ロール認証サービス
利用マニュアル

第 1.6 版 2018 年 5 月 17 日

情報連携推進本部

変更履歴

版数	日付	内容
第1.0版	2013年4月1日	新規作成
第1.1版	2013年5月13日	離籍者の認証について追記しました
第1.2版	2013年8月2日	PHP用ライブラリについて追記しました
第1.3版	2015年2月24日	申請可能な属性情報に追記しました
第1.4版	2017年1月26日	CASサーバが返す情報に追記しました
第1.5版	2018年3月13日	Javaのweb.xml記載に誤記があったのを修正しました
第1.6版	2018年5月17日	権限委譲のユーザ設定部分を削除

目次

1. ロール認証サービスとは

1.1. 背景

名古屋大学では、ウェブアプリケーションを対象としたシングルサインオンの CAS 認証サービス（https://auth.mynu.jp/cas/）を提供しています。

CAS 認証サービスでは、名古屋大学 ID とパスワードによって、ユーザ認証のみ行っていますが、ロール認証サービス（https://auth.nagoya-u.ac.jp/cas/）では、名古屋大学 ID とパスワードに加えてロールおよびロールホルダーでの認証をできるようにし、2013 年 4 月にサービス開始しました。

本マニュアルは、ロール認証サービス（https://auth.nagoya-u.ac.jp/cas/）について記述したものです。

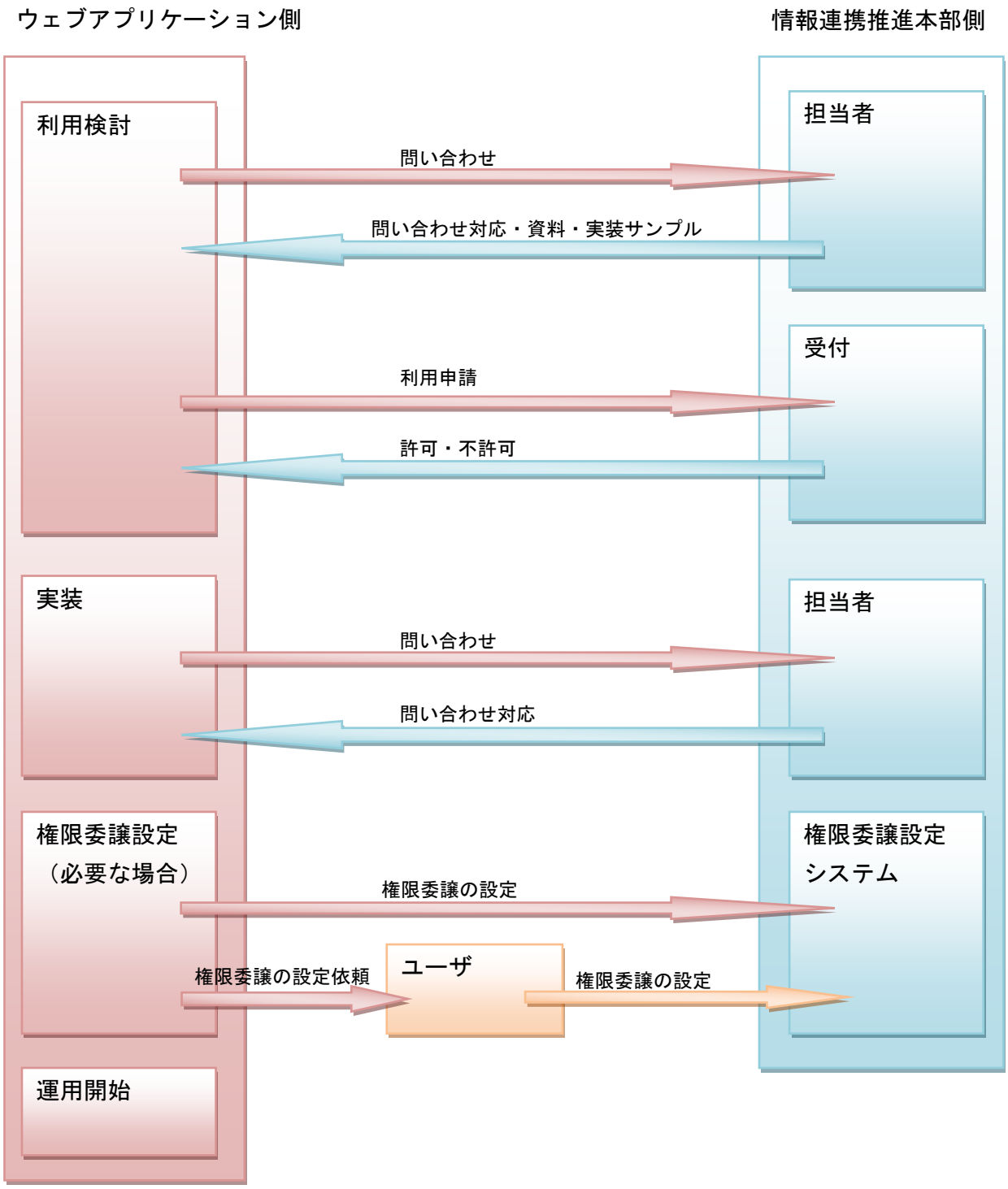
1.2. ロール認証サービスでできること

ロール認証サービスを利用するウェブアプリケーションは次のことを実現できます。

- 名古屋大学 ID とパスワードによる認証
- 名古屋大学離籍者は自動でログインを制限
- 許可したロールをもつユーザのみにログインを制限
- ロール認証サービスを利用するウェブアプリケーション同士のシングルサインオン
- ログインユーザの属性情報（名古屋大学 ID、氏名、所属、身分、ロール情報など）の取得と利用
- ウェブアプリケーション側でロール情報と権限を紐づけることによる権限管理
- 権限委譲による代理ログイン

1.3. ロール認証サービス利用までの流れ

ロール認証サービスを利用するためには、利用申請とウェブアプリケーション側の実装が必要です。次に利用までの流れを示します。



利用申請時に次の項目を指定します。

- ウェブアプリケーションの URL
- ウェブアプリケーションへのログインを許可するロールとロールホルダー
- ウェブアプリケーションが利用する、ユーザの属性情報
- シングルサインオンの許可、不許可
- 離籍者の認証許可、不許可
- 権限委譲の許可、不許可

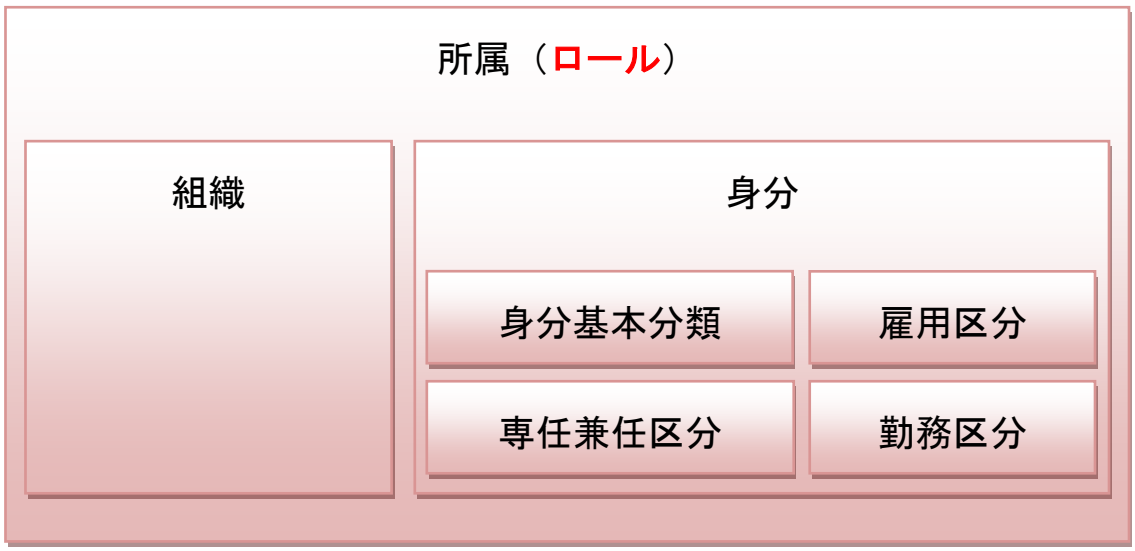
1.4. 制限事項

ウェブアプリケーションは、認証サーバと HTTPS で通信できる必要があります。
離籍者は権限委譲ができません。また、在籍者から離籍者への権限委譲もできません。

1.5. ロールとは

ロール認証サービスでは、所属をロールとして定義します。

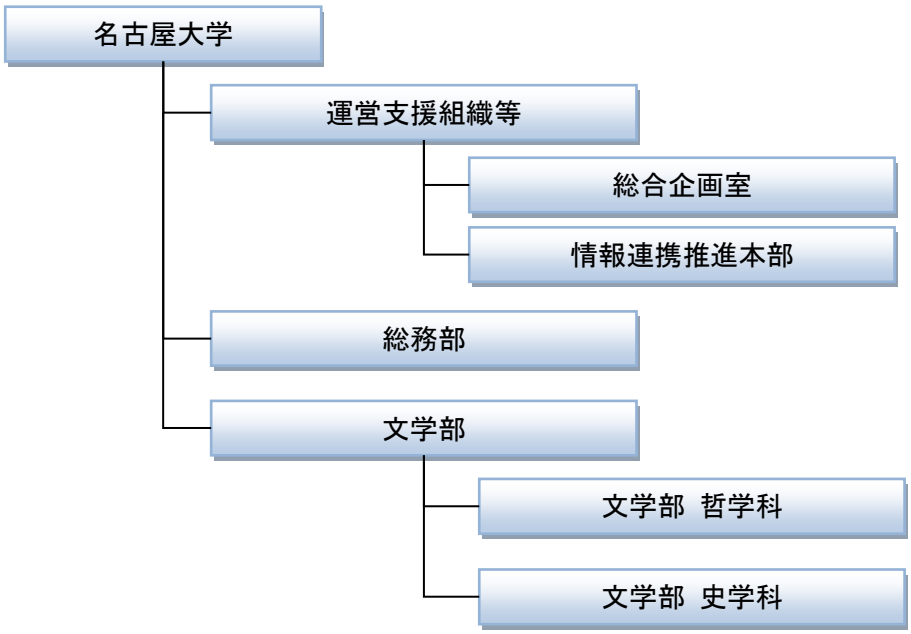
なお、所属は、組織と身分の組合せであり、身分は身分基本分類、雇用区分、勤務区分、専任兼任区分の組合せです。



