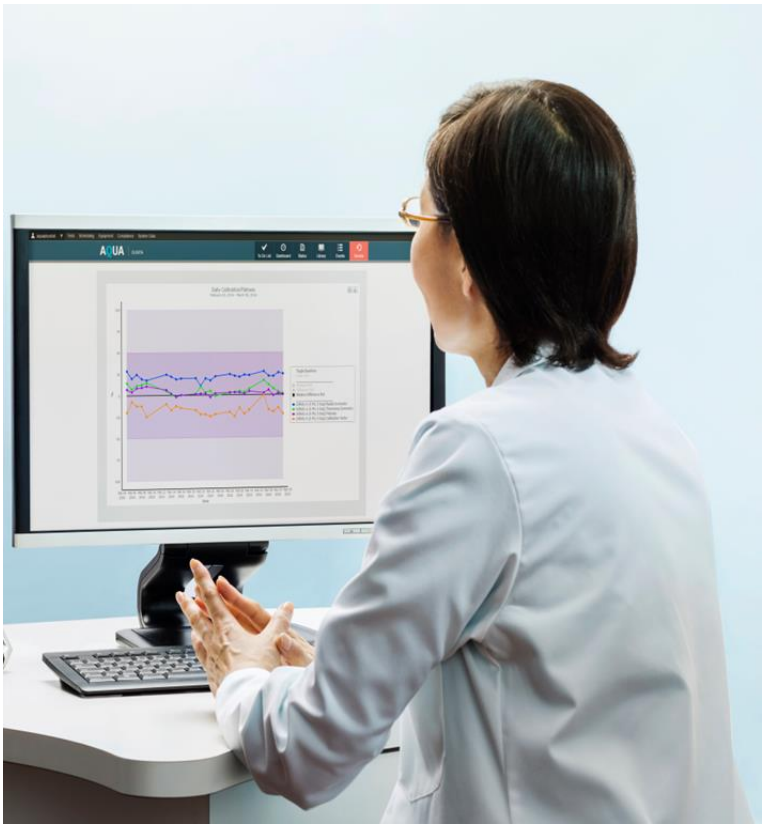
AQUA

テスト作成ガイド　AQUA 3.00



Document ID : E013338 / 03
Publication date: 2021-08-15

著作権について

商標について

エレクタ製品の商標はすべてエレクタに帰属します。

他の商標の承認

エレクタは、本書で使用している他社の商標および登録商標を認めます。

参考文献

エレクタは、本書で言及しているすべての文書を機器に同梱しているわけではありません。エレクタは、どの文書を機器に添付するか決定する権利を留保します。

連絡先情報

 MANUFACTURER Elekta, Inc. 1450 Beale Street Suite 205 St. Charles, MO 63303 USA Phone: +1.800.878.4267	**European Union Authorized Representative** EC REP Elekta, Inc. Kungstensgatan 18, Box 7593 SE-103 93 Stockholm Sweden Phone: +46 8 587 254 00 Fax: +46 8 587 255 00 eu_ar@elekta.com	**Australia Sponsor Information** Elekta, Inc. is the sponsor of this device. All inquiries in relation to this product to be made to the following: Sales.Ausnz@elekta.com **Elekta, Inc.** Suite 10.02, Level 10 16 Arthur Street North Sydney NSW 2060 +61 (0) 2 8907 1800 A.B.N. 49 109 006 966
TECHNICAL SUPPORT Contact your local Elekta representative for technical support or visit www.elekta.com.		

目次

1. AQUA テスト作成するにあたって

1.1 注意事項

テストの作成には細心の注意が必要です。カスタムテストでは、システム・クラッシュやデータ破損を引き起こす可能性があります。ユーザー定義テストは、ユーザーのニーズに基づくテストのカスタマイズを容易にするため、または臨床以外のマシン品質テスト目的のためのものです。実装時（新規テストの場合）および各システムアップグレード後（既存テストの場合）に、当該テストの正確性、正しさ、および機能性を検証する全責任は、カスタム QA テストの作成者が負います。新規および既存のカスタムテストの検証は、管理者である AQUA ユーザーがテストの状態をドラフトからファイナルモードに変更したときに確認されます。

1.2 作成前に知っておくべきこと

AQUA テスト作成では、パーソナライズされたテストを作成できます。この機能は多機能であり、利便性が高いため、管理者である AQUA ユーザーは、カスタマイズされたテストを作成、使用、維持する際には注意が必要です。

警告 1 

カスタマイズされたテストは、人間の判断に取って代わるものではありません。臨床使用前に、カスタマイズされたテストの安全性と有効性を検証し、妥当性を確認してください。

警告 2 

ユーザー名とパスワードをアクセス可能な場所に保存しないでください。この行為により、ユーザー検証が開示される可能性が高まり、臨床データに対する不正アクセスや悪意のある操作につながる可能性があります。

警告 3 

AQUA テスト作成の汎用性により、複数の方法を使用してパラメータに値を割り当てることができます。臨床で使用する前に、AQUA で割り当てられたすべての値が正しいことを検証する必要があります。

1.3 テスト作成対象者

AQUA Test Creation では、新しいテストを柔軟に作成できるだけでなく、各施設のニーズに合わせてテストをカスタマイズすることもできます。また、ワークフローの確立と合理化にも役立ちます。ただし、XML スクリプトと Java プログラミングの知識が必要です。

AQUA Test Creation を使用するには、これらの分野に精通している必要があります：

- AQUA マシン QA 管理システム
- マシン QA の地域／施設別プロトコル、標準、およびワークフロー
- マシン QA に関する安全性の問題

注意：テスト XML スクリプトを作成または修正する前に、関連資料を読 み、資料に記載されている安全関連のメッセージを理解してください。

1.4 バージョン互換性

注意：Elekta は、AQUA の現在のバージョンのユーザー定義テストが将来のリリースと互換性があることを保証しません。ただし、上位互換性のためにあらゆる試みを行います。

2. AQUA テスト作成の概要

2.1 AQUA テスト作成について

すべての AQUA テストは、XML スクリプトによって定義されます。施設要件に応じて、新規または修正したテスト XML スクリプトをアップロードすることで、テストを作成または修正できます。ユーザー定義のテストは、AQUA ユーザーインターフェイスのテスト XML スクリプトエディタを使用して修正することもできます。複雑なアルゴリズムについては、Java プラグインの形式で関数を実装し、その関数を新しいテスト XML スクリプトで使用できます。
テストの作成は AQUA の基本ライセンスに含まれており、追加のライセンスは必要ありません。

注意: ビーム処方を使用した自動画像ベースのテストは、テスト作成の一部ではありません。

3. XML テスト作成ガイド v3.0

3.1 XML テストスクリプト作成ガイド

このドキュメントでは、XML テスト スクリプトのセクションと要素について説明します。また、AQUA のテストスクリプトの高度な機能の設定情報についても説明します。このドキュメントの情報は、AQUA 3.00 に適用されます。

3.1.1 スクリプト概要

XML テスト スクリプトには、AQUA でテストを定義するために必要なすべての情報が含まれています。 テストに関するメタデータ、ユーザーに表示されるフォーム、合格、不合格、または警告を生成するために使用されるテスト ロジックが保持されます。 XML スクリプトでは大文字と小文字が区別されることに注意してください。

メタデータ セクションには次の情報が含まれます。

- テストの説明
- TPI フラグと呼ばれる、テストがサードパーティのデバイスと統合されているかどうかを示す true/false フラグ
- テストを実行するときに使用される手順と仕様
- テストで使用されるコアパラメータのリスト
- パラメータ間の関係（ベースラインや許容値など）
- プロット特性

テスト スクリプトは、次に示すように、バージョンとエンコーディングを含む xml タグで始まる必要があります。この後に <test> ノードが続きます。<test> ノード内には、<metadata> ノード、<form> ノードがあり、その後にオプションでロジック要素と追加の <form> ノードが続きます。 テスト スクリプト ファイルが AQUA にロードされると、<metadata> ノードがスクリプトから削除されることに注意してください。 説明、手順、および仕様フィールドは、AQUA のテスト データベース エントリのそれぞれの部分に読み込まれます。

```
<?xml version="1.0" encoding="UTF-8"
<test>
```

```
    <metaData>
        <description> . . . </description>
        <tpi>...</tpi>
        <parameterSet> . . . </parameterSet>
        <parameterRelnSet> . . . </parameterRelnSet>
        <plotData> . . . </plotData>
        <procedure> . . . </procedure>
        <specification> . . . </specification>
    </metaData>
    <form> . . . </form>
    Logic block
    <form>. . . </form>
</test>
```

テストノードがサーバーによって処理されると、各ノードがリストの順序で実行されます。<form> ノードには通常、ユーザー入力を必要とするコントロール (数値フィールド、ボタンなど) が含まれます。ユーザーの操作が必要な場合、サーバーは UI を通じて提供される <form> ノードをクライアントに送信します。ユーザーがフォームに情報を入力し、制御をサーバーに渡すと、スクリプトの次のノードに実行が移ります。上に示したテストスクリプトは、ユーザーが情報を入力し、いくつかのロジックが実行され、結果がユーザーに返されるシナリオの典型的なモデルです。

サーバーは、UI を通じてレンダリングされる <form> ノードをクライアントに送信します。

3.1.2 メタデータ

3.1.2.1 <description>

<description>ノードにはテストを説明するテキストが記述され、最大 255 文字まで記述できます。<description>ノードの内容は空のままでも構いません。

3.1.2.2 <procedure>

<procedure> ノードには、テスト手順を説明するテキストが含まれます。<procedure> ノードの内容は、AQUA UI ではプレーンテキストとして表示されます。テキストは、AQUA UI に組み込まれている HTML テキストエディタを使用して更新および書式設定できます。テストに添付された画像への参照は、構文を使用して <procedure> ノードで定義できます： [Image name="fileName"]という構文を使用します。テスト変数の値には、次の構文を使用してアクセスできます：procedure テキストの@variableName という構文を使用します。

