アイデンティティフェデレーションにおける グループ管理 UI 提供のための SCIM 拡張の詳細設計と実装

西村健[1]　清水さや子[1]　古村隆明[1,2]

概要：様々なサービスでグループの機能が実現されている．サービスごとに作成されるグループは，それぞれのサービス内に限定され，他のサービスで使用することが難しい．ある 1 つのグループを複数のサービスで使用したい場合，グループの共通化が求められる．これに対して，著者らはグループを属性プロバイダーとして管理するためのサービスとして，「mAP Core」を開発し運用している．一般ユーザは，グループ管理のためのサービスが独立しているという認識がない場合が多く，利用するサービス経由でグループ管理をしようとするときに，mAP Core 上のインターフェースが異なりすぎて，混乱を招くことが多々発生していた．さらに，サービスによっては，mAP Core と連携する際に，大幅なシステム改修が発生する場合もあった．そこで，インターフェースを統一し，システム改修を容易にするために，サービスが呼び出すための API（Application Programming Interface）の開発が求められた．API の開発は，サービスとの連携を容易にできるよう，SCIM（System for Cross-domain Identity Management）をベースとし，その上で mAP Core の API を実装した．一般的な SCIM 実装では，トラステッド DB の管理者などが操作者となるが，一方 mAP Core ではアイデンティティフェデレーション上で提供されているものであるため，操作者はサービス管理者となり，操作者が持つ権限が自明でないという課題がある．この課題を解決するため，本研究では，SCIM サーバにロールとアクセス権の処理を導入し，操作者ごとに権限の整理を行うことを実現した．本論文では，SCIM の拡張で用いたロールとアクセス権などに関する詳細設計と，詳細設計にもとづいて実装した mAP Core API について紹介する．

キーワード：認証認可，グループ，アイデンティティフェデレーション，属性プロバイダー，API，SCIM

Detailed Design and Implementation of SCIM Extension for Providing Group Management UI in Identity Federation

TAKESHI NISHIMURA[†1]　SAYAKO SHIMIZU[†1]
TAKAAKI KOMURA[†1,2]

Abstract: Group functions are implemented in various services. Groups created for each service are limited within each service and are difficult to use in other services. If you want to use one group for multiple services, group sharing is required. In contrast, the authors develop and operate "mAP Core" as a service for managing groups as attribute providers. General users often do not recognize that the services for group management are independent, and when trying to manage groups via the services they use, the interfaces on mAP Core are too different, causing confusion. A lot of things were happening. Furthermore, depending on the service, there were cases where major system modifications occurred when linking with mAP Core. Therefore, in order to unify the interface and facilitate system modification, it was required to develop an API (Application Programming Interface) for calling services. The development of the API was based on SCIM (System for Cross-domain Identity Management) to facilitate linking with services, and on top of that, the mAP Core API was implemented. In a typical SCIM implementation, the administrator of the trusted DB is the operator, but mAP Core is provided on the identity federation, so the person who accesses the API of mAP Core is the service administrator. There is a problem that the authority possessed by the operator is not self-evident. In order to solve this problem, in this research, we introduced the role and access right processing to the SCIM server, and realized to organize the rights for each operator. This paper introduces the detailed design of the roles and access rights as the SCIM extension, and the mAP Core API implemented based on the detailed design.

Keywords: Authentication Authorization, Group, Identity Federation, Attribute Provider, API, SCIM

1. はじめに

近年，情報技術の発展や Covid-19 の対応などにより，教育，研究や業務などにおいて，より一層多くの仕組みのオンライン化が進められている［1］［2］［3］．オンラインサービスの増加にともない，重要となるのがアイデンティティフェデレーションである．アイデンティティフェデレーションは多くの国で構築され，日本では，2010 年に，国立情報学研究所を中心に「学認」と呼ばれる学術認証フェデレーションが構築され，活動を開始している［4］［5］［6］．アイデンティティフェデレーションでは，認証情報の管理は，組織内やアプリケーションベンダーなどが提供するサービス（Service Provider［SP］）側では行わず，組織ごとのアイデンティティプロバイダー（Identity Provider［IdP］）側

1 国立情報学研究所
National Institute of Informatics
2 京都大学
Kyoto University

で行われる．一方，認可情報は，①SP 側でアクセスを許可するユーザリストとして保持するか，②IdP 側に格納されている属性の値に基づくか，①と②を組み合わせて指定するのが一般的であった．