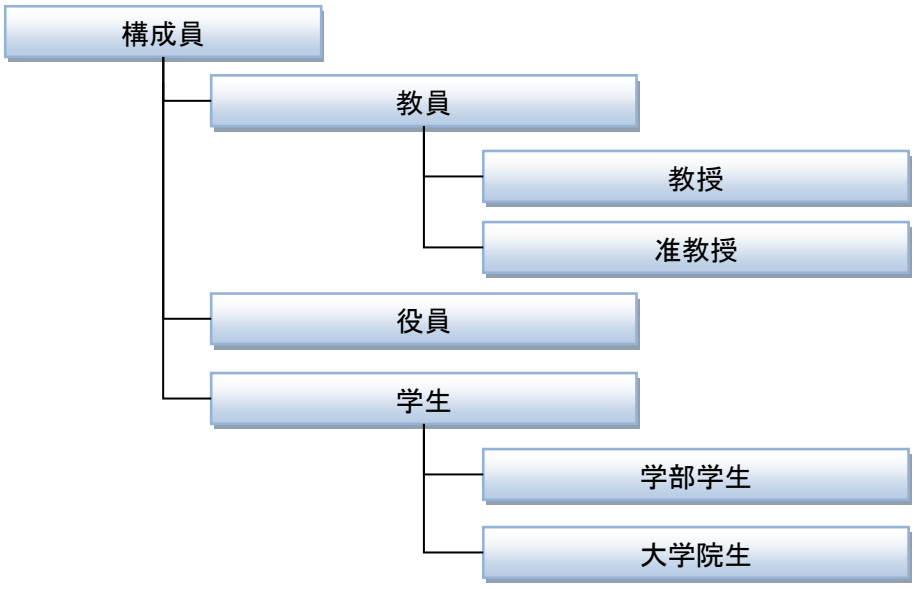
また、組織、身分基本分類、雇用区分、勤務区分、専任兼任区分は階層になっており、階層で上位にある項目は、下位の項目を含む意味を持ちます。

