OZ1 JP-LINKアダプターユーザーガイド

バージョン *1.2*

内容

はじめに 3
ダッシュボード 3
　ダッシュボードビュー 3
データベース 4
　データベースビュー 4
　データベースの追加・編集 5
サブシステム 7
　サブシステムビュー 7
　サブシステムの追加・編集 8
　サービスの追加・編集 11
　サービステストビュー 20
サービス 22
　サービスビュー 22
　　サービスセレクトビュー 23
　サービスビューの追加・編集 24
　サービステストビュー 24
　サービスを削除する 24
証明書 25
　証明書の表示 25
　　アダプターサーバー証明書 26
　　クライアント証明書 26
　証明書の追加・編集 27
Query Logs 29
　Query Logs リストビュー 30
　Query Log view 31
Settings (設定) 32
　設定画面 32
　ユーザー 32
　　ユーザー一覧表示 33
　　ユーザービューの追加・編集 33
　監査ログ 34

はじめに

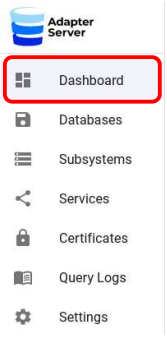
このドキュメントは、OZ1 Universal JP-LINK Web アプリケーションの機能およびそのユーザインタフェースについて説明します。

OZ1 Universal JP-LINK Web アプリケーション(以下、OZ1アダプター)は、X-Roadサービスを管理するーシステムであり、データベースのコンテンツをX-Roadプロトコル上のWebサービスとして、他の X-Road ユーザーと共有するための動的なアダプタ・サーバーです。
OZ1アダプターは、管理者がデータベース接続を定義し、そのデータベース接続を使用するX-Roadサービスを作成・設定することができます。
X-Roadサービスの作成、開始、停止、変更、削除を行うことができ、これらのX-Roadサービスをサブシステムに編成することができます。各X-Roadサービスは、データベースのSQLクエリをX-Roadのリクエストおよびレスポンススキーマに関連付けます。

他に、システムユーザー／監査ユーザの追加・削除、証明書の導入とこれを利用したセキュリティサーバーとのセキュアな通信の実現、クエリログの管理、監査ログ（アクションログ）の管理を行うこともできます。

※もし、ご利用中にバグや機能不備等を見つけられましたら、ご報告いただけますと幸いです。
特に重大であると判断したものは迅速に対応いたします。
また、マニュアルに対するご指摘もあわせていただけますと幸いです。

ダッシュボード



異なる UXA システムの現在の状態を表示する、UXA のデフォルトのホームページです。

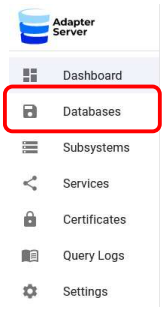
ダッシュボードは、メニューの”Dashboard ”からアクセスできます。

ダッシュボードビュー

ダッシュボードをクリックすると、それぞれのシステムビューに移動します。

1. システム名とそのシステム内のアイテム数
2. ユーザーシステムと総ユーザー数のうちの有効数

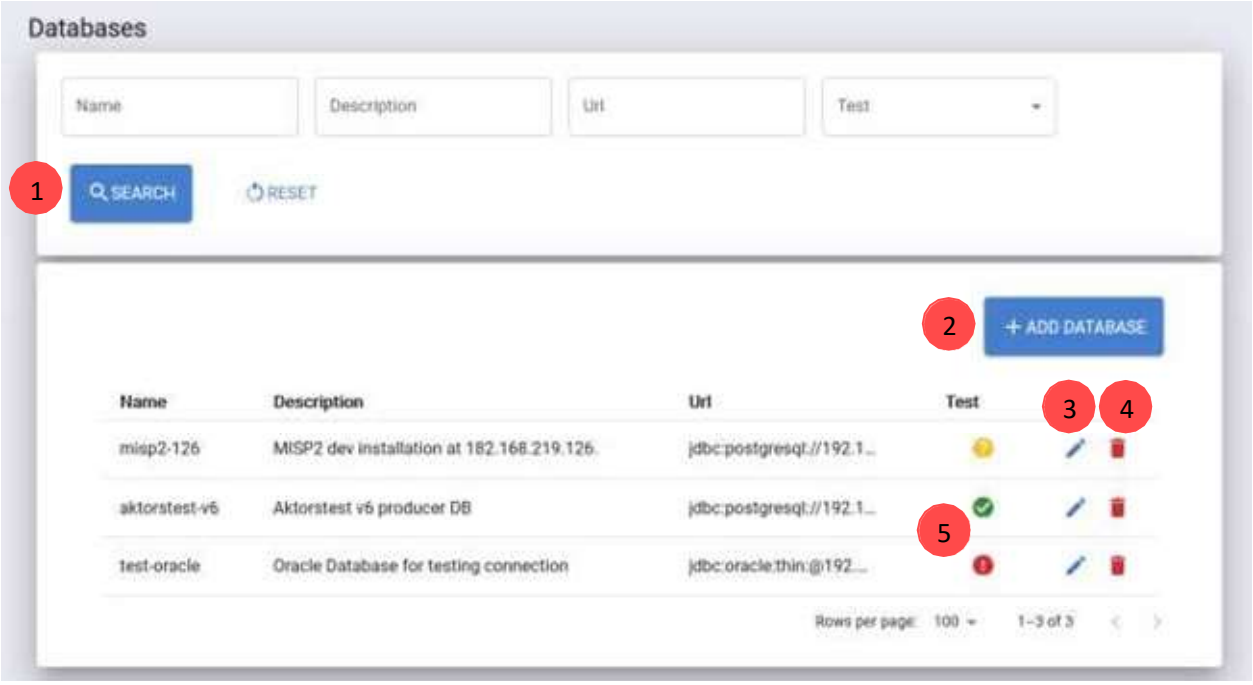
データベース



データベースは、サービスを定義するために使用されます。

データベース・ビューは、メニュー項目「データベース」からアクセスできます。

データベースビュー

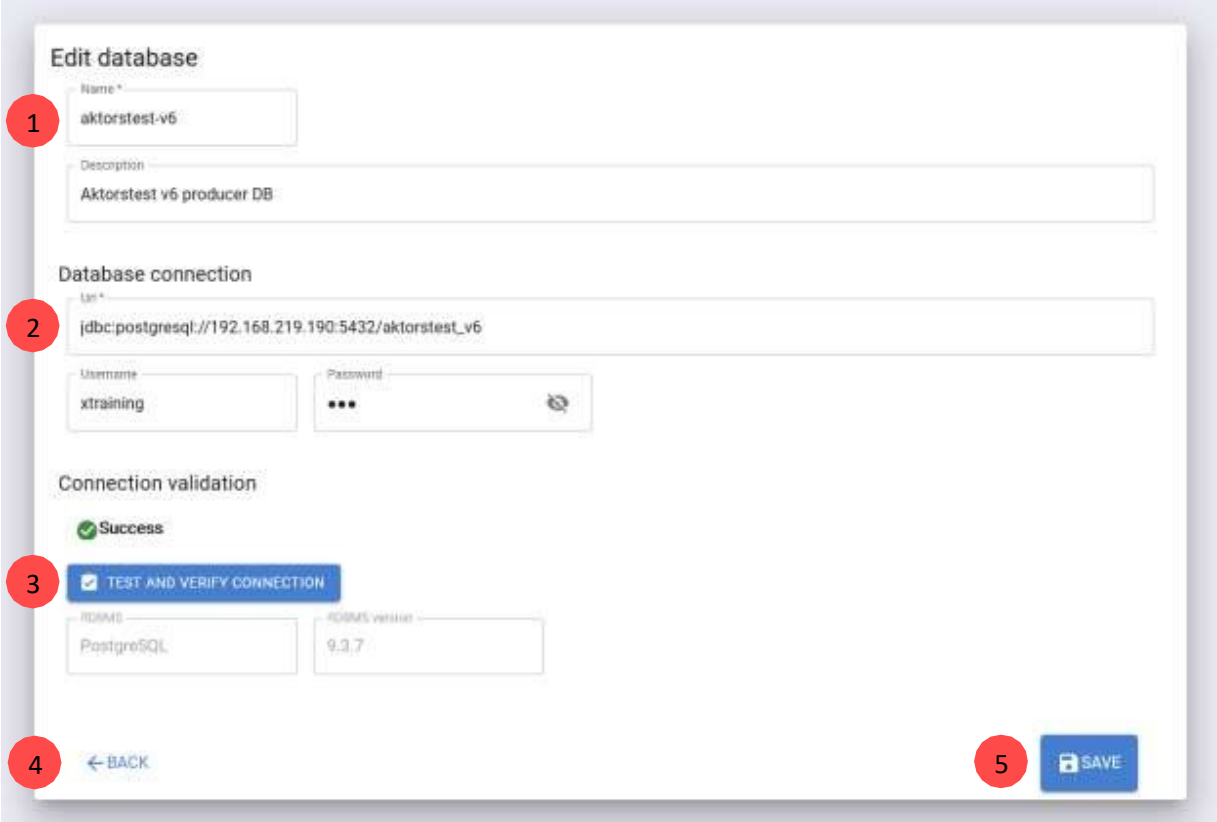


画像 *1* データベースビュー

番号の説明はそれぞれ以下の通りです。

1. データベースの検索/フィルタリング、検索ボタンをクリックするとフィルタが適用される
2. 新規データベースの追加
3. 与えられたデータベースの編集
4. 依存するサービスがない場合、与えられたデータベースを削除する
5. データベース接続テスト結果
    a. – 接続テストに成功
    b. – テスト未実施
    c. – 接続テストに失敗

データベースの追加・編集



画像 *2* データベース追加・編集画面

番号の説明はそれぞれ以下の通りです。

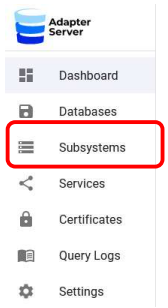
1. データベース情報です。名前は一意でなければなりません。
2. JDBC の URL（MySQL の例）、データベースのユーザー名とパスワードです。データベースの URL とユーザー名の組み合わせは一意でなければなりません。
3. 指定されたURL とユーザーでデータベース接続をテストします。接続が成功すると、RDBMS および RDBMS バージョンのフィールドが自動的に入力されます。
4. データベース一覧へ戻る
5. 新しいデータベースの作成、または既存のデータベースへの変更の保存

サポートRDBMSとそのバージョン

RDBMS製品名	動作確認済バージョン	備考
MySQL	8.x	
	5.7	
	5.6	
MSSQL	Azure SQL Database	
	Azure Synapse Analytics	
	Azure SQL Managed Instance	
	SQL Server 2019	
	SQL Server 2017	
	SQL Server 2016	
	SQL Server 2014	
	SQL Server 2012	
PostgreSQL	14	
	13	
	12	
	11	
	10	
	9 *	非推奨
	8 *	非推奨
MariaDB	All Version	
OracleDB	Database 21.x	
	Database 19.x	
	Database 18.3	
	Database 12.2	
	Database 12.1	
	Database 11.2.0.4	

* 旧バージョンのPostgreSQL対応はユーザーが提出したバグレポートに依存しているため、信頼できない可能性があります。動作は確認しておりますが、これらのバージョンの利用は推奨しておりません。