SP の増加および多様化により，SP の認可で使用するユーザの集合は，重複する場合も多く，SP ごとに管理するのでは効率が悪かった．そのため，著者らは，アイデンティティフェデレーション上で，認可に使用するユーザのリストをグループとして作成するためのサービスとして「mAP Core」を開発し，運用してきた[7]. mAP は，member Attribute Provider の略であり，Core は，Web 上のユーザーインターフェース（以下，UI とする）を SP 側に委ねたいという将来的な展望から，mAP Core と名付けている．現状の mAP Core では，グループの管理を mAP Core 上で行う．このためグループ管理の操作が必要な際には，利用する SP から mAP Core の UI へリダイレクトが行われる．しかし，一般ユーザにとっては，グループ管理のための別のサービスが存在するという認識がない場合が多く，利用する SP と mAP Core 上で操作するための UI や操作性が異なりすぎて，混乱を招くことが頻繁に発生していた．また，SP 側では mAP Core と連携する際に，大幅なシステム改修が発生する場合もあった．そのため，これらの課題を解決するために，mAP Core の API（Application Programming Interface）の開発が求められた（以下，mAP Core API と呼ぶ）．

アイデンティティフェデレーション上で mAP Core API を実現する際に使用する技術としては，LDAP［8］，SCIM（System for Cross-domain Identity Management）[9]，Grouper［10］などのプロトコルが考えられるが，本研究では，これらを比較し，SP 側との連携を容易に実現できる仕組みである SCIM を用いることとした．SCIM は Microsoft や AWS など多くのサービスで利用されている技術であるが，一般的な実装では SCIM の操作者は，トラステッド DB の管理者などである．しかし，mAP Core は，アイデンティティフェデレーション上で実装されていることより，API にアクセスする人は連携する SP の管理者であり，それぞれの操作者が持つ権限が自明でないという課題がある．mAP Core API では，多くの SP が利用できるようサーバ対クライアントが「一対多」になるようにして，ユーザごと，かつ，SP ごとに権限を設定できなければならない．本研究では，これらの課題を解決するための手法として，SCIM を拡張し，ロールとアクセス権限の処理を導入することで，操作者ごとに権限の整理を行うことを実現した．

2 章では，関連技術について述べ，3 章では SCIM を用いた mAP Core API の詳細設計について述べる．4 章では mAP Core API の実装について述べ，5 章ではまとめを述べる．

2. 関連技術

2.1 アイデンティティフェデレーション

アイデンティティフェデレーションは，多くの国で構築されている．日本でも，2010 年より，国立情報学研究所が中心になって「学術認証フェデレーション[学認（GakuNin）]」という名称で研究開発され［4］，現在，約 300 機関が参加している．

学認では，Shibboleth［11］を用いた認証を行っている．学認への参加に際しては，参加申請と，参加機関ごとにユーザの認証を処理する IdP の構築が必要になる．IdP は，機関ごとにユーザのアカウントを管理する．SP におけるログイン時に，SP は該当ユーザのアカウント情報を管理する IdP へ認証要求を行う．IdP は認証を行い SP に結果を返す．その後，SP 側で管理している認可情報に基づき，認可を行う．一般的な SP の認可情報は，SP 側でアクセスを許可するユーザリストとして保持するか，IdP サーバに格納されている属性の値に基づくか，これらを組み合わせて指定するのが一般的である．SP の数は，年々増加し，電子ジャーナルや E-Learning ［12］，研究データ基盤［13］など，2022 年 11 月現在では，100 以上のサービスが存在し，サービス内容や認可方法，UI も多様化している．

2.2 属性プロバイダー

2.2.1 mAP Core の概要

SP の増加や多様化に伴い，SP の認可の範囲も多様化し，IdP 上で保持する属性情報だけでは認可に不足するようになり，また SP の認可のために保持するユーザリストが他の SP と重複することが多くなるなど，SP に対する認可の効率化に向けた検討が求められた．

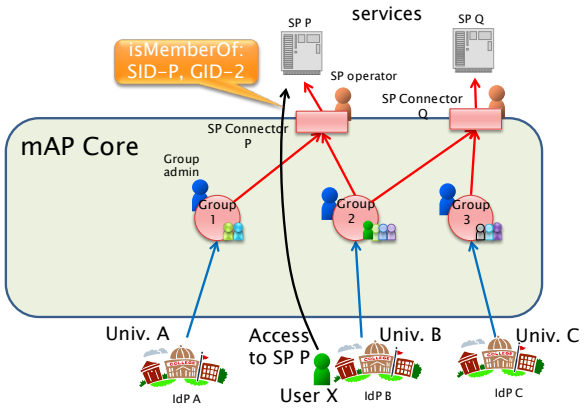

図 1　mAP Core の概要
Figure 1　Implementation of mAP Core

そこで，著者らは，属性プロバイダーとして，IdP が保持するユーザのリストを基に，任意の「学認 ID の集合」であるグループを作成でき，そのグループ情報を SP での認可