次に例を示します。

組織の例

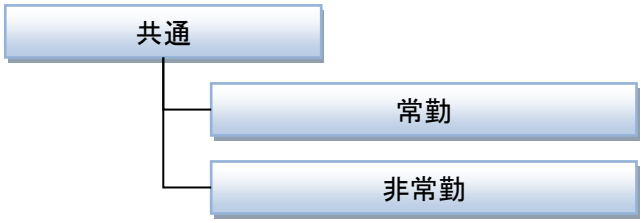


身分基本分類の例

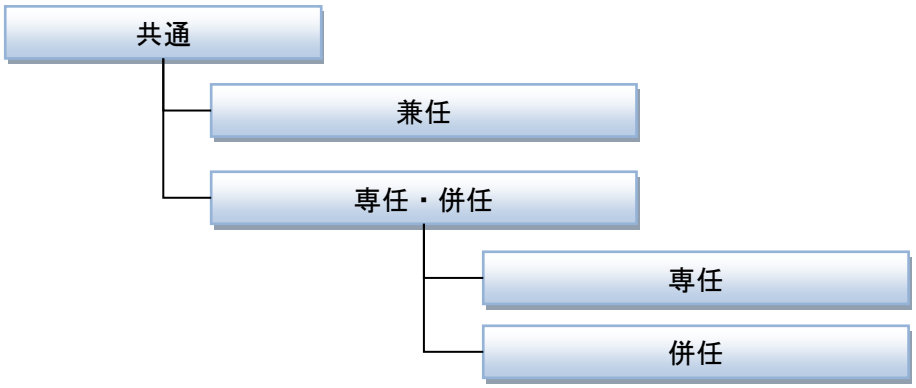


雇用区分の例

勤務区分の例



専任兼任区分の例

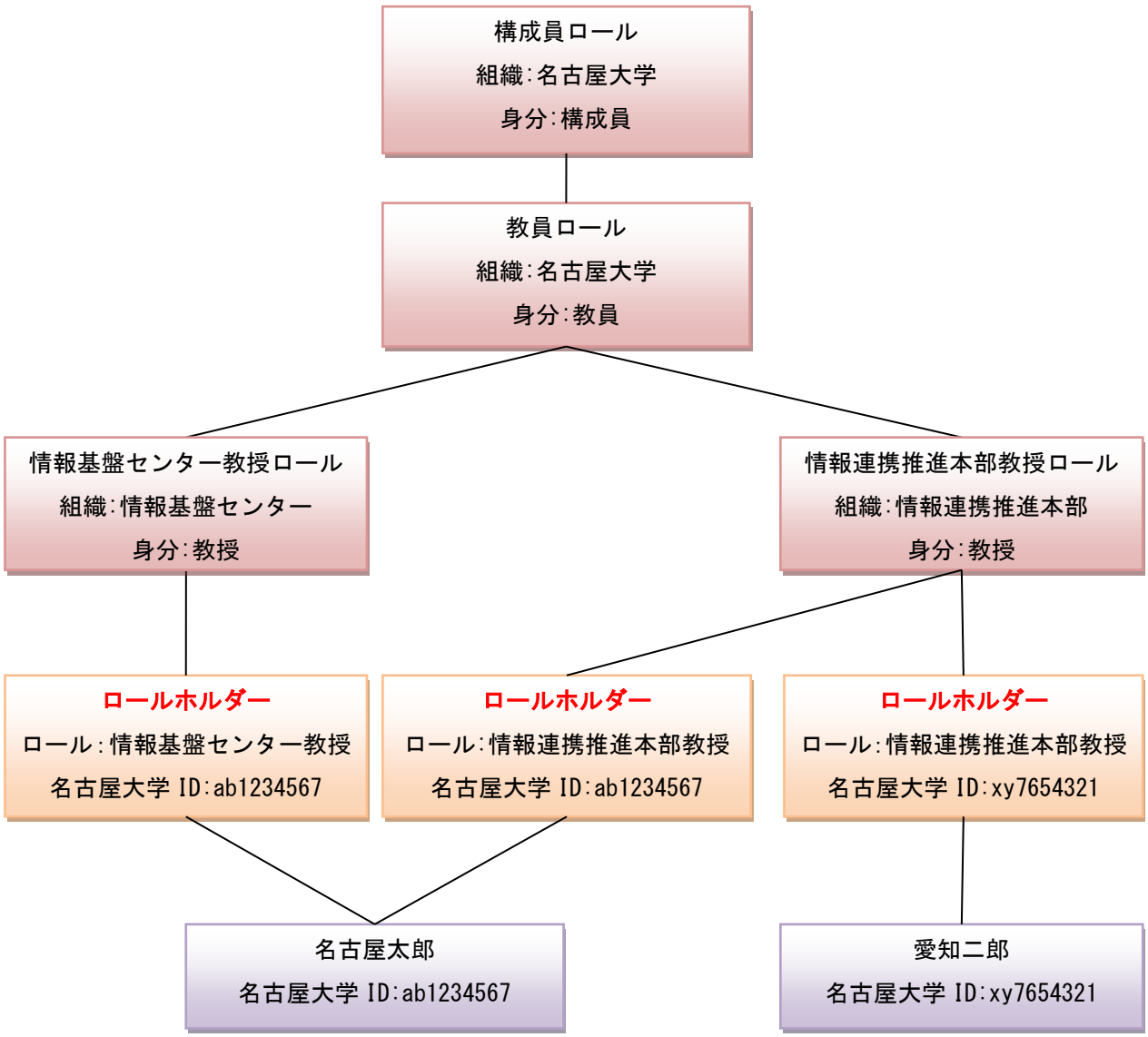


例えば、学生全員を示すロールは、組織[名古屋大学]、身分基本分類[学生]、雇用区分[共通]、勤務区分[共通]、専任兼任区分[共通]の組合せで表します。

1.6. ロールホルダーとは

ロール認証サービスでは、名古屋大学 ID により一意に決まる人が、どの所属（ロール）に含まれるかを示したものを、ロールホルダーと定義します。

概念図を次に示します。



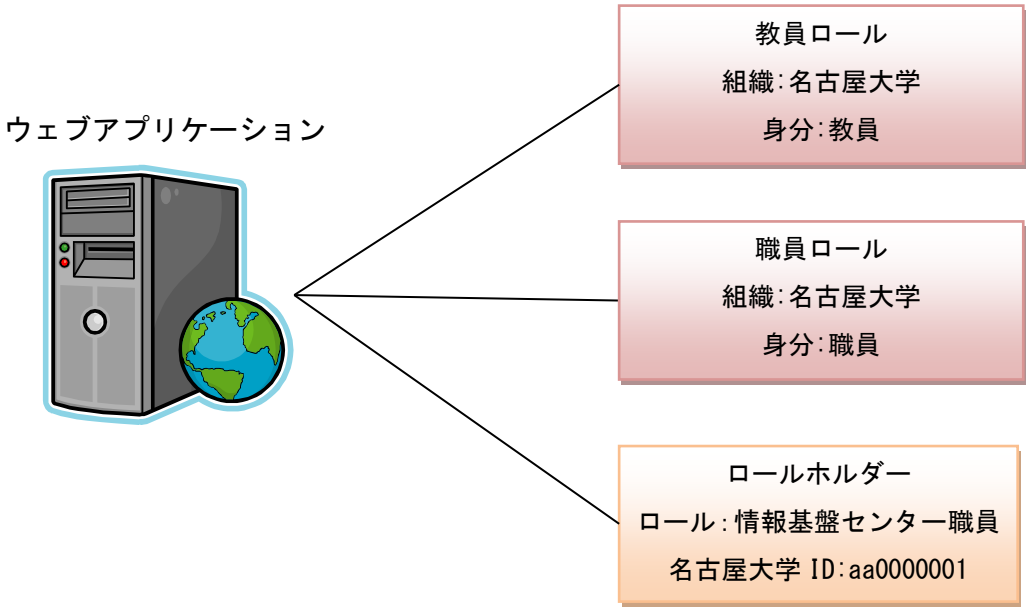
1.7. ロール認証の仕組み

ロール認証サービスを利用するウェブアプリケーションには、ログインを許可するロール、またはロールホルダーを設定します。

ウェブアプリケーションにログインできる人は、ウェブアプリケーションに許可したロールに含まれる人、または許可したロールホルダーと同一である人です。

なお、ログインできる人を制限しない場合は、「名古屋大学構成員」のロールを、ウェブアプリケーションに対して許可することで実現できます。

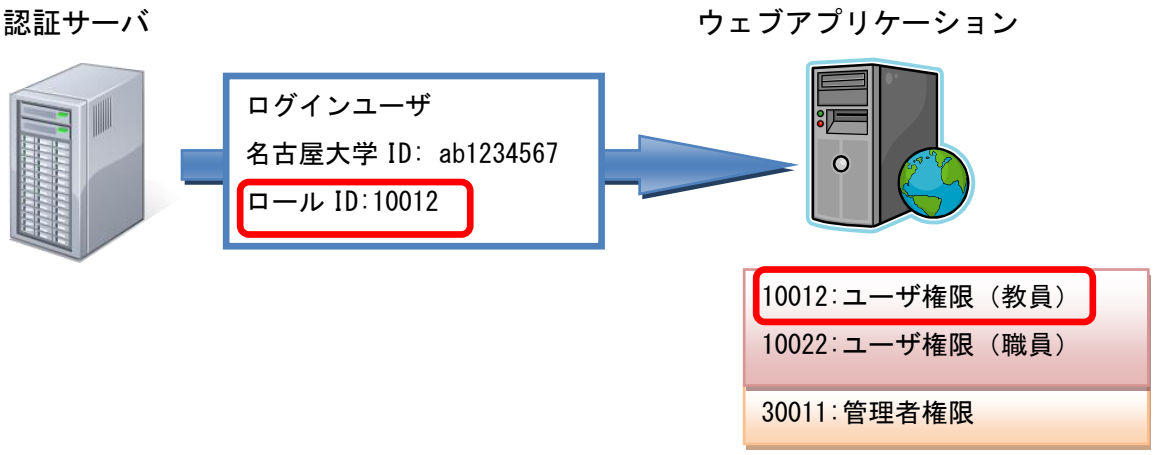
例えば、ウェブアプリケーションに教員と職員に対して利用許可をし、かつ情報基盤センター職員のある 1 名を管理者として利用許可する場合は、次の図のように、ロールとロールホルダーを設定します。



ウェブアプリケーションが権限によって動作を変えたい場合、ロール認証サーバから受け取ることができるログインした人のロール ID、またはロールホルダーID を使用できます。

例えば、上図を例にすると、ログインした人のロール ID が、教員ロール ID か職員ロール ID の場合は、通常の利用権限で動作させ、ログインした人のロールホルダーID が、管理者用のロールホルダーID の場合は、管理者権限で動作させるように実装します。

ログインユーザのロール ID が認証サーバから、ウェブアプリケーションに渡され、ユーザ権限と対応付ける場合の例を次に示します。



1.8. 権限委譲の仕組み

ロール認証サービスでは、権限委譲による代理ログインの仕組みがあります。

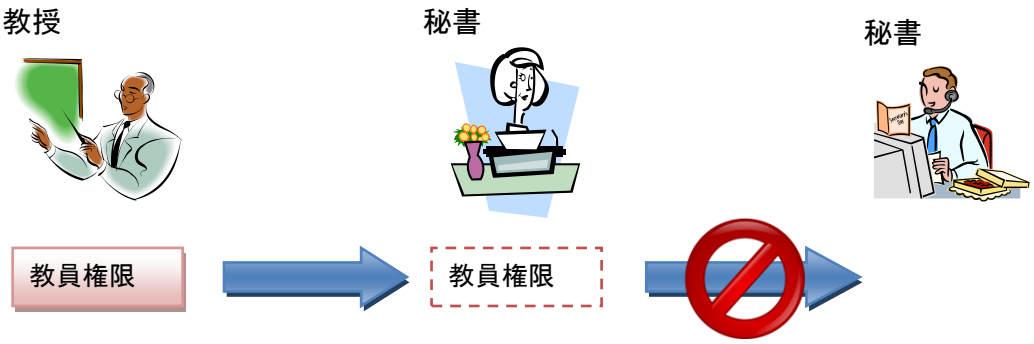
ウェブアプリケーション毎に、権限委譲を許可するか許可しないかを決めます。

ウェブアプリケーションの管理者は、誰から誰へ権限委譲するか自由に設定できます。また、本人はウェブアプリケーションを指定し、誰へ権限委譲するかを設定できます。

なお、権限委譲は 1 段階のみ許します。委譲された権限を他の人へ委譲することはできません。

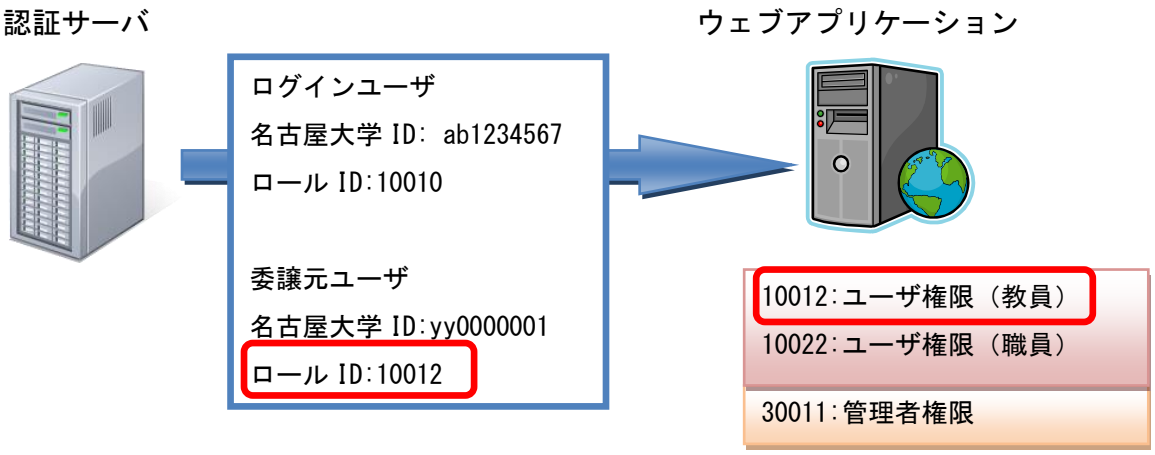
注意事項

権限委譲は身分を制限せず設定できますので、アプリケーションの特性によって、教員から TA へは委譲しない等、ルールを明確にし、利用者への周知徹底を行ってください。



権限委譲を許すウェブアプリケーションに、権限委譲されたユーザがログインした場合、ログインユーザの属性情報に加えて、委譲元ユーザの属性情報が渡されます。

ウェブアプリケーションが権限によって動作を変えたい場合、ログインユーザの属性情報、委譲元ユーザの属性情報を使い、ユーザの選択、権限との対応付けなどを実装します。


認証サーバ
ウェブアプリケーション
ログインユーザ
名古屋大学 ID: ab1234567
ロール ID:10010
委譲元ユーザ
名古屋大学 ID:yy0000001
ロール ID:10012
10012:ユーザ権限（教員）
10022:ユーザ権限（職員）
30011:管理者権限

2. CAS 認証メカニズム

2.1. CAS 認証メカニズムを理解する必要性

ロール認証サービスは、CentralAuthenticationService（CAS）を拡張して実現しています。

よってロール認証サービスをウェブアプリケーションが実装するには、CAS 認証メカニズムの理解が必要です。また、認証でエラーになる場合の調査においても、CAS 認証メカニズムを理解していることが解決に役立ちます。

2.2. ウェブアプリケーションにユーザが初めてアクセスする場合

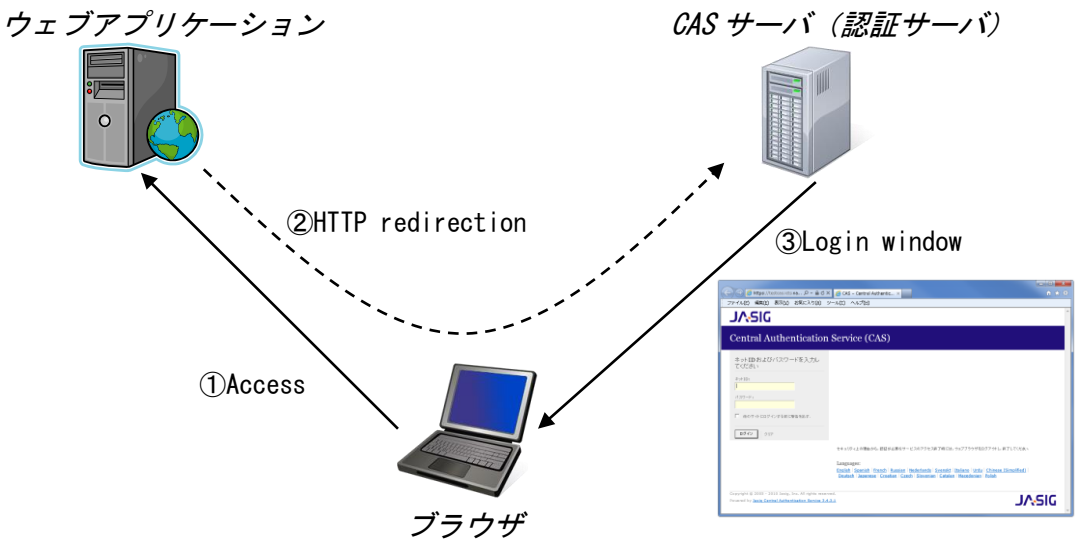
2.2.1. CAS 認証その 1

ユーザは Web ブラウザで、利用するウェブアプリケーションの URL（例えば、http://test.jp/index）にアクセスします（図の①）。

ウェブアプリケーション側は CAS サーバへ HTTP リダイレクション機能を使って、アクセスを転送します（図の②）。その際、service パラメータを用いて、認証すべきサービスの URL を伝えます。（例：https://auth.nagoya-u.ac.jp/cas/login?service=http://test.jp/index）

