Standards and Guidelines for RESTful APIs in TIER

Standards and Guidelines that apply to all TIER RESTful APIs

- Common invalid requests

Any resource/method that has these situations should use these common error codes

TIER result code	HTTP response code	Success?	Request description
ERROR_METHOD_NOT_AVAILABLE	405	false	Method that is not available for a resource
ERROR_PAGING_INVALID	400	false	If paging is not sent in correctly
ERROR_MULTIPLE_PARAMS	400	false	If multiple request parameters were sent when one was expected
ERROR_INVALID_REQUEST_BODY	400	false	If a body was sent in the request and not expected
ERROR_ID_EXPECTED	400	false	If an ID of a resource was expected and not sent in, e.g. if deleting a group but the group ID was not specified. Note, this request could also generate ERROR_METHOD_NOT_AVAILABLE. It is up to the implementer, this response code might be more helpful
ERROR_INVALID_PATH	404	false	If too many URL strings are passed in. e.g. if this GETs a group: /Groups/id:abc, then this would generate this response error: /Groups/id:abc/something. Similarly, if a path element has an invalid path (e.g. if the format of an ID in the path is not correct), this the same condition. If a misspelled or mistaken path element is specified (e.g. /Gruops) then this is the same condition. Note: if the path is valid but the resource is not found (e.g. cant find the group), then a more specific TIER result code should be used.
ERROR_INVALID_PARAM	400	false	If a parameter required a specific format or list of values and is invalid. Note, if a more specific ERROR_ result code is desired that could be used instead
ERROR_NOT_AUTHORIZED	403	false	If the resource exists but it not allowed to be read / updated / inserted / deleted by the authenticated user
ERROR_EXCEPTION	500	false	If an unexpected exception was thrown

Misc

Common parameters

Parameter name	Values	Description
indent	true|false	if the output should be formatted to be easy to read

Standards and guidelines that apply to all TIER APIs

Category	Description of guideline /standard	Constraints/exceptions	Supporting work /issues/resources
Relation to SCIM			
API operations	For all API operations that can be found in SCIM, follow the SCIM protocol as defined in RFC7644		1. See SCIM at: https://tools.ietf.org/html/rfc7643 and https://tools.ietf.org/html/rfc7644 2. See initial TIER group operations at: https://spaces.at.internet2.edu/display/DSAWG/Initial+Set+of+TIER+Group+Management+APIs 3. See also: https://bugs.internet2.edu/jira/browse/TIERAPI-1
Schema	Relationship of TIER schema with SCIM-defined schema, RFC7643	1. Each schema element used in TIER will be defined within a TIER schema. 2. Any element defined in SCIM that would be useful as is in TIER will have a TIER equivalent element with an identical definition. 3. Elements not defined in SCIM will be newly defined in the TIER schema specifications.	On making schema compliant and extensible, see: https://spaces.at.internet2.edu/display/DSAWG/Ignoring+Unrecognized+Schema+Fragments+in+a+Received+Resource+Representation
Resource types	Follow SCIM Protocol Section 6 when defining additional resource types		
Requests			
HTTP verbs	Customary definition of HTTP verbs (e.g., GET, PUT, POST, DELETE) will be followed.	In some cases it will be necessary to specify different nuances in different contexts.	See: https://www.ietf.org/rfc/rfc2616.txt
URI syntax	Follow REST conventions	1. The major version of the client is in the URL:https://groups.institution.edu/tierGroups/v1 2. ~~SCIM will be followed with respect to plurals~~	
Resource (URI) syntax	Resource reference syntax	1. Resource names should be camel case starting with a capital letter. 2. SCIM will be followed with respect to plurals 3. If you have a resource that has a sub-resource, then you need to specify that sub-resource. Do this: /Users/some:id/Roles/role:id not:/Users/some:id/role:id e.g. /myTierServer/Users/id:abc/Groups/groupName:edu:institution:whatever 4. The major version of the client is in the URL:https://groups.institution.edu/tierGroups/v1	

