INFRASTRUCTURE AND INTEGRATED TOOLS FOR PERSONALIZED LEARNING OF READING SKILL

D8.2 – iRead Core Infrastructure API

Document identifier	**iRead_D8.2_Core_Infrastructure_APIs_v2.0**
Date	**22/12/2017**
WP	**WP8**
Partners	**NTUA, ULBS**
WP Lead Partner	**NTUA**
Document status	**Final**

Grant Agreement number:

731724 — iRead H2020-ICT-2016-2017/H2020-ICT-2016-1

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No 731724

Deliverable Number	D8.2
Deliverable Title	iRead Core Infrastructure API
Deliverable version number	V1.0
Work package	WP8
Task	Task 8.1 iRead System Architecture
Nature of the deliverable	Report + Prototype (R+P)
Dissemination level	Public
Date of Version	2017-12-22

Author(s)	P. Georgantas
Contributor(s)	Eugen Cojocaru, George Tsatiris, Chrysanthi Raftopoulou, Kostas Karpouzis
Reviewer(s)	I.Mihu, C.Mihu, D. Elliott
Abstract	This deliverable describes the iRead core infrastructure API, ie the necessary calls to access the iRead data and services, the data required to invoke them and those returned, and examples of how to utilize the API to authenticate and authorize a user, and retrieve features and content to work with. All required code to build the API has been uploaded and maintained at https://git.image.ece.ntua.gr/gitlab/iread/infrastructure
Keywords	iRead API, user profile read and write, authorization, authentication

Document Status Sheet			
Issue	**Date**	**Comment**	**Author**
V0.1	2017-11-03	First Draft	Panagiotis Georgantas
V0.2	2017-11-10	Changes in Resources API	Eugen Cojocaru
V0.3	2017-11-20	Changes in Profiles API	George Tsatiris
V1.0	2017-11-22	Integration of comments	Panos Georgantas
V1.1	2017-11-30	Changes in Modes API	Chrisanthi Raftopoulou
V1.2	2017-12-08	Integration of comments	Panagiotis Georgantas
V2.0	2017-12-22	Additional text and flow charts	Panagiotis Georgantas Kostas Karpouzis

Table of content

1. Executive Summary 6
2. The iRead infrastructure 7
3. Implementation examples 8
3.1. User authentication and authorization 8
3.2. User creation 9
3.3. Retrieving next features and content 10
4. Authentication API 11
4.1. Get access token 11
4.2. Refresh access token 12
5. User Management API 14
5.1. Create/Update a user 14
5.2. Delete a user 16
5.3. Search Users 17
5.4. Register/Update a client application 19
5.5. Delete an existing Application 21
5.6. Get info about an Application 22
5.7. Create/Update a group 23
5.8. Delete a group 24
5.9. Get info about a group 25
5.10. Add users or groups to group 26
5.11. Remove users or groups from group 27
5.12. Get group members 28
5.13. Add permission to users/groups 29
5.14. Remove permissions from users/groups 31
6. Models API 32
6.1. Create/Update Domain Model 32
6.2. Delete Domain Model 35
6.3. Get Domain Models 36
6.4. Get Domain Model Features Information 38
6.5. Set DomainModels Feature Information 40
7. Profiles API 41
7.1. Create/Update User Profile for Domain Model 41

7.2. Delete Profile 43
7.3. Get User Profiles 44
7.4. Get User Profile Features 45
7.5. Set User Profile Features competence 47
7.6. Get Next Profile Features 48
7.7. Unlock User Profile Feature 50
8. Logger API 51
8.1. Log a user action 51
8.2. Search user action logs 53
8.3. Log application messages 56
8.4. Search application logs 57
8.5. Get last resources used by the user 59
9. Resources API 62
9.1. Get Words From a Dictionary 62
9.2. Get Sentences 65
9.3. Upload Text Resources 67
9.4. Get Text Resources 69
9.5. Upload Multimedia Resources 71
9.6. Get Multimedia Resources 73
9.7. Upload File 75
9.8. Download File 75
9.9. Delete File 76

1. Executive Summary

This document defines the APIs offered by iRead core Infrastructure to iRead applications. The APIs are implemented as REST services and are organized in five groups:

1. **User Management**: These APIs can be used to manage users within the iRead system. Users have a set of system defined optional attributes (e.g.name, age) and can also have any number of other searchable attributes. Furthermore, a user can have any number of application defined preferences. Users can be organized into groups and permissions can by granted to users or groups using the respective APIs. iRead access control system follows a restrictive policy where by default only the user who created an object has access to this object. Any additional required access must be explicitly granted using the API.

2. **Models**: Domain models can be managed in iRead using the APIs in this group. Models can be defined, updated, searched, enabled or disabled. APIs are provided for getting information or updating specific features in a domain model.

3. **User Profiles:** A profile is essentially an instantiation of a domain model for a user. A set of profile specific attributes and preferences can be defined along with the profile. Using the API it is possible to create, edit, delete or search the profiles of a specific user and update the user's progress by setting his competence level on one or more features. Additional APIs exist for getting the currently available features for this profile or explicitly unlock a feature.

4. **Logger:** iRead core Infrastructure offers a log facility. Using the logger API applications can log and search user actions (e.g. logins, progress, used resources, etc) or application messages (e.g. login failures, alerts, etc). A specific API is offered in order to search and fetch the last resources used by the user according to previously logged actions.