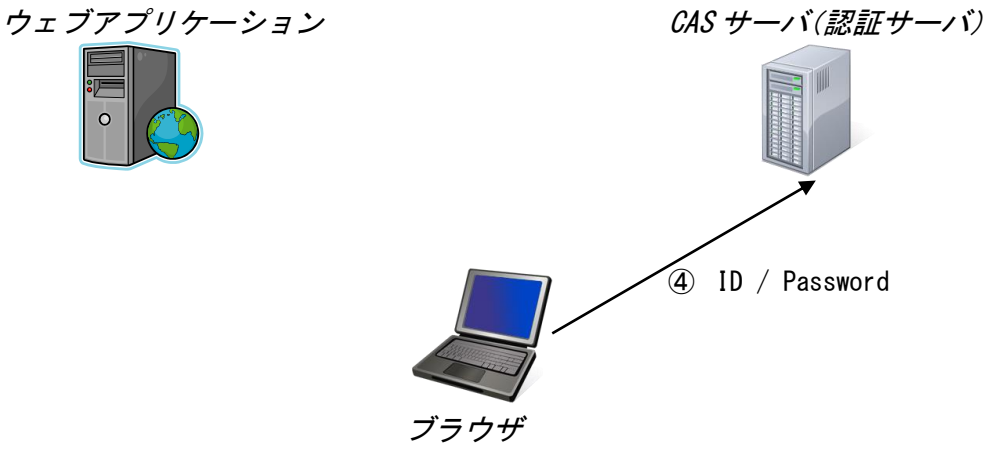
CAS サーバは、ブラウザに保存されている TicketGrantingCookie（TGC）を確認し、TGC がない場合は、CAS 認証がまだ終わっていないと判断し、認証画面を表示します（図の③）。

なお、service パラメータに設定する URL は、URL エンコードする必要があります。



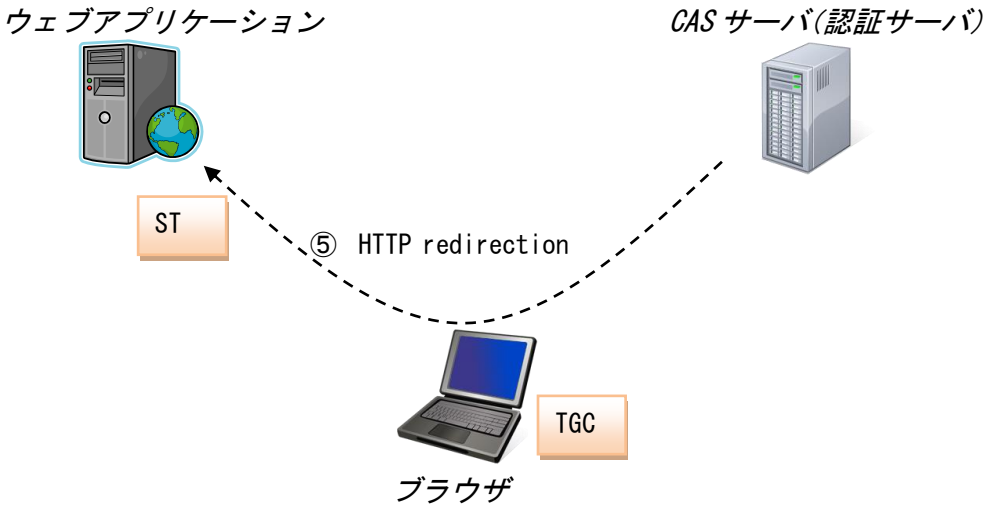
2.2.2. CAS 認証その２

ユーザは認証画面で、名古屋大学 ID とパスワードを入力し送信します（図の④）。



2.2.3. CAS 認証その３

ユーザが正しく認証されると、CAS サーバはブラウザに対し TGC を発行するとともに、URL パラメータ ticket に、ServiceTicket（ST）をセットし、再度呼び出されたアウェブアプリケーションへ HTTP リダイレクトをします（図の⑤）。（例：http://test.jp/index?ticket=ST-668-ySrjgIgdRiOVeTWdO7OdZzIwzlOtiz9ZO9vtbKcXuAreVhzH90-cas）

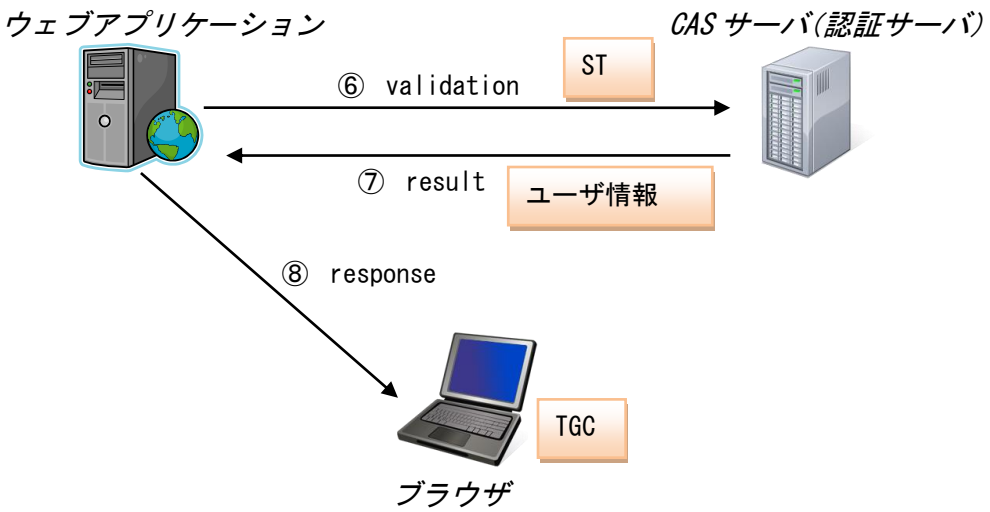


2.2.1. CAS 認証その４

ウェブアプリケーションは取得した ST を検証するため、CAS サーバに対して ST を送信します（図の⑥）。（例：https://auth.nagoya-u.ac.jp/cas/serviceValidate?service=http://test.jp/index&ticket=ST-668-ySrjgIgdRiOVeTWdO7OdZzIwzlOtiz9ZO9vtbKcXuAreVhzH9O-cas）

CAS サーバでは ST の有効性を検証し、その結果をウェブアプリケーションに送信します（図の⑦）。その際、ログインユーザの属性情報（名古屋大学 ID、氏名、所属、身分、ロール情報など）を XML 形式で送信します。XML の形式については「2.5CAS サーバが返す情報」に示します。

ウェブアプリケーションは CAS サーバからの情報に基づいて、ユーザにサービスを提供します（図の⑧）。

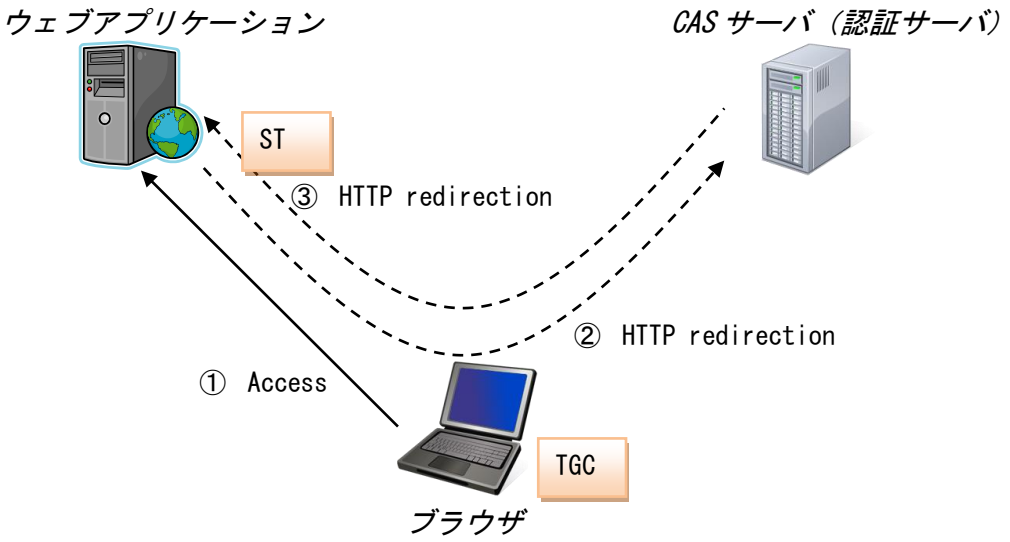


2.3. シングルサインオンでユーザがアクセスする場合

ユーザは Web ブラウザで、利用するウェブアプリケーションの URL にアクセスします（図の①）。

ウェブアプリケーション側は CAS サーバへ HTTP リダイレクション機能を使って、アクセスを転送します（図の②）。その際、service パラメータを用いて、認証すべきサービスの URL を伝えます。

CAS サーバは、ブラウザに保存されている TGC を確認し、TGC があるため認証画面を表示せず、URL パラメータ ticket に、ST をセットし、再度呼び出されたウェブアプリケーションへ HTTP リダイレクトをします（図の③）。



その後の動作は、「2.2.1 CAS 認証その４」と同様です。

2.4. ログアウトする場合

ブラウザが、CAS サーバのログアウト用 URL（https://auth.nagoya-u.ac.jp/cas/logout）にアクセスすると、ロール認証サービスからログアウトします。