3.1.2.3 <specification>

<specification> ノードには、テストが正常に完了し、PASS 結果が得られるために必要な条件を記述したテキストが含まれます。<specification> ノードの内容は、AQUA UI ではプレーンテキストとして表示されます。テキストは、AQUA UI に組み込まれている HTML テキストエディタを使用して更新および書式設定できます。テストに添付された画像への参照は、構文を使用して <specification> ノードで定義できます： [Image name="fileName"]という構文を使用します。テスト変数の値には、構文を使用してアクセスできます： specification テキストの@variableName という構文を使用します。

注意：Elekta は、テスト手順や仕様で使用される安全でないハイパーリンクや画像ファイルによって引き起こされる損害については責任を負いません。

3.1.2.4 <parameterSet>

<parameterSet> ノードには、すべてのテスト パラメータを記述するメタデータが含まれます。アップロード処理では、このメタデータを使用して AQUA でのテスト定義を完成させます。パラメータのタイプには、test、input、output、equipment、calibration、source、reference、vector があります。これらのタイプの詳細については、セクション 3.1.3.6：テストパラメータの概要を参照してください。

例 メタデータセクション

```
<metaData>
  <description>Simple HT Timer Test.</description>
```

```
<procedure>Enter the HT Timer value. [Image name="HTTimerPicture"].
</procedure>
<specification>
HT Timer value must be within @failTolerance of the baseline: @baseline
</specification>
<parameterSet>
  <parameter>
    <type>test</type>
    <name>failTolerance</name>
    <label>Fail Tolerance</label>
    <format>numeric</format>
    <plottable>true</plottable>
    <description></description>
  </parameter>
  <parameter>
    <type>test</type>
    <name>baseline</name>
    <label>Baseline</label>
    <format>numeric</format>
    <plottable>true</plottable>
    <description></description>
  </parameter>
  <parameter>
    <type>input</type>
    <name>htTimerValue</name>
    <label>HT Timer Value</label>
    <format>numeric</format>
    <plottable>true</plottable>
    <description></description>
  </parameter>
</parameterSet>
</metadata>
```

3.1.2.5 <parameterReInSet>

<parameterRelnSet> ノードはパラメータ間の関係を定義します。

次の例では、「左」パラメータ、関係名、および「右」パラメータとしてマークされたパラメータとして構造化された入力パラメータのベースラインを識別します。ここで、左パラメータは右パラメータに対する関係名です。例えば、左側のパラメータ (radialSymmetryBaseline) の下には、右側のパラメータ (radialSymmetry) へのベースラインがあります。

```
<parameterReln>
  <left>radialSymmetryBaseline</left>
  <name>baseline</name>
  <right>radialSymmetry</right>
</parameterReln>
```

可能なパラメータ関係タイプ (<name>) は次のとおりです。

- Baseline (ベースライン)
- absolute upper warn tolerance（絶対上限警告許容値）
- absolute lower warn tolerance（絶対低限警告許容値）
- relative upper warn tolerance（相対的な上限警告許容値）
- relative lower warn tolerance（相対的な低限警告許容値）
- absolute upper fail tolerance（絶対的な上限の Fail 許容値）
- absolute lower fail tolerance（絶対的な下限の Fail 許容値）
- relative upper fail tolerance（相対的な上限の Fail 許容値）
- relative lower fail tolerance（相対的な下限の Fail 許容値）
- plot against（～に対してプロットする）
- property
- parameter map（パラメータマップ）

「absolute」tolerance は、許容値とベースラインが同じ単位を使用する場合に使用されます。

「relative」tolerance は、許容値がベースラインのパーセンテージとして指定される場合

に使用されます。
「Plot against」 この関係は、X 軸と Y 軸を定義します（‘X’ plost against ‘Y’）。

```
<parameterReln>
    <left>MLC leaf number</left>
    <name>plot against</name>
    <right>Y2 MLC bank 100</right>
</parameterReln>
```

「property」関係については、二次機器パラメータ（Secondary Equipment Parameters）のセクションで説明されています。「parameter map」の関係については、マップされたテストの設定の仕方のセクションで説明されています。

3.1.2.6 <plotData>

plotData 要素は、プロットビューの構成を記述します。パラメータに関連付けられたベースラインと許容値は、自動的にビューに含まれます。1 つまたは複数の plotView ノードを plotData 要素内で定義できます。

<type>「line chart」を使用して、パラメータを日付に対してプロットできます。

この例では、プロットビューの名前は Symmetry で、パラメータ RadialSymmetry と transverseSymmetry が含まれています。 これらのパラメータは、テスト実行日を X 軸にプロットされます。

```
<plotData>
    <plotView>
        <name>Symmetry</name>
        <type>line chart</type>
        <parameters>
            <parameter>radialSymmetry</parameter>
            <parameter>transverseSymmetry</parameter>
        </parameters>
```

```
    </plotView>
</plotData>
```

「by parameter」タイプを使用して、パラメータを相互にプロットできます。 この場合にプロットされるパラメータは「vector」タイプとして定義する必要があります。

```
<plotView>
    <name>Y1 MLC Bank 100</name>
    <type>by parameter</type>
    <default>false</default>
    <parameters>
        <parameter>Y1 MLC bank 100</parameter>
    </parameters>
</plotView>
```

3.1.3 フォーム要素

テストのレイアウトはフォーム要素によって定義されます。これらは、Controls、Presentation、Menu、Panel、Layouts の各グループに大別されます。これらの各グループに含まれる要素の詳細については、「XML Reference」のセクションを参照してください。入力フォームの典型的なレイアウトには以下が含まれます：

- 重要なテストパラメータのラベルと値のペア
- ユーザー入力用のテーブル/ラベルと値のペア
- 行の区切り
- メニュー（計算/キャンセル）

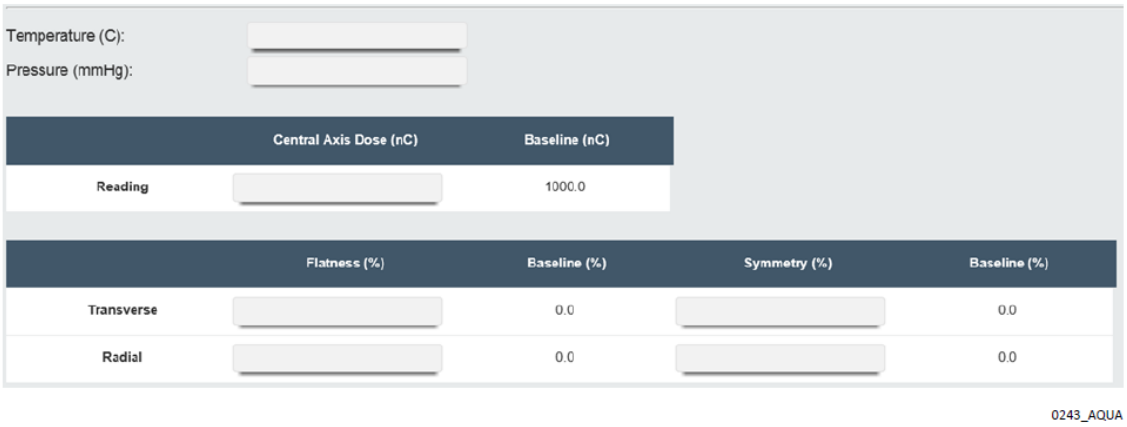


図 3.1 入力フォームの典型的なレイアウト

結果の典型的なレイアウトは、結果が色で強調表示された入力フォームのようなものです：

- 重要なテストパラメータのラベルと値のペア
- ユーザー出力と結果を示す表/ラベルと値のペア
- テストのステータス
- 行の区切り
- メニュー

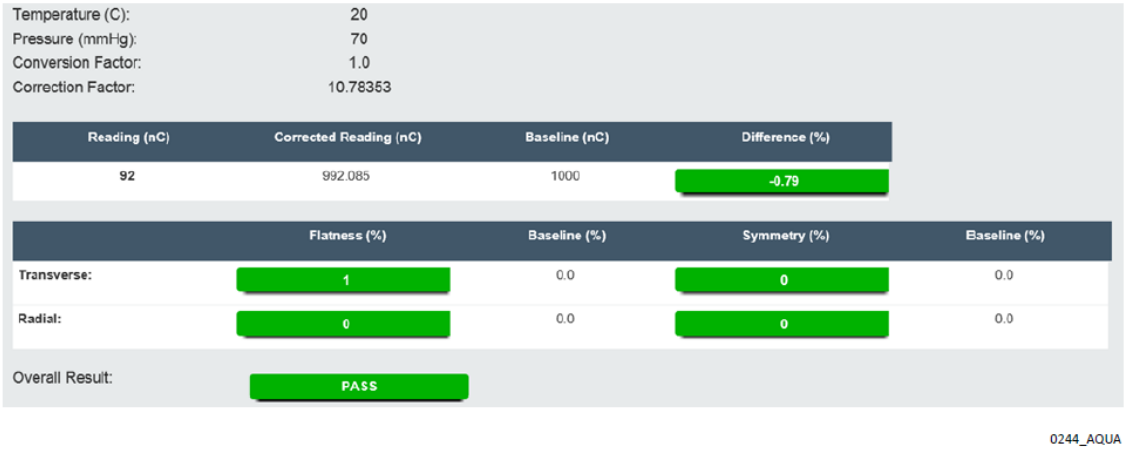


図 3.2 結果の典型的なレイアウト

いくつかの一般的な例を以下に示します。

例: **ラベルと値のペア**

```
<horizontal/>
    <comment value="Beam:" width="200"/>
```

```
    <comment value="@energy" width="200"/>
<vertical/>
```