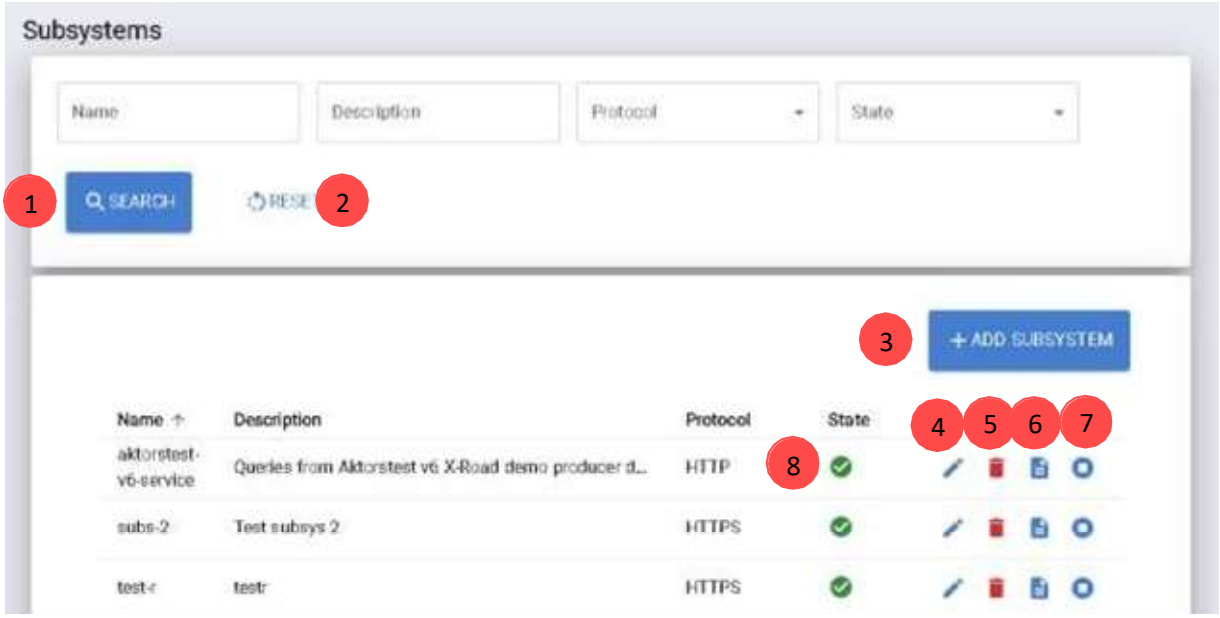
サブシステム



サブシステムは、SECURITY SERVER サービスのグループを表し、それらのサービスがリストアップされ、記述されている 1 つのWSDL ファイルに対応しています。サブシステムは、リストアップ、作成、編集、および削除の操作を行えます。既存のサブシステムは開始および停止することができ、そのサービスは追加および削除することができます。

サブシステム・ビューは、「サブシステム」メニューからアクセスできます。

サブシステムビュー



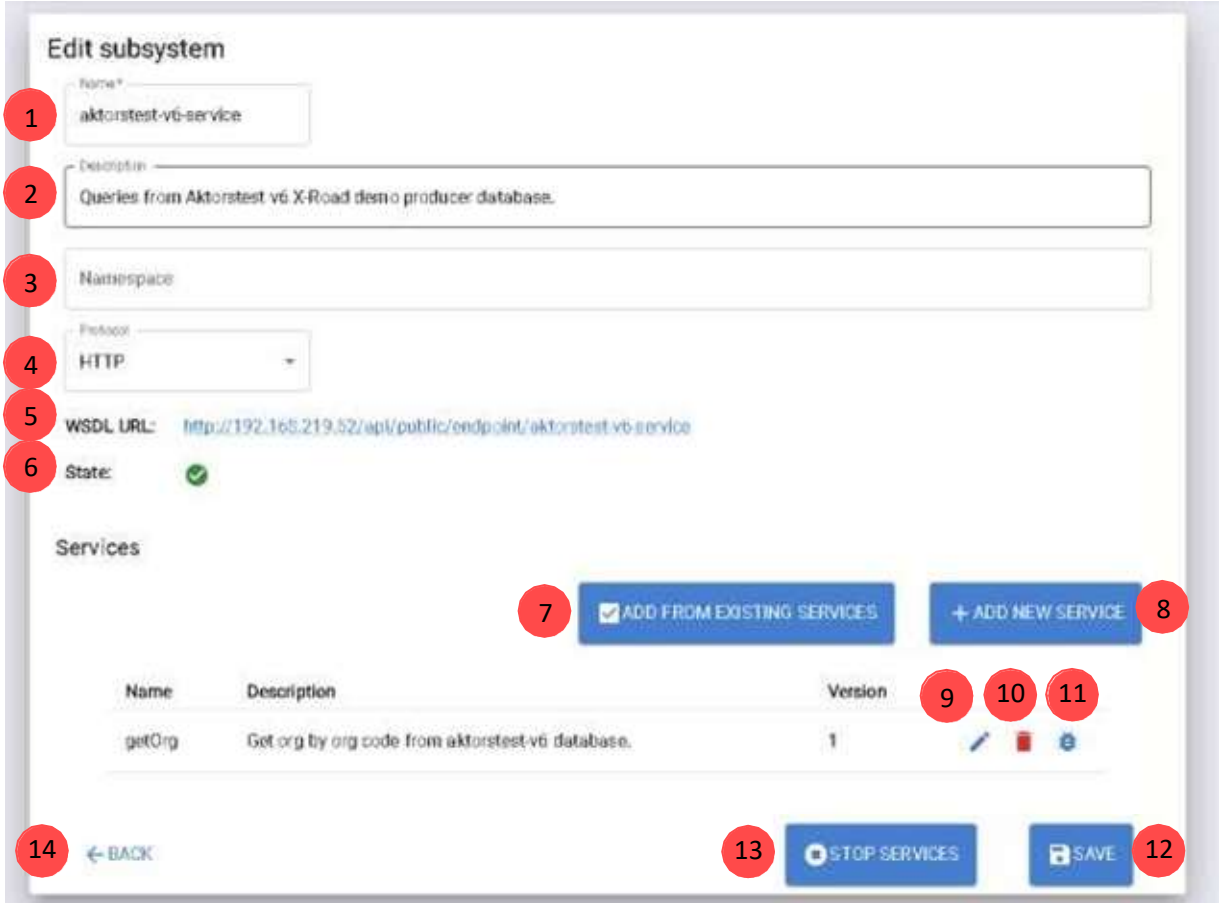
画像 *3* サブシステムビュー

番号の説明はそれぞれ以下の通りです。

1. サブシステムのフィルタリングされたリストの検索
2. 検索フィルターフォームを初期値に戻す
3. サブシステムの新規作成
4. サブシステムの編集

5. サブシステムとそのすべてのサービスを削除します。アイコンをクリックすると、最初に確認ダイアログが表示されます。サブシステムを削除すると、そのサブシステムに関連するすべてのサービスも削除されます。
6. サービスの WSDL をブラウザの新しいタブに表示します。
7. サブシステムの状態を変更します。 アクティベートしてサービスを開始し、ディアクティベートしてサービスの実行を停止します。
8. サブシステムの状態を示します。非アクティブの 場合となり、 アクティブの場合となります。

サブシステムの追加・編集



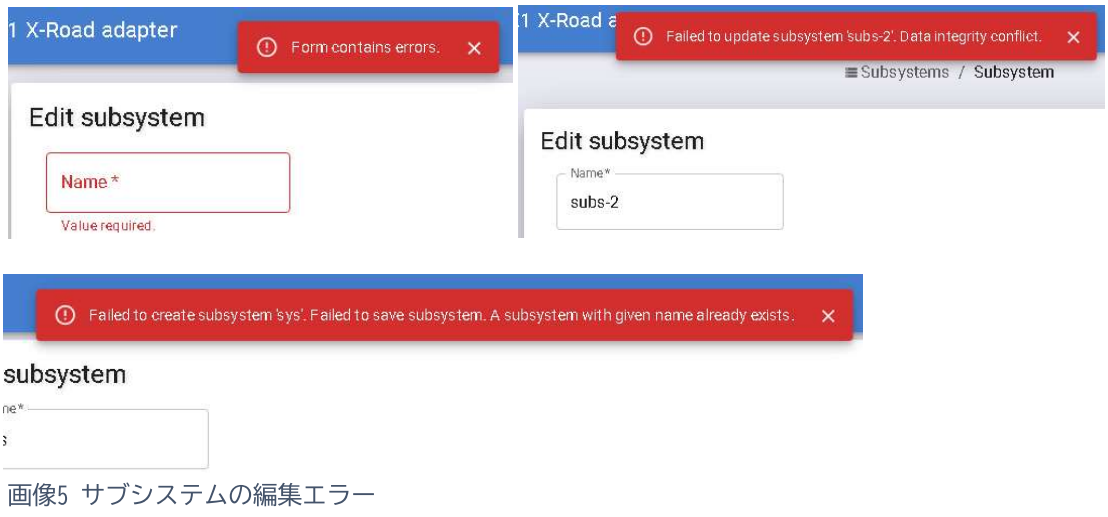
画像 *4* サブシステムの追加/編集画面

1. サブシステムの短い技術名。これは、現在のサブシステムを参照するために、WSDL URL パスで使用されます。この値は必須で、アプリケーション全体で一意でなければなりません。すでに存在する名前で新しいサブシステムを保存しようとすると、「データ整合性の衝突」エラーが表示されます。
2. サブシステムの説明を記述します。値は任意です。
3. SECURITYSERVERサービスのターゲットネームスペースとなります。空にすることもできます。その場合、ターゲットネームスペースは、http://<SUBSYSTEM_NAME>.oz1.life/producerのようにサブシステム名から自動的に導き出されます。名前空間のURLは、実際のウェブ上の場所を指す必要はなく、XMLスキーマ要素のセットを区別するために使用されるだけであることに注意してください。

4. SECURITY SERVER の SOAP リクエストエンドポイントとWSDL の URL プロトコル、HTTP または HTTPS のいずれかを選択します。
5. WSDL URL は、設定されたURL プロトコル、UXA ウェブアプリケーションのURL、およびサブシステム名から派生したものです。WSDL URL への HTTP GET メソッドによるリクエストは、レスポンスとしてWSDL コンテンツを返し、POST メソッドはSECURITY SERVER SOAP サービスエンドポイントにリクエストデータを送信します。
6. サブシステムの状態：非アクティブの 場合となり、アクティブの場合となります。サブシステムの状態は、サブシステムが保存された後、［サービス開始／サービス停止］ボタンから設定されます。
7. 現在のサブシステムへの既存サービスの選択を可能にする、サービス選択ビューを開きます。選択されたサービス間の関係は、選択操作中は永続化されません。そのため、サブシステムでは、新しいサービスのセットを選択した後に「保存」アクションを実行する必要があります。
8. 新しいサービスを作成して、現在のサブシステムに追加します（サービスを作成した後に、サブシステムの保存ボタンをクリックする必要があります）。
9. サブシステムサービスの編集
10. サブシステムからサブシステム・サービスを削除します。これはサービスを削除するのではなく、サブシステムからの関連付けを解除します。サービスは、サービス選択リストまたは「サービス」メニュー項目から引き続きアクセスできます。
11. サービスのテスト アイコン SECURITY SERVER SOAP サービスのテスト画面にリダイレクトし、SOAP リクエストを変更してサービスのエンドポイントに送信して疎通確認を行うことができます。テストのためには、サブシステムの起動が必要です。「Start services」ボタンで起動してください。サブシステムが起動していない場合、サービステストビューにリンクを示すアイコンは表示されません。

    サブシステムとそのサービスの関連付けを保存します。サブシステムにサービスが追加または削除されると、選択された関連付けをデータベースに保存するよう保存ボタンが赤い色で SAVE 強調表示されます。

    サブシステムを保存の際、エラーが発生する場合があります。最も基本的なものとして検証エラーであり、例えば、必須フィールドが空のままで、ユーザーが保存しようとすると、以下の左図のようなエラーメッセージが表示されます。