に利用可能とする属性プロバイダー「mAP Core」を開発し運用している（図 1）．また，mAP Core と連携し，ユーザが SP の利用時に，所属機関ごとに利用できる SP と，所属グループごとに利用できる SP を一覧で表示できるゲートウェイサービスである「学認クラウドゲートウェイサービス」を開発し，運用を行っている［7］．

2.2.2 mAP Core の課題

mAP Core のグループの管理を行いたい場合は，利用するサービスからリダイレクトして mAP Core 上で操作を行う必要がある．しかし，一般ユーザには，グループ管理のための別のサービスが存在するという認識がない場合が多く，利用する SP と mAP Core 上の UI が異なりすぎて，混乱を招くことが頻繁に発生していた．また，SP 側では，もともとグループ管理機能を持っているサービスの場合，mAP Core と連携すると，もともと保持しているグループに関する DB と異なる場所に DB を持たなければならなくなるなど，大幅なシステム改修が発生する場合もあった．また，グループ管理機能を持っていないサービスにおいても，作成されたグループとの連携の際にシステム改修が必要なサービスが多かった．そこで，これらの課題の解決のための新たなシステムの開発が求められた．

そこで，著者らは，サービス側の UI を用いてグループ管理を行うことができ，グループに関する DB の大幅な改修も不要で，スムーズにグループ管理が行うことができる新たな仕組みとして，mAP Core API の開発を行った．

2.3 mAP Core API の設計に向けた関連技術

mAP Core API を設計する際，API サーバとなる mAP Core 内の実装を容易にするだけではなく，アイデンティティフェデレーション上の各サービスが API を容易に利用できるようにするために，仕様を軽量にし，一般に利用されている API やプロトコルとできる限り互換性を持たせることを重視した．

このような方針のもと，以下の既存技術について調査した結果，mAP Core API は，SCIM を元にして，mAP Core に必要な拡張を SCIM の独自スキーマ内で表現する方法を採用することとした．

- LDAP
- Grouper
- SCIM

以下に，各既存技術に関する詳細を示す．

一つ目の LDAP はディレクトリ管理のプロトコルとして古くから存在し，現在も多くの組織で利用されている［14］．LDAP は相互運用性も高く，ACL などによる認可の仕組みも十分に整っているが，LDAP を用いて管理する対象が X.500 モデルに基づくディレクトリ情報である事が問題になった．

mAP Core の核となるグループ管理用の情報は RDB を用いて既に実装されており，LDAP の扱うディレクトリとは異なる構造を持っているため，LDAP のプロトコルで扱おうとするとデータ表記が不自然なものになる．また，LDAP の認可機能は，LDAP のディレクトリ情報に強く紐付く形で定義・利用されている．これらの点から，mAP Core との親和性が低いと判断し，LDAP は検討対象から外すこととした．

二つ目の Grouper は Shibboleth と共に InCommon［15］で開発されており，グループ情報を集中管理して，様々な外部サービスに対してプロビジョニングを行うツールである［16］［17］．Grouper と mAP Core は共にグループ情報の集中管理という同じ目的を持つが，前者は他サービスへプロビジョニングを行うことが主目的であるのに対し，後者は SAML 属性プロバイダーとして動作し，利用者が SAML 認証を行ったタイミングでグループ情報を付加することが主目的である事が，両者の大きな違いであり，mAP Core を独自に実装した理由でもある．

Grouper では，Grouper への情報登録・変更用に WebAPI が提供されている．JSON，XML，XHTML など複数のデータ表記法に対応し，SOAP と REST に沿った二種類の設計方針に基づいた，複数の WebAPI が提供されていた．本 API は仕様を軽量にするという方針のため，Grouper の WebAPI のうち，参考にするのは JSON + REST 形式にした．

三つ目の SCIM も JSON + REST 形式で，アカウント情報をクラウドサービス間などでプロビジョニングするために 2011 年に登場した新しいプロトコルで，相互運用性の向上と複雑性の減少を掲げて設計された．グループ管理のためのグループ属性も SCIM の標準スキーマに含まれている［18］［19］［20］［21］．

Grouper API と SCIM との最大の違いは，前者は Grouper で定義されているデータベースを制御するために設計されているのに対し，後者は特定のデータ構造を前提としていない点であった．Grouper API は，Grouper 内部で利用されている属性名や構造が JSON にも出現しており，Grouper の設計を知らないと意味を理解できない情報やデータ構造を含んでいる．一方 SCIM では，属性名は直接的で，データ構造もシンプルでわかり易い．また，Amazon Web Services SSO，Microsoft Azure Active Directory，Oracle Identity Cloud Service，IIJ ID サービス，Slack など多くのクラウドサービスが SCIM でのプロビジョニングを受け付けおり，SCIM に関する仕様や使用例などの情報が広く公開されているなどの点も考慮して，mAP Core API では SCIM を元にして API を実装することとした．