5. **Resources:** iRead offers dictionary, sentences, texts and multimedia files as resources. Using the Resources API it is possible to search through these resources and fetch the necessary information.

2. The iRead infrastructure

The iRead infrastructure (Figure 1) stores all information pertaining to the different functionalities of the iRead system (domain models, user profiles, usage information and logs, etc.) and provides secure and interoperable means to access and update them via API calls. This deliverable describes the API calls used by developers, the methods used to invoke each of them, and the data returned upon successful invocation (usually pending authorization approval).

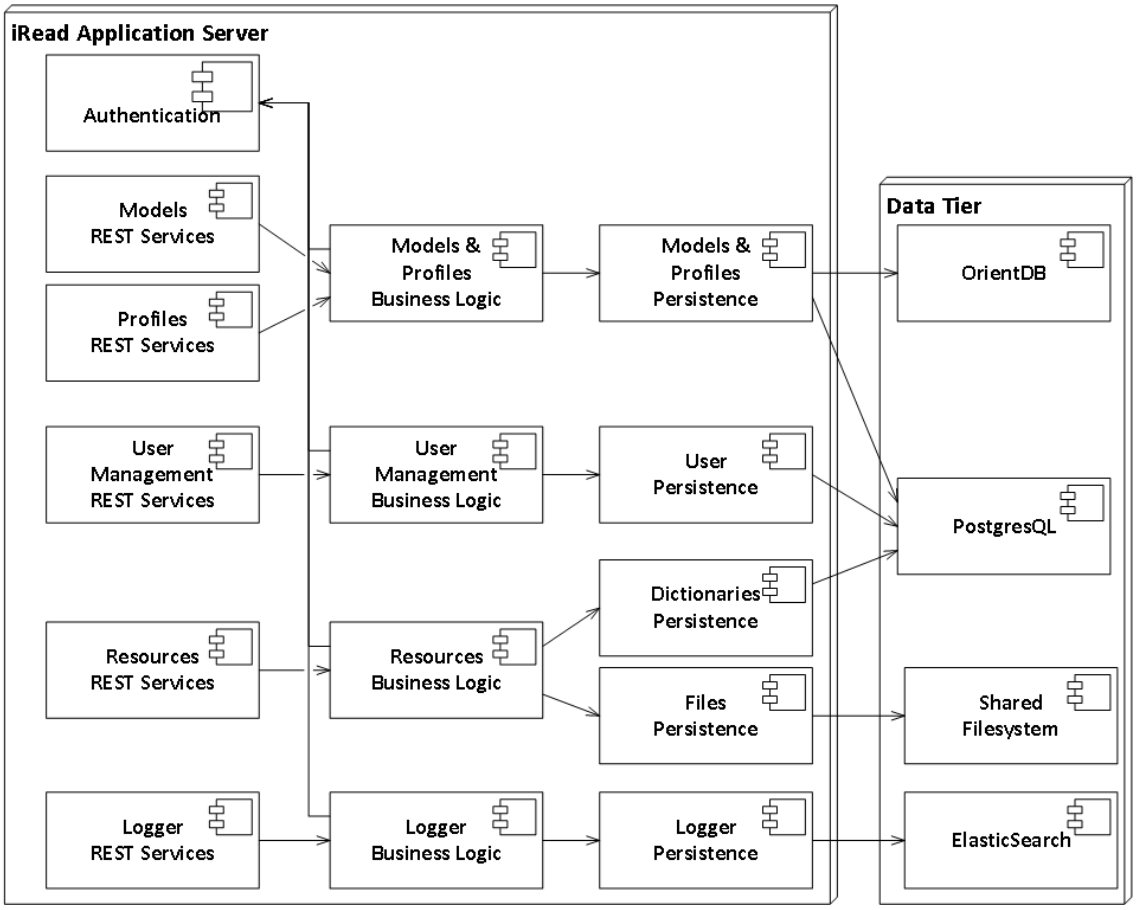

Figure 1: Core infrastructure components

In the following, we describe some common situations of iRead API utilization and the sequence of processes invoked by game and application developers. As shown in Figure 1 above, the iRead infrastructure is logically divided in the Application Server, a set of software components hosted in the cloud, which enables game and app developers to securely access and modify models, profiles, and services, and the Data Tier, a set of relational and graph databases which store static user information and user and domain models, respectively. Since the information which different applications may use is not standardized, we opted to use ElasticSearch and JSON notation to store access logs, effectively making it easier for developers to store and retrieve information irrespectively of any static database schemas. All required code to build the API has been uploaded and maintained at https://git.image.ece.ntua.gr/gitlab/iread/infrastructure

3. Implementation examples

This section presents some examples of how to use the iRead API to implement core functionalities needed across different applications. These relate to authentication and authorization, creating new iRead users, and requesting new content and feature(s) to work with for a given profile.