Object references	Objects can be referred to in various ways without having to do superfluous lookups. Objects should have a primary identifier. This identifier should not change. Therefore it should be opaque to allow for renames. Objects can be referred to by other unique identifiers as well. Unique identifiers that are not the primary identifier can change. Prefixes specify how the objects are referred to.	1. Prefixes must be alphanumeric. These prefixes therefore cannot contain colons or whitespace. 2. It is recommended that institutions use prefixes with part of the prefix related to the institution (e.g. the pennkey at penn could have a prefix "pennkey") 3. The TIER-defined prefixes are: a. id - this is the unchanging primary identifier of the object b. name - if applicable, this is the system name of the object (e.g. the name which doesn't change much and which might not be opaque, e.g. the group name) c. index - if applicable, this is the numeric index of the object, e.g. the posix ID of the group d. uniqueAttribute - this will lookup one object by any of the id or any unique attribute. if it finds multiple objects, it will return an error e. loginId - if applicable, this will lookup an object by a netId f. eppn - if applicable, this will lookup an object based on the scoped netId or whatever is used for eppn g. other prefixes will begin with "tier" h. to deal with potential edge cases, a prefix should not contain the delimiter (e.g, ":"), and everything after the actual delimiter should be considered the value Examples: URL of user by opaque id: https://people.institution.edu/entities/id:1234567 URL of user by netid: https://people.institution.edu/entities/netId:jmith URL of groups for a user by eppn: https://groups.institution.edu/entities/eppn:jsmith@institution.edu/groups URL to determine if Person p is in a Group g https://groups.institution.edu/groups/name:edu:institution:community:employees/members/uniqueAttribute:1234543	1. See recent discussion at bottom of: https://spaces.at.internet2.edu/display/DSAWG/TIER+API+SCIM+common+elements 2. Regarding, first example: URL of user by opaqueid: https://people.institution.edu/entities/id:1234567 Note to discuss whether Entity can cover both Users and Groups
Searches and queries	Filtering syntax for searches and queries follows SCIMsection 3.4.2.2		
Pagination	Follow SCIM section 3.4.2.4 on pagination		
Parameter passing and syntax	Parameters can be passed to resource requests in the body of the POST or as URL parameters (e.g. ?paramName=paramValue)	1. Parameter names defined in TIER APIs not found in SCIM must begin with "tier" followed by ".", followed by the desired parameter name. 2. Parameter names should be camel case starting with lower case. 3. Parameter values should be camel case starting with lower case. 4. Parameter booleans should be "true" or "false". 5. Parameter names and values are case sensitive. 6. If a resource allows a site to add a parameter name, the site name should begin with the camel case concatenation of the two-level domain name with a period at the end (e.g. "wiscEdu.localParameter"). 7. If a resource allows a site to add a parameter value, the value should begin with the camel case concatenation of the two-level domain name with a period at the end. (e.g. "wiscEdu.localValue").	
Responses			
Response format	JSON	1. In some cases we will define constraints on string values, e.g. for calendar dates, the JSON type will be string, but TIER will constrain the string values to conform to ISO 8601 'calendar date' 2. xml and other formats can be provided by the server but are not required	
Response content	Representations (such as users, groups)	1. Different clients (identified by authenticating credential) might see different responses a) This might be due to the privileges that the clients have on the data b) This might be due to a server configuration that returns more optional data elements	Example of representation of user in response: https://spaces.at.internet2.edu/display/DSAWG/TIER+API+SCIM+group+member
Metadata	Standardized elements of response metadata		