画像5 サブシステムの編集エラー

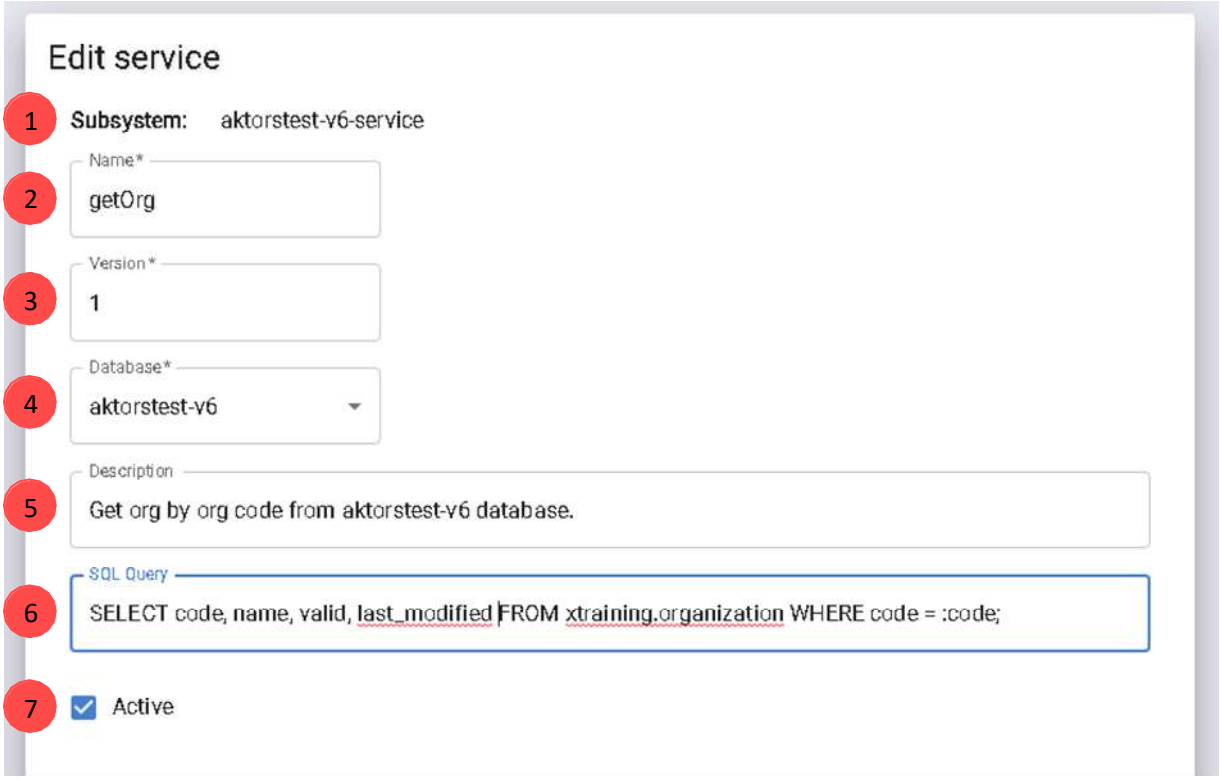
上：検証エラー　下：名前が一意でないことによるデータ整合性エラー

サブシステムはUXA アプリケーション全体で一意の名前を持つ必要があります。
既に存在する名前で保存しようとすると、一般的なデータベースの競合エラーメッセージが表示され保存できません。(画像5下参照)
(一意性の制約は、関連するサブシステム内のサービス名にも適用されます。)

他にもデータベーススキーマの文字列長や文字列形式の制約違反などにより、”Data integrity conflict”エラーが発生する可能性があります。このエラーの正確な原因は、Webアプリケーションのシステムログファイルに記載されており、そこにはエラーとそのスタックトレースが記録されているため、このエラーの詳細を知りたい場合にはWebアプリケーションのシステムログファイルを参照してください。

12. サブシステムの状態を変更するための Start/Stop サービスボタンです。サービスと WSDL エンドポイントは、アクティブに起動されたサブシステムでのみアクセスでき、そうでない場合は *SOAP-ENV:Fault* でラップされたエラーメッセージが返されます。新しいサブシステムが少なくとも1つのサービスと一緒に保存されている場合、保存後のデフォルトの状態は非アクティブ（サービスがまだ開始されていない状態）となります。
    サービスの開始をユーザーに促すため、ボタンは赤い背景で  強調表示されます。
13. サブシステムを保存せずに、サブシステム・リストに戻ります。

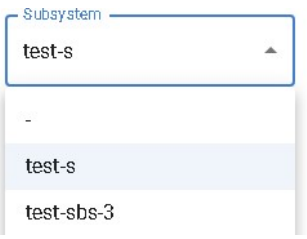
サービスの追加・編集



画像 *6* サービス追加・編集画面セクション *1/3* – サービスデータフィールド

1. 現在の操作が関連付けられているサブシステム。サービスが特定のサブシステム・ビューから編集用に開かれた場合、サブシステムの値は固定されます。

   サービス」のリストビューから編集ビューを開いた場合、サブシステムは、定義されたすべてのサブシステムの選択フィールドから選ぶことができます。



画像 *7* サービス編集画面のサブシステム選択フィールド（「サービス」リストから移動します）

2. サービスの名称。サービス名は、SECURITY SERVER SOAP メッセージ内の 2 つの異なる場所に設定されます。

   a. これは SECURITY SERVER リクエストの SOAP メッセージヘッダの *xrd:service/iden:serviceCode* 要素に設定されており、リクエストが来ると UXA サービスエンドポイントで解析され

る。解析された *serviceCode* 要素の内容は、現在のSECURITY SERVER リクエストをUXA システムで定義された応答サービスと関連付けるために使用される。言い換えれば、*serviceCode* ヘッダはUXA システムのサービス名に対応する。

b. サービス名は、SECURITY SERVER リクエストの *SOAP-ENV:Body* 要素内で直接、オペレーションスキーマのルート要素の XML 要素タグ名としても使用される。同様に、SECURITY SERVER レスポンスボディのルート XML 要素のタグ名も、”Response ”を追加することで、サービス名から派生する。

つまり，サービス名は，XML 仕様のタグ名形式に準拠していなければなりません。文字で始まること，” ”や”: ”などの特殊文字は使用できません。 一般的にはキャメルケース形式が使用されますが、アンダースコアベースの形式も選択肢の一つとなります。

また、サービス名は、サブシステム内で一意でなければなりません。同じサービスの複数のバージョンを、サブシステム内で一度にアクティブにすることはできません。

3. バージョン番号 - サービスへの後方互換性のない変更を記録するための整数です。SECURITY SERVER SOAP メッセージヘッダ *xrd:service/iden:serviceVersion* 要素に対応しています。SECURITY SERVER メッセージの *serviceVersion* ヘッダーの内容を UXA のバージョン番号と互換性のある整数形式に変換するために、接頭辞 ”v ”を削除している。バージョン番号は、受信した SECURITY SERVER SOAP リクエストをUXA システムで定義されたサービスにマッピングするために、サービス名と併せて使用されます。

4. SQL クエリが実行されるデータベース。Select-element には、システムで定義されているすべてのデータベースがオプションとして表示されます。

5. サービスの説明を記述します。主に、サービスの開発者と利用者のためのドキュメントとして機能します。WSDL 内のオペレーションの *wsdl:documentation/xrd:title* 要素に追加されます。

6. サービス「SECURITY SERVER」のレスポンスのために、選択されたデータベースからデータを取得する、入力パラメータ・プレースホルダ付きの SQL クエリです。

   SECURITY SERVER のリクエスト／レスポンスメッセージとSQL クエリとのマッピングは、以下のように***Input parameters*** と ***Output parameters*** のセクションでXML スキーマを宣言することで実現します。

   基本的に、入力パラメータ *Name* のテキストフィールドの値は、”: ”を前置することで、クエリパラメータのプレースホルダーとして SQL クエリで使用することができます。

   出力に関しては、SQL クエリの結果のカラム名が、定義された出力パラメータの *Name* フィールドの値に直接マッピングされます。

   たとえば、次のようなクエリがあります。

```
SELECT code, name, valid, last_modified FROM xtraining.organization WHERE code = :code
```

上記の SQL クエリは、*xtraining.organization* テーブルの 4 つのカラム（*code*, *name*, *valid*, *last_modified*）のエントリを検索し、*xtraining.organization* テーブルのカラム *code* が *code* という入力パラメータと一致する結果の行のみを含みます。

プレースホルダーの":"から始まるプレフィックス構文は、入力パラメータのプレースホルダーをデータベースのカラム名等と区別するために使用されます。

SQLクエリの結果のカラム名は、同じ *Name* フィールド値を持つ Output パラメータ（*code*, *name*, *valid*, *last_modified*）に直接マッピングされます。
この特性のためにCOUNT関数やSUM関数等、SQL実行結果に明示的な列名が割り当てられない関数を利用された場合、値を取得することができません。これらの関数を利用する場合には、明示的にAS句にて列名を定義し、マッピングしてください。

また、単純に異なるカラム名でデータを取得したい場合にもAS句を使用できます。

例えば、以下のクエリは、portalテーブルの全レコード件数をカウントし、countPortalsカラムとして出力結果を返します。アプリケーションはcountPortalsというOutput パラメータを定義しマッピングすることで、この出力結果を受け取ることができます。

```
SELECT count(*) AS countPortals FROM portal
```

IN 句を使用することもできます。

```
SELECT code, name, valid, last_modified FROM xtraining.organization WHERE code IN :code
```

注意： IN句の後に定義する入力パラメータ用のプレースホルダーを () で囲わず、上記の例の通りに記載ください。実際のIN句に必要な () はアプリケーションにて補完します。

IN 句の引数（上記の例では:code）は、Input パラメータを"Array"型で宣言する必要があります。加えて、:codeパラメータの配列を格納する為の特殊なInput パラメータが必要になる点にご注意ください。名称は任意ですが、格納されているパラメータ（上記の例では:code）と関連のある分かりやすい名称にすることを推奨します。
上記の例ですと、実際の設定は次のようになります：

リクエストWSDLの記述例は次のようになります：

```
<codes>
  <code>val1</code>
  <code>val2</code>
  <code>val3</code>
</codes>
```

codesパラメータ内にcodeパラメータを1つ以上列挙する形式であることに注意してください。

INSERT/UPDATE/DELETE ステートメントもサポートされています。以下の例では、更新された

行の *id* 列の値を指定して、*query_name* テーブルの *description* 列を希望の値に更新しています。

```
UPDATE query_name SET description = :description WHERE id = :qNameId
```

description と *qNameId* はどちらも入力パラメータで、前の例と同様にリクエストから取得されますが、ステートメントの出力は宣言されていません。INSERT/UPDATE/DELETE のSQL クエリは、影響を受けた行の数を整数として返します。この数値は、サービスの出力パラメータとして公開することができます。そのためには、出力パラメータセクションで、任意の名前を付けた Number 型の出力パラメータを 1 つ宣言します。SQL ステートメントのカウント結果は、SECURITY SERVER サービス応答メッセージの対応するフィールドに書き込まれます。

DATE および TIMESTAMPのデータ型において、SOAPメッセージの形式はそれぞれ次のようになります。

- DATE型: yyyy-MM-dd（例: 2022-02-28）
- TIMESTAMP型: yyyy-MM-ddTH:mm:ss.S（例: 2022-02-28T16:57:48.98）

X-Road SOAPメッセージのヘッダーの情報は、特別なプレースホルダー名を使うことで、その値を取得することができます。
例えば、次のようなSQLを記述することが可能です：

```
SELECT code, name FROM organization WHERE code = :HEADER_CLIENT_MEMBER_CODE
```