また、service パラメータに、URL を指定すると、ロール認証サービスからログアウトした後、指定した URL にリダイレクトされます。（例：https://auth.nagoya-u.ac.jp/cas/logout?service=http://test.jp/logout）

2.5. CAS サーバが返す情報

2.5.1. ST 検証成功の場合

XML の要素を次に示します。

XML 要素	複数	省略可	内容
cas:serviceResponse			
cas:authenticationSuccess			
cas:user			ユーザ ID。名古屋大学 ID または全学 ID
cas:attributes			属性情報
cas:XXXX（XXXX は属性名）	○	○	NagoyaUnivID など申請した属性を列挙する **※申請した属性名にセミコロン「;」を含む場合は、アンダーバー2 つ「__」に変換しています**
cas:syozoku_group			所属情報
cas:syozoku	○		兼務も含め、ユーザの所属を全て列挙する
cas:syozoku_id			所属 ID
cas:bumon_id			部門 ID
cas:bumon_name_jp			部門名称
cas:bumon_name_full_jp			部門名称全体
cas:bumon_name_en			部門名称英語（未使用）
cas:bumon_name_full_en			部門名称全体英語（未使用）
cas:mibun_id			身分 ID
cas:mibun_name_jp			身分名称
cas:mibun_name_en			身分名称英語（未使用）
cas:senken_kbn_cd			専任兼任区分コード
cas:senken_kbn_label			専任兼任区分名称
cas:enrollment			在籍：T 非在籍：F
cas:roleholders		○	ロールホルダーリスト 認証できたロールホルダーがある場合のみ
cas:roleHolder	○		認証できたロールホルダーを列挙する
cas:id			ロールホルダーID
cas:name			ロールホルダー名称
cas:syozoku_id			所属 ID（cas:syozoku_group の対応する ID が入る）
cas:roles		○	ロールリスト 認証できたロールがある場合のみ
cas:role	○		認証できたロールを列挙する
cas:id			ロール ID
cas:name			ロール名称
cas:syozoku_id_group			
cas:syozoku_id	○		ロールに含まれる所属 ID を列挙する （cas:syozoku_group の対応する ID が入る）
cas:delegationOfAuthorityGroup		○	委譲元ユーザリスト 権限委譲を許可したウェブアプリケーションのみ
cas:delegationOfAuthority	○		権限委譲された場合、委譲元ユーザを列挙する
cas:user			上記要素と同じ
cas:attributes			上記要素と同じ
cas:roleholders			上記要素と同じ
cas:roles			上記要素と同じ

XML の例を次に示します。

```
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
 <cas:authenticationSuccess>
  <cas:user>zz0000000</cas:user>
  <cas:attributes>
   <cas:NagoyaUnivID>zz0000000</cas:NagoyaUnivID>
   <cas:fullName__lang-ja>名大 太郎</cas:fullName__lang-ja>
   <cas:syozoku_group>
    <cas:syozoku>
     <cas:syozoku_id>1</cas:syozoku_id>
     <cas:bumon_id>2</cas:bumon_id>
     <cas:bumon_name_jp>学術情報開発研究部門</cas:bumon_name_jp>
     <cas:bumon_name_full_jp>学術情報開発研究部門</cas:bumon_name_full_jp>
     <cas:bumon_name_en>aaabbbcc</cas:bumon_name_en>
     <cas:bumon_name_full_en>aaabbbcc</cas:bumon_name_full_en>
     <cas:mibun_id>10</cas:mibun_id>
     <cas:mibun_name_jp>准教授</cas:mibun_name_jp>
     <cas:mibun_name_en>ccc</cas:mibun_name_en>
     <cas:senken_kbn_cd>01</cas:senken_kbn_cd>
     <cas:senken_kbn_label>専任</cas:senken_kbn_label>
     <cas:enrollment>T</cas:enrollment>
    </cas:syozoku>
   </cas:syozoku_group>
  </cas:attributes>
  <cas:roleholders>
   <cas:roleHolder>
    <cas:id>23</cas:id>
    <cas:name>学術情報開発研究部門准教授 ab0123456</cas:name>
    <cas:syozoku_id>1</cas:syozoku_id>
   <cas:roleHolder>
  </cas:roleholders>
  <cas:roles>
   <cas:role>
    <cas:id>12</cas:id>
    <cas:name>学術情報開発研究部門准教授</cas:name>
    <cas:syozoku_id_group>
      <cas:syozoku_id>1</cas:syozoku_id>
    </cas:syozoku_id_group>
   </cas:role>
  </cas:roles>
  <cas:delegationOfAuthorityGroup>
   <cas:delegationOfAuthority>
    <cas:user>zz0000001</cas:user>
    <cas:attributes>
     <cas:NagoyaUnivID>zz0000001</cas:NagoyaUnivID>
     <cas:fullName__lang-ja>名大 次郎</cas:fullName__lang-ja>
     <cas:syozoku_group>
      <cas:syozoku>
       <cas:syozoku_id>1</cas:syozoku_id>
       <cas:bumon_id>2</cas:bumon_id>
       <cas:bumon_name_jp>学術情報開発研究部門</cas:bumon_name_jp>
       <cas:bumon_name_full_jp>学術情報開発研究部門</cas:bumon_name_full_jp>
       <cas:bumon_name_en>aaabbbcc</cas:bumon_name_en>
       <cas:bumon_name_full_en>aaabbbcc</cas:bumon_name_full_en>
       <cas:mibun_id>10</cas:mibun_id>
```

```
        <cas:mibun_name_jp>准教授</cas:mibun_name_jp>
        <cas:mibun_name_en>ccc</cas:mibun_name_en>
        <cas:senken_kbn_cd>01</cas:senken_kbn_cd>
        <cas:senken_kbn_label>専任</cas:senken_kbn_label>
        <cas:enrollment>T</cas:enrollment>
       </cas:syozoku>
      </cas:syozoku_group>
     </cas:attributes>
     <cas:roles>
      <cas:role>
       <cas:id>12</cas:id>
       <cas:name>学術情報開発研究部門准教授</cas:name>
       <cas:syozoku_id>1</cas:syozoku_id>
      </cas:role>
     </cas:roles>
    </cas:delegationOfAuthority>
   </cas:delegationOfAuthorityGroup>
  </cas:authenticationSuccess>
</cas:serviceResponse>
```

2.5.2. ST 検証失敗の場合

XML の要素を次に示します。

XML 要素	複数	省略可	内容
cas:serviceResponse			
cas:authenticationFailure			失敗の理由

XML の例を次に示します。

```
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
    <cas:authenticationFailure code="INVALID_TICKET">
        Ticket ST-1856339-aA5Yuvrxzpv8Tau1cYQ7 not recognized
    </cas:authenticationFailure>
</cas:serviceResponse>
```

2.5.3. 申請可能な属性情報

属性と属性名を示します。値の範囲やコードについては、情報連携推進本部のウェブページ「http://www.icts.nagoya-u.ac.jp/ja/info/nuid.html」を参照してください。

区分	属性		属性名
学生・職員共通	名古屋大学 ID		NagoyaUnivID
	ミドルネーム	（ベース属性）	（middleName）
		漢字	middleName;lang-ja
		ローマ字	middleName;lang-en
		カタカナ	middleName;lang-ja;phonetic
	名	（ベース属性）	（givenName）
		漢字	givenName;lang-ja
		ローマ字	givenName;lang-en
		カタカナ	givenName;lang-ja;phonetic
	姓名	（ベース属性）	fullName
		漢字	fullName;lang-ja
		ローマ字	fullName;lang-en
		カタカナ	fullName;lang-ja;phonetic
	プライマリの所属部局	（ベース属性）	（department）
		漢字	department;lang-ja
		コード	departmentNumber
	プライマリの所属学科(専攻)，掛	（ベース属性）	（section）
		漢字	section;lang-ja
		コード	sectionNumber
	学生番号，職員番号，あるいは学務システム用番号		employeeNumber
	在学・在職中か否かを表すフラグ		Enrollment
学生のみ	入学年度		nagAdmissionYear
	学年		nagGrade
職員のみ	プライマリの職名	（ベース属性）	（title）
		漢字	title;lang-ja
		コード	titleCode
	プライマリの職種	（ベース属性）	（employeeType）
		漢字	employeeType;lang-ja
		コード	employeeTypeCode
	性別		Gender

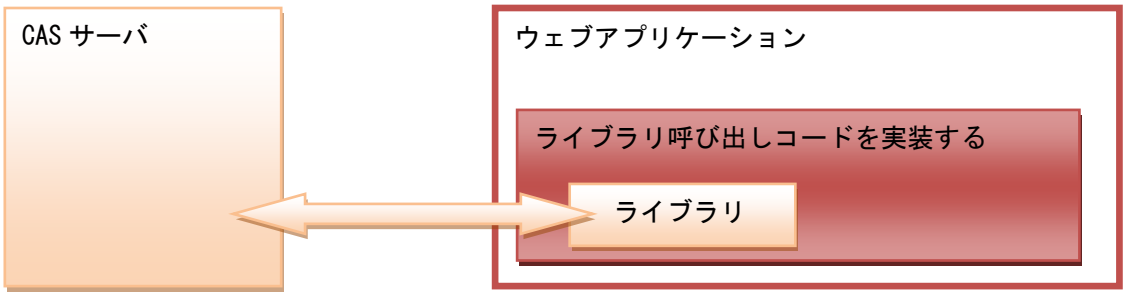
※属性名に括弧がついたベース属性は取得できません。

3. ロール認証の実装方法

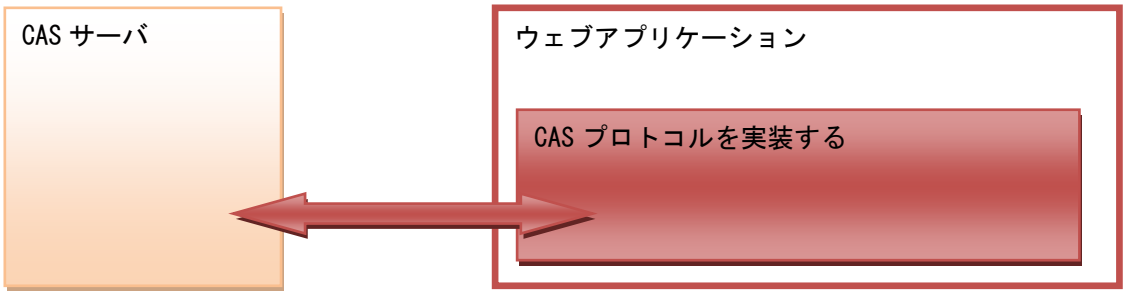
ウェブアプリケーションにロール認証を実装する場合、クライアント用ライブラリを利用する方法と、CAS プロトコルを実装する方法があります。