Response codes	HTTP response codes will not be overridden but enriched	HTTP response codes are not entirely unambiguous with respect to all resource operations so supplemental response information will be provided. TIER response codes should be capital letters (could have numbers) where words are separated by underscores. Successful response codes should start with SUCCESS (or be equal to SUCCESS. Unsuccessful response codes should start with ERROR	1. TIER discussion thread: [tier-api] HTTP response code 2. Tracked in: https://bugs.internet2.edu/jira/browse/TIERAPI-2 3. Examples of TIER response codes here: https://spaces.at.internet2.edu/display/DSAWG/TIER+API+SCIM+group+member 4. See recent discussion at bottom of: https://spaces.at.internet2.edu/display/DSAWG/TIER+API+SCIM+common+elements
TIER HTTP headers	1. X-TIER-success: required, boolean, true or false 2. X-TIER-resultCode: required, string, corresponds to the result code in the body. 3. (optional) X-TIER-requestId: unique URL for the request that can be used for troubleshooting, auditing, or logging 4. (optional) X-TIER-responseDurationMillis: integer, number of milliseconds that the server took to process the request.	Headers should be named: X-TIER-someName where the last string is lower-case camelcase Related to numbered items in preceding column: 1. true means the TIER API server successfully handled the request. Even if the response is false there might be a body to parse Generally this is implemented on the server with exception handling. If there is an exception and any database transactions are rolled back then the response is false. Note that if looking up a resource the HTTP status code might be 404 but X-TIER-success is still true. If a batched request is partially successful then this response code should be true if it can be returned. The body should give details about the failure. 2. There should only be a handful of valid response codes for a given request resource and HTTP method Note: If the request requires authentication and none is sent, a 401 HTTP status code will be returned and there might not be TIER headers since this response could be sent by the web server.	
Securing APIs			
	HTTPS		
	Techniques for securing API operations are currently external to the API.	Future versions of this guideline may be more prescriptive. (Reference auth types suggested are HTTP Basic and SSL certificates. If the request requires authentication and none is sent, a 401 HTTP status code will be returned. In such cases, there might be no TIER headers since this status code could have been sent by the web server.	Assumes API is passed the value of the authenticated identity.
API specification			
	Swagger specification and compliant tools.		1. See swagger spec: http://swagger.io/specification/ 2. Swagger tools: http://swagger.io/tools/

Meta

Each response should have a "meta" attribute with the following structure. Note, some error conditions will prevent this attribute from being sent back. See the SCIM definition

Field	Type	Required	Description

resourceType	String	true	The name of the resource type of the resource. This attribute has a mutability of "readOnly" and "caseExact" as "true". e.g. for groups it is "Group"
created	DateTime	false	The "DateTime" that the resource was added to the service provider
lastModified	DateTime	false	The most recent DateTime that the details of this resource were updated at the service provider. If this resource has never been modified since its initial creation, the value MUST be the same as the value of "created".
location	String URI	true	The URI of the resource being returned. This value MUST be the same as the "Content-Location" HTTP response header (see Section 3.1.4.2 of [RFC7231]).
version	String	false	The version of the resource being returned. This value must be the same as the entity-tag (ETag) HTTP response header (see Sections 2.1 and 2.3 of [RFC7232]). This attribute has "caseExact" as "true". Service provider support for this attribute is optional and subject to the service provider's support for versioning (see Section 3.14 of [RFC7644]). If a service provider provides "version" (entity-tag) for a representation and the generation of that entity-tag does not satisfy all of the characteristics of a strong validator (see Section 2.1 of [RFC7232]), then the origin server MUST mark the "version" (entity-tag) as weak by prefixing its opaque value with "W/" (case sensitive).
tierCanonicalLocation	String URI	false	If this response refers to another object, that URI is specified here. For example i you are requesting /Groups/id/Members/id, the response is a User or Group or System, and the canonical location would be the Group or User or System URI
tierSuccess	boolean	true	Same value as X-TIER-success
tierServiceRootUrl	String URI	true	The root URI for this service: https://groups.institution.edu/groupsApp/tierApiAuthz
tierServerVersion	String	true	Same version as URL though could have a build number on end after dot: e.g. v1.123
tierResultCode	String	true	Same value as X-TIER-resultCode, e.g. SUCCESS
tierRequestId	String	true	Same value as X-TIER-requestId
tierResponseDurationMillis	int	false	Same value as X-TIER-responseDurationMillis
tierErrorMessage	String	false	If this is an error this can hold the free form error message
tierHttpStatusCode	int	true	http status code of the response
tierWarning	String	false	free form warnings for example if superfluous params were sent
tierDebugMessage	String	false	could be sent to give debug info for this request

Archive: Numbered list of standards and guidelines for RESTful APIs in TIER

Archived Comments