3.1. User authentication and authorization

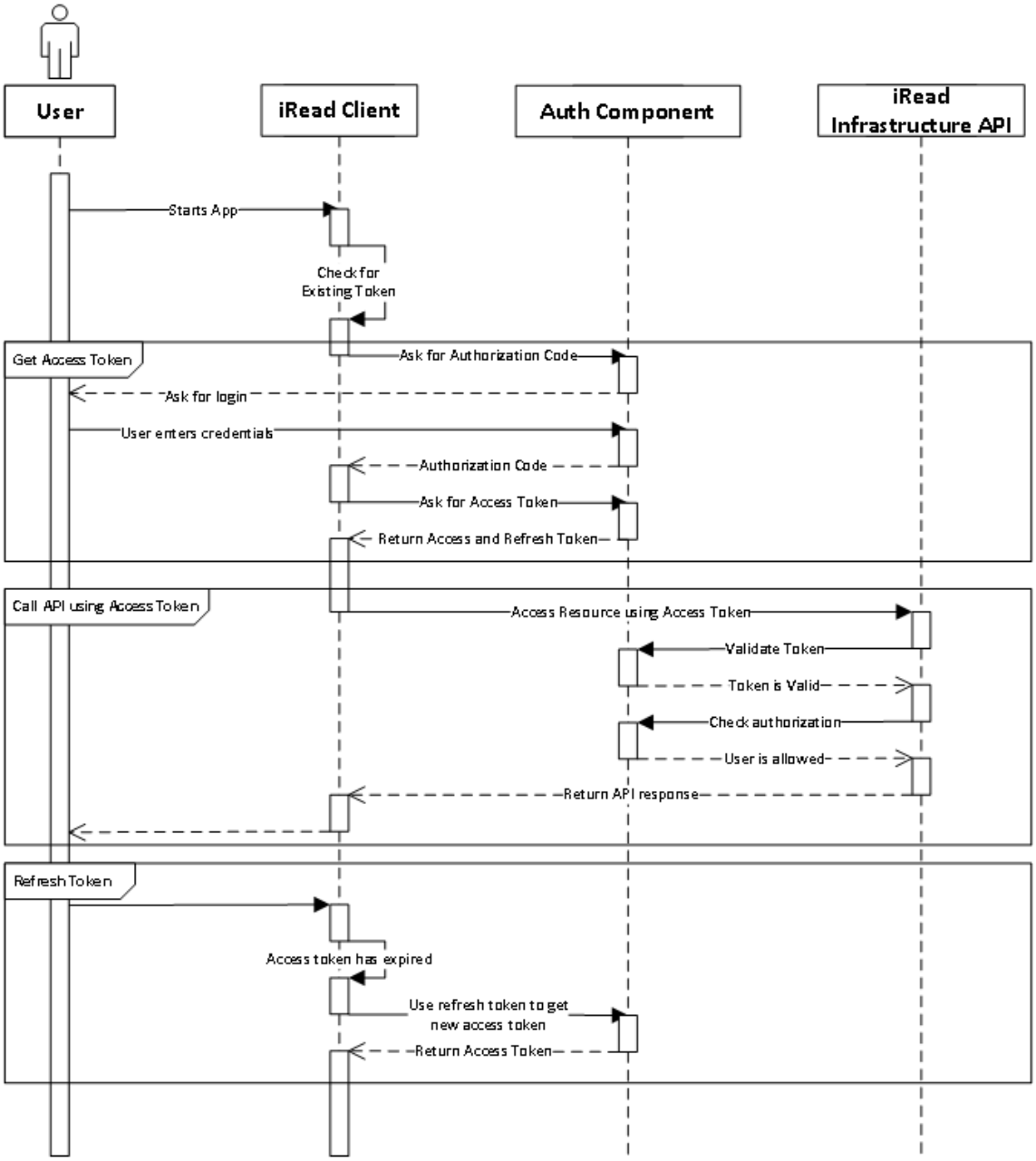

Figure 2: User authentication and authorization process flow

This figure describes the sequence of API calls required to authorize and authenticate a user. This process is essentially based on issuing an access token, following entry of user credentials; this token is then used to provide authorization for subsequent user (and application) requests and includes any user permission level associated with the particular user account (e.g. read/write access to a set or one particular profile)