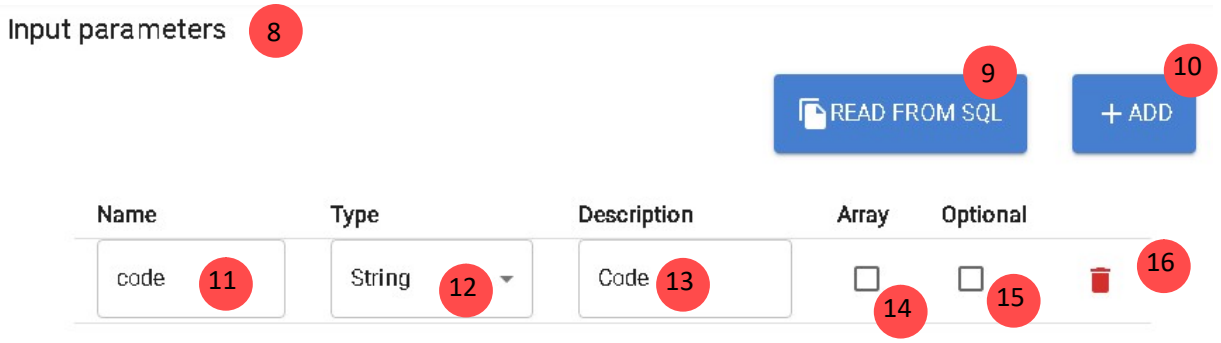
パラメータの値は、X-Road のリクエストメッセージヘッダから直接読み込まれます（上記の例の場合、:HEADER_CLIENT_MEMBER_CODEなので、リクエスト元のメンバーコード）。

以下が、使用可能な特別なプレースホルダーのリストです。

プレースホルダ名	取得元タグ (/SOAP-ENV:Envelope/SOAP-ENV:Header/*)	説明
基本情報		
:HEADER_PROTOCOL_VERSION	xrd:protocolVersion	プロトコルバージョン
:HEADER_MESSAGE_ID	xrd:id	メッセージID
:HEADER_USER_ID	xrd:userId	ユーザID
データプロバイダーの情報（サービス提供側）		
:HEADER_SERVICE_XROAD_INSTANCE	xrd:service/iden:xRoadInstance	XROADインスタンスID
:HEADER_SERVICE_MEMBER_CLASS	xrd:service/iden:memberClass	メンバークラス
:HEADER_SERVICE_MEMBER_CODE	xrd:service/iden:memberCode	メンバーコード
:HEADER_SERVICE_SUBSYSTEM_CODE	xrd:service/iden:subsystemCode	サブシステムコード
:HEADER_SERVICE_SERVICE_CODE	xrd:service/iden:serviceCode	サービスコード
:HEADER_SERVICE_SERVICE_VERSION	xrd:service/iden:serviceVersion	サービスバージョン
サービスプロバイダーの情報（サービス利用側）		
:HEADER_CLIENT_XROAD_INSTANCE	xrd:client/iden:xRoadInstance	XROADインスタンスID
:HEADER_CLIENT_MEMBER_CLASS	xrd:client/iden:memberClass	メンバークラス
:HEADER_CLIENT_MEMBER_CODE	xrd:client/iden:memberCode	メンバーコード
:HEADER_CLIENT_SUBSYSTEM_CODE	xrd:client/iden:subsystemCode	サブシステムコード
代表者情報		
:HEADER_REPRESENTED_PARTY_CLASS	repr:representedParty/repr:partyClass	代表メンバクラス
:HEADER_REPRESENTED_PARTY_CODE	repr:representedParty/repr:partyCode	代表メンバコード

注意：
これらの特別なプレースホルダーは大文字小文字を区別します。
よって、:HEADER_USER_IDと:header_user_idは異なるプレースホルダーとして評価されます。
また、これらの特別なプレースホルダーはアプリケーションによって利用用途が固定されている為、利用者はこれらの特別なプレースホルダーをカスタムして使うことは出来ません。

7. サービスの状態
   Active (Active にチェックが入っている*)*はサービスが利用可能であることを意味します。*Not active (Active* にチェックが入っていない*)*は、サービスはサブシステム内にとどまっているが、利用できないことを意味します（*SOAP- ENV:Fault* が返されます）。



画像 *8* サービス追加・編集 フォームの入力パラメータ部

8. サービス編集フォームの「入力パラメータ」セクションとなります。このセクションでは、SECURITY SERVER リクエストメッセージのスキーマを宣言することができる。このスキーマは、SECURITY SERVER リクエストメッセージから特定のデータ値を抽出し、それをプレースホルダーとして SQL クエリに渡すために使われる。

   SECURITY SERVER リクエストメッセージがサブシステムのエンドポイントに到着し、UXA バックエンドでマッチするサービス宣言が見つかった後、入力パラメータセクションの設定に基づいて SECURITY SERVER リクエストメッセージボディからそれぞれの入力パラメータ値が解析され、適切な SQL データタイプに変換され、パラメータ化された SQL クエリに渡されます。解析されたパラメータ値は、":"接頭辞付きのパラメータ名で参照できます。

9. SQL クエリに入力された内容からリクエスト・パラメータを解析し、リクエスト・パラメータ・テーブルを自動的に作成します。既存のフィールドがある場合には、既存のフィールドに書き込むためにユーザーの確認を求めます。

   IN句を利用したプレースホルダは自動的にArray型としてマークされ、XML-element型のラッパーで囲まれる。(詳細は6.を参照ください)

   ただし、データ型を反映する機能は現在ありません。データ型はデフォルトで *String* が選択されます。必要に応じて、適切なデータ型に変更してください。

   パラメータが SQL クエリから解析されてリストに追加されても、サービスと関連するパラメータが自動的には保存されません。保存するには、*Save*（または *Save and test*）ボタンを押下する必要があります。

10. フォームに新しい入力パラメータを追加します。このボタンをクリックすると、入力パラメータテーブルの最後に新しいフィールドが追加されます。

    ユーザーは、要素に適切な名前、データタイプ、説明を挿入し、それに応じて SQL クエリを調整し、「保存」（または「保存してテスト」）をクリックします。

    追加されたパラメータは，サービスリクエストボディのルート要素の最初の子要素として解釈されます．後者の要素（*SOAP-ENV:Body* 要素の最初で唯一の子要素）は、タグ名がサービ

ス名と同じ値に固定されており、変更できないため、入力パラメータの表には表示されません。

表の中に ”XML Element ”タイプのパラメータが宣言されている場合、追加された要素は、”XML Element ”タイプの最も近い上記入力パラメータの子要素として解釈されます。

11. 入力パラメータ名- 必須。 両方の役割を果たします。

    a. SOAP リクエストメッセージに含まれるXML 要素のタグ名で、ここからテキストコンテンツが解析され、SQL データタイプに変換され、パラメータとしてSQL クエリに渡される。

    b. SQL クエリで入力データを参照するためのプレースホルダー（”:”-プレフィックス付きの名前の値）。

    このため、*Name* は XML タグ名の文字列形式を満たす必要があります。例えば、特殊文字を含んではならず、数字で始まっていてはいけません。また、名前は SECURITY SERVER リクエスト内で一意でなければならず、プレースホルダーへの結合が曖昧にならないようにする必要があります。

12. 入力パラメータのデータタイプ。主に、パラメータ化された SQL クエリを実行するために、リクエストのデータフィールドを SQL データタイプに変換するために使用される。また、WSDL の XML スキーマで宣言的に使用され、スキーマ内のXML 要素の子と親の関係を解釈します。

    注意事項としてリクエストパラメータには正しいデータタイプを設定してください。そうしないと SQL 文がエラーとなる可能性があります。

    例えば、次のような SQL クエリがあります。

    ```
    SELECT count(*) AS countPortals FROM portal WHERE id > :minId
    ```

    入力パラメータ *minId* の *Type* が *String* に設定されている状態でサービスを実行すると、SOAP レスポンスに以下の faultstring が返されます。「*ERROR: operator does not exist: integer > character varying... ”*となっています。*minId* のデータタイプを *Number* に変更すると、サービスは正常に結果データを返します。

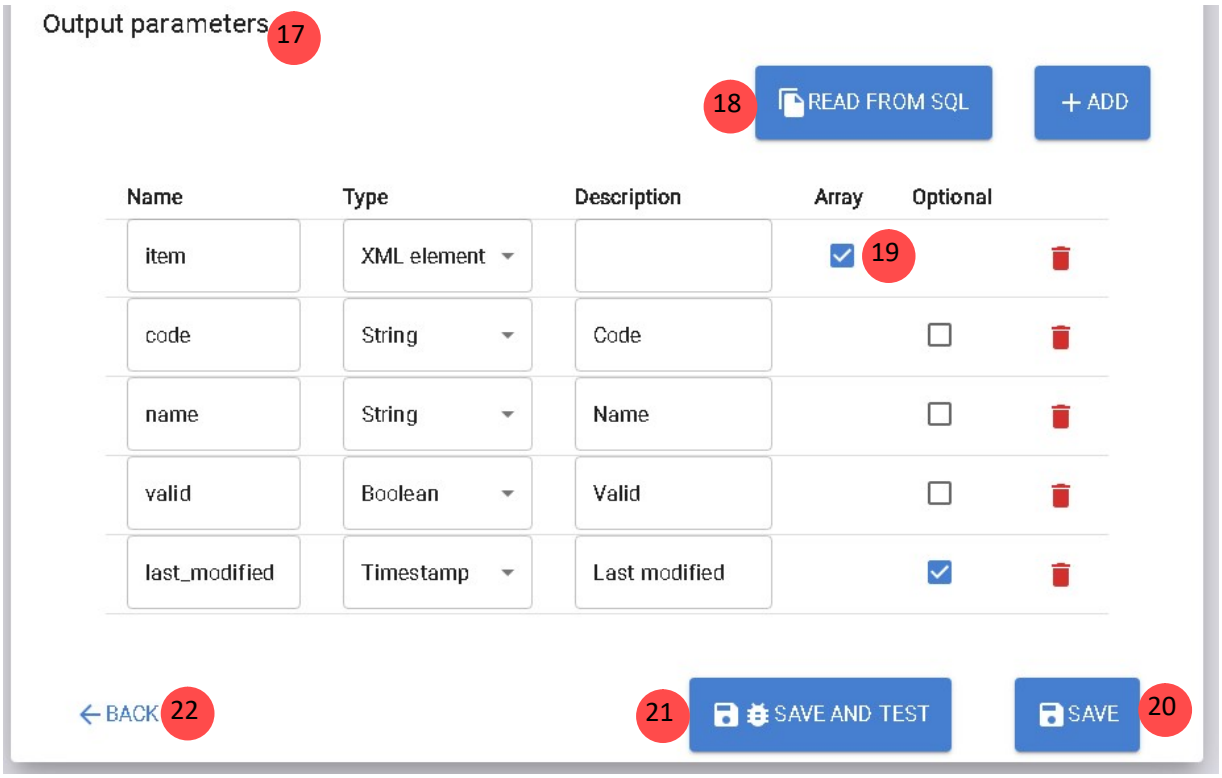
13. 入力パラメータの説明。この値は必須ではありま せんが 、*xsd:element/xsd:annotation/xsd:appinfo/xrd:title* パスでWSDL メッセージスキーマに追加されるため、サービス開発者からサービス消費者に対して説明の役割を果たします。

    いくつかのサードパーティツール（MISP2 X-Road クライアントなど）は、これらの説明を解釈し、サービスの説明から解析されて自動生成されたサービスクライアントフォームのラベルを追加するために使用します。

14. リクエスト・スキーマで入力パラメータを複数エントリ（またはエントリなし）可能にする ”Array ”プロパティ。このプロパティは、IN-statement パラメータに使用される。リクエストスキーマで配列にできるのは、アトミックデータ型（XML 要素のラッパータイプ以外のすべてのデータ型）だけである。