3.mAP Core API の詳細設計

3.1 SCIM と SCIM への拡張の必要性

本節では SCIM の概要と，SCIM には存在しなかったロールに基づくアクセス権の管理機能の必要性について述べる．

SCIM の典型的な用途は，組織内のトラステッド DB から，利用するサービスに対してアカウント情報やグループ情報をプロビジョニングする事である．
サービス側が SCIM サーバ，トラステッド DB 側が SCIM クライアントとして動作するが，一般的な用途では，SCIM サーバから見て SCIM クライアントは一つに固定される（図 2 左）．

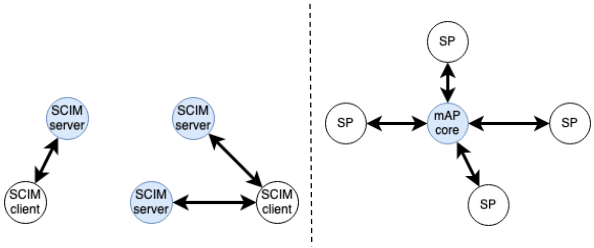

図 2　左:典型的な SCIM のサーバ・クライアントの関係
右: mAP Core API でのサーバ・クライアントの関係
Figure 2　Left: typical SCIM server-client relationship
Right: server-client relationship in mAP Core API

なお，クラウドサービスでは，テナントやドメインといった単位で管理者が設定され，管理者はその範囲内でアカウントやグループ情報を管理することになる．クラウドサービス全体では SCIM クライアントが多数存在する環境であっても，特定のテナントやドメインだけに注目すると，SCIM サーバに対する SCIM クライアントはやはり一つに固定されている．

SCIM サーバに対して SCIM クライアントが一つに固定されていれば，クライアントに全権を委ねて良く，SCIM で交換する情報に対して複雑なアクセス制御を行う必要がないため，仕様も動作もシンプルに保つことができる．SCIM はそのようにシンプルに設計されている．

一方，mAP Core が想定するように，アイデンティティフェデレーション上の様々なサービスがグループ情報を統合管理しようとする局面で SCIM を利用する場合，mAP Core が SCIM サーバ，様々なサービスが SCIM クライアントとして動作することになる．

この状況では，一つの SCIM サーバに対して，多数の SCIM クライアントがアクセスを行うことになる（図 2 右）．各サービスは様々な管理主体のものが含まれることを想定しているため，SCIM で書き込まれた情報に，全ての SCIM クライアントが対等にアクセスできてしまう事は望ましくない．

例えば，サービス A で作成したグループ a の情報に対して，サービス A では自由にアクセスを許すが，無関係なサービス B ではログインしたユーザがグループ a に所属しているかどうかやグループ a のメンバー一覧の情報は取得できない，といった制御を行いたい場合がある．

そのため，サービス管理者ごとにロールを割り当て，ロールに基づいたアクセス制御を mAP Core 内に実装することとした．SCIM のプロトコルとして大きな変更はなく，アクセスを試みた管理者のロールによって，読み出せる属性や書き込める属性が増減するような変更になる．

3.2 オブジェクトの種類とそれぞれの関係

本 API の操作対象として，SCIM で規定される User および Group に加えて，Service を導入する．これは，アイデンティティフェデレーション上に存在し mAP Core と連携する SP を表すものである．

Group には任意の数の User が紐付けられ，ユーザがグループに所属していることを表す．内部的に一般メンバーとグループ管理者の区別があり，後者のみメンバーの追加削除などグループに対する操作を行えるものとする．

Service にも任意の数の User が紐付けられ，ユーザがサービスの管理者であることを表す．

Service と Group の間にも紐付けがあり，紐付けが存在することはグループがサービスを利用することを表している．これにより，グループに関する情報はそれが利用すると宣言されたサービスのみに提供され，無関係なサービスには提供されないということが可能となる．すなわちフェデレーション上の任意のサービスが全グループの情報を取得できるといったプライバシーの問題を防ぐことができる．詳細は Appendix で示す．

3.3 ロールとアクセス権

認証を受けたアカウントには，「システム管理者」「グループ管理者」「サービス管理者」といったロールがあり，そのロールに従って，アクセスできるデータやその範囲が規定される．表 1 に mAP Core API に規定されるロールの一覧を示す．

以降，認証には学認を用いるものとし詳細は割愛しているが，アカウントにはフェデレーション内で一意の ID が割り当てられており認証時に IdP から送信されるものとする．