3.2. User creation

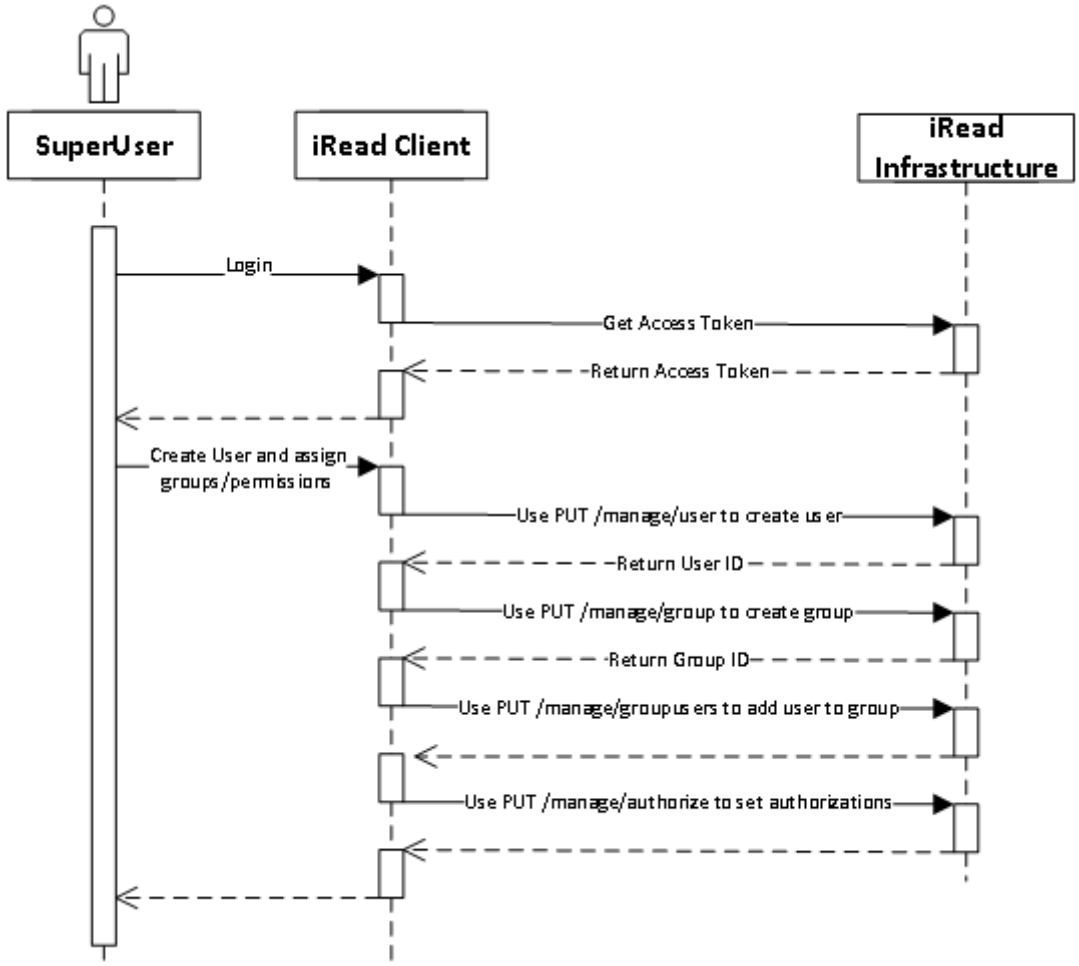

Figure 3: User creation process flow

This figure describes the sequence of API calls needed to create a new user (following authentication), and assign them to particular user groups (roles) and authorize them, usually for read or write access to profiles of particular users or user groups (e.g. students of a school where a teacher works).