15. スキーマフィールドの”Optional”プロパティとなります。チェックされていない場合、要素はSECURITY SERVER リクエストに必須となります。要素がないとSOAP-Fault になります。” Optional ”プロパティがチェックされている場合，SQL クエリのプレースホルダを初期化する際に，欠落している要素はNULL にバインドされます。これは WSDL にも反映されており、オプションの要素は *minOccurs="0"*属性で記述されます。
入力パラメータは、”Array ”または ”Optional ”のいずれかを指定することができる。

16. セットしたパラメータを削除します。保存ボタンが押下されるまで変更は行われません。



画像 *9* サービス追加・編集 ”フォームの出力パラメータ部

17. サービス編集画面 出力パラメータセクションは、入力パラメータセクションと似ていますが、データは逆方向に流れます：入力された SECURITY SERVER SOAP リクエストデータをSQL クエリパラメータにバインドする代わりに、出力パラメータセクションはSQL クエリの結果をSECURITY SERVER SOAP レスポンスにバインドします。表に記載されているパラメータは、SECURITY SERVER SOAP レスポンスメッセージボディのXML スキーマを表しています。

    リストへのパラメータの追加や削除、パラメータテーブルの Name、Type、Description、Optional 列のセマンティクスなど、ほとんどの仕組みは入力パラメータの場合と同等なので、よろしければ「入力パラメータ」の項の関連する説明を参照してください。

18. SQL クエリフィールドから出力パラメータを解析する方法も、入力パラメータを解析する方法と同様です。ただし、パラメータ名は SQL クエリの結果カラム定義セクションから見つけます。しかし、基本的には単純な SQL 文の出力パラメータ宣言に対してのみ動作し、複雑なSQLの場合には正確な解析が行われない場合もあります。

19. ”Array”プロパティは、出力パラメータテーブルの最初の行にのみ設定することができます。設定されている場合、対応するXML 要素は、XML メッセージの中で一度だけでなく、何度でも出現することを示します。

    例えば、与えられたSQL クエリが複数の行を返す可能性がある場合、「Array」プロパティを持つスキーマフィールドは Output parameters の下で宣言する必要があります。Array ”プロパティがチェックされている場合、”Optional ”プロパティは非表示になります。”Array ”プロパティは、*minOccurs=”0 ”*スキーマ属性も使用しているため、”Optional ”の上位互換と考えられます。

    SQL クエリが複数の行を返したものの、Output parameters テーブルに配列要素が宣言されていない場合、SOAP-Fault として faultstring *”Cannot assemble service response from SQL query results”* (SQL クエリは複数の行を返しましたが、応答メッセージのスキーマに配列定義がありません。) で返されます。

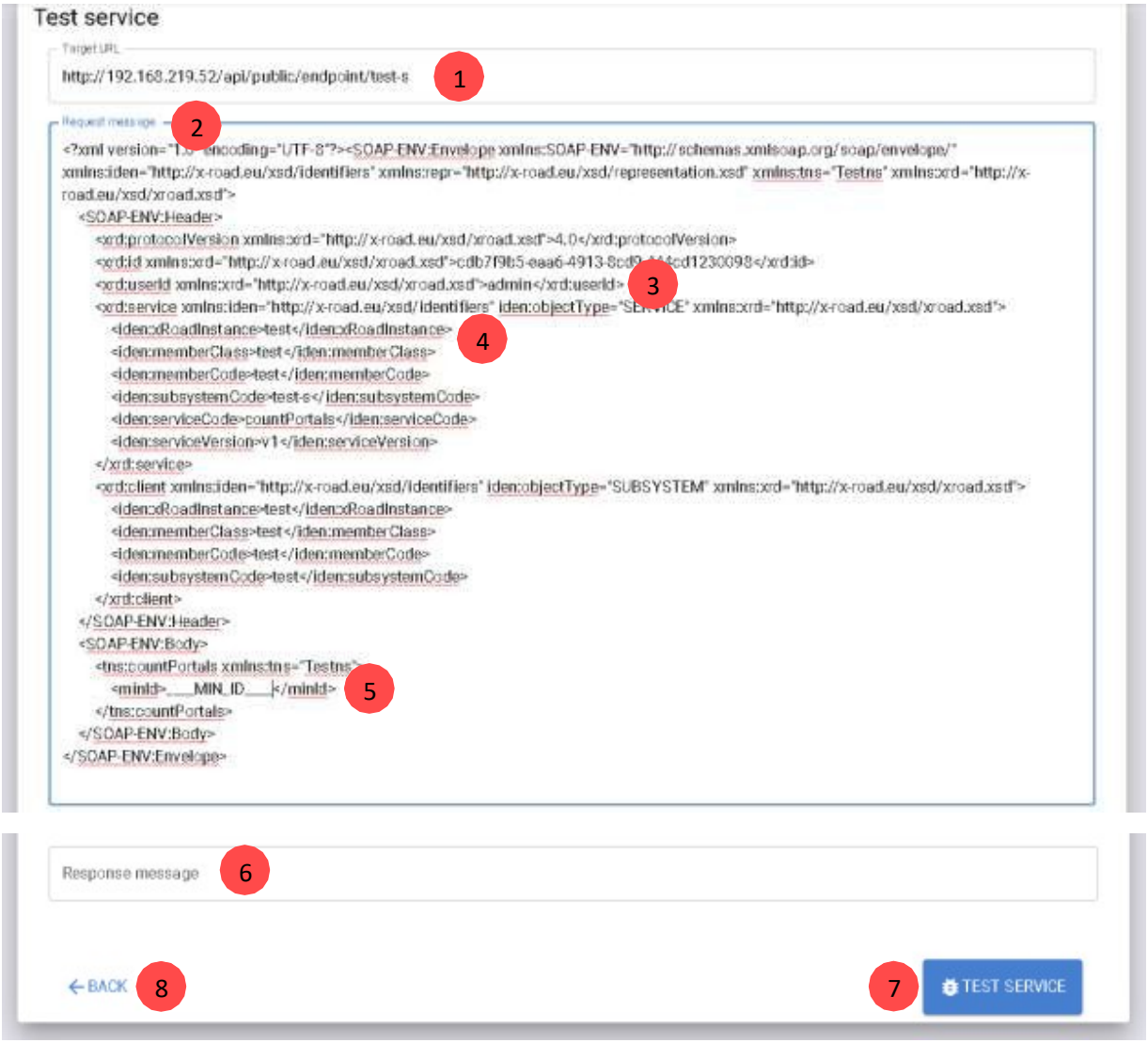
20. サービス宣言をその入力および出力パラメーターと共にデータベースに保存し、サブシステムの追加/編集ビューに戻って、サブシステム・サービスの下に新しく保存されたサービスを追加します。サブシステム・サービスの関連付けを持続させるには、「サブシステムの追加／編 集」ビューで「 保 存 」ボタンを再度押す必要があります。ユーザーが（サブシステムの追加／編集ビューではなく）「サービス」メニュー項目の下のリストからサービスの追加／編集ビューに移動した場合は、「保存」ボタンを押した後も現在のビューに留まり、左の「戻る」ボタンでサービスリストに戻ることができます。

21. サービスを保存し、サービス・テスト・ビューに移動します。基礎となるサブシステムがまだ保存されていない場合、このボタンは選択できません。

22. 前の画面に戻ります。ユーザーのホーム画面であるサブシステムの追加/編集ビューに戻るか、ユーザーのホーム画面である場合はサービスビューに戻ります。

サービステストビュー

SOAP リクエストメッセージを編集してサービスエンドポイントに送信し、返ってきた SOAP レスポンスメッセージを確認することができます。



画像 *10* サービステスト風景

1. SOAP テストリクエストメッセージが送信される、サービスサブシステム名とプロトコルから得られるサービスエンドポイントの URL です。テキストフィールドの値は、このビューに遷移すると自動的に入力されます。
2. サービスエンドポイントに送信されるサービス SOAP リクエストメッセージとなります。
3. SECURITY SERVER の *userId* ヘッダーフィールドには、UXA ウェブアプリケーションにログインしたユーザーのユーザー名が自動的に入力されます。

4. テスト時の挙動では、サービスはSECURITY SERVER を経由せずに、サービスの SOAP エンドポイントに対して直接テストされます。そのため、テスト用のクエリがクエリログで簡単に区別できるよう、SECURITY SERVER のヘッダ値のほとんどは「test」に設定されています。

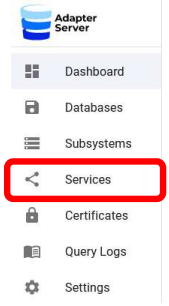
5. サービスリクエストボディのスキーマ構造が追加され、データ要素の値にタグ名由来のプレースホルダーが追加されています。サービスのテストを実行する前に、これらを置き換え必要があることに注意してください。

6. テストサービスが実行され、SOAP クエリがレスポンスを返した後に、SECURITY SERVER クエリのレスポンスメッセージが追加されるレスポンスメッセージフィールドです。

7. *Request* メッセージフィールドの内容を *Target URL* に送信するボタンです。結果は *Response* メッセージフィールドに表示されます。

8. ユーザーが見ていたビュー(Add/Edit Service画面)に戻ります。

サービス



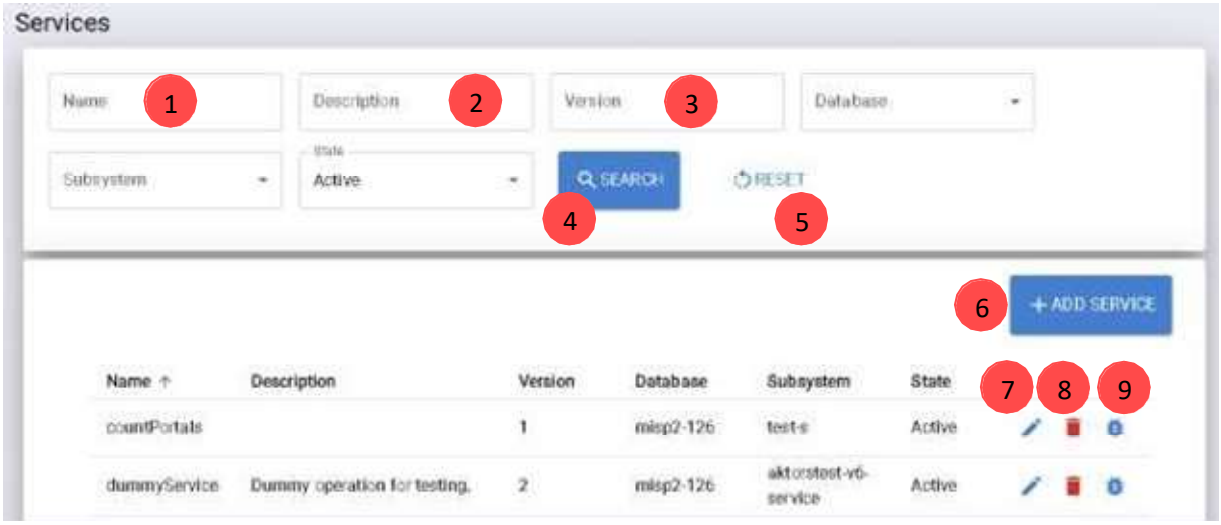
サービスは、データベースの SQL クエリを表し、入力パラメータが読み込まれ、結果が HTTP または HTTPS プロトコル上のSECURITY SERVER SOAP メッセージフォーマットで送り返されます。

サービスは、「サブシステム(subsystems)」メニュー項目を最初にクリックし、アイコンをクリックして編集用の特定のサブシステムを選択することで到達できるサブシステム編集ビューからでも表示できます。

既存のすべてのサービスのリストは「サービス(Services)」メニュー項目からも表示できます。

サービス編集画面では、SQL クエリとそれに対応するリクエストおよびレスポンスの XML 構造を定義できます。
既存のサブシステムにサービスを追加・削除することができます。サービスは、サービステストビューでテストできます。