Example: **テーブル**

```
<space/>
    <vertical startTable="" />
        <horizontal param="inputLine" />
        <space width="200"/>
        <comment value="Flatness (%)" width="200"/>
        <comment value="Baseline (%)" width="200"/>
        <comment value="Symmetry (%)" width="200"/>
        <comment value="Baseline (%)" width="200"/>
        <vertical/>
        <horizontal param="inputLine" />
        <comment value="Transverse: " width="200"/>
        <number name="transverseFlatness" validate="@double" width="200"/>
        <comment value="@transverseFlatnessBaseline" width="200"/>
        <number name="transverseSymmetry" validate="@double" width="200"/>
        <comment value="@transverseSymmetryBaseline" width="200"/>
        <vertical/>
        <horizontal param="inputLine" />
        <comment value="Radial: " width="200"/>
        <number name="radialFlatness" validate="@double" width="200"/>
        <comment value="@radialFlatnessBaseline" width="200"/>
        <number name="radialSymmetry" validate="@double" width="200"/>
        <comment value="@radialSymmetryBaseline" width="200"/>
        <vertical/>
    <vertical endTable="" />
<space/>
```

例:**メニュー**

```
<menu domain="SAVE,CANCEL" labels="Calculate,Cancel" name="status"/>
```

3.1.3.1 手順と仕様開示パネル

AQUA v3.0 では、手順と仕様開示パネルがテストフォームに自動的に表示されます。 以前のバージョンから AQUA v3.0 にアップグレードした場合、カスタマイズされたテストで手順と仕様開示パネルが 2 回表示されることがあります。
2 番目の開示パネルを削除するには、XML スクリプトを編集して対応するフォーム要素を削除する必要があります。通常、手順と仕様開示パネルは、入力フォームと出力フォームの上部に表示されます。 XML 構文は次のようになります。

```
<group name="procedureGroup"/>
    <comment value="@procedureContent" width="1260"/> <endGroup/>
<group name="specificationGroup"/>
    <comment value="@procedureSpecification" width="1260"/> <endGroup/>
```

手順と仕様開示パネルの 2 セットを表示するテストフォームの入力フォームと出力フォームから、このコードを削除するだけです。 この編集を AQUA UI で行う場合は、必ず [Save] ボタンをクリックして変更を保存してください。

3.1.3.2 フォームノード内での動的パラメータの使用

動的パラメータは、値を定義せずにテストスクリプトで使用されます。 実際には、変数の割り当ては暗黙的に実行されます。
たとえば、変数 transverseFlatnessBaseline は、スクリプト内で明示的に定義することも、次のように動的パラメータとして定義することもできます。

```
<comment name="transverseFlatnessBaseline" width="200" value="2.1" />
<comment value="@transverseFlatnessBaseline" width="200" />
```

3.1.3.3 フォーム以外のノード（ロジックブロック）での動的パラメータの使用

動的パラメータ値は、テスト実行時に対応する同名の変数に代入されます。実際、動的パラメータは、<set> 要素や <constant> 要素に相当します：

```
<constant variable_name="some_value" />
<set name="variable_name" value="some_value" />
```

たとえば、transverseFlatnessBaseline という動的パラメータは、次のようにテストスクリプトロジックで使用できます：

```
<set name="range">
    <subtract>
    <value name="transverseFlatnessBaseline" />
    <value name="transverseFlatness" />
    </subtract>
</set>
```

3.1.3.4 テスト完了のためのスクリプト

すべてのテスト スクリプトは、テストの最終ステータス (PASS、FAIL、WARNING のいずれか) を報告するフォームで終わる必要があります。 テストの最終結果を保持するために、「status」という特別な変数を使用されます。
さらに、ドメイン値 SAVE および CANCEL を含むメニュー要素を含める必要があります。
次のスクリプトのスニペット (テンプレート) は、すべてのテスト スクリプトで使用できます (変数 result と resultColour は、特定のテストで使用する変数に置き換えます)。

```
<horizontal />
<comment value="Overall Status: "width="200" />
<colourComment name=“colourComment1” width="100" value="@result"
colour="@resultColour" />
<hidden name="status" value="@result" />
<vertical/>
<menu name="status" domain="SAVE,CANCEL" labels="Save,Cancel"/>
```

メニュー要素のみが含まれている場合、[SAVE] を選択すると、常に PASS のテスト ステータスが生成されます。

3.1.3.5 ファイルのアップロード

管理者用[Tests] ウィンドウの File Upload ツールを使用すると、テストまたはテストのグループの XML スクリプトと、テスト手順または仕様に含める画像を含む zip ファイルをアップロードできます。 [ポリシー] オプションを選択すると、ポリシーもアップロードすることもできます。

zip ファイルには以下が含まれている必要があります。

- テスト用のすべてのファイルが含まれるフォルダー
- 各テストの XML テストスクリプト
- .jpg または .png 形式の関連画像
- TestList.txt という名前の単一のテキストファイル

TestList.txt ファイルには、各テストと画像ごとに、AQUA で表示されるテスト名/画像名 (Testname または ImgName) と XML/画像ファイル名 (XMLfilename または ImgFileName) をアスタリスク（*）で区切った行があります。

ImgName1*ImgFileName.jpg
Testname1*XMLfilename1.xml
Testname2*XMLfilename2.xml
など

テストスクリプトを処理する前に、TestList.txt ファイルで画像を参照し、テストスクリプトを処理する前に画像を AQUA に認識させることが重要です。

画像は、次の形式を使用してテストスクリプトの Procedure(手順)または Specification(仕様)セクションに埋め込むことができます:

[Image name="ImgName1"]

子テストは、TestList.txt の親テストの前にリストする必要もあります。

注意 : テストを含むファイルをアップロードすると、同じ名前の既存のテストが上書きされ、AQUA に新しいバージョンのテストが作成されます。

注意：zip ファイルの作成には 7-Zip の使用をお勧めします。

注意：AQUA のバージョン 3.0 では、各「colourComment」タグに一意の名前が必要です（XML Reference の <colourComment>を参照)。 次のエラーが表示される場合は、「colorComment」タグに「name」属性を追加し、zip ファイルを再度アップロードします。

Error: A <colourComment> tag is missing a 'name' attribute

3.1.3.6 テストパラメータの概要

テストパラメータは、実行時に提供されるか、テスト実行中に生成されるテストスクリプトの動的要素を指します。 テスト仕様のこのセクションは、テストの中核となる要素の全体像と、レポート作成時にこれらの要素がどのように扱われるかを示しています。 テストには、次のいずれかの <type> タグがあります:

- Equipment(機器): これらは、テストされるユニットの特性を指します。 複数の機器パラメータを使用したテストは、同じテストに対して、可能なテストインスタンスのデカルト積が得られます。
- Source(線源): 通常、線源強度テストなど、現在の放射線源に関する情報がテストで使用される HDR ユニットに使用されます。 ソース要素のリスト (データ例を含む) は次のとおりです。

表 3.1 線源要素とサンプルデータ

Source element（線源要素）	Example data (例)
Source Isotope（線源同位体）	Iridium-192
Source Half-Life（線源半減期）	73.827
Source Serial Number（線源のシリアル番号）	A233-D567
Source Reference（線源リファレンス）	48.01
Source Reference Date（線源参照日）	7/28/2013 08:00:00
Source Expiry Date（線源の有効期限）	10/28/2013

ソース要素がテストで使用される場合、テスト対象のユニットの現在のソースのみがテスト用に選択されます。ソース要素をテスト定義に含める場合は、スクリプトの変数名として

使用される一意の名前を含めます（例えば、線源同位体には sourceIsotope という名前を使用します）。

- **Calibration**（校正）: 二次機器からの校正係数を必要とするテスト。 利用可能なキャリブレーション要素は次のとおりです。

表 3.2 校正要素とサンプルデータ

Calibration element（校正要素）	Example data（例）
Calibration Factor（校正係数）	48.79
Calibration Serial Number（校正シリアル番号）	34556767
Calibration Date（校正日）	7/28/2013
Calibration Units（構成単位）	Gy/C
Calibration Expiry（校正有効期限）	7/28/2014
Calibration Equipment（校正機器）	FG65-G-996

- 校正要素がテストで使用される場合、テスト対象のユニットの現在の校正のみがテストに選択されます。 校正機器は、各テストで選択された機器の名前を含む特別なパラメータです。 これは、テストウィンドウに表示し、各テストで使用された機器が履歴記録に確実に記録されるようにするのに役立ちます。
  校正要素がテスト定義に含まれる場合、テストスクリプトで使用される名前 (変数) は、必要な機器のタイプを特定する必要があります。たとえば、電位計の校正係数は、calFactorElec という名前にすることができます。これは、1 つのテストで複数の校正係数が必要な場合に必要です。
- **Test**（テスト）: これらは、ベースラインや許容値など、特定のテストのインスタンスに固有のパラメータを指します。
- **Input**（入力）: テストの実行中に手動で入力される、テストで使用されるパラメータ。 入力パラメータの指定は、明示的なレポートのラベル付けやプロット効果のタグ付けに使用されます。
- **Output**（出力）: これらはテストからの出力を表します。 入力パラメータと同様に、出力パラメータの指定は、明示的なレポートのラベル付けやプロット効果のタ

グ付けに使用されます。

- **Vector**（ベクトル）：カンマ区切りの数値リストを定義します。
- **Secondary**（セカンダリ）：テストの一部として使用される電位計などのセカンダリマシンを選択できます。
- **Component**（コンポーネント）：マシンのコンポーネント情報を参照する機能。
- **Reference**（参照）：マシンのリファレンス情報を参照する機能。
- **Map** （マップ）：マップされたテスト結果を別のテスト結果に直接渡す機能。

3.1.3.7 複数実行テスト（*multi-run tests*）の設定

複数実行テストは、テスト間の親子関係を定義します。 これらは、複数のテストを同時に実行する必要があるが、結果を別のテストに保存する必要がある場合（レポート目的など）に便利です。
複数実行テストの設定については、以下で詳しく説明します。

In parent test (親テストでは):