3.3. Retrieving next features and content

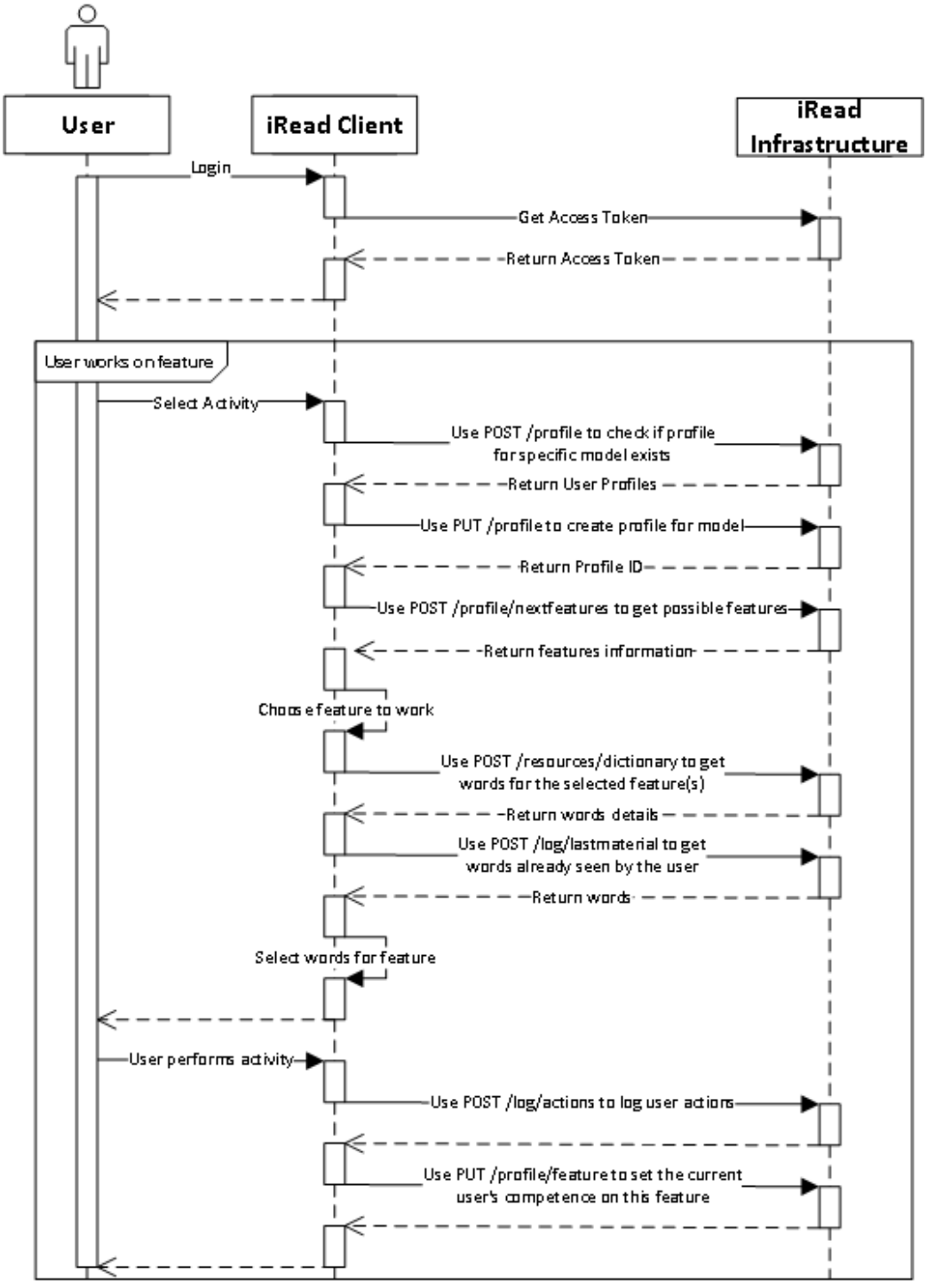

Figure 4: Retrieving next features and content process flow

This figure describes an API call sequence to retrieve the next profile feature to work with (for a given user profile) and then retrieve content for that material (filtered to remove content already accessed by the user).

4. Authentication API

4.1. Get access token

Description	Get access token	
URL	/auth/token	
Method	POST	
URL Params	-	
HTTP Headers	-	
Body Type	x-www-form-urlencoded	
Form Data	grant_type = password username = <username> password = <password> client_id = <applicaition-id> client_secret = <client-secret>	 The appid returned when creating an application The appsecret returned when creating an application
Success Response Code	200	
Success Response Data	{ “accessToken” : [string], “expires_in": [int], "refresh_expires_in" : [int], "refresh_token": [string], "token_type”: “Bearer” }	The token to use in the Authorization Header in all API calls Token expiration in seconds Refresh Token expiration in seconds Token to use when refreshing the accessToken

Error Response Code	400 or or 401 or 500
Error Response Data	-

4.2. Refresh access token

Description	Get access token	
URL	/auth/token	
Method	POST	
URL Params	-	
HTTP Headers	-	
Body Type	x-www-form-urlencoded	
Form Data	grant_type = refresh_token refresh_token= <refresh_token> client_id = <applicaition-id> client_secret = <client-secret>	The refresh token acquired in last access token request The appid returned when creating an application The appsecret returned when creating an application
Success Response Code	200	
Success Response Data	{ “accessToken” : [string], “expires_in": [int], "refresh_expires_in" : [int], "refresh_token": [string], "token_type”: “Bearer” }	The token to use in the Authorization Header in all API calls Token expiration in seconds Refresh Token expiration in seconds Token to use when refreshing the accessToken

Error Response Code	400 or or 401 or 500
Error Response Data	-

5. User Management API

5.1. Create/Update a user

Description	Create/Update a user	
URL	/manage/user	
Method	PUT	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "uid": [string] "username" : [string], "password": [string], "firstname" : [string], "lastname" : [string], "email": [string], "gender" :[string], "birthdate": [string], ..., "preferences": { [preference_name] : [preference_value] , } }	 User's iRead internal id. Required when updating the user. User's username. Required when creating a user. User's password in plain text(will be hashed in iRead). Required when creating a user Optional. Optional. Optional. Optional. Optional. User attributes. Arbitrary key-value pairs for use by the applications User preferences. Arbitrary key-value pairs for use by the applications Optional.

Success Response Code	200
Success Response Data	{ "uid" : [string] }
Error Response Code	403 or 500
Error Response Data	{ "error" : [string] }
Comments	This API is used to create or edit users in iRead. When called without a uid parameter it creates a new user. When a uid exists, it updates the existing user's information For the users there is set of predefined attributes and client applications can add their own. For the linguistic models the following info should be included for the user: "mother_language": [string], "school": [string], "classroom" : [string] There is also a facility for saving user preferences. Apart from the conceptual difference between user attributes and user preferences, user preferences are not searchable. Deleting a value is done by setting it to "" (empty string)

5.2. Delete a user

Description	Delete an existing user	
URL	/manage/user	
Method	DELETE	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "uid" : [string], }	The user's id. If missing, the logged in user
Success Response Code	200	
Success Response Data	{ "result" : "success" }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	

5.3. Search Users

Description	Search Users	
URL	/manage/user	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "_start" : [int], "_limit" : [int], "username" : [string], "uid" : [string] "key" : "value" }	Fetch results starting at this position. Optional. Default 0. Max results. Optional. Default 100 The user's username. Optional. The user's id. Optional. fetch users having attribute key=value. Optional.
Success Response Code	200	
Success Response Data	{ "_start" : [int], "_limit" : [int], "_size" : [int], [{ "password": [string], "firstname" : [string], "lastname" : [string],	Results starting at this position. Results count. Total resultset size.

	"email": [string], "gender" :[string], "birthdate": [string], ..., "preferences": { [preference_name] : [preference_value] , }, ...] }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	
Comments	This API is used to search users in iRead and get their info. Returned users are the users that the current user is authorized to view and match the search criteria. At least one key-value pair must exist, the key being username or any other user attribute.	