次に実装する範囲の概要を示します。

クライアント用ライブラリを利用する場合



CAS プロトコルを実装する場合



3.1. Java 用のライブラリを使用する場合

3.1.1. 概要

Java 用のライブラリは、サーブレットのフィルターとして使用します。

必要なライブラリをプロジェクトに入れ、web.xml ファイルにフィルターを設定します。

3.1.1. 必要なライブラリ

ロール認証サービス用に名古屋大学で拡張したライブラリを使用します。

cas-client-core.jar	CAS クライアントの本体
nucas-client-api.jar	ロール認証サービス用に拡張した API
commons-logging.jar	ログ出力用

3.1.2. フィルターとして使用するクラス

フィルターとして使用するクラスを次に示します。

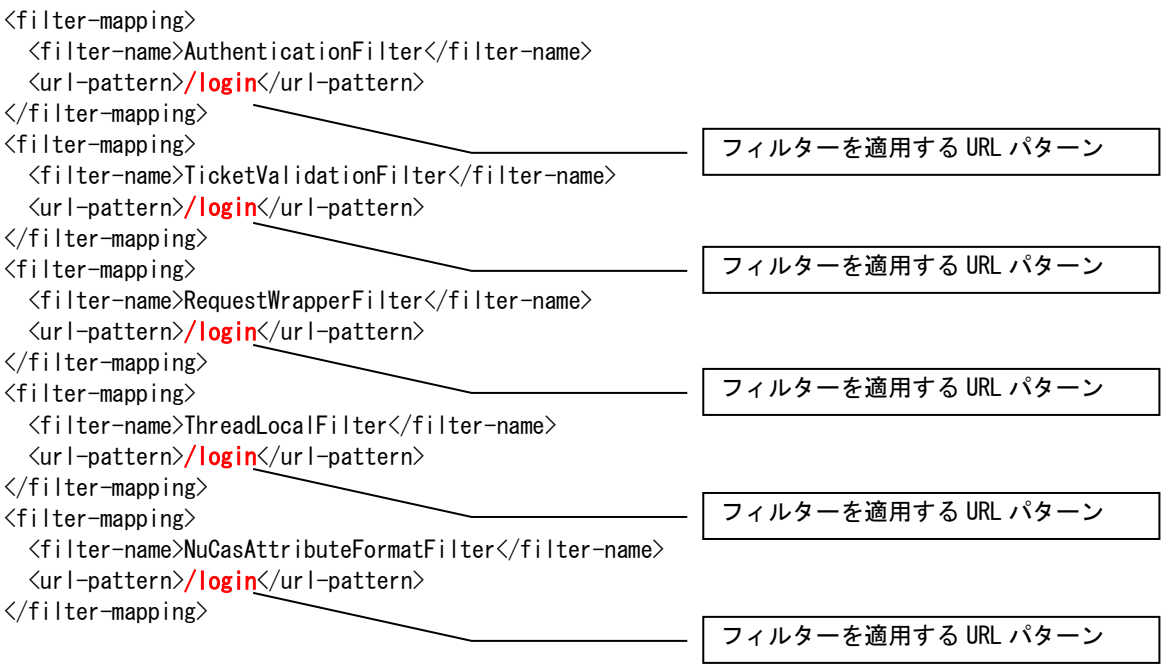
順序	クラス名	必須	内容
1	org.jasig.cas.client.authentication .AuthenticationFilter	○	ロール認証を行う。
2	org.jasig.cas.client.validation .Cas20ProxyReceivingTicketValidationFilter	○	検証を行う。
3	org.jasig.cas.client.util .HttpServletRequestWrapperFilter	○	属性情報を request 情報から取得できるようにする。
4	org.jasig.cas.client.util .AssertionThreadLocalFilter	○	検証結果を ThreadLocal に保持する。
5	nu.cas.client.util .NuCasAttributeFormatFilter	○	属性名にアンダーバー2 つ「__」がある場合、セミコロン「;」に変換する。

3.1.1. web.xml の記述

web.xml に追加する記述例を次に示します。また、ウェブアプリケーション毎に変更する箇所を赤字で示します。下記では申請したウェブアプリケーションの URL が「http://webapplication.jp/login」の場合を例にしています。

```
<filter>
  <filter-name>AuthenticationFilter</filter-name>
  <filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>
  <init-param>
    <param-name>casServerLoginUrl</param-name>
    <param-value>https://auth.nagoya-u.ac.jp/cas/login</param-value>
  </init-param>                                                    CAS サーバのログイン URL
  <init-param>
    <param-name>serverName</param-name>
    <param-value>http://webapplication.jp/</param-value>
  </init-param>                                                    ウェブアプリケーションの URL
</filter>
<filter>
  <filter-name>TicketValidationFilter</filter-name>

<filter-class>org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter</filter-class>
  <init-param>
    <param-name>casServerUrlPrefix</param-name>
    <param-value>https://auth.nagoya-u.ac.jp/cas/</param-value>
  </init-param>                                                    CAS サーバの URL
  <init-param>
    <param-name>serverName</param-name>
    <param-value>http://webapplication.jp/</param-value>
  </init-param>                                                    ウェブアプリケーションの URL
</filter>
<filter>
  <filter-name>RequestWrapperFilter</filter-name>
  <filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</filter-class>
  <init-param>
    <param-name>roleAttribute</param-name>
    <param-value>NagoyaUnivID</param-value>
  </init-param>                                                    request.isUserInRole メソッド
  <init-param>                                                     で検証するユーザ属性
    <param-name>ignoreCase</param-name>
    <param-value>false</param-value>
  </init-param>                                                    検証する文字列について大文字、小文字の区別
</filter>                                                          true=区別しない／false=区別する
<filter>
  <filter-name>ThreadLocalFilter</filter-name>
  <filter-class>org.jasig.cas.client.util.AssertionThreadLocalFilter</filter-class>
</filter>
<filter>
  <filter-name>NuCasAttributeFormatFilter</filter-name>
  <filter-class>nu.cas.client.util.NuCasAttributeFormatFilter</filter-class>
</filter>
```



3.1.1. ソースコードの記述

フィルターを通過した時点で、ロール認証が成功しています。認証で異常が発生した場合は、Exception となります。

CAS サーバが返す情報をウェブアプリケーションで利用する場合、Assersion を使って取得する場合と、NuCasClientUtil を使って取得する場合があります。

ログインユーザの検証をする場合

```
boolean boo = request.isUserInRole("zz1000000");
```

ログインユーザの属性情報を取得する場合

```
Assertion assertion=AssertionHolder.getAssertion();
Map<String, Object> map = assertion.getPrincipal().getAttributes();
map.get("NagoyaUnivID");
```

ログインユーザの所属情報を取得する場合

```
Assertion assertion=AssertionHolder.getAssertion();
List<Map<String, Object>> list = assertion.getSyozokuList();
list.get(0).get("bumon_id");
```

ログインユーザのロールリストを取得する場合

```
Assertion assertion=AssertionHolder.getAssertion();
List<NuCasRoleDetail> list = assertion.getRoleIdList();
list.get(0).getId();   # ロール ID の取得
list.get(0).getName(); # ロール名称の取得
```

ログインユーザのロールホルダーリストを取得する場合

```
Assertion assertion=AssertionHolder.getAssertion();
List<NuCasRoleDetail> list = assertion.getRoleHoldIdList();
list.get(0).getId();   # ロールホルダーID の取得
ist.get(0).getName();  # ロールホルダー名称の取得
```

ログインユーザの情報のみ取得する場合

```
NuCasUserBean user = NuCasClientUtil.getLoginUserBean(AssertionHolder.getAssertion());
```

委譲元ユーザの情報のみ取得する場合

```
List<NuCasUserBean> list = NuCasClientUtil.getIjyoUserBeans(AssertionHolder.getAssertion());
```

ログインユーザ、委譲元ユーザ含め、認証成功した全てのユーザの情報を取得する場合

```
List<NuCasUserBean> list =
 NuCasClientUtil.getAllUserOfAuthenticationSuccess(AssertionHolder.getAssertion());
```

ログインユーザが認証成功しているかの検証

```
boolean result = NuCasClientUtil.isAuthenticationSuccess(AssertionHolder.getAssertion());
```

NuCasUserBean からユーザ名を取得する場合

```
NuCasUserBean bean = NuCasClientUtil.getLoginUserBean(AssertionHolder.getAssertion());
String userName = NuCasClientUtil.getUserFullName(bean);
```

NuCasUserBean から属性情報を取得する場合

```
NuCasUserBean bean = NuCasClientUtil.getLoginUserBean(AssertionHolder.getAssertion());
Map<String, Object> attr = bean.getAttributes();
```

NuCasUserBean からロールリストを取得する場合

```
NuCasUserBean bean = NuCasClientUtil.getLoginUserBean(AssertionHolder.getAssertion());
List<NuCasRoleDetail> list = bean.getRoleList();
```

NuCasUserBean からロールホルダーリストを取得する場合

```
NuCasUserBean bean = NuCasClientUtil.getLoginUserBean(AssertionHolder.getAssertion());
List<NuCasRoleHoldDetail> list = bean.getRoleHoleList();
```

NuCasUserBean から所属情報を取得する場合

```
NuCasUserBean bean = NuCasClientUtil.getLoginUserBean(AssertionHolder.getAssertion());
List<NuCasSyozokuBean> list = bean.getSyozokuList();
```