1. メタデータの <multirun> タグで子テスト（childe tests）を定義します。 すべてのフォームとロジック要素は親にあり、結果は子テストに保存されます。

```
<description>Field Coincidence</description>
<scripttype>XML</scripttype>
<multirun>
  <child>Light vs Radiation Field Coincidence</child>
  <child>Radiation Field Size</child>
</multirun>
<parameterSet>
  <parameter>
     <type>equipment</type>
     <name>fieldsizeY</name>
     <label>Centered Y Field Width</label>
     <format>alpha</format>
      <plottable>false</plottable>
  </parameter>
```

2. 結果フォームに隠しタグを追加して、子テストに必要なパラメータを保存します。

```
<hidden name="multiRun" value="true"/>
```

3. 結果フォームで、子テストに必要な値 (ベースライン、許容値、ステータスを含む) を保存します。

```
<hidden name="y2OpticalField-LightvsRadiationFieldCoincidence"
value="@y2OpticalField"/>
<hidden name="y1OpticalField-LightvsRadiationFieldCoincidence"
value="@y1OpticalField"/>
<hidden name="x2OpticalField-LightvsRadiationFieldCoincidence"
value="@x2OpticalField"/>
<hidden name="status-LightvsRadiationFieldCoincidence"
value="@coincidenceStatus"/>
```

- 「y2OpticalField」は、子テストのパラメータ名です。
- 「LightvsRadiationFieldCoincidence」は、スペースを含まない子テスト名です。
- 「@y2OpticalField」は、親テストのパラメータを指します。
- 「status」は、テストの PASS/FAIL ステータスを保存するために使用されるキーワードです。

In child test (子テストでは)：

4. メタデータでスクリプトタイプを NONE に設定し、子が実行できないようにします。

```
<metaData>
  <description>Radiation Field Size</description>
  <scripttype>None</scripttype>
```

5. すべてのパラメータ（機器パラメータを除く）に対して出力するパラメータの種類を設定します。

6. 子テストに対応するプロットビューを定義します。

3.1.3.8 セカンダリ機器の設定

セカンダリ機器とは、テストで使用されるがテストインスタンス名には反映されない機器を指します。使用された機器を追跡したり、テスト中に使用された機器に関する詳細情報を保存したりするために、セカンダリパラメータを使用できます。
セカンダリパラメータは、マシンタイプ、マシングループ、またはマシングループタイプのいずれかを指定する追加のタグを使用して、他のパラメータと同様の方法で定義します。この追加タグには、テストインスタンスの設定中またはテスト実行中に選択する値を入力します。

例: マシンタイプに基づく選択

```
<parameter>
  <type>secondary</type>
  <name>wellchamber</name>
  <label>Well Chamber</label>
  <format>alpha</format>
  <plottable>false</plottable>
  <description>Secondary Equipment</description>
  <machinetype>Well Chamber</machinetype> </parameter>
</parameter>
```

例: マシングループに基づく選択

```
<parameter>
  <type>secondary</type>
  <name>secondaryLinac</name>
  <label>Secondary Linac</label>
  <format>alpha</format>
  <plottable>false</plottable>
  <description>Secondary Equipment</description>
```

```
  <machinegroup>Elekta All</machinegroup>
</parameter>
```

例: マシングループタイプに基づく選択。 一般的な使用例は、機器がペアとして校正される場合で、マシングループタイプとして「calibration」を使用します。

```
<parameter>
  <type>secondary</type>
  <name>calPair</name>
  <label>Calibration Pair</label>
  <format>alpha</format>
  <plottable>false</plottable>
  <description>Secondary Equipment</description>
  <machinegrouptype>calibration</machinegrouptype>
</parameter>
```

[Edit Test Instance] ウィンドウで、セカンダリ機器を選択して保存するオプションがあります。空白のままにすると、テスト実行中にセカンダリ機器を選択するよう促すポップアップウィンドウが表示されます。選択した機器の名前は、セカンダリタイプパラメータに保存されます。
さらに、機器コンポーネント、線源および校正情報など、選択した二次デバイスに関する詳細も表示したり、テスト実行で使用したりすることができます。「component」、「source」、または「calibration」パラメータをセカンダリ機器に結び付けるパラメータ関係は、キーワード「property」を使用して定義する必要があります。

例: 校正パラメータとセカンダリパラメータのパラメータ関係

```
<parameterSet>
    <parameter>
        <type>secondary</type>
        <name>calPair</name>
        <label>Calibration Pair</label>
```

```
        <format>alpha</format>
        <plottable>false</plottable>
        <description>Secondary Equipment</description>
        <machinegrouptype>calibration</machinegrouptype>
    </parameter>
    <parameter>
        <type>calibration</type>
        <name>calFactorForPair</name>
        <label>Calibration Factor</label>
        <format>numeric</format>
        <plottable>false</plottable>
        <description>Calibration factor for Calibration Pair</description>
    </parameter> </parameterSet>
...
<parameterRelnSet>
    <parameterReln>
        <left>calFactorForPair</left>
        <name>property</name>
        <right>calPair</right>
    </parameterReln>
</parameterRelnSet>
```

例: マシンタイプに基づく選択

```
<parameter>
  <type>secondary</type>
  <name>wellchamber</name>
  <label>Well Chamber</label>
  <format>alpha</format>
  <plottable>false</plottable>
  <description>Secondary Equipment</description>
  <machinetype>Well Chamber</machinetype>
```

</parameter>

3.1.3.9 マシンコンポーネントとマシンリファレンスの設定

マシンコンポーネントおよびマシンリファレンス情報には、必要に応じて、component および reference パラメータタイプを介してテストスクリプトからアクセスできます。
component パラメータタイプの場合、<label> タグはコンポーネントの名前にマップされ、オプションの <meta> タグは指定されたコンポーネントのメタデータにマップされます。 テストが実行されると、コンポーネントの値またはコンポーネントのメタデータがパラメータに保存されます。

例: コンポーネントパラメータ

```
<parameter>
    <type>component</type>
    <label>MLC</label>
    <meta>Y Field Size</meta>
    <name>fieldsizeY</name>
    <format>alpha</format>
    <plottable>false</plottable>
</parameter>
```

同様に、reference パラメータタイプの場合、<label> タグには Machine Reference Type の名前が含まれ、<format> タグは、Machine Reference Type に関連付けられたデータと同じ形式 (数値、アルファ、ベクトルなど) である必要があります。 <selectable> タグも存在し、true に設定されている必要があります。
<selectable> タグを true に設定すると、テストインスタンスの [Edit Test Parameters] ウィンドウのドロップダウンリストから Machine Reference を選択できるようになります。

例: 参照パラメータ

```
<parameter>
    <type>reference</type>
    <name>refData</name>
    <label>Photon PDD Curve</label>
    <format>vector</format>
    <plottable>true</plottable>
    <description>Reference Profile at the given beam energy</description>
    <selectable>true<selectable>
</parameter>
```

コンポーネントおよび/または参照パラメータを設定すると、出力パラメータと同様の方法で使用できます。

3.1.3.10 マップされたテストの設定

マップされたテストにより、ユーザーは現在のテスト実行で以前のテスト結果にアクセスできます。 マッピングは、テストインスタントレベルで、テスト内、または 2 つの異なるテスト間で設定できます。

注意 : ユーザーは、Mapped Tests 機能を使用してアクセスしたデータの正確性を検証する責任があります。すべてのテスト入力はユーザーによって検証される必要があります。

3.1.3.10.1 シングルテスト

1 つのテストのパラメータ間のマッピングは、「パラメータマップ」関係を使用して設定できます。例えば、テストに「result」と「baseline」という 2 つの出力パラメータがあるとすると、ユーザーは前回のテスト実行時の「result」値を「baseline」パラメータに保存するよう定義することができます。最初のテスト実行では、'baseline' パラメータ値は NULL になります。

注意：この関係は、常に 2 つの出力パラメータの間で定義する必要があります。

```
<parameterReln>
    <left>baseline</left>
    <name>parameter map</name>
    <right>result</right>
</parameterReln>
```

「baseline」の値は、以下に示すように、<form> タグ内の入力パラメータのデフォルト値として設定できます:

```
<horizontal/>
<comment value="Input: " width="200"/>
<number name="inputValue" validate="@double" value="@baseline"
width="200"/> <vertical/>
```

詳細については、「セクション 3.1.3.10.3: マッピングされたテストに関する注意事項 (シングルテストと異なるテスト)」を参照してください。

3.1.3.10.2 各種テスト

各種テストのパラメータ間のマッピングは、「parameter map」関係を使用して設定することもできます。 パラメータ関係の <right> は、テストを識別するために「parameterNameTestNameWithoutSpaces」を使用して定義されます。 たとえば、テスト A からは「baseline」、テスト B からは「result」という名前の 2 つの出力パラメータがあるとします。テスト A では、以下に示すように関係を定義します。

```
<parameterReln>
    <left>baseline</left>
    <name>parameter map</name>
    <right>result-TestB</right>
</parameterReln>
```

さらに、「結果」値を取得するテスト B のテスト インスタンスを定義する必要があります。これは、以下に示すマップ パラメータタイプを使用して実装されます。 テスト A のテス

ト インスタンスが作成されると、ユーザーはマップパラメータに使用するテスト B のテスト インスタンスを選択します。

注意：テスト A は依存テストと呼ばれ、テスト B はマップされたテストと呼ばれます。

```
<parameter>
    <type>map</type>
    <name>mappedInstance</name>
    <label>Mapped Test Instance</label>
    <format>alpha</format>
    <plottable>false</plottable>
    <mappedTest>Test B</mappedTest>
</parameter>
```

例 1 テスト「Previous Test Run Demo 1」にパラメータが含まれている場合：

```
<parameter>
    <type>output</type>
    <name>baseline</name>
    <label>Baseline</label>
    <format>numeric</format>
    <plottable>false</plottable>
</parameter>
<parameter>
    <type>output</type>
    <name>result</name>
    <label>Result</label>
    <format>numeric</format>
    <plottable>false</plottable>
</parameter>
```