5.4. Register/Update a client application

Description	Register/Update a client application	
URL	/manage/app	
Method	PUT	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "appId": [string], "name": [string], "description": [string], "contact": [string], }	The application id. Required when updating the application Required when creating the application Required when creating the application Required when creating the application
Success Response Code	200	
Success Response Data	{ "appid": [string], "appsecret": [string] }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	
Comments	This API is used to define new applications in iRead. The returned appid is used to identify the application within iRead. The app secret is used as a password for the application and is returned only on creation.	

	Deleting a value is done by setting it to "" (empty string)

5.5. Delete an existing Application

Description	Delete an existing application	
URL	/manage/app	
Method	DELETE	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "appid": [string], }	The application id
Success Response Code	200	
Success Response Data	{ "result" : "success" }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	

5.6. Get info about an Application

Description	Get info about a client application	
URL	/manage/app	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "appId" : [string] }	The application id.
Success Response Code	200	
Success Response Data	{ "appid" : [string], "name" : [string], "description": [string], "contact": [string], }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	

5.7. Create/Update a group

Description	Create/Update a group	
URL	/manage/group	
Method	PUT	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "gid" : [string], "name" : [string], "description": [string] }	The group id. Required when updating the group
Success Response Code	200	
Success Response Data	{ "gid" : [string] }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	

5.8. Delete a group

Description	Delete an existing group	
URL	/manage/group	
Method	DELETE	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "gid" : [string], }	The group id.
Success Response Code	200	
Success Response Data	{ "result" : "success" }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	

5.9. Get info about a group

Description	Get info about a group	
URL	/manage/group	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "gid" : [string] }	The group id.
Success Response Code	200	
Success Response Data	{ "gid" : [string], "name" : [string], "description": [string] }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	

5.10. Add users or groups to group

Description	Add users or groups to group	
URL	/manage/groupusers	
Method	PUT	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "gid" : [string], "ids": [[string],...] }	The group id. An array of users or groups to add to the group
Success Response Code	200	
Success Response Data	{ "result" : "success" }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	

5.11. Remove users or groups from group

Description	Remove users or groups from group	
URL	/manage/groupusers	
Method	DELETE	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "gid" : [string], "ids": [[string],...] }	The group id. An array of users or groups to remove from the group
Success Response Code	200	
Success Response Data	{ "result" : "success" }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	

5.12. Get group members

Description	Add users or groups to group	
URL	/manage/groupusers	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "gid" : [string], }	The group id.
Success Response Code	200	
Success Response Data	{ "gid" : [string], "ids": [[string],...] }	An array of users or groups belonging to the group
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	

5.13. Add permission to users/groups

Description	Add permissions to users/groups	
URL	/manage/authorize	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "grantee_id" : [string], "permissions": [{ "object_id": [string], "permissions": [[string],...] }, ...] }	The group/user id to grant permissions to. The object id to which permissions are applied. An array of permissions to add for this object.
Success Response Code	200	
Success Response Data	{ "result" : "success" }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	

Comments	This API is used to grant permissions to a user or group. The grantee_id can be the username or gid of a group. The object_id (the object to which the permissions are applied) can be one of: - username. When granting permissions on a user, permissions can be one of: - FULL: all rights - READ: read user info - READ_CONTACT: read user contact info (name, email) - WRITE : change user info - CREATE_PROFILE: Can create profiles for this user - VIEW_ALL_LOGS: Can view all logs for this user - VIEW_ALL_PROFILES: View all user profiles. - group id. When granting permissions on a group, permissions can be one of: - FULL: all rights - READ: see group info and members - profile_id: - FULL: Read/Write all info on user's profile. - READ: Read all info from user's profile. - application_id: - FULL: all rights - VIEW_LOGS: View application logs. In all cases, the user that creates another user,group,profile or application gets automatically granted FULL permissions on this object

5.14. Remove permissions from users/groups

Description	Remove permissions from users/groups	
URL	/manage/authorize	
Method	DELETE	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "id": [string], "permissions": [{ "object_id": [string], "permissions": [[string],...] }, ...] }	The group/user id to remove permissions from. The object id for which permissions are assigned. An array of permissions to add for this object.
Success Response Code	200	
Success Response Data	{ "result" : "success" }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	

6. Models API

6.1. Create/Update Domain Model

Description	Create/Update Domain Model	
URL	/model	
Method	PUT	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "modelId" : [string], "enabled" : [boolean], "features" : [{ "id": [int], "unlockValue": [numeric], "minValue": [int], "maxValue": [int], "thresholdPercent": [numeric], ... }, ...], "edges": [{	 The model to update. Required when updating. Whether the model is available for creating profiles. Default: True. Model's features The feature's id. The required competence for unlocking the feature. The feature's competence min value. The feature's competence max value. The required threshold of incoming edges for unlocking the feature. Model's edges connecting features