サービスビュー



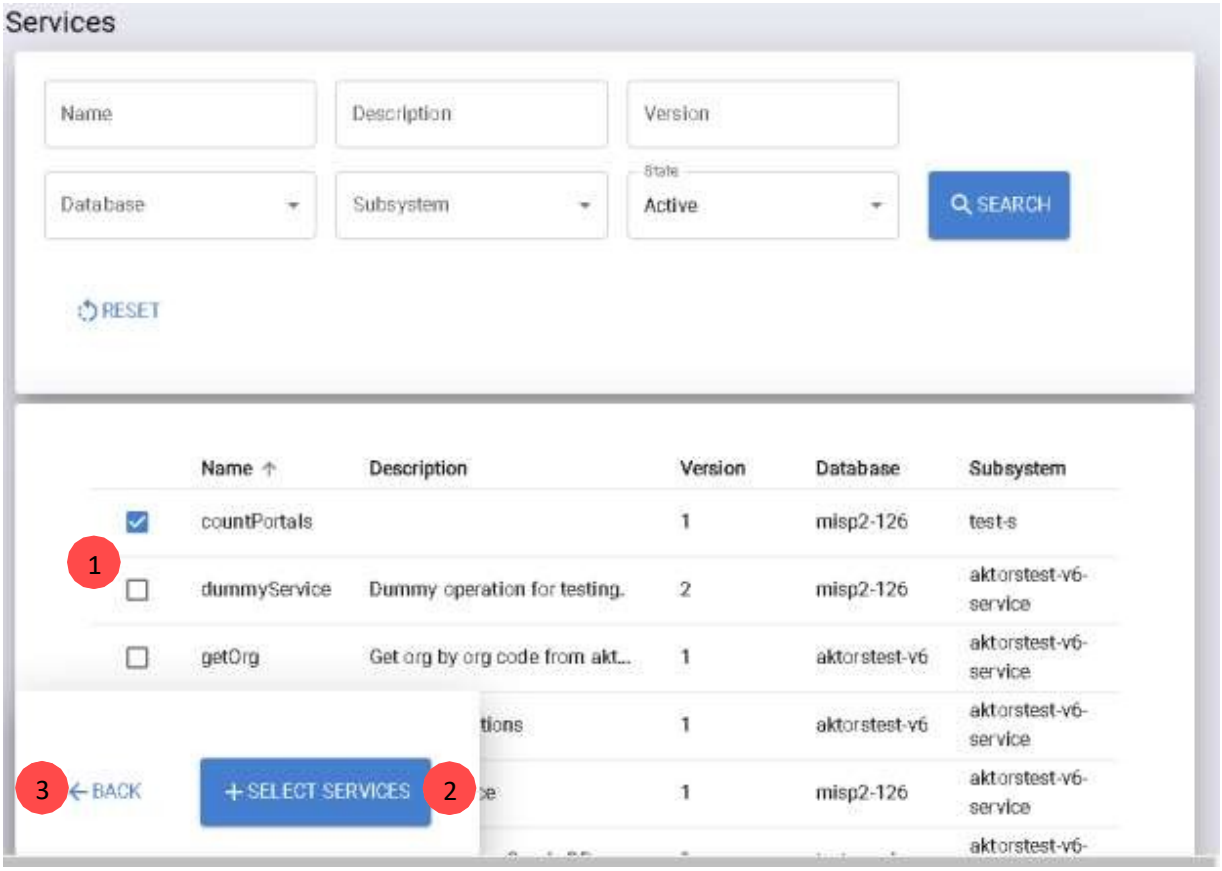
画像 *11* サービスビュー

1. サービス名フィールドフィルター、単語の先頭を指定して検索（SECURITY SERVER *serviceCode* に対応）します。
2. サービス説明をフィルタして検索します。
3. サービスバージョンフィールドフィルター：整数（SECURITY SERVER serviceVersion から 'v' の接頭辞を除いたものが該当します）

4. 設定したフィルターをもとに検索を行います。
5. フィルターを初期値に戻します。
6. 「サービスの追加」画面に移動します。
7. サービスの編集を行います。
8. サービスを削除します（最初に確認ダイアログが表示されます）。
9. サービスをテストします。

サービスセレクトビュー

サービス選択ビューは、すべてのサービスのリストからサブシステムへのサービスを選択/選択解除するときに表示されます。このビューは通常の［サービス］ビューと似ていますが、アクション列（編集/削除/テストボタン付き）が削除され、代わりにチェックボックスが最初の列に表示されています。



画像 *12* サービスセレクトビュー

1. サブシステム・サービスを選択するか否かのチェックボックスとなります。

2. 選択したサービスをサブシステムに追加し、サブシステム・ビューに戻り、操作を続行します(選択したサービスの関連付けをサブシステムで永続化するには、保存が必要です)。
3. サブシステム・ビューに戻ります。

サービスビューの追加・編集

サービスの追加・編集については、「サブシステム」の章で詳細を説明しています。

サービステストビュー

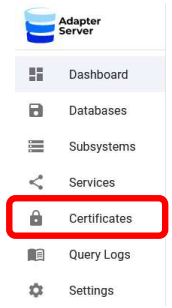
サービステストビューについては、「サブシステム」の章で詳細を説明しています。

サービスを削除する

サービスとその入出力パラメータを削除するには、”Services ”メニュー項目から”Services ”をクリックしてください。確認ダイアログで「OK」を選択すると、そのサービスが削除されます。

サブシステムのサービスリストでボタン  を押下すると、サブシステムからサービスが削除されますが、「サービス」ビューからはアクセス可能です。サブシステムビューでサブシステム全体が削除された場合、その関連サービスは、サービスを所有するサブシステムとともにすべて削除されます。

証明書



証明書は、HTTPS 対応サブシステムへのアクセスを許可された信頼できる SSL 証明書のエントリを表します。これらは、SOAP エンドポイントにクエリを送信するセキュリティ・サーバー証明書で構成されます。

証明書の表示は、「証明書」メニューから行うことができます。

「証明書」メニュー項目には、関連するメタデータを含む証明書を管理する機能が含まれます。証明書は、保存、削除、再アップロードによる変更、表示、一覧表示が可能です。

UXA Web アプリケーションサーバーの証明書も「証明書」ビューで確認したり、エクスポートしたりできます。

証明書の表示

画像 *13* 証明書ビュー。サーバーのアクティブな証明書（*1*）とクライアントの証明書（*2*）

アダプターサーバー証明書

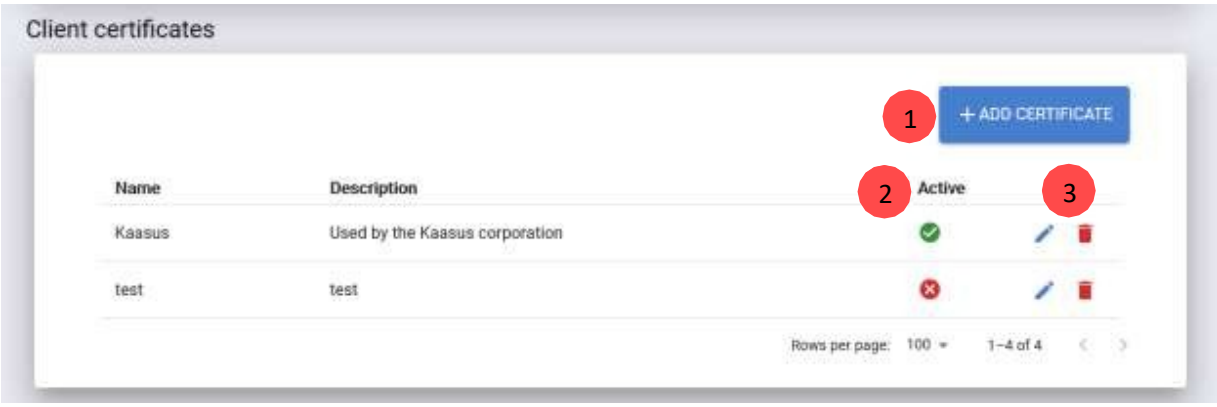
画像 *14* サーバー証明書表示パネル

1. 証明書を一意に識別するための証明書のハッシュ（またはサムプリント）。
2. 証明書の有効期間終了日
3. サーバー証明書を.pem ファイルでダウンロードする
4. サーバー証明書の詳細情報を表示する

画像 *15* サーバー証明書の詳細情報

クライアント証明書

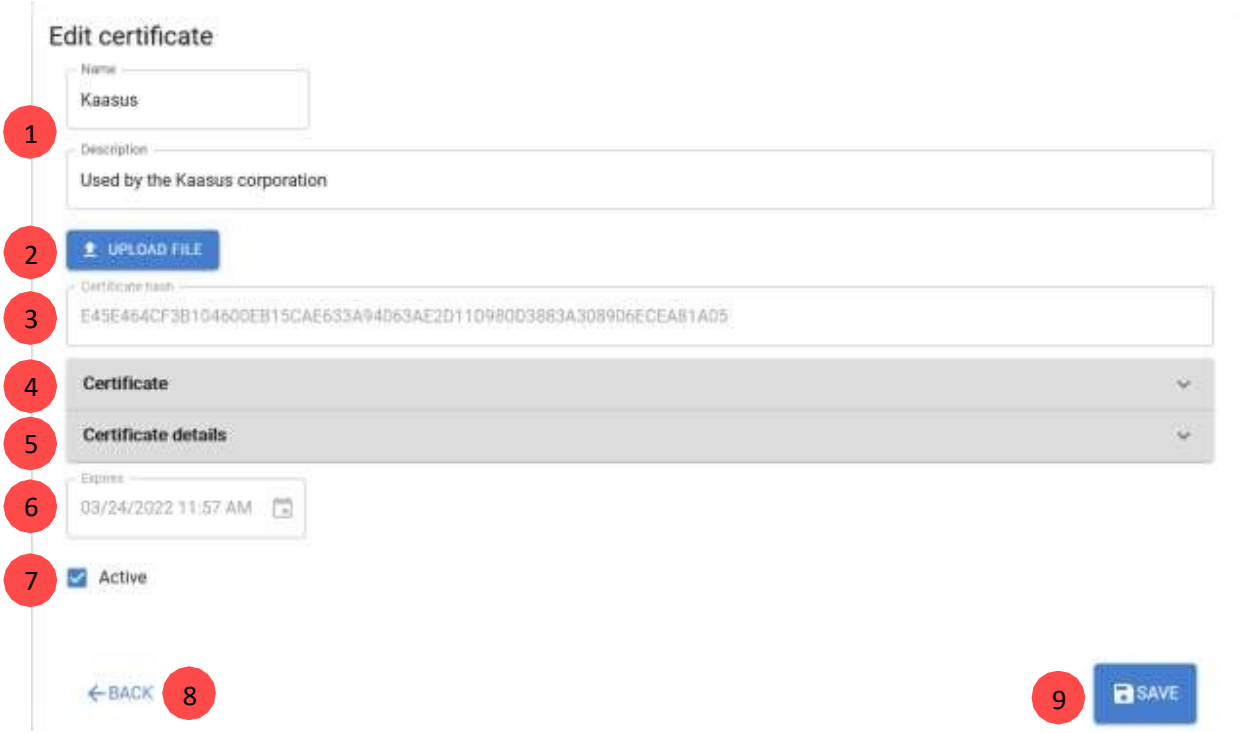
ここに追加され、アクティブになっている証明書は信頼されており、これらの証明書を使用した HTTPS 接続が受け入れられます。



画像 *16* クライアント証明書の一覧

1. 新しいクライアント証明書の追加
2. 証明書がアクティブな場合。アクティブでない証明書を使用した HTTPS 接続は拒否される
3. 与えられた証明書の編集または削除