<parameterRelnSet> には次のようになります。

```
<parameterReln>
    <left>baseline</left>
    <name>parameter map</name>
    <right>result</right>
</parameterReln>
```

そうすると、最初のテストインスタンスの後に実行されるテストインスタンスは、同じテストインスタンスの前回の実行の「result」と等しい「baseline」パラメータが設定されます。そのため、テストインスタンスの実行で「result」が 20 と計算された場合、同じテスト インスタンスの次回の実行は「baseline」が 20 で始まります。最初の実行では、ベースラインは「NULL」になります。

例 2 上記の「Previous Test Run Demo 1」の定義と、別のテスト「Previous Test Run Demo 2」にパラメータがあると仮定します：

```
<parameter>
    <type>output</type>
    <name>baseline</name>
    <label>Baseline</label>
    <format>numeric</format>
    <plottable>false</plottable>
</parameter>
<parameter>
    <type>output</type>
    <name>result</name>
    <label>Result</label>
    <format>numeric</format>
    <plottable>false</plottable>
</parameter>
<parameter>
    <type>map</type>
    <name>mappedInstance</name>
```

```
    <label>Mapped Test Instance</label>
    <format>alpha</format>
    <plottable>false</plottable>
    <mappedTest>Previous Test Run Demo 1</mappedTest>
</parameter>
```

そして関係性：

```
<parameterReln>
    <left>baseline</left>
    <name>parameter map</name>
    <right>result-PreviousTestRunDemo1</right>
</parameterReln>
```

次に、「Previous Test Run Demo 2」のインスタンスのテストパラメータを設定するときに、「Previous Test Run Demo 1」のソーステストインスタンスを選択します:

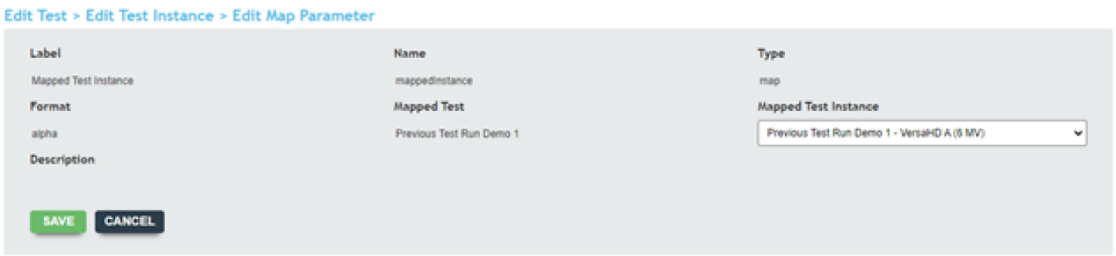


図 3.3 Edit Map Parameter ウィンドウ

また、「Previous Test Run Demo 2」の実行では、テストパラメータとして入力された「Previous Test Run Demo 1」のテストインスタンスの直近の実行の "result" に "baseline"値がプリセットされます（上記）。
詳細については、セクション 3.1.3.10.3「マップされたテストの注意事項（シングルテストおよび各種テスト用）」を参照してください。

3.1.3.10.3 マップされたテストの注意事項（シングルテストおよび各種テスト用）

前回のテスト実行のタイムスタンプは、キーワード「mappedTestRunTestNameWithoutSpaces」を使用して取得できます。 たとえば、タイムスタンプを表示するには、次のコード スニペットを <form> ノード内で使用できます:

```
<horizontal/>
<comment value="Mapped Test Run: " width="200"/>
<comment value="@mappedTestRun-TestB " width="200"/>
<vertical/>
```

- <left> パラメータ値は、前回のテスト実行が PASS だった場合にのみ入力されます。FAIL の場合は、最後に実行された PASS が使用されます。テストの実行がない場合、初期値は NULL になります。
- マップされたテストがアップロード後にドラフトされた場合、依存テストが実行されると、マップされたテストは NULL になります。
- 依存テストが最終的な場合 (ドラフトモードではない場合など)、マップされたテスト実行はドラフトではない実行になります。
- 一部のテストでは、フォームがプラグインで定義されており、XML スクリプトで定義されているパラメータの一部がありません。 これらのパラメータは、別のテストへの入力として使用する前に、<parameterSet> ブロック内の <parameter> として定義する必要があります。

3.1.3.11 スクリプトに関する補足説明

- スクリプト内であらかじめ設定されている変数をフォームで使用する場合は、@記号を使用します (たとえば、value="@variableValue")。
- 初期化されていない変数の値は使用しないでください。
- 複数実行するテストやマップされたテストに使用する場合を除き、要素名に「-」を含めることはできません。
- 数値は小数点以下 6 桁が切り捨てられます。
- contextTestDate は、実行中のテストで参照すると、現在のユーザー コンテキストに関連付けられた日付を取得できます。 たとえば、<set name="testDate"

value="@contextTestDate" /> となります。

- 要素 <comment> および <constant> はデータベースに保存されません。

3.1.4 xml リファレンス

3.1.4.1 メタデータ:<metadata>

メタデータセクションには次の情報が含まれます。

- テストの説明
- テストを実行するときに使用される手順と仕様
- テストで使用されるコアパラメータのリスト
- パラメータ間の関係 (ベースラインや許容値など)
- プロット特性

3.1.4.2 <description>

<description> ノードには、テストの対応する要素のテキストが含まれます。

3.1.4.3 <procedure>

<procedure> ノードには、テストの対応する要素のテキストが含まれます。

3.1.4.4 <specification>

<specification> ノードには、テストの対応する要素のテキストが含まれます。

3.1.4.5 <parameterSet>

<parameterSet> ノードには、すべてのテストパラメーターを説明するメタデータが含まれています。 アップロードプロセスでは、このメタデータを使用して、AQUA でのテスト定義を完了させます。 例えば：

<description>Simple HT Timer Test.</description>
<procedure>Enter the HT Timer value.</procedure>
<specification>
HT Timer value must be within @failTolerance of the baseline: @baseline
</specification>

```
<parameterSet>
    <parameter>
        <type>test</type>
        <name>failTolerance</name>
        <label>Fail Tolerance</label>
        <format>numeric</format>
        <plottable>true</plottable>
        <description></description>
    </parameter>
    <parameter>
        <type>test</type>
        <name>baseline</name>
        <label>Baseline</label>
        <format>numeric</format>
        <plottable>true</plottable>
        <description></description>
    </parameter>
    <parameter>
        <type>input</type>
        <name>htTimerValue</name>
        <label>HT Timer Value</label>
        <format>numeric</format>
        <plottable>true</plottable>
        <description></description>
    </parameter>
</parameterSet>
```

3.1.4.6 <parameter>

シングルテストパラメータを記述します。

3.1.4.7 <type>

考えられるパラメータのタイプは次のとおり：

- **Test**（テスト）：これらは、ベースラインや許容値など、特定のテストのインスタンスに固有のパラメータを指します。
- **Input**（入力）：テストの実行中に手動で入力される、テストで使用されるパラメータ。 入力パラメータの指定は、明示的なレポートのラベル付けやプロット効果のタグ付けに使用されます。
- **Output**（出力）：これらはテストからの出力を表します。 入力パラメータと同様に、出力パラメータの指定は、明示的なレポートのラベル付けやプロット効果のタグ付けに使用されます。
- **Equipment**(機器):
  これらは、テストされるユニットの特性を指します。 複数の機器パラメータを使用したテストは、同じテストに対して、可能なテストインスタンスのデカルト積が得られます。
  正しくマッピングするには、装置パラメータのラベルが機器コンポーネントのタイプに対応している必要があります。
- **Calibration**（校正）：二次機器からの校正係数を必要とするテスト。 利用可能なキャリブレーション要素は次のとおりです。

表 3.3 線源要素と値

Element（要素）	Value（値）
Calibration Factor（校正係数）	48.79
Calibration Serial Number（校正シリアル番号）	34556767
Calibration Date（校正日）	7/28/2013
Calibration Units（構成単位）	Gy/C
Calibration Expiry（校正有効期限）	7/28/2014
Calibration Equipment（校正機器）	FG65-G-996

校正要素がテストで使用される場合、テスト対象のユニットの現在の校正のみがテ

ストに選択されます。 校正機器は、各テストで選択された機器の名前を含む特別なパラメータです。 これは、テストウィンドウに表示し、各テストで使用された機器が履歴記録に確実に記録されるようにするのに役立ちます。
校正要素がテスト定義に含まれる場合、テストスクリプトで使用される名前 (変数) は、必要な機器のタイプを特定する必要があります。たとえば、電位計の校正係数は、calFactorElec という名前にすることができます。これは、1 つのテストで複数の校正係数が必要な場合に必要です。

- **Source** (線源): 通常、線源強度テストなど、現在の放射線源に関する情報がテストで使用される HDR ユニットに使用されます。 ソース要素のリスト (データ例を含む) は次のとおりです。

表 3.4 線源要素とサンプルデータ

Element（要素）	Example data (例)
Source Isotope（線源同位体）	Iridium-192
Source Half-Life（線源半減期）	73.827
Source Serial Number（線源のシリアル番号）	A233-D567
Source Reference（線源リファレンス）	48.01
Source Reference Date（線源参照日）	7/28/2013 08:00:00
Source Expiry Date（線源の有効期限）	10/28/2013

ソース要素がテストで使用される場合、テスト対象のユニットの現在のソースのみがテスト用に選択されます。ソース要素をテスト定義に含める場合は、スクリプトの変数名として使用される一意の名前を含めます。例えば、線源同位体には sourceIsotope という名前を使用します。