これらのロールについて各オブジェクトが持つ Attribute へのアクセス権，情報取得に加え上書き・追加・削除の権限についての整理を行った．整理した結果の詳細を Appendix に示すが，簡易的に説明すると，何らかの関係性がある場合に閲覧の権限が与えられ，より強い関係性がある場合に操作の権限が与えられるものである．

表 1　オブジェクトとロール
Table 1　objects and roles

オブジェクト	Role	プログラム内の role 名
“User”	システム管理者	User_SystemAdmin
	ユーザ本人	User_UserHimself
	組織(IdP)管理者	User_IdpAdmin
	招待主	User_Invitation
	グループ管理者 (所属グループ)	User_GroupAdmin
	サービス管理者 (利用中サービス)	User_ServiceAdmin
“Group”	システム管理者	Group_SystemAdmin
	グループ管理者	Group_GroupAdmin
	サービス管理者 (紐付け+Admin)	Group_ServiceAdmin
	サービス管理者 (紐付け有り)	Group_ServiceGroupAdmin
	利用者(グループ参加)	Group_GroupMember
	他者(groupType==Public)	Group_Others
“Service”	システム管理者	Service_SystemAdmin
	サービス管理者	Service_ServiceAdmin
	他者	Service_Others

Appendix の詳細についていくつか補足を行う．

表 Appendix の付表 2 中の「利用者」「他者」に“（A）/（B）/（C）”と表示されているものについては，「memberListVisibility」Attribute，つまりグループに対するメンバー一覧の可視性の値によってアクセス権が決まるという記述であり，（A）/（B）/（C）の詳細は以下の場合にどのような権限があるかを表示している．

（A）: memberListVisibility == "Public"
（B）: memberListVisibility == "Private"
（C）: memberListVisibility == "Hidden"

なお，Public，Private，Hidden は，グループのメンバーリストの公開範囲を以下とする．

Public - メンバーでなくても閲覧可能
Private - メンバーに対してのみ公開
Hidden - メンバーであっても公開しない
（他のメンバーが誰かわからない）

また，R というアクセス権記述は，対象ユーザが自身の情報をサービスに送信することへの同意を別途取得しており，これを反映するものである．これはサービス管理者に対する記述であり，以下の情報の送信同意を反映することができるようになっている．

- name （氏名）
- mail （メールアドレス）
- ePPN （ユーザ ID）
- isMemberOf （所属するグループ ID）

3.4　ロール判定

認証で得られたアカウントとアクセスしようとしているオブジェクトから，その人のロールが決まる．オブジェクト毎に決まるロールの一覧は，表 1 に示した通りである．

一つのアカウントは，複数のロールを持ち得る．ロールはアカウントと対象オブジェクトをパラメーターとしてその関係性として求められる．

以下表 1「プログラム内の role 名」を使って詳細を述べる．

User_SystemAdmin/Group_SystemAdmin/Service_SystemAdmin は対象オブジェクトによらずシステム管理者に割り当てられるロールである．

User_IdpAdmin は対象 User が所属する IdP の管理者のロールである．IdP の管理者は別途管理されているものとする．

User_Invitation はユーザ本人ではなくサービスが代理で作成した User オブジェクトについてそのサービス管理者のロールである．

User_GroupAdmin は対象 User が所属するグループのいずれかのグループ管理者であるというロールである．

User_ServiceAdmin は対象 User が所属するいずれかのグループで利用しているサービスの管理者であるというロールである．

User を対象とする API については，上記に含まれないアカウントは閲覧を含めた操作を行うことができない．

Group_GroupAdmin は対象 Group の管理者であるというロールである．

Group_ServiceAdmin/Group_ServiceGroupAdmin は対象 Group が利用するサービスの管理者であるというロールである．デフォルトではグループに対する閲覧が許可されるが，フラグでサービスに対してメンバーの追加削除などより範囲の操作を許容することを可能としており，これをロールとして区別している．

Group_GroupMember は対象 Group のメンバーであるというロールである．

Group_Others は，Group を対象とする API について，上記に含まれない「他者」でも閲覧等可能な Public なグループがあり，その場合に割り当てられるロールである．

Service_ServiceAdmin は対象 Service の管理者であるというロールである．

Service については基本的に公開としており利用中かそ

うでないかなどに関わらず閲覧等可能になっており，上記に含まれない場合に一律 Service_Others というロールが割り当てられる．

4. mAP Core API の実装

4.1 実装環境

mAP Core API の実装では，リソースは AWS-RDS 上に，OS は Cent OS（64bit），Web サーバは Apache HTTP Server を用いる．mAP Core API は PHP を用いて実装されている（表 2）．mAP Core API は以下の要件を満たす RESTful API［22］をベースとして実装している．

