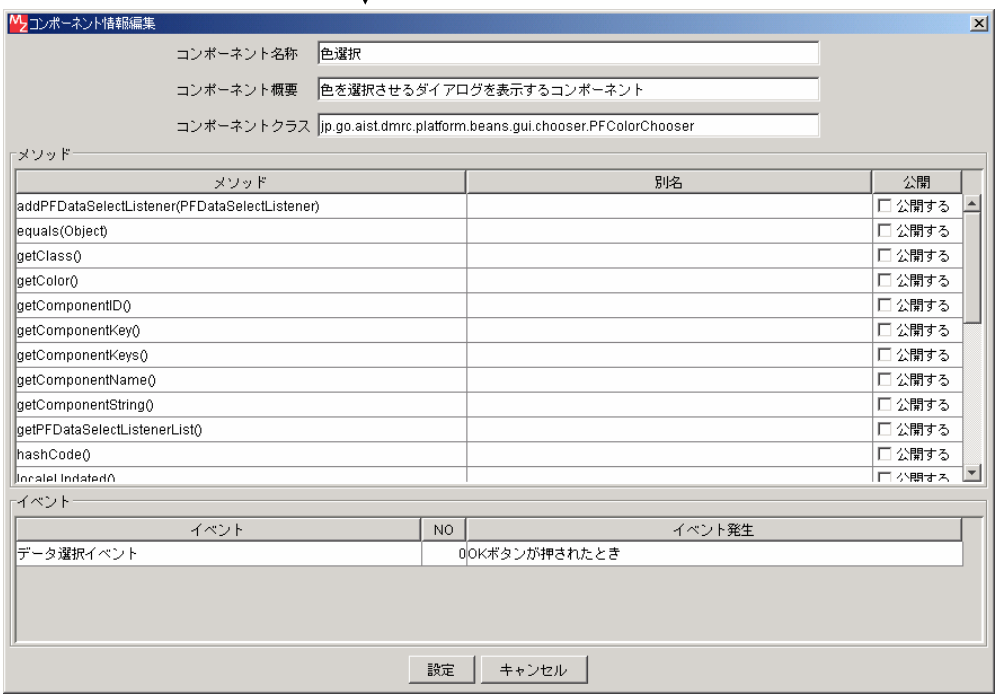


10.2.3. イベント番号の削除

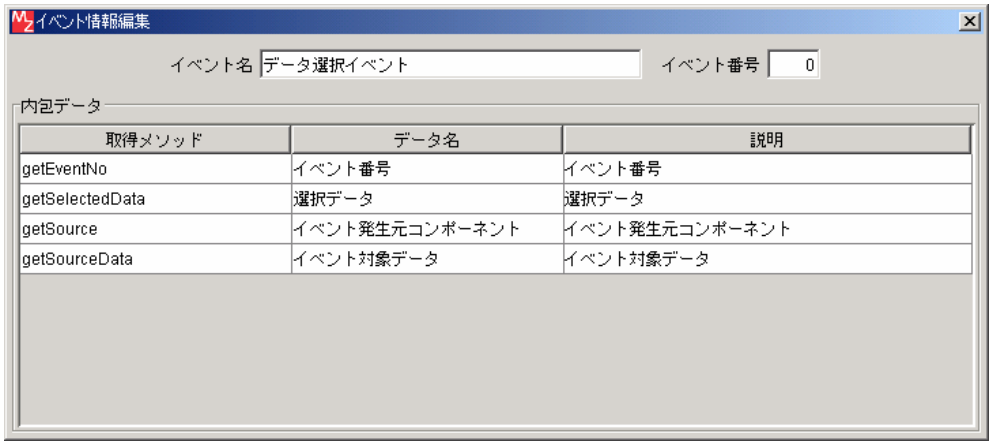
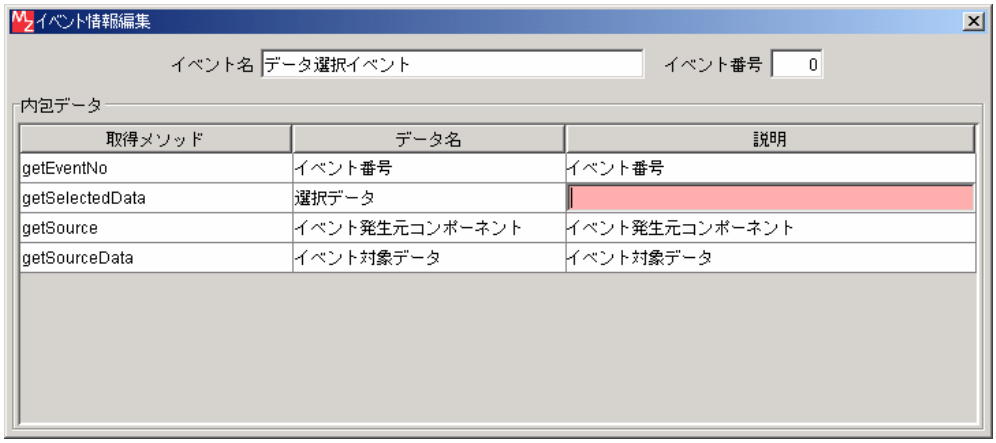
画面	コンポーネント情報編集画面
手順	削除するイベント番号の“NO”セル上でマウス右クリックし、[イベント削除]を選択