- **Secondary**（セカンダリ）: セカンダリ タイプのパラメータは、テストインスタンスをテスト中に使用される測定機器にリンクするために使用されます。セカンダリ装置を参照することで、校正情報や機器インターフェースを利用することができます。 詳細については、セクション 1.1.3.1.7 を参照してください。

- **Component**（コンポーネント）: コンポーネントタイプパラメータは、与えられたマシンコンポーネントに定義されたメタデータ値にアクセスするために使用されます。 たとえば、Jaws または MLC に定義されたフィールドサイズ設定です。 詳細については、セクション 1.1.3.1.8 を参照してください。マシンのコンポーネント情報を参照する機能してください。
- **Reference**（参照）: 参照タイプのパラメータは、マシンに保存されたマシン参照データにアクセスするために使用されます。 マシン参照データの例としては、ビーム プロファイル、PDD、ビーム品質指定などがあります。 詳細については、セクション 1.1.3.1.8 を参照してください。
- **Map** （マップ）: マップは、AQUA 内で現在のテストを別のテストにリンクするために使用されます。 テストをマッピングすることで、あるテストの結果を別のテストに渡すことができます。 たとえば、年次測定の結果が、同様の日次/月次測定のベースラインにすることができます。詳細については、セクション 1.1.3.1.9 を参照してください。

3.1.4.8 <name>

スクリプト内で参照するパラメータの名前を定義します。

3.1.4.9 <label>

ウィンドウに表示されるラベルです。

3.1.4.10 <format>

サポートされているタイプ:

- Numeric
- Alpha

3.1.4.11 <plottable>

- true
- false

3.1.4.12 <description>

パラメータのテキストによる説明。

3.1.4.13 <unit>

- 現在サポートされている単位を取得するには、[System Data] > [Units] に進みます。
- ユニットが定義されていない場合は、インターフェイスから新しいユニットを作成し、テストで使用できます。

3.1.4.14 <machinetype>

- 現在サポートされているマシンタイプを取得するには、[Equipment] > [Machine Type] を選択します。
- マシンタイプが定義されていない場合は、インターフェイスから新しいマシンタイプを作成し、テストで使用できます。

3.1.4.15 <parameterReInSet>

<parameterRelnSet> ノードはプロット機能をサポートするパラメータ間の関係を定義し、<plotData> ノードはプロット ビューの設定を記述します。

3.1.4.15.1 <parameterReIn>

単一パラメータの関係を定義します。 次の例では、入力パラメータのベースラインを識別します。

```
<parameterReln>
    <left>radialSymmetryBaseline</left>
    <name>baseline</name>
    <right>radialSymmetry</right>
</parameterReln>
```

3.1.4.15.2 <name>

可能な関係タイプは次のとおりです。

- Baseline (ベースライン)
- absolute upper warn tolerance（絶対上限警告許容値）
- absolute lower warn tolerance（絶対低限警告許容値）
- relative upper warn tolerance（相対的な上限警告許容値）
- relative lower warn tolerance（相対的な低限警告許容値）
- absolute upper fail tolerance（絶対的な上限の Fail 許容値）
- absolute lower fail tolerance（絶対的な下限の Fail 許容値）
- relative upper fail tolerance（相対的な上限の Fail 許容値）
- relative lower fail tolerance（相対的な下限の Fail 許容値）
- plot against（～に対してプロットする）
- property
- parameter map（パラメータマップ）

3.1.4.15.3 <left> <right>

「is-a」関係の左側と右側を定義します。 次の例では、radialSymmetryBaseline (<left> タグの下) は、対称性 (<right> タグの下) の「is-a」ベースライン (<type>) です。

```
<parameterReln>
    <left>radialSymmetryBaseline</left>
    <name>baseline</name>
    <right>radialSymmetry</right>
</parameterReln>
```

3.1.4.16 <plotData>

3.1.4.16.1 <plotView>

plotData 要素は、プロットビューの設定を記述します。この例では、プロット ビューの名前は Symmetry で、パラメータ RadialSymmetry と transverseSymmetry が含まれています。 パラメータに関連するベースラインと許容差は、自動的にビューに含まれます。

```
<plotData>
    <plotView>
        <name>Symmetry</name>
        <type>line chart</type>
        <parameters>
            <parameter>radialSymmetry</parameter>
            <parameter>transverseSymmetry</parameter>
        </parameters>
    </plotView>
</plotData>
```

3.1.4.16.2 <name>

プロットの名前を定義します。

3.1.4.16.3 <type>

- パラメータ別（パラメータ同士をプロットする）
- 折れ線グラフ（日付に対してパラメータをプロットする）

3.1.4.16.4 <parameters>

すべての <parameter> タグが含まれます。

3.1.4.16.5 <parameter>

プロットできるパラメータを定義します。

3.1.4.16.6 <default>

この plotView がデフォルトのプロットかどうかを指定します。

```
<default>true</default>
```

テストを実行すると、デフォルトのプロットが画面下部の小さなウィンドウに表示されます。 プロットには、テスト実行の有効日（Effective Date）を最終日とし、それよりも数日

前の日付の値が表示されます。前の日数は、テスト インスタンスのスケジュールされた頻度と、[System Data] > [Time Periods] にある Default Plot Days に基づいて決定されます。1 つのテストにつき 1 つの<default> プロットのみを設定する必要があります。

3.1.5 フォーム要素 ：<form>

すべてのフォーム要素は、セクション 3.1.6: **フォーム要素の属性**にリストされている属性を持つことができます。

3.1.5.1 Controls:

3.1.5.1.1 <command>

指定されたテキスト値を使用してテキスト ウィジェットを作成します。 この値はフォーム出力に表示されます。

<comment value="Shown text" />

テキストを左揃えにするには、テキストの後にコロンを追加します。

<comment value="Shown text:" />

3.1.5.1.2 <colourComment>

データを保持し、指定した色の背景を含むことができるテキスト・ウィジェットを作成します。色の設定は GREEN, RED, BLUE, YELLOW です。

<colourComment name="colourComment1" value="text" colour="YELLOW" />

注意：AQUA のバージョン 2.0 では、各「colourComment」タグに固有の名前が必要です (上記の例を参照)。 次のエラーが表示された場合は、「name」属性を追加し、zip ファイルを再アップロードします。 Error: A <colourComment> tag is missing a 'name' attribute.

3.1.5.1.3 <hidden>

非表示ウィジェットは、データを保持するために使用できますが、フォームに値を表示する

ことはできません。

<hidden name="variableName" value="variableValue" />

3.1.5.1.4 <readOnly>

データを保持できるテキスト ウィジェット。

<readOnly name="widgetName" value="Shown Text" />

3.1.5.1.5 <longInteger>

64 ビット整数の入力データを受け取るために使用します。ウィジェットに値を置くと、その値がデフォルトとしてフォームに表示されます。

<longInteger name="variableName" value="123" />

3.1.5.1.6 <integer>

32 ビット整数の入力データを受け取るために使用します。ウィジェットに値を置くと、その値がデフォルトとしてフォームに表示されます。

<integer name="variableName" value="123" />

3.1.5.1.7 <number>

浮動小数点入力データを受け取るために使用します。ウィジェットに値を置くと、その値がデフォルトとしてフォームに表示されます。

<number name="variableName" value="123" />

3.1.5.1.8 <string>

英数字データの入力ボックス。

<string title="Tester Name" name="testerName"/>

3.1.5.1.9 <data>

日付セレクタ。 フォームに日付ウィジェットが表示されます。

<date name="testDate" title="Test Date" />

3.1.5.1.10 <dateRange>

日付範囲セレクタ。

3.1.5.1.11 <listBox>

ドロップダウン メニューまたは複数選択リスト ボックスのいずれかを作成します。 属性 domainClass または domain を使用できます。 複数選択リストを選択するには、「multi」属性を使用します。

<listBox domain="PASS,FAIL,SKIP" />
<listBox domainClass="Machine" multi="true" />

3.1.5.1.12 <checkbox>

<checkBox name="truePos" value="true" domain="true" labels="True Position" />

3.1.5.1.13 <radioButton>

一連のリンクされたラジオボタンを作成します。ラベルを使用して、ドメインの外観を上書きします。

<radioButton domain="PASS,FAIL" labels="Pass,Fail" />

3.1.5.2 Presentation:

3.1.5.2.1 <line>

水平線を作成します。

<line width="800" />

3.1.5.2.2 <space/>

空白行を作成します。

3.1.5.2.3 <blank/>

空白を作成します。

3.1.5.3 Menus:

3.1.5.3.1 <menu>

フォームの下部に使用されます。 SAVE ドメイン オプションが常に存在する必要があります。 CANCEL はオプションですが、推奨されます。

<menu name="status" domain="SAVE,CANCEL" labels="Save,Cancel"/>

入力フォームと出力フォームを持つ手動入力テストの場合、出力フォームのメニューに Revise オプションを追加することが可能です。Revise ボタンを使用すると、テスト実行を保存する前に入力フォームに戻り、値を更新することができます。

<menu name="status" domain="SAVE,CANCEL" labels="Save,Cancel"/>

3.1.5.4 Panels:

3.1.5.4.1 <border>

単一の黒いピクセルの境界線を持つパネルを作成します。 パネルにウィジェットを挿入するには、グループ属性を使用します。 グループはパネルの前に作成する必要があります。

3.1.5.4.2 <group><endGroup>

Group は endGroup と組み合わせて使用されます。 これらのタグの間にあるウィジェットは、ウィジェット グループに配置され、それらを含めるために DisclosurePanel に割り当てることができます。

```
<group name="myGroup" />
    <comment value="My first comment" width="800" />
```

```
    <comment value="My second comment" width="800" />
<endGroup />
```

3.1.5.4.3 <disclosurePanel>

拡大・縮小するパネルを作成します。title 属性の代わりに header 属性を使用します。パネルにウィジェットを挿入するには group 属性を使います。グループはパネルの前に作成する必要があります。

```
<disclosurePanel name="spec" header="Mine" group="myGroup" />
```