1) クライアント，サーバ，およびリソースで構成されるクライアントサーバ・アーキテクチャ．
2) クライアントとサーバ間の通信がステートレスであること．サーバにクライアントのコンテンツが要求をまたいで保存されることはない．セッション状態に関する情報はサーバではなくクライアントが保持する．
3) クライアントとサーバ間のやりとりの一部を不要にするためのキャッシュ可能なデータ．
4) コンポーネント間で統一されたインターフェース．これにより，アプリケーションのニーズに固有の形式ではなく，標準化された形式で情報が転送される．
5) 階層化されたシステム制約．クライアントとサーバ間のやりとりは，階層レイヤーによって仲介できる．

表 2　構成モジュールとパッケージ
Table 2　Configuration modules and packages

モジュール	パッケージ
Web サーバ	Apache HTTP Server　2.4.6
mAP Core API V2	PHP（cakePHP）
リソース	AWS-RDS　5.4.16
OS	CentOS (64bit)　7.9.2009

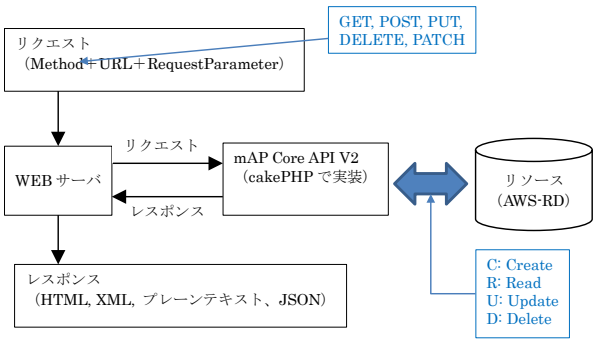

図 3　mAP Core API の動作原理図
Figure 3　Operation principle diagram of mAP Core API

一般的な REST 用に設計された API へのデータのリクエストは，通常ハイパーテキスト転送プロトコル(HTTP)を介して送信され，リクエストが受信されると，API は，DB に対して操作し，操作結果を様々な形式（HTML, XML, プレーンテキスト，JSON）でメッセージ（レスポンス）として返すものである（図 3）．

4.2 SCIM Schema

SCIM の実装は，SCIM schema と RDB table との対応表を作成し，SCIM オペレーションを RDB の CRUD にマッピングすることによって実現した．

mAP Core API は，mAP Core で管理している各種オブジェクトを SCIM Schema というモデルで提供する REST API である．図 4 のとおり，DB で管理されているリソースは SCIM Schema で抽象化され，その構造は隠蔽される．これにより，ユーザは，DB 間の関係を意識することなく，平易な SCIM Schema を理解し，操作 Method を適用すればよい．

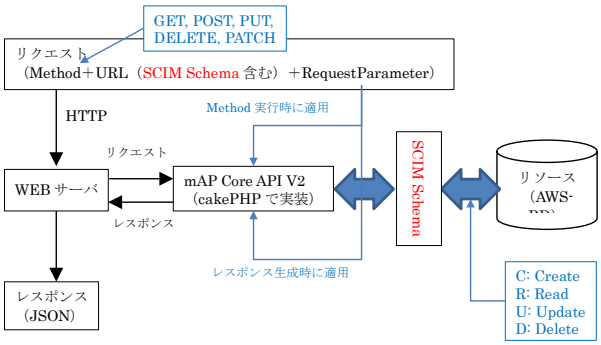

図 4　SCIM Schema を提供する mAP Core API
Figure 4 mAP Core API providing SCIM Schema

mAP Core API で定義される Schema はオブジェクトと同義で User / Group / Service の 3 種類あり，それぞれが Attribute の階層構造を持つ．例えば Group に対する members のように，一つの上位 Attribute に対して下位 Attribute は複数が対応し，下位 Attribute を array で表す．

4.3 API リクエストの処理手順

図 4 で示したように，mAP Core API の処理は，次のように行われる．

1. リクエスト（Method+URL+RequestParameter）と認証ヘッダ（Authorization Header）を https プロトコルで Web サーバに送る．
2. Authorization ヘッダを解析し，アクセスアカウントを認証する．
3. URL から Method の操作対象を決める．
4. RequestParameter を Method のオプションと操作結果からレスポンスに変換する時のオプションに分離する．
5. 操作対象に対して，Method をオプションと共に適用する．

① 認証されたアカウントから Role を決定する．
② URL と Method オプションは，SCIM Attribute 名を含むので，これを対応する内部 DB のテーブル名やフィールド名に変換する．
③ Role に基づき，内部 DB のテーブルを操作対象として Method に対応する CRUD 処理を行う．