ログインユーザの在籍状態（enrollment）を取得する場合

```
String enrollment = NuCasClientUtil.getEnrollment(AssertionHolder.getAssertion());
```

ロール ID からログインユーザの在籍状態（enrollment）を取得する場合

```
String enrollment = NuCasClientUtil.getEnrollmentByRole(AssertionHolder.getAssertion(), role_id);
```

ロールホルダーID からログインユーザの在籍状態（enrollment）を取得する場合

```
String enrollment = NuCasClientUtil.getEnrollmentByRoleHolder(AssertionHolder.getAssertion(),
roleholder_id);
```

3.1.2. ログアウト

ウェブアプリケーションのログアウト処理を行った後、ロール認証サーバのログアウト URL「https://auth.nagoya-u.ac.jp/cas/logout」にリダイレクトします。

ロール認証サーバのログアウト後、ウェブアプリケーションの画面を再度表示させたい場合、「https://auth.nagoya-u.ac.jp/cas/logout?service=http://test.jp/logout」の様に、service パラメータで URL を指定できます。

3.2. PHP 用のライブラリを使用する場合

3.2.1. 概要

PHP 用のライブラリは、API として使用します。
必要なライブラリをディレクトリに入れ、認証 API をコールします。

3.2.2. 必要なライブラリ

ロール認証サービス用に名古屋大学で拡張したライブラリを使用します。

CAS	CAS クライアントの本体を含むディレクトリ
CAS.php	ロール認証サービス用の API
config.php	CAS サーバの設定

3.2.3. 認証の仕方

必要なファイルをインクルードし、認証用 API を実行します。

```
// 必要なファイルをインクルード
include_once('config.php');
include_once('CAS.php');

// デバッグ出力する場合（引数でファイル名を指定可能）
phpCAS::setDebug();

// CAS クライアントの初期設定
phpCAS::client(CAS_VERSION_2_0, $cas_host, $cas_port, $cas_context);

// 証明書の検証についてどちらかの行をコメントアウト
//phpCAS::setCasServerCACert($cas_server_ca_cert_path);
phpCAS::setNoCasServerValidation();

// 認証 API の実行
phpCAS::forceAuthentication();
```

3.2.4. ソースコードの記述

認証 API を通過した時点で、ロール認証が成功しています。
CAS サーバが返す情報をウェブアプリケーションで利用する場合、API を使って取得します。

ログインユーザの属性情報を取得する場合
```
$attributes = phpCAS::getAttributes();
$attributes["NagoyaUnivID"];
```

ログインユーザの所属情報を取得する場合
```
$syozokuList = phpCAS::getSyozokuList();
```

ログインユーザのロールリストを取得する場合
```
$roleIdList = phpCAS::getRoleIdList();
```

ログインユーザのロールホルダーリストを取得する場合
```
$roleHoldIdList = phpCAS::getRoleHoldIdList();
```

委譲元ユーザの情報のみ取得する場合
```
$ijyoUser = phpCAS::getIjyoUserList();
```

ログインユーザが認証成功しているかの検証
```
$authenticationSuccess = phpCAS::isAuthenticationSuccess();
```

ログインユーザの在籍状態（enrollment）を取得する場合
```
$enrollment = phpCAS::getEnrollment();
```

ロール ID からログインユーザの在籍状態（enrollment）を取得する場合
```
$roleEnrollment = phpCAS::getEnrollmentByRole($roleId);
```

ロールホルダーID からログインユーザの在籍状態（enrollment）を取得する場合
```
$roleHolderEnrollment = phpCAS::getEnrollmentByRoleHolder($roleHoderId);
```

3.2.5. ログアウト

ウェブアプリケーションのログアウト処理を行った後、ロール認証サーバのログアウトをするために API を実行します。。

ロール認証サーバのログアウト後、ウェブアプリケーションの画面を再度表示させたい場合、service パラメータで URL を指定できます。

```
// CAS サーバからのログアウト
phpCAS::logout();
// CAS サーバからログアウト後に、特定 URL へリダイレクトする場合
//phpCAS::logout(array('service'=>'http://mysite/'));
```

3.3. Apache 用のライブラリを使用する場合

3.3.1. 概要

Apache の基本認証にロール認証サービスが使用できます。

使い方は「https://wiki.jasig.org/display/CASC/mod_auth_cas」に記載されていますので、本マニュアルでは必要な部分のみ記載します。

なお、CAS サーバから返る属性情報を使用したい場合は、モジュールのソースコードを変更する必要があります。

3.3.2. モジュールの作成

ソースファイルをウェブからダウンロードし、コンパイルします。

```
$ apxs -i -lssl -lcurl -c mod_auth_cas.c
```

3.3.3. モジュールの配置と設定

作成した mod_auth_cas.so を modules に配置します。例えば/etc/httpd/modules です。

httpd.conf に追加します。

```
LoadModule auth_cas_module modules/mod_auth_cas.so
CASDebug                On
CASCertificatePath      /usr/share/purple/ca-certs/
CASValidateServer       On
CASLoginURL             https://auth.nagoya-u.ac.jp/cas/login
CASValidateURL          https://auth.nagoya-u.ac.jp/cas/serviceValidate
CASCookiePath           /tmp/
CASAllowWildcardCert    On
CASValidateDepth        3
```

認証をかける場所を指定します。

```
Alias /mod_cas /var/www/html/mod_cas
<Directory "/var/www/html/mod_cas">
    AuthType CAS
    Require valid-user
</Directory>
```

3.4. その他の言語の場合

「2 CAS 認証メカニズム」に示すプロトコルを実装すると、ロール認証サービスを利用できます。

4. 権限委譲の実装方法

4.1. 取得する属性情報について

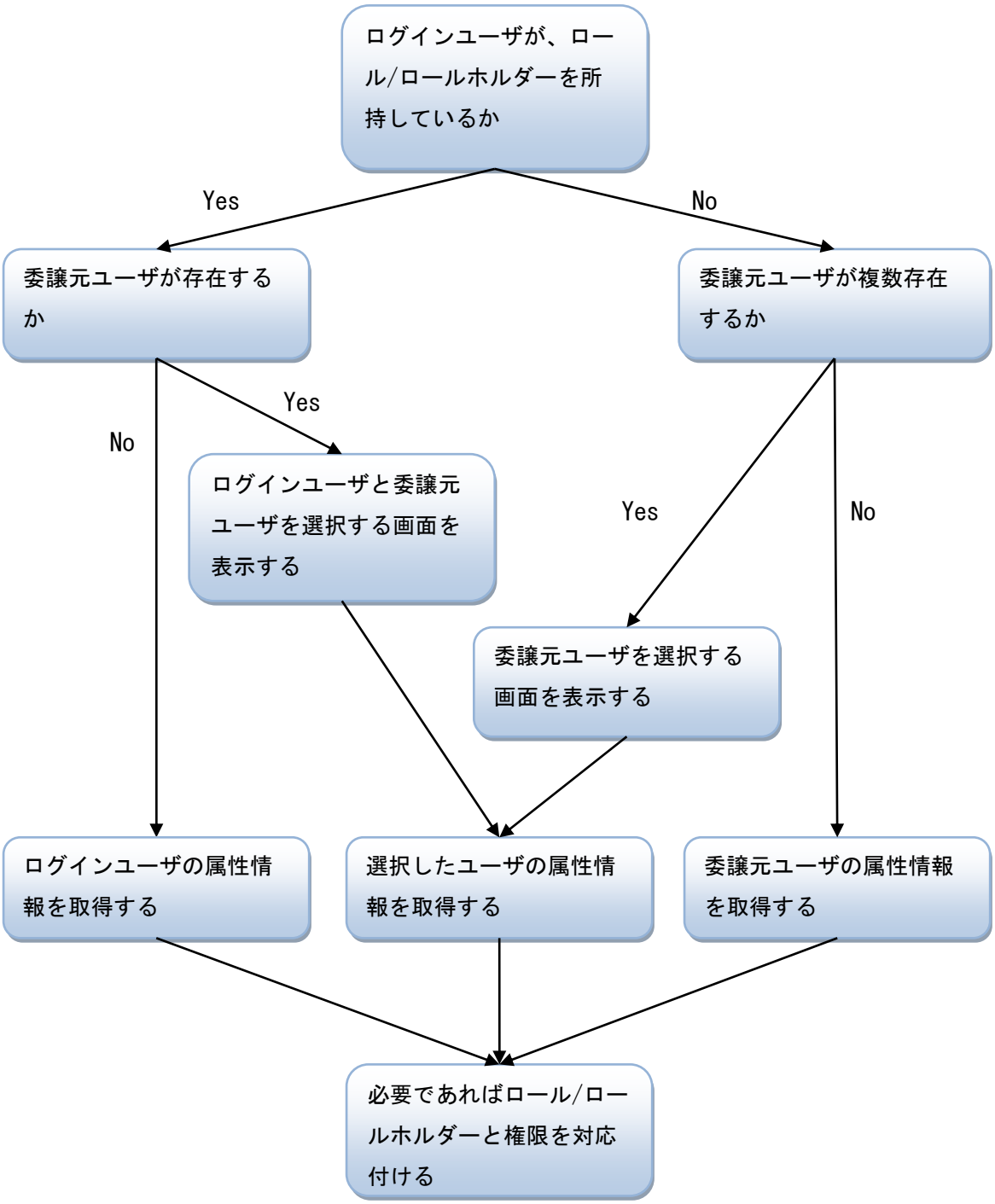
権限委譲を許可するウェブアプリケーションでは、ログインユーザと委譲元ユーザのロール/ロールホルダーの所持状況により、取得する属性情報が異なります。

ウェブアプリケーションを利用できるロール/ロールホルダー所持状況による、取得する属性情報の組合せを次に示します。

<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="2">ログインユーザが
アプリケーション利用可能ロール/ロールホルダーを</td></tr>
<tr><td>所持する</td><td>所持しない</td></tr>
<tr><td rowspan="2">委譲元ユーザがアプリケーション利用可能ロール/ロールホルダーを</td><td>所持する</td><td>ログインユーザの属性情報・ロール/ロールホルダー
＋
委譲元ユーザの属性情報・ロール/ロールホルダー</td><td>ログインユーザの属性情報
＋
委譲元ユーザの属性情報・ロール/ロールホルダー</td></tr>
<tr><td>所持しない</td><td>ログインユーザの属性情報・ロール/ロールホルダー</td><td>ログインエラー</td></tr>
</table>