イベント番号が削除される



10.2.4. イベント内包データの設定

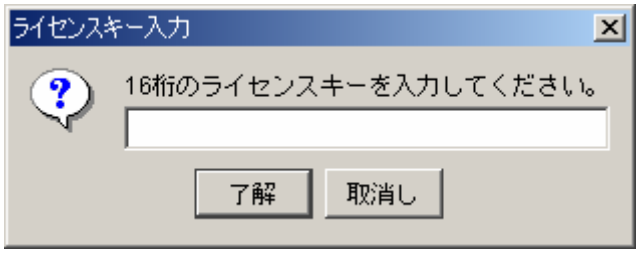
画面	コンポーネント情報編集画面
手順	①設定するイベント番号の“イベント”セル上でマウスを左クリックする → イベント情報編集画面が表示される ※イベント情報が登録されていない場合、下図に示す説明がデフォルトとして表示される。 
	②イベント内包データの説明を、“説明”セルに入力する デフォルトでは、データ名と同じ文字列が入っている （セルを左ダブルクリックすることで入力可能） 
	③右上の[×]ボタンで終了（確定）

11. アプリケーションのライセンス管理

MZ Platform 上で構築されたアプリケーションには、有償アプリケーションや期間限定体験版などライセンスを必要とするアプリケーションが提供されることがあります。これらはアプリケーションデータとともに、ライセンスキー文字列が提供されます。

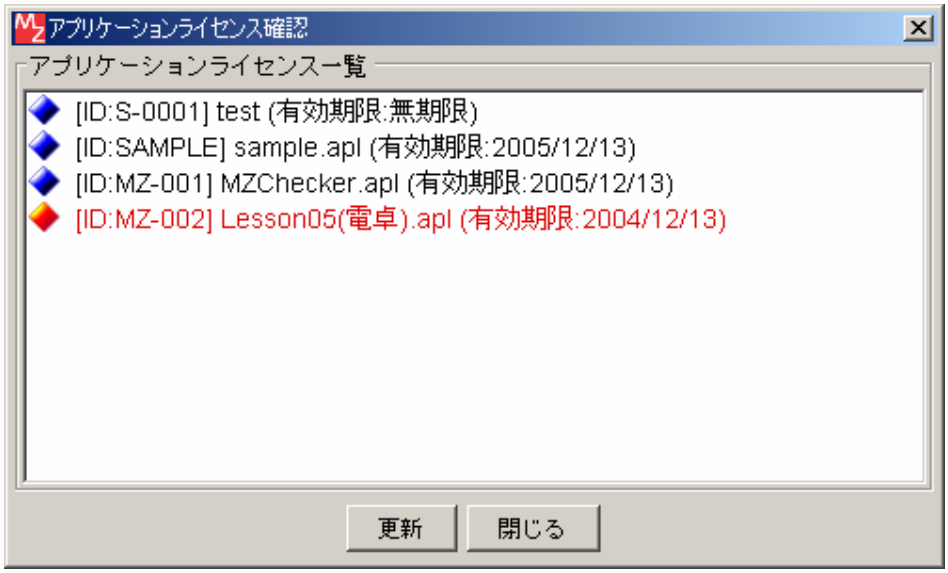
1)アプリケーションライセンスの登録

ライセンスが必要なアプリケーションを使用する際には、アプリケーションビルダーでもアプリケーションローダーでも初回使用時に以下のライセンスキー入力画面が表示されます。アプリケーション提供側から提示されたライセンスキー文字列を入力してください。一度入力したライセンスは保存され、2 回目以降の使用時にはライセンスキー入力は不要となります。



2)アプリケーションライセンスの確認／更新

現在登録されているアプリケーションライセンスは、アプリケーションビルダー上で確認することが可能です。また、有効期間を延長した新たなライセンスキーを入手した場合、ライセンスの更新を行うことが可能です。

画面	アプリケーションビルダー メイン画面
手順	①メニューバーから［ヘルプ]－[アプリケーションライセンス情報］を指定する → 登録済みのライセンスが有効期限とともに表示される 
	②更新対象のアプリケーションをダブルクリックするか、選択後に[更新]ボタンを押下 → 新たなライセンスキーを入力する画面が表示される