6. 操作結果にレスポンス変換オプションを適用し，JSON 形式レスポンスを生成する．
7. クライアントに Status Code と共にレスポンスを返す．

5. まとめ

本研究では，アイデンティティフェデレーション上において実装されている属性プロバイダー, mAP Core の API 開発において，SCIM 技術を拡張して実現した．実現するにあたっては，SCIM 技術には存在しなかったロールに基づくアクセス権の管理機能を導入し SCIM を拡張することで，操作者ごとに権限の整理を行うことを実現した．

本研究においては，現時点では，SCIM 技術を拡張して mAP Core API の開発を行った段階である．今後，mAP Core API と，さまざまな SP との連携を行っていくことで，ユーザにグループ管理の際に混乱を招かないようなインターフェースの提供ができ，SP 側のグループに関する DB の大幅な改修が不要になるということを実証していく予定である．また，本研究が有効に機能する SP を明らかにするために SP の分類を示すことを検討している．

mAP Core API を使用することで，mAP Core のメリットを活かしつつ, SP に対するグループ管理がスムーズに行えるようになることを期待する．また，mAP Core API を利用する SP が増え, 1 つのグループが複数のサービスで使われることによる, SP 上におけるグループ管理の負担の軽減を期待する．

参考文献

［1］ 加藤 大弥，砂原 秀樹：研究科単位での学内サービスの ID 統一化に向けた IdP の導入および運用についての検討．インターネットと運用技術シンポジウム 2021 論文集，pp.17-23（2021）．
［2］ 土屋 雅稔，中村 純哉：豊橋技術科学大学における在宅勤務に対応した業務システムの構築．学術情報処理研究，25 巻 (2021)，1 号，pp.55-65（2021）．
［3］ 大向 一輝, 飯野 勝則，片岡 真，塩崎 亮，村上 遥：学術情報システムの標準化技術．情報の科学と技術，71 巻 (2021) 4 号，pp.152-158（2021）．
［4］ 学認：学認, 学認（オンライン），入手先 <https://www.gakunin.jp> (参照 2022-11-01).
［5］ 西村健, 中村素典, 山地一, 佐藤周行, 大谷誠, 岡部寿男, 曽根原登：多様なポリシーを反映可能な認証フェデレーション機構の実現，電子情報通信学会論文, 2013/6 Vol. J96-D No. 6（2013）．
［6］ 西村 健, 中村 素典, 山地 一禎, 大谷 誠, 岡部 寿男, 曽根原 登：日本における学術認証フェデレーション“学認”の展開，大学 ICT 推進協議会 2011 年度年次大会 論文集，D6-5（2011）．
［7］ 西村 健，清水 さや子，古村 隆明，坂根 栄作，合田 憲人：属性プロバイダーのゲートウェイサービスへの展開と運用評価．マルチメディア，分散，協調とモバイル（DICOMO2022）シンポジウム論文，pp.223-242（2022）．
［8］ RFC Editor：Lightweight Directory Access Protocol (LDAP)： The Protocol, The Protocol RFC 4511, LDAP (online): available from <https://www.rfc-editor.org/rfc/rfc4511> (accessed 2022-11-01).
［9］ Morey J. Haber, Darran Rolls: System for Cross-Domain Identity Management (SCIM), Identity Attack Vectors, pp.159-161(2020).
［10］ InCommon：Grouper, Grouper (online), available from <https://incommon.org/software/grouper/> (accessed 2022-11-01).
［11］ Shibboleth Consortium：Shibboleth, available from <https://www.shibboleth.net/> (accessed 2022-11-01).
［12］ 上田浩，古川雅子，浜元信州，中村素典，山地一禎：学認 LMS: Moodle における Shibboleth 認証とアプリケーション間認証の実現，電子情報通信学会，第 9 回バイオメトリクスと認識・認証シンポジウム(SBRA2019)講演論文集（2019）．
［13］ 込山悠介，林正治，加藤文彦，大向一輝，山地一禎：学術機関に向けた研究データの管理と共有のための共通基盤の開発，情報処理学会研究報告，Vol.2019-IOT-47, No.18（2019）．
［14］ RFC Editor：Lightweight Directory Access Protocol (LDAP)：Directory Information Models, LDAP (online), available from <https://www.rfc-editor.org/rfc/rfc4512> (accessed 2022-11-01).
［15］ InCommon：InCommon (online), available from <https://incommon.org/> (accessed 2022-11-01).
［16］ Grouper：Grouper Wiki Home (online), available from <https://spaces.at.internet2.edu/display/Grouper/> (accessed 2022-11-01).
［17］ Grouper：Grouper Web Services (online), available from <https://spaces.at.internet2.edu/display/Grouper/Grouper+Web+Services> (accessed 2022-11-01).
［18］ SCIM：SCIM System for Cross-domain Identity Management (online), available from <https://www.simplecloud.info/> (accessed 2022-11-01).
［19］ RFC Editor：System for Cross-domain Identity Management:Definitions, Overview, Concepts, and Requirements (online), available from <http://www.rfc-editor.org/rfc/rfc7642> (accessed 2022-11-01).
［20］ RFC Editor：System for Cross-domain Identity Management: Core Schema (online), available from <http://www.rfc-editor.org/rfc/rfc7643> (accessed 2022-11-01).
［21］ RFC Editor：System for Cross-domain Identity Management: Protocol (online), available from <http://www.rfc-editor.org/rfc/rfc7644> (accessed 2022-11-01).
［22］ Fielding, Roy Thomas：Architectural styles and the design of network -based software architectures,University of California, Irvine ProQuest Dissertations Publishing(2000).