3.1.5.5 Layouts:

3.1.5.5.1 <common>

特に指定がない限り、次のウィジェットに共通のパラメータを与えます。

3.1.5.5.2 <vertical/ >

以下のウィジェットを縦方向に配置します。

3.1.5.5.3 <horizontal/ >

以下のウィジェットを横方向に配置します。

3.1.6 フォーム要素の属性

これらは上記で説明したフォーム要素の属性を表します。

- **name**: ウィジェットの名前を設定し、変数名として使用します。
- **title**: 太字で表示されるウィジェットのタイトルを設定します。
- **title_width**: タイトル部分の幅をウィジェットとは別に設定します。
- **header**: これは開示パネルにのみ使用されます。
- **noTitle**: ウィジェットからタイトルを削除します (ウィジェットが予想よりも「高く」表示される可能性があります。
- **value**: ウィジェットの値を設定します。
- **validate**: 通常は @notnull とともに使用され、続行するにはウィジェットに入力する必要があることを示します。 その他: @double、値は null 以外の数値である

必要があります。

<number name="radialSymmetry" validate="@double" />

- **width**: ウィジェットの幅を設定します。
- **height**: ウィジェットの高さを設定します。
- **size**: ウィジェットのサイズ（幅、高さ）を設定します。
- **condition**: これは、別のウィジェットの名前を指定して、equal と組み合わせて使用されます。
- **equal**: これは条件と組み合わせて使用されます。 指定されたウィジェットの値が目的の値である場合、このウィジェットが表示されます。

  <radioButton name="radioB1" domain="YES,NO" />

  <textbox name="example" condition="radioB1" equal="YES" />
- **conditonEquals**: condition と equal の代わりに使用されます。

  <radioButton name="radioB1" domain="YES,NO" />

  <textbox name="example" conditionEquals="radioB1,YES" />
- **domainClass**: 指定したクラスのドメインを設定します。返される値はID値です。
- Machine
- MachineType
- MachineStatus
- Test
- PlotableTest
- TimePeriod
- Tag
- TimeGranularity
- TestGroup
- MachineGroup
- EventType
- Reports
- **domain**: ドメイン値を設定します。 これは文字列のリストにすることができます。
- **labels**: ユーザーに表示されるウィジェットのラベルを設定します。

3.1.7 ロジック要素

表 3.5 ロジック要素のフィールドと説明

フィールド	**説明**
<constant>	これは、テストで定数値を宣言する方法です。 その形式は標準の変数設定とは異なります。 <constant name="value" />
<set>	これにより、変数の値を割り当てることができます。 <set name="variableName" value="variableValue" /> <set name="variableName2"> <value name="variableName" /> </set>
<add> <subtract> <divide> <multiply> <abs>	加算と乗算は結合しており、任意の数の値を取ることができます。 <add> <value>1</value> <value>2</value> <value name="variableName" /> </add>
<if> <then>	if 文の作成に使用されます。 <if> <greaterThan> <value name="variableName1" /> <value name="variableName2" /> </greaterThan> <then> <set name="result" value="someResult" /> </then> </if>

フィールド	説明
<greaterThan> <greaterThanEquals> <lessThan> <lessThanEquals> <equals>	<if> <greaterThan> <abs> <subtract> <value name="transverseSymmetryBaseline" /> <value name="transverseSymmetry" /> </subtract> </abs> <value name="failTolerance" /> </greaterThan> <then> <set name="overallResult" value="FAIL"/> </then> </if>
<function> <input> <return>	テスト XML 内から多くの Java 関数を呼び出すことができます。これらは、ロジックが複雑すぎて XML のロジック機能では処理できない場合や、アルゴリズムが一般的に使用される場合（例えば、値のリストの平均を求めるなど）のために開発されました。 例 次の行を<form>要素に含めることができます。これは、カンマで区切られた値の文字列としてパラメータ読み取りに保存される 3 つの入力フィールドをユーザーに表示します。 <number name="readings" width="75" /> <number name="readings" width="75" /> <number name="readings" width="75" /> このパラメータが平均関数に渡されると、その結果はパラメータ mean に格納される単一の平均値になります。

フィールド	説明
	<function name="mean"> <input name="readings" value="args"/> <return name="mean"/> </function> **mean**：配列の算術平均を計算します。 • **Input** • **args**:数値の配列 • **Output** • **mean**:平均値 **maxDiff**：ベースライン値と値の配列の間の最大差の絶対値を計算します。 • **Input** • **args**:数値の配列 • **base**:比較する基準値 • **Output** • **maxDiff**:算出された最大差 **maxPercentDiff**：ベースライン値と値の配列の間の最大差異パーセントを計算します。 • **Input** • **args**:数値の配列 • **base**:比較する基準値 • **Output** • **maxDiff**:算出された最大パーセント差 **minimumValue**：値の配列から最小値を見つけます。 • **Input** • **args**:数値の配列 • **Output** • **minValue**:最小値

フィールド	説明

	maximumValue：値の配列から最大値を見つけます。
	• **Input**
	  • **args**:数値の配列
	• **Output**
	  • **maxValue**:最大値
	standardDeviation：値のリストの標準偏差を計算します。
	• **Input**
	  • **args**:数値の配列
	  • **means**:平均値（指定しない場合は平均値が計算されます）
	• **Output**
	  • **standardDeviation**:算出された標準偏差
	squareRoot：値の平方根を計算します。
	• **Input**
	  • **square**:二乗値を指定します。
	例
	<function="squareRoot">
	<input name="userinputnumber" value="square"/>
	<return name="squareRoot"/>
	</function>
	この例では、userinputnumber は数値型の入力パラメータであり、値は常に「square」に設定されます。この関数は userinputnumber の平方根を計算します。

フィールド	説明

	concat：2 つの値を 1 つのカンマ区切りの値に連結します。
	• **Input**
	  • **numinputs**:入力値の数
	  • **input_1**:最初の値
	  • **input_2**: 2 番目の値
	• **Output**
	  • **functionStatus**:Pass もしくは Fail
	  • **functionMessage**:Fail の場合、エラーメッセージ
	  • **output**:カンマで区切られた連結値
	例
	<function name="concat">
	<input name="numInputs"/>
	<input name="input_1"/>
	<input name="input_2"/>
	
	<return name="functionStatus"/>
	<return name="functionMessage"/>
	<return name="output"/>
	</function>
	halfLifeDecayFunction:開始日と崩壊率を日数で指定して、現時点の崩壊係数を計算します。
	• **Input**
	  • **startDate**:崩壊開始日
	  • **materialDecayRate**:崩壊率(日数ベース)
	• **Output**
	  • **decayFactor**:現在の崩壊係数を算出

フィールド	説明

	determineROI_PDD: PDD データをトリミングして Dmax→Full Depth の範囲で比較を行う機能です。
	• **Input**
	  • **referenceLabels**: 基準 PDD のデータラベル（測定位置）からなるベクトルパラメータ
	  • **referenceData**: 基準 PDD のデータ値（相対線量）からなるベクトルパラメータ
	  • **measuredLabels**: 測定された PDD のデータラベル（測定位置）からなるベクトルパラメータ
	  • **measuredData**: 測定された PDD のデータ値（相対線量）からなるベクトルパラメータ
	  • **normalizationLocation**: 正規化の深さ（データラベルと同じ単位)
	  • **normalizationValue**: 正規化の値（通常は 100)
	• **Output**
	  • **ROIPDDResult**: ROIPDDResult が 0 の場合、エラーが発生した。
	  • **ROIPDDMessage**: 関数によって生成されたエラーメッセージの内容
	  • **trimmedReferenceLabels**: ROI 内の参照データラベルからなるベクトル
	  • **trimmedReferenceData**: ROI 内の参照データ値からなるベクトル
	  • **trimmedMeasuredLabels**: ROI 内の測定データラベルからなるベクトル
	  • **trimmedMeasuredData**: ROI 内の測定データ値からなるベクトル

フィールド	説明

	determineROI_Profile
	• **Input**
	  • **referenceLabels**: 基準 PDD のデータラベル（測定位置）からなるベクトルパラメータ
	  • **referenceData**: 基準 PDD のデータ値（相対線量）からなるベクトルパラメータ
	  • **measuredLabels**: 測定された PDD のデータラベル（測定位置）からなるベクトルパラメータ
	  • **measuredData**: 測定された PDD のデータ値（相対線量）からなるベクトルパラメータ
	  • **normalizationLocation**: 正規化の深さ（データラベルと同じ単位)
	  • **normalizationValue**: 正規化の値 (通常は 100)
	• **Output**
	  • **ROIPDDResult**: ROIPDDResult が 0 の場合、エラーが発生した。
	  • **ROIPDDMessage**: 関数によって生成されたエラーメッセージの内容
	  • **trimmedReferenceLabels**: ROI 内の参照データラベルからなるベクトル
	  • **trimmedReferenceData**: ROI 内の参照データ値からなるベクトル
	  • **trimmedMeasuredLabels**: ROI 内の測定データラベルからなるベクトル

3.1.8 キーワード

表 3.6 キーワードのフィールドと値

フィールド	**値**
multirun	これは、このテストが他のテストを参照し、結果として複数回の実行を保存する必要があることを示すために使用します。フォーム内の非表示タグに配置する必要があります。 <hidden name="multiRun" value="true" />
status	すべてのテスト スクリプトは、テストの最終ステータス (PASS、FAIL、WARNING のいずれか) を報告するフォームで終わる必要があります。テストの最終結果を保持するために、「status」という特別な変数が使用される。さらに、ドメイン値 SAVE および CANCEL を含むメニュー要素を含める必要があります。次のスクリプト スニペット (テンプレート) は、すべてのテスト スクリプトで使用できます (変数 result および resultColour は、特定のテストで使用される変数に置き換えられます)。 <horizontal /> <comment value="Overall Status: " width="200" /> <colourComment name=”colourComment1” width="100" value="@result" colour="@resultColour" /> <hidden name="status" value="@result" /> <vertical/> <menu name="status" domain="SAVE,CANCEL" labels="Save,Cancel"/> メニュー要素のみが含まれている場合、[SAVE] を選択すると、常に PASS のテスト ステータスが生成されます。