	"sourceId": [int], "targetId": [int], "weight": [numeric], "unlockValue": [numeric] }, ...], "groups": [{ "name": [string] "items": [[id], ...] }, ...] }	Source feature Destination feature Edge Weight Source feature competence required to unlock this edge Groups of features group name feature ids
Success Response Code	200	
Success Response Data	{ "modelid" : [string], }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	
Comments	This API is used to create or update a domain model. The attributes mentioned above are the minimum set included in all all domain models. For the linguistic domain Models an example of the returned information is shown here: { "features" : [{ "id": 2,	


```
    "unlockValue": 0.75,
    "minValue": 0,
    "maxValue": 10,
    "thresholdPercent": 0.5,
    "linguisticLevel": "Phonology",
    "category": "GPC",
    "difficultyLevelIndex": 1,
    "progressionName": "Y1",
    "type": "Phoneme",
    "description": "/s/ as snake",
    "examples": "snake",
    "frequencyInChildText": "UNSET",
    "positionInWord": "ANY",
    "exception": "",
    "naturalCurriculumPlace": "Y1",
    "letterPhonemeMapping": "1-1",
    "relatedWordDifficulty": 1
  },
  .....
 ]
}
```


When updating a model, it is possible to either:
- provide the complete set of features/edges/groups must be specified.
- set it to enabled/disabled.

6.2. Delete Domain Model

Description	Delete Domain Model	
URL	/model	
Method	DELETE	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "modelId": [string] }	The model to delete
Success Response Code	200	
Success Response Data	{ "result" : "success" }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	

6.3. Get Domain Models

Description	Get Domain Models	
URL	/model	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "modelId": [string] }	The model to fetch. Optional.
Success Response Code	200	
Success Response Data	{ "modelId": [string], "enabled": [boolean] "features": [{ "id": [int], "unlockValue": [numeric], "minValue": [int], "maxValue": [int], "thresholdPercent": [numeric], ... }, ...], "edges": [{	

	"sourceId": [int], "targetId": [int], "weight": [numeric], "unlockValue": [numeric] }, ...], "groups": [{ "name": [string] "items": [[id], ...] }, ...] }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	
Comments		

6.4. Get Domain Model Features Information

Description	Get Domain Model specific features information	
URL	/model/feature	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "modelid" : [string], "features": [[int],...] }	The model Id The features to fetch. Optional
Success Response Code	200	
Success Response Data	{ "features" : [{ "id": [int], "unlockValue": [numeric], "minValue": [int], "maxValue": [int], "thresholdPercent": [numeric], ... }, ...] }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	

Comments	This API is used to fetch information about the features in a profile. The attributes mentioned above are the minimum set included in all all domain models. For the linguistic domain Models an example of the returned information is shown here: { "features" : [{ "id": 2, "unlockValue": 0.75, "minValue": 0, "maxValue": 10, "thresholdPercent": 0.5, "linguisticLevel": "Phonology", "category": "GPC", "difficultyLevelIndex": 1, "progressionName": "Y1", "type": "Phoneme", "description": "/s/ as snake", "examples": "snake", "frequencyInChildText": "UNSET", "positionInWord": "ANY", "exception": "", "naturalCurriculumPlace": "Y1", "letterPhonemeMapping": "1-1", "relatedWordDifficulty": 1 }, ...]}

6.5. Set DomainModels Feature Information

Description	Set Domain Model specific features information	
URL	/model/feature	
Method	PUT	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "modelid" : [string], "features": [{ "id": [int], "unlockValue": [numeric], "minValue": [int], "maxValue": [int], "thresholdPercent": [numeric], ... }, ...]}	The user's id. If missing returns info about the logged in user the profie Id The feature's competence to set
Success Response Code	200	
Success Response Data	{"result" : "success" }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	

7. Profiles API

7.1. Create/Update User Profile for Domain Model

Description	Create/Update User Profile For Domain Model	
URL	/profile	
Method	PUT	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "profileId": [string], "uid" : [string], "modelId": [string], ..., "preferences": { [preference_name] : [preference_value] , ... } }	The profile to update. Required when updating a profile. The user's id. If missing, the logged in user The source model for the profile. Required When creating a profile. User attributes. Arbitrary key-value pairs for use by the applications User preferences. Arbitrary key-value pairs for use by the applications Optional.
Success Response Code	200	
Success Response Data	{ "profileId" : [string], }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	

Comments	This API is used to create or update for a user a new profile based on a domain model. When called without a profileId parameter it creates a new user. When a uid exists it updates the existing user's information User attributes specific to the profile can be added, as well as profile specific preferences, in analogy with the global user attributes and preferences. Deleting a value is done by setting it to "" (empty string)

7.2. Delete Profile

Description	Delete User Profile	
URL	/profile	
Method	DELETE	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "profileId": [string] }	The profile to delete
Success Response Code	200	
Success Response Data	{ "result" : "success" }	
Error Response Code	403 or 500	

7.3. Get User Profiles

Description	Get User Profiles	
URL	/profile	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "uid" : [string], "profileId": [string] }	The user's id. If missing returns info about the logged in user The profile to fetch. Optional.
Success Response Code	200	
Success Response Data	[{ "profileId": [string], "modelId": [string], ..., "preferences": { [preference_name] : [preference_value] , } }, ...]	The source model for the profile. Required When creating a profile. User attributes. Arbitrary key-value pairs for use by the applications User preferences. Arbitrary key-value pairs for use by the applications
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	
Comments	This API is used to get the existing profiles of a user. The returned profiles are only the profiles that the current user/application is allowed to access. Any profile specific user attributes/preferences are returned	