証明書の追加・編集



画像 *17* 証明書ビューの追加/編集

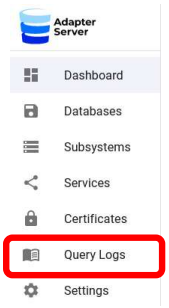
1. 証明書の名前と説明

2. *.pem* または*.cer* をベース 64 エンコードした証明書をアップロードしてください。証明書は一意でなければなりません。
3. アップロードされた証明書のハッシュ値
4. Base64 でエンコードされた証明書です。クリックすると開きます
5. 証明書の詳細はこちらクリックすると開きます
6. 証明書の有効期間終了日
7. 証明書が有効であれば、UXA での HTTPS 接続に使用できます。
8. 証明書の表示に戻る
9. クライアント証明書の保存または作成

画像 *18Base64* エンコードされた証明書

画像 *19* 証明書の詳細

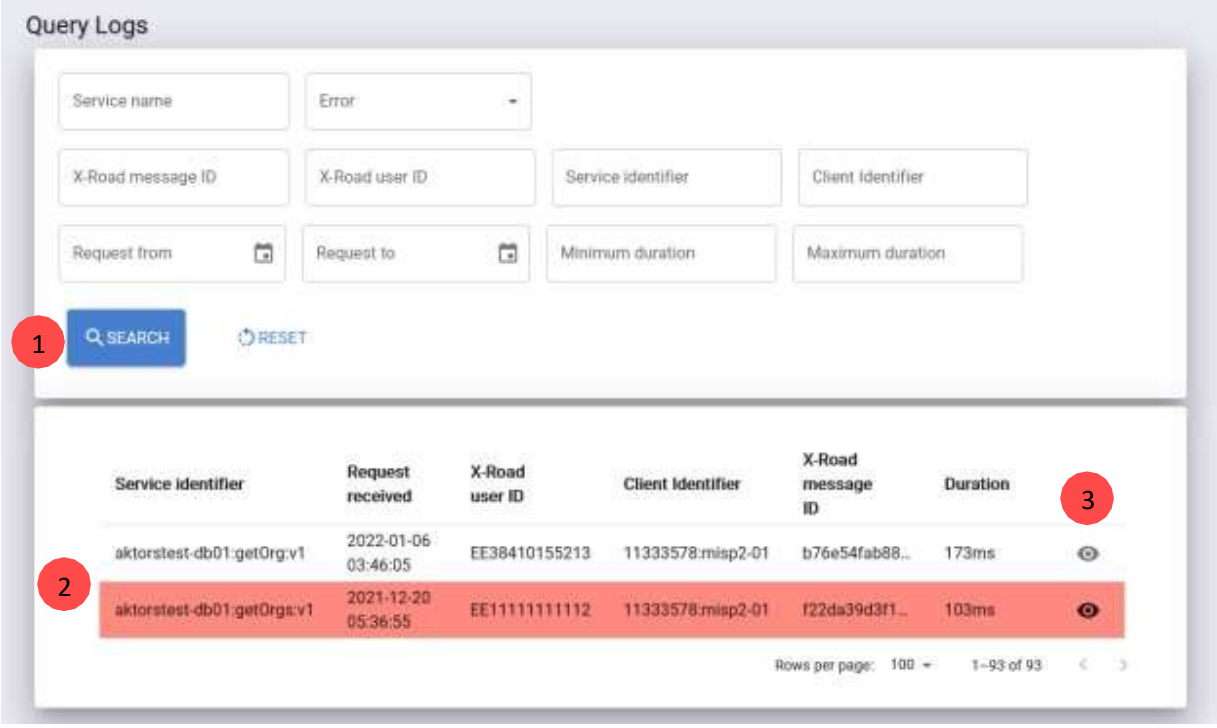
Query Logs



UXA で定義されたサービスに対して行われたすべてのリクエストは、クエリログとして記録されます。

Query Logs ”は、”Query Logs ”メニューからアクセスできます。

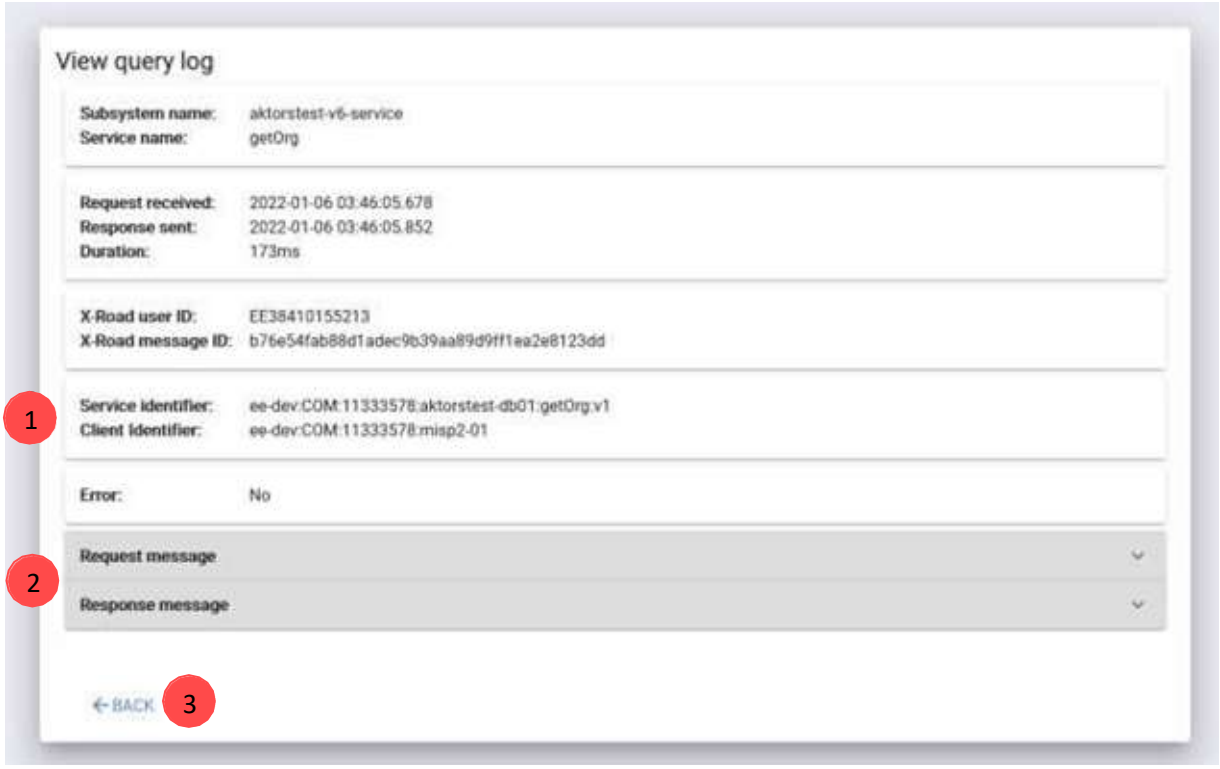
Query Logs リストビュー



画像 *20* Query Logs 表示

1. 検索フィルターを適用し、関連するログのみを表示
2. 赤色の背景の行は、リクエストのエラーを示し、白色の背景の行は、リクエストが成功したことを示します。
3. 詳細を見るには[Query Log view]を開く

Query Log view

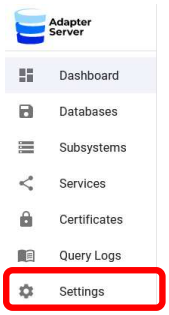


画像 *21* Query Log view 詳細

1 サービスとクライアントの識別子の略記法
   1.1 サービス識別子
   "*service_xroad_instance:service_member_class:service_member_code:service_subystem_code:service_code:service_version*"
   1.2 クライアントの識別子
   "*client_xroad_instance:client_member_class:client_member_code: client_subystem_code*"
2 Query Log リクエストとレスポンスのメッセージ。クリックすると開く
3 Query Logs のリストビューに戻る

画像 *22* クエリリクエストとレスポンスメッセージ

Settings（設定）



設定からUXA ユーザーの操作、監査ログの閲覧、サーバー構成ファイルの閲覧が可能

設定画面は、”Settings ”メニューから表示できます。

Setting画面

画像 *23* 設定画面

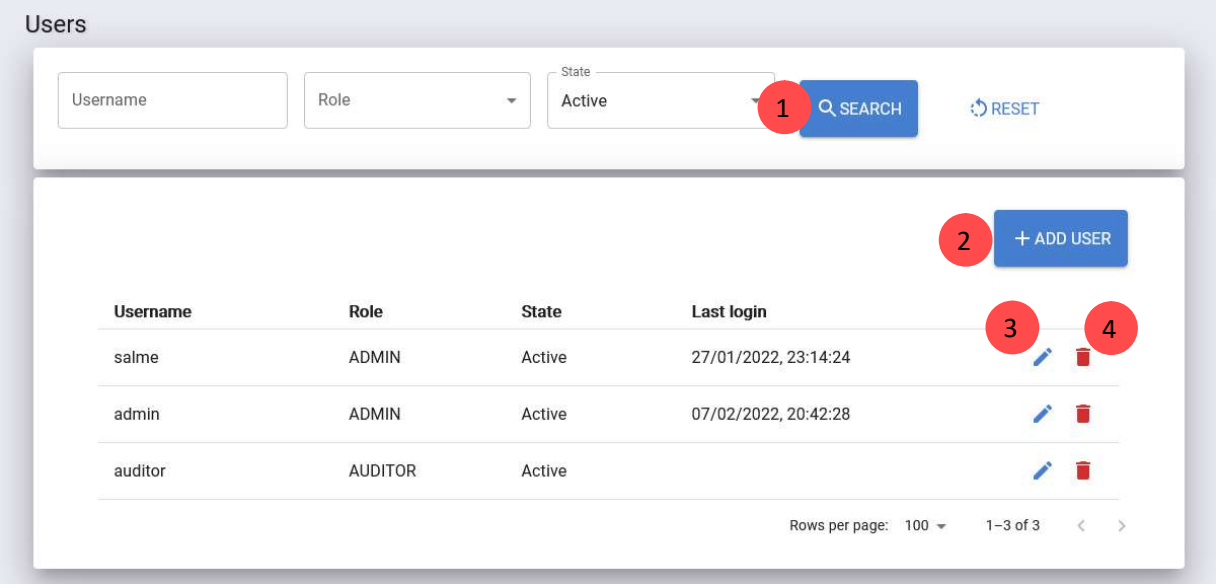
1. UXA のユーザーを表示・管理するための[ユーザービュー]
2. [監査ログ表示] – ウェブアプリケーション内でユーザーが行ったアクションを検索・表示する
3. サーバーの設定とプロパティを表示します。クリックの詳細を表示します。