4.2. 実装する処理の流れ

ウェブアプリケーションが実装する処理の例を次の図に示します。



4.3. Java の実装例

別途提供する「javaCasClientDelegationSample」を参照してください。

5. 権限委譲の設定方法

誰から誰へ権限委譲するかは、ウェブアプリケーション管理者が行います。

注意事項

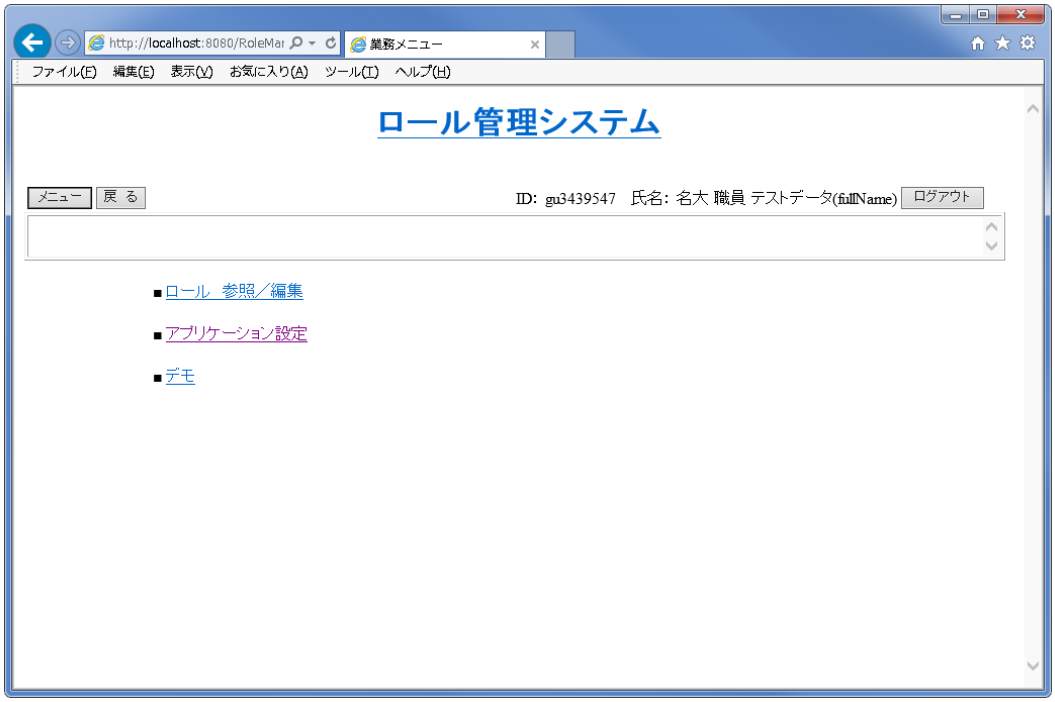
権限委譲は身分を制限せず設定できますので、アプリケーションの特性によって、教員から TA へは委譲しない等、ルールを明確にし、利用者への周知徹底を行ってください。

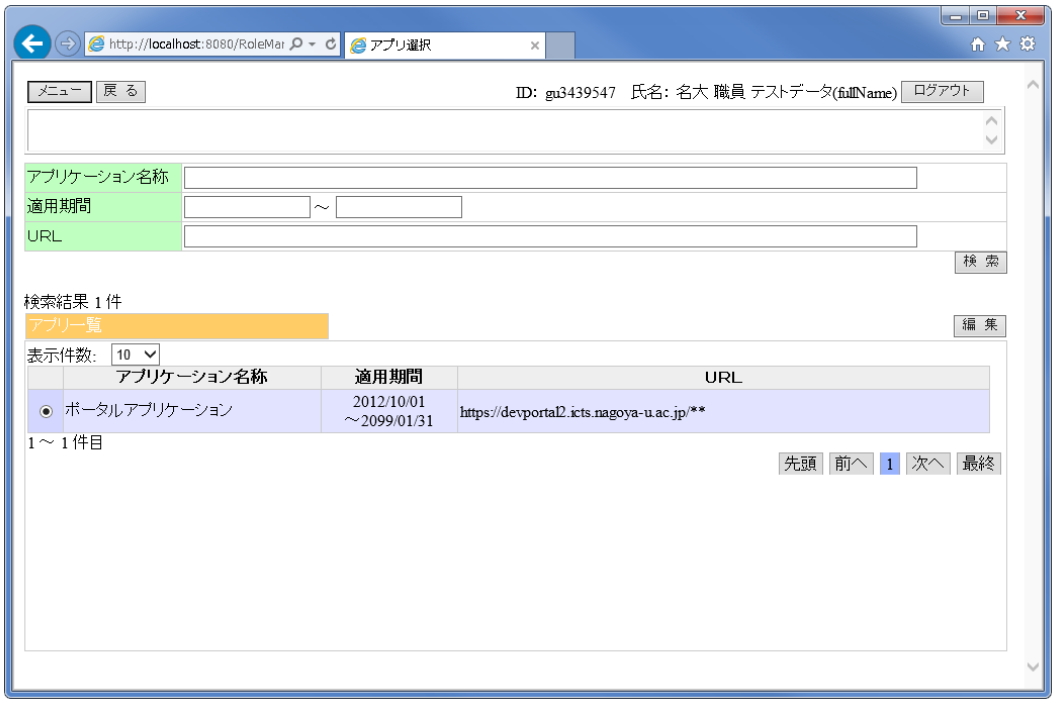
また、アプリケーション管理者は、不正な権限委譲が設定されていないか、定期的に権限委譲の状況を確認してください。

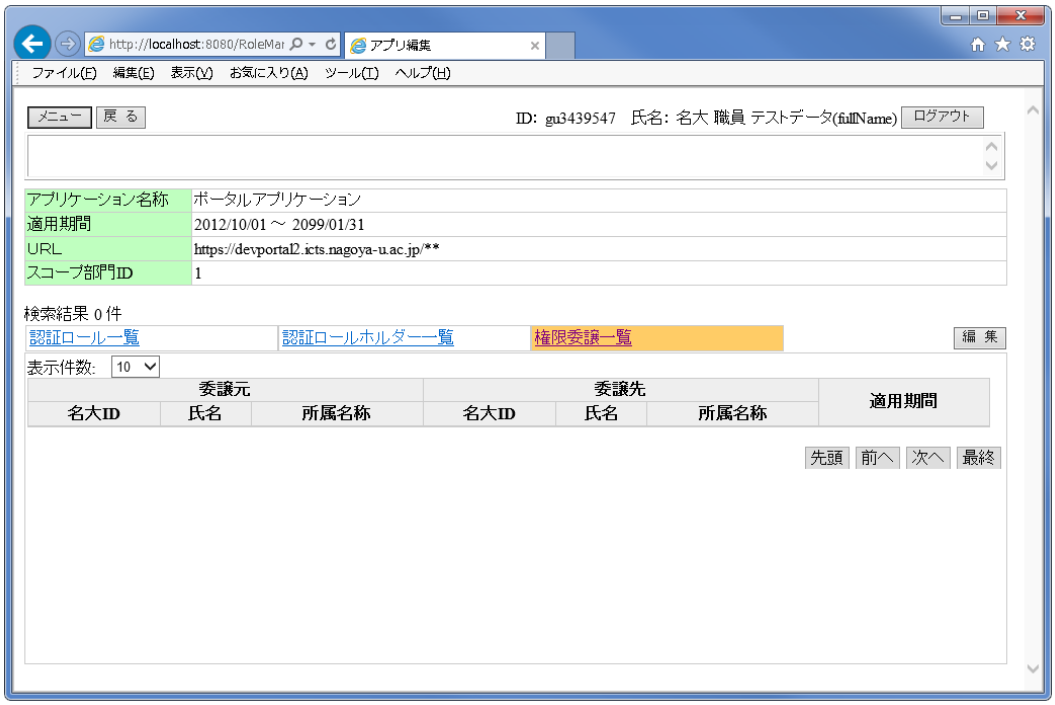
5.1. アプリケーション管理者による権限委譲の設定

「https://auth.nagoya-u.ac.jp/RoleManagement/」をブラウザで開きます。

名古屋大学 ID でログインすると、管理しているアプリケーションの一覧が表示されますので、権限委譲を設定するアプリケーションを選択します。




http://localhost:8080/RoleMar
アプリ選択
メニュー
戻 る
ID: gu3439547 氏名: 名大 職員 テストデータ(fullName)
ログアウト
アプリケーション名称
適用期間
～
URL
検 索
検索結果 1 件
アプリ一覧
編 集
表示件数: 10
アプリケーション名称
適用期間
URL
ポータルアプリケーション
2012/10/01 ～2099/01/31
https://devportal2.icts.nagoya-u.ac.jp/**
1 ～ 1 件目
先頭
前へ
1
次へ
最終


http://localhost:8080/RoleMar
アプリ編集
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
メニュー
戻 る
ID: gu3439547 氏名: 名大 職員 テストデータ(fullName)
ログアウト
アプリケーション名称
ポータルアプリケーション
適用期間
2012/10/01 ～ 2099/01/31
URL
https://devportal2.icts.nagoya-u.ac.jp/**
スコープ部門ID
1
検索結果 0 件
認証ロール一覧
認証ロールホルダー一覧
権限委譲一覧
編 集
表示件数: 10
委譲元
委譲先
適用期間
名大ID
氏名
所属名称
名大ID
氏名
所属名称
先頭
前へ
次へ
最終

5.2. ロール ID・ロールホルダーID の確認方法

ウェブアプリケーションの管理者は、ロール認証サービスの管理画面「https://auth.nagoya-u.ac.jp/RoleManagement/」にログインし、ウェブアプリケーションに対して許可したロール、およびロールホルダーの ID を確認できます。