Appendix　各オブジェクト（ユーザ，サービスなど）が持つ Attribute に対して，各ロールが持つアクセス権（r :read，w:write，d:delete，a:append を示す）

付表 1　User
Appendix 1　User

attribute name	システム管理者	ユーザ本人	組織(IdP)管理者	招待主	グループ管理者（所属グループ）	サービス管理者（利用中サービス）
.	rwd	rwd	rwd	rwd	r	r
schemas[]						
id	r	r	r	r	r	r
externalId	rw	rw	rw	rw	r	r
userName	r	rw	r	r	r	R
preferredLanguage	rwd	rwd	rwd	rwd	r	R
meta.resourceType	r	r	r	r	r	r
meta.created	r	r	r	r	r	r
meta.lastModified	r	r	r	r	r	r
meta.createdBy	r	r	r	r	r	r
eduPersonPrincipalNames[]	rd	rd	rd	r	r	R
eduPersonPrincipalNames[].eduPersonI	r	rd	r	r	r	R
eduPersonPrincipalNames[].idpEntity:	r	rd	r	r	r	R
emails[]	rwda	rwda	rda	r	-	R
emails[].value	rwd	rwd	rd	r	-	R
groups[]	rda	rwda	rda	ra	r	R
groups[].value	r	r	r	ra	r	r
groups[].$ref	r	r	r	r	r	r

付表 2　Group
Appendix 2　Group

attribute name	システム管理者	グループ管理者	サービス管理者（紐付け+Admin）	サービス管理者（紐付け有り）	利用者（グループ参加）	他者 groupType==Public	他者 groupType==Private
.	rwd	rwd	r	r	r	r	-
		上位グループ管理者					
schemas[]							
id	r	r	r	r	r	r	-
externalId	rw	rw	r	r	r	r	-
displayName	rwd	rwd	rwd	r	r	r	-
public	rwd	rwd	r	r	r	r	-
description	rwd	rwd	rwd	r	r	r	-
suspended	rw	rw	rw	-	-	-	-
deleted	rw	rw	rw	-	-	-	-
memberListVisibility	rw	rw	r	r	r	r	-
meta.resourceType	r	r	r	r	r	r	-
meta.created	r	r	r	r	r	r	-
meta.lastModified	r	r	r	r	r	r	-
members[]	rwda	rwda	rwda	r	rd / rd / d	r / a / -	-
members[].$ref	r	r	r	r	r / r / -	r / - / -	-
members[].type	r	r	r	r	r / r / -	r / - / -	-
members[].display	r	r	R	R	r / r / -	r / - / -	-
members[].value	r	r	r	r	r / r / -	r / - / -	-
members[].labels[]	rwad	rwad	rwad	r	r / r / -	r / - / -	-
administrators[]	rwda	rwda	rwda	r	r	r	-
administrators[].$ref	r	r	r	r	r	r	-
administrators[].display	rwd	rwd	Rwd	R	r	r	-
administrators[].value	r	r	r	r	r	r	-
services[]	rwda	rwda	rwda	rd	r	-	-
services[].$ref	r	r	r	r	r	-	-
services[].display	rwd	rwd	rwd	r	r	-	-
services[].value	r	r	r	r	r	-	-
services[].administratorOfGrou	rw	rw	rw	r	r	-	-

付表 3　Service
Appendix 3 Service

attribute name	システム管理者	サービス管理者	他者
.	rwd	rwd	r
schemas[]			
id	r	r	r
serviceName	rw	rw	r
serviceUrl	rw	rw	r
supended	rw	rw	r
deleted	w	w	-
meta.resourceType	r	r	r
meta.created	r	r	r
meta.lastModified	r	r	r
entityIds[]	rwda	rwda	r
entityIds[].value	rwd	rwd	r
administrators[]	rwda	rwda	r
administrators[].$ref	r	r	r
administrators[].display	rwd	rwd	r
administrators[].value	r	r	r
groups[]	rwda	rwda	r
groups[].$ref	r	r	r
groups[].display	rwd	rwd	r
groups[].value	r	r	r