3.1.9 メタデータ：<メタデータ>

メタデータ セクションには次の情報が含まれます。

- テストの説明
- テストを実行するときに使用される手順と仕様
- テストで使用されるコアパラメータのリスト
- パラメータ間の関係 (ベースラインや許容値など)
- プロット特性

3.2 エレクタの実績テストに基づく新しいテストの作成

Elekta の実績があるテストは編集できません。Elekta の実績のあるテストに基づいて、新しいテストを作成できます。

1. Tests > Test List > Edit Test のウィンドウで、変更するテストを選択します。
2. EXPORT をクリックし、編集用の XML を保存する場所を選択します。
3. テスト XML を編集します。
4. アップロードするテストと同様に、XML を TestList.txt ファイルと一緒にディレクトリに配置します。必ず、元のテストとは異なる名前を付けます。
5. Tests > File Upload を選択して、新しいテストをアップロードします。

注意：編集するテストが親テストの場合、すべての子テストをエクスポートします。TestList.txt ファイルのオリジナルとは異なる名前を使用して、それらを新しいテストの.zip ファイルに含めます。それぞれの名前の末尾を "CHD "にすることをお勧めします。親テストの<multirun>セクションを編集します。正しい新しい子の名前を参照するように、各 <child> 要素の名前を変更します。

4. AQUA v3.0 Plugin 機能ガイド v1.0

4.1 AQUA v3.0 Plugin の紹介

4.1.1 AQUA v3.0 Plugin 機能ガイドー目的

このドキュメントでは、AQUA 3.0 で使用する関数プラグインの作成方法について説明します。

4.2 AQUA v3.0 Plugin

プラグインは、ユーザーが開発し、AQUA インストールにリンクされるカスタム Java ライブラリです。 プラグインに含まれるカスタマイズされた関数は、XML テスト スクリプトから参照され、AQUA 内で実行できます。 ユーザーは、Java を使用したソフトウェアの開発と、Java コードを JAR ファイルにコンパイルすることに精通していることを前提としています。
このドキュメントの対象は、スカラー入出力（integer、double、string）を含む関数に限定されており、画像などのより複雑なデータ型は扱っていません。

- プラグインを開発して JAR ファイルにコンパイルする
- JAR ファイルを AQUA インストールに追加する
- XML テスト スクリプトを通じてプラグイン関数を呼び出す

4.2.1 開発

カスタムプラグインの開発に必要なライブラリは、AQUA インストール ディレクトリにあります。（例 : C:¥Program Files (x86)¥Apache Software Foundation¥Tomcat 8.5¥webapps¥AQA ¥WEB-INF¥lib）必要なライブラリは次のとおりです。

- AQAModel.jar
- GWTServer.jar

プラグインを作成するには、クラスを拡張するクラスを作成します。
com.uhn.GWTServer.common.Plugin

プラグインクラスの構造を説明するために、TestFunctionPlugin という名前のプラグインの例が付録に含まれています。プラグインファイルの名前は、ライブラリ名として参照され

ます。（例：TestFunctionPlugin.jar）

本文書で言及される他の要素を特定するために、この例を使用します。

- Package Name
- Class Name
- Function Name
- Function Output Parameter
- Function Input Parameter

プラグインクラスには、プラグインが初期化されるとき（Tomcat を介して AQUA が起動されるとき）に実行されるコードを含む init()メソッドと、プラグインがシャットダウンされるとき（Tomcat を介して AQUA が終了されるとき）に実行される cleanup()メソッドがあります。これらのメソッドにコードを記述する必要はありません。
プラグインに含める固有の関数ごとに、Function Name は、XML テスト スクリプト内から関数を呼び出すときに使用する名前です。

```
public TestFunction add() {
  return new AddFunction();
}
```

このメソッドは、カスタム プラグイン関数の実装を含む TestFunction オブジェクトを返します。この実装には、実行中の XML テスト スクリプトから入力を受け取る 1 つ以上の 関数入力パラメータ呼び出し（getDoubleInput、getIntegerInput、getStringInput など）と、実行中の XML スクリプトに情報を返す 1 つ以上の 関数出力パラメータ呼び出し（addParameterReturn など）が含まれます。

```
private class AddFunction extends BaseFunction {
  public void execute() {
    Double left = getDoubleInput("left");
    Double right = getDoubleInput("right");
    addParameterReturn("sum", left + right);
  }
  @Override
  public void onFailure(Error e) {
    throw new UnsupportedOperationException("Not supported yet.");
```

```
  }
  @Override
  public void onFailure(Exception e) {
    throw new UnsupportedOperationException("Not supported yet.");
  }
}
```

入力を受け取るコール（getDoubleInput など）は、AQUA XML テスト実行環境から指定された変数 の値を取得します。同様に、出力パラメータの呼び出しは、環境に出力を返します。プラグインのコードは JAR ファイルにコンパイルされます。

カスタムプラグインのリリースバージョンは、以下の関数で定義されます。

```
  @Override
  public void version(){
    return "1.0.0"; // ex:plug-in release
    version 1.0.0
    }
```

プラグインのバージョンは、次のような形式になっています：

Major.Minor.Non-Functional

Major は、プラグインリリースにおける重要な機能変更を表します。例えば、新しいテスト入力パラメータの導入などです。

Minor は、プラグインリリースのマイナーな機能変更を表します。例として、テスト機能ロジックの更新が挙げられます。入出力パラメータのシグネチャは変更されません。

Non-Functional は非機能的な変更です。例としては、インラインコメントの追加やリファクタリングがあります。

このプラグインバージョンを使用して、社内のプラグインリリース管理プロセスに従ってカスタムプラグイン開発を管理してください。

注意：AQUA 3.00 より前に開発されたカスタム プラグインの場合は、AQUA 3.00 で使用する前に、前の手順に従ってバージョン関数の実装を追加し、プラグインをコンパイルし、AQUA インストール ライブラリ ディレクトリに再デプロイします。

4.2.2 AQUA v3.0 Plugin – コンポーネント構成

新しいプラグインを構成する前に、(AQUA を実行している) Apache Tomcat サービスを停止します。
コンパイルされたプラグイン JAR ファイルは、AQUA インストール ライブラリ ディレクトリに配置されます。
（例えば、C:¥Program Files (x86)¥Apache Software Foundation ¥Tomcat 8.5¥webapps¥AQA¥WEB-INF¥lib)
Component.xml ファイルには、AQUA システムで使用されるプラグインへの参照を含む、AQUA 設定の詳細が含まれています。このファイルは、WEB-INF¥classes フォルダーにあります。
（例えば、C:¥Program Files (x86)¥Apache Software Foundation¥Tomcat 8.5¥webapps¥AQA¥WEB-INF¥classes)

このファイルは、クライアント・プラグインへの参照を含むように変更されます。ライブラリ名は Plugins というエントリに追加されます。このエントリは、カンマ（スペースなし）で区切られた複数のプラグインを参照できます。

次の例では、プラグイン TestFunctionPlugin が、以前に plugin1 と plugin2 が追加されたリストに追加されます。

```
<entry key="Plugins">plugin1,plugin2,TestFunctionPlugin</entry>
<entry key="plugin1">com.client.plugin1.plugin.MyFunctions1</entry>
<entry key="plugin2">com.client.plugin2.plugin.MyFunctions2</entry>
<entry key="TestFunctionPlugin">com.uhn.TestFunctionPlugin.plugin.MathFunctions</entry>
```

ライブラリ名をパッケージ名にリンクする <entry> が、component.xml ファイルに追加されます。
JAR ファイルをライブラリディレクトリにコピーし、設定ファイルを変更した後、Tomcat で AQUA アプリケーションを再起動する必要があります。
設定されたプラグインのバージョンは、AQUA の System Settings 表示の System Data にも含まれます。

4.2.3 AQUA v3.0 Plugin – Test XML

クライアント プラグイン関数は、次のようにテスト XML スクリプトから呼び出すことができます。

```
. . .
<set name="left" value="25.2" />
<set name="right" value="10.4" />
<function name="add" plugin="TestFunctionPlugin">
   <input name="left"/>
   <input name="right"/>
   <return name="sum"/>
</function>
. . .
```

<function> 要素の name 属性は関数名を参照し、plugin 要素はライブラリ名を参照します。 <input> と <return> 要素は、クラスで定義された Function Input Parameter および Function Output Parameter と一致します。

4.3 AQUA v3.0 Plugin – 付録

```
package com.uhn.TestFunctionPlugin.plugin;                    1

import com.uhn.GWTServer.common.Features;
import com.uhn.GWTServer.common.Plugin;
import com.uhn.webtools.AQA.server.engine.TestFunction;
import com.uhn.webtools.AQA.server.functions.BaseFunction;

public class MathFunctions extends Plugin {                   2

 @Override
 public Features init() {
   return null;
 }

 @Override
 public void cleanup()  {
 }

 public TestFunction add() {                                  3
   return new AddFunction();
 }

 private class AddFunction extends BaseFunction {
   public void execute() {
     Double left = getDoubleInput("left");                    4
     Double right = getdoubleInput("right");
     addParameterReturn("sum"), left + right);
   }                                5

   @Override
```

```
    public void version (){
      return "1.0.0"
    }

    @Override
    public void onFailure(Error e) {
      throw new UnsupportedOperationException("Not supported yet.");
    }
    @Override
    public void onFailure(Exception e) {
      throw new UnsupportedOperationException("Not supported yet.");
    }
  }
}
```

表 4.1　凡例

No.	**説明**
1	Package Name
2	Class Name
3	Function Name
4	Function Input Parameter
5	Function Output Parameter
6	Version of plugin release