ユーザー

アプリケーションへのアクセス権を持つユーザーとユーザーロールを管理します。

Setting画面のUserリンク(画像23 ①)からアクセスできます。

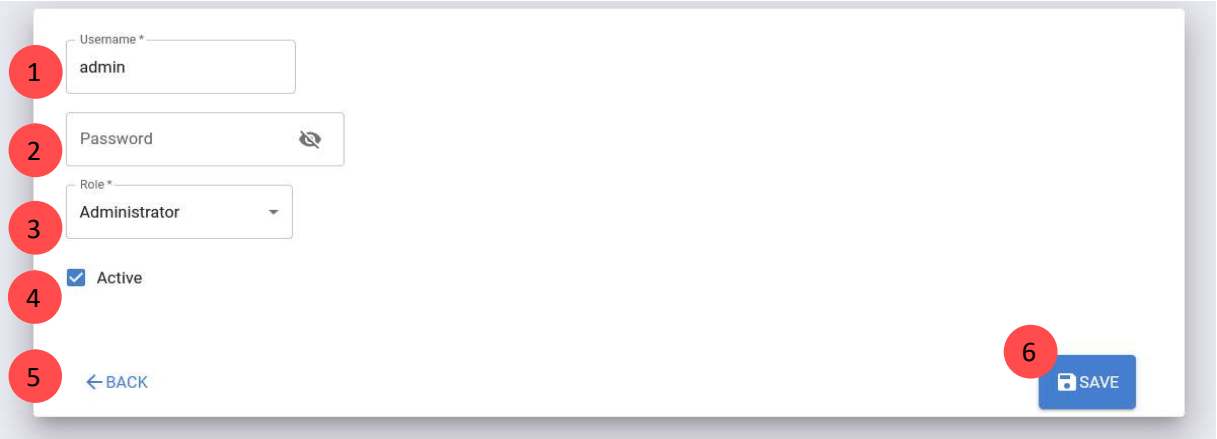
ユーザー一覧表示



画像 *24* ユーザー一覧表示

1. 検索フィルターを適用し、関連するユーザーのみを表示
2. ユーザー追加ビューを開く
3. 与えられたユーザービューを開く
4. 既存のユーザーを非アクティブにする。（ユーザーにはソフト削除しか実装されていません。ユーザーは、監査ログが関連づけられるため、システムから完全に削除することは許可されていません）

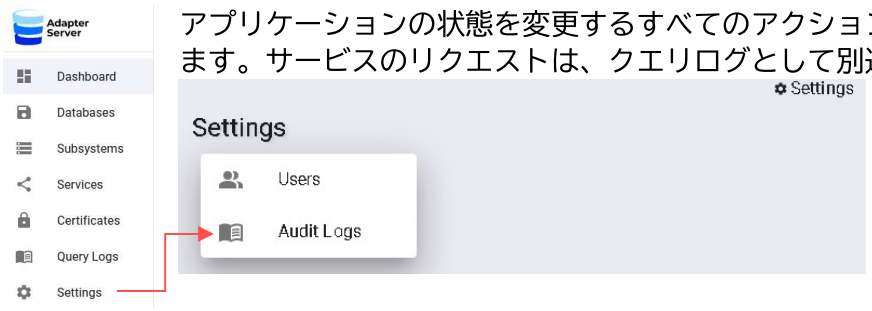
ユーザービューの追加・編集



画像 *25* ユーザービューの追加・編集

1. ユーザー名 - 作成時にのみ設定可能で、その後の編集で変更することはできません。Usernameは、Audit Logの一貫性を保つために変更を禁止しています。
2. ユーザーパスワード - 初期作成時に設定する必要があります。パスワードがシステム上にハッシュ化されて保存されます。
3. ユーザーロール - 管理者はシステムの変更と閲覧が可能ですが、監査者はシステムの変更はできず、閲覧のみ可能です。
4. 有効チェック - アクティブユーザーとは、ログイン可能なユーザーであることを示します。非アクティブユーザーは、論理削除されたものと見做され、ログインすることができません。非アクティブユーザーはInactiveフィルターが適用されている場合のみ、ユーザーリストに表示されます。
5. ユーザー一覧表示に戻る
6. ユーザーの保存または作成

監査ログ



アプリケーションの状態を変更するすべてのアクションは、監査ログとして記録されます。サービスのリクエストは、クエリログとして別途記録されます。

監査ログは「Setting」＞「Audit Logs」から表示することができます。

監査ログ一覧ビュー

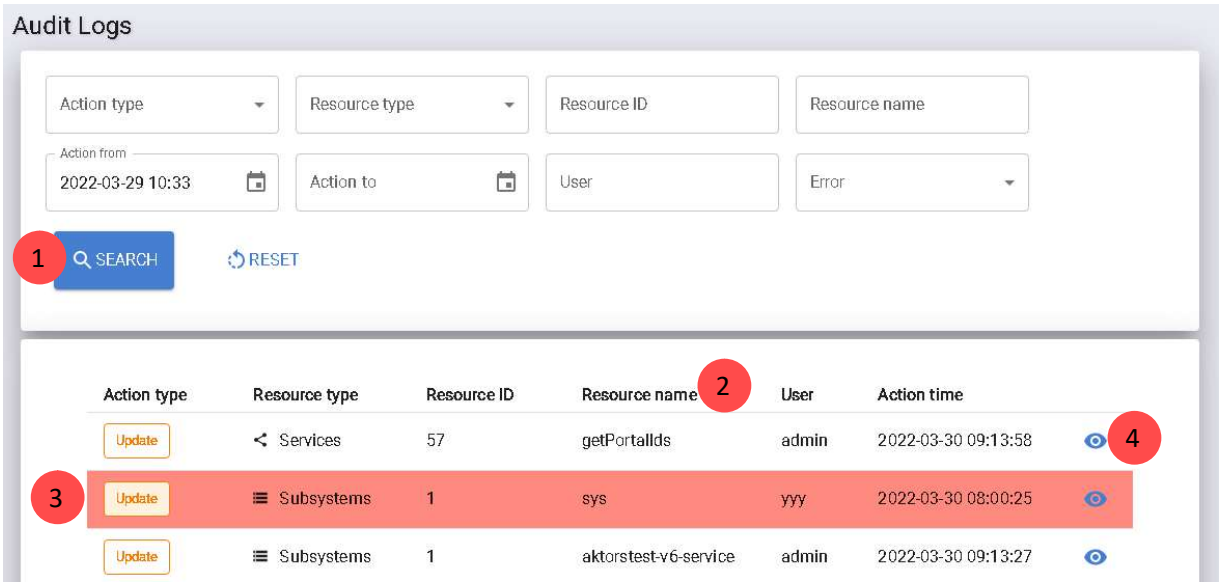


Image 1 Audit Logs list view

1. 検索 - 指定された検索フィルタを適用し、監査ログエントリのリストを取得する監査ログリストを検索します。フィルタは、監査ログの各項目に対応し、各項目の定義については監査ログビューのセクションを参照ください。

2. ヘッダ – 各項目のヘッダをクリックすることによって並べ替えることができます。
3. 監査ログ – 赤い背景のログは、該当のアクションがエラーで終了したことを示しています。HTTP エラーの内容は、監査ログの詳細(API レスポンスステータス)で確認することができます。背景が白の行は、アクションが成功したことを示します。
4. ログの詳細表示 – 各監査ログ項目の詳細表示を開きます。

監査ログ詳細

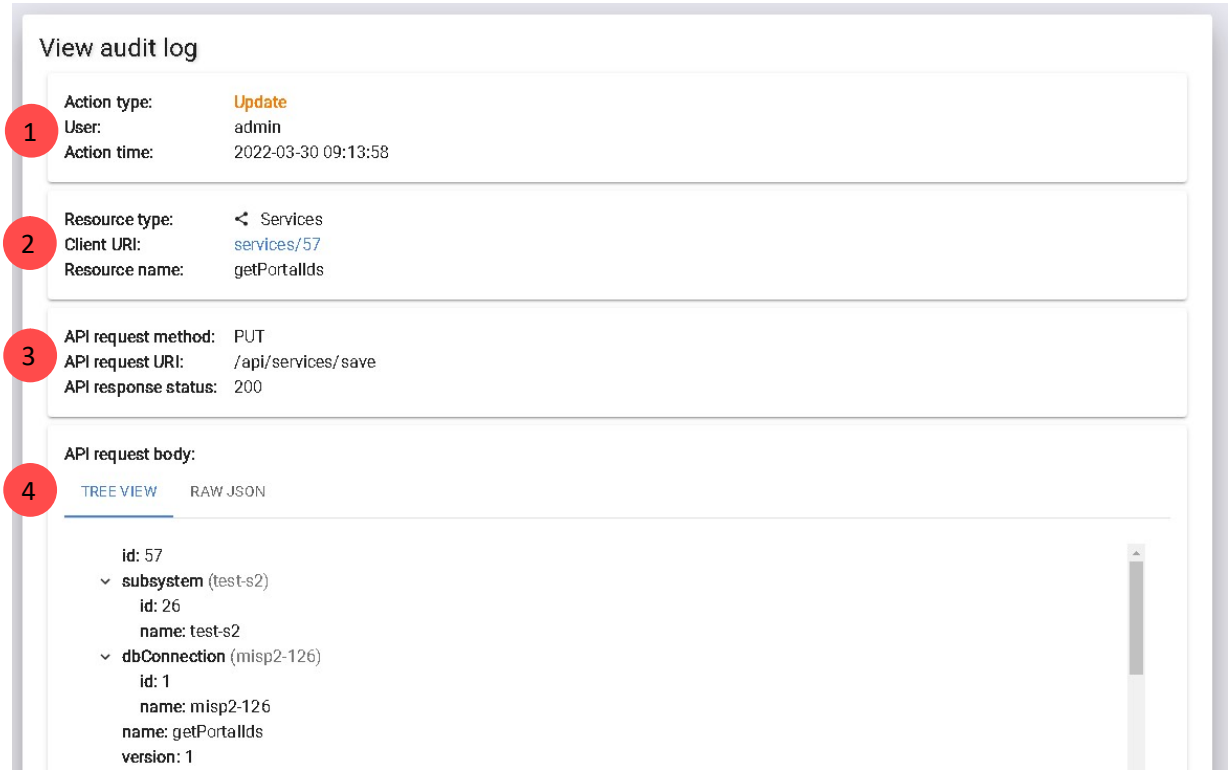


Image 2 Detailed Audit Log view

1. 一般的情報:
    - *アクション種別* – 作成、更新、削除、ログイン、DB テスト
    - *ユーザ* – 当該ログに関連するアクションを実行したユーザー名
    - *実行時間* – アクションが実行された時間
2. リソース情報:
    - *リソース種別* – 現在記録されているアクションが適用されたリソース種別。リソースはそれぞれアプリケーションのメニュー項目に対応しています。利用可能なリソースは、データベース、サブシステム、サービス、証明書、およびユーザーです。
    - *クライアント URI* – 関連するリソース編集ビューへのリンクとして表示される URI。リソースが削除されている場合、リンクは表示されますが、移動しようとするとエラーとなります。サービスの“/”の後の数字はリソース ID を表し、監査ログ一覧ビュ

ーの検索フィルターでフィルタリングする際に指定することができ、表示結果をそのリソースのみに限定することができます。

- *リソース名* - リソースに関連付けられたリソースの名前です。この値は、主に特定の監査ログを検索するために使用されます。

3. API 情報：

   アプリケーションの API リクエストデータを、レスポンス HTTP ステータスコードとともに表示します。リクエストメソッドと API リクエスト URI は証跡として残していますが、ユーザーのアクションを追跡してシステムログと関連付ける場合を除き、多くの情報は追加されません。上記のリソースセクションの値は、API リクエストパラメータから導き出されたものです。

   - *レスポンスステータス* - アクションがエラーで終了したかどうかを判断します。APIレスポンスステータスが400以上の監査ログは、監査ログ一覧上にはエラーとして表示されます。

4. *API リクエストボディ* - 現在の監査ログに記述されているアクションを実行するために、アプリケーションがサーバーに送信した API リクエストの詳細です。フィールドデータは内部的に JSON 形式で保持されており、デフォルトではツリービューで表示されます。Raw JSON タブをクリックすると、JSON 形式データが表示されます。

注意：ツリービューはサブツリーが閉じられている場合でもオブジェクトを認識できるよう、アプリケーションにて注釈を付与して表示しています。括弧内のテキスト、例えば画像21の *(test-s2)* は、リソース名を表示しています。上述の通り、アプリケーションにて注釈を保管していない生データを参照したい場合には、RAW JSONタブよりJSON形式にてデータを参照ください。