7.4. Get User Profile Features

Description	Get User Profile features	
URL	/profile/feature	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "uid" : [string], "profileId" : [string], "features": [[int],...] }	The user's id. If missing returns info about the logged in user the profie Id The features to fetch. Optional
Success Response Code	200	
Success Response Data	{ "features" : [{ "id": [int], "competence": [int], "unlockValue": [numeric], "minValue": [int], "maxValue": [int], "thresholdPercent": [numeric], }, ...] }	 The feature's id. The user's current competence on this feature. The required competence for unlocking the feature. The feature's competence min value. The feature's competence max value. The required threshold of incoming edges for unlocking the feature.
Error Response Code	403 or 500	

Error Response Data	{ "error" : [string] }	
Comments	This API is used to fetch information about the features in a profile. The attributes mentioned above are the minimum set included in all profiles for all domain models. For the linguistic domain Models an example is shown here: { "features" : [{ "id": 2, "competence": 3 "unlockValue": 0.75, "minValue": 0, "maxValue": 10, "thresholdPercent": 0.5, "linguisticLevel": "Phonology", "category": "GPC", "difficultyLevelIndex": 1, "progressionName": "Y1", "type": "Phoneme", "description": "/s/ as snake", "examples": "snake", "frequencyInChildText": "UNSET", "positionInWord": "ANY", "exception": "", "naturalCurriculumPlace": "Y1", "letterPhonemeMapping": "1-1", "relatedWordDifficulty": 1 }, ...] }	

7.5. Set User Profile Features competence

Description	Set User Profile features competence	
URL	/profile/feature	
Method	PUT	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "uid" : [string], "profileId" : [string], "features": [{ "id": [int], "competence": [int] }, ...] }	The user's id. If missing returns info about the logged in user the profie Id The feature's competence to set
Success Response Code	200	
Success Response Data	{ "result" : "success", }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	

7.6. Get Next Profile Features

Description	Get Next Profile features	
URL	/profile/nextfeatures	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "uid" : [string], "profileId" : [string], "groups" : boolean }	The user's id. If missing returns info about the logged in user the profie Id Whether to include groups of features or not. Optional, default false
Success Response Code	200	
Success Response Data	{ "features" : [{ "id": [int], "competence": [int], "unlockValue": [numeric], "minValue": [int], "maxValue": [int], "thresholdPercent": [numeric], ... }, ...], groups: [	group of features

	{ "groupname" : [string], "features": [[int], ...], "group_availability" : [decimal], }, ...] }	 groupname array feature ids fraction of available features in groups
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	
Comments	This API is used to fetch the next features of a profile the user can work with, the features that are currently enabled according to the profiles' traversal algorithm and the competence the user has achieved on various features so far.	

7.7. Unlock User Profile Feature

Description	Unlock User Profile Feature	
URL	/profile/unlockfeatures	
Method	PUT	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "uid" : [string], "profileId" : [string], "features": [[int], ...] }	The user's id. If missing returns info about the logged in user the profie Id The features to unlock
Success Response Code	200	
Success Response Data	{ "result" : "success", }	
Error Response Code	403 or 500	
Error Response Data	{ "error" : [string] }	
Comments	This API is to make available one or more specific features in a profile that are not yet available as dictated by the user's progress. For example a teacher may specify that he wants his students to work on a specific feature that is not yet available to the students. By calling this API the selected features will be included in the subsequent calls to /profile/nextfeatures	

8. Logger API

8.1. Log a user action

Description	Log a user action	
URL	/log/actions	
Method	PUT	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	[{ "uid" : [string], "applicationid": [string], "time_start": [string], "time_end": [string], "tags": [[string],...], "features": [{ "model_id": [int], "feature_id" : [int], "resources" : [{ "id":[string], "type": [string], "result": [string], "content": [string] }, ...	 The user's id. If missing, the logged in user The application id. Time in ISO 8601:2004 format. Time in ISO 8601:2004 format. searchable tags for this action. features related to this action resources used for this feature. Optional resource id. One of content or id must exist resource type: WORD,SENTENCT,TEXT,MULTIMEDIA. Optional result for this resource. e.g. success, failure, good... Application dependent. for words, the actual word. One of content, id must exist

	] }, ...], "data": { ... } }, ...]	Extra non searchable application specific data
Success Response Code	200	
Sucess Response Data	{ "logid" : [string] }	
Error Response Code	403 or 500	
Error Response Data	{ error : [string] }	
Comments	This API is used to log actions and progress by a user. It includes an array of tags attribute which can contain the following values: - LOGIN - LOGOUT - COMPONENT_START - COMPONENT_END - PROGRESS - Any other application defined tags An action can contain information about a feature, in which case it is possible to include the resources used for working on this feature. A section data is included for logging application/domain specific data.	

8.2. Search user action logs

Description	Get user's last logged actions	
URL	/log/actions	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "_start" : [int], "_limit" : [int], "uid" : [string], "applicationid": [string], "time_start": [string], "time_end": [string], "tags": [[string],...], "features": [{ "model_id": [int], "feature_id" : [int], }, ...], "resources" : [{ "id":[string], "type": [string], "result": [string], "content": [string] },	 Fetch results starting at this position. Optional. Default 0. Max results. Optional. Default 100 The user's id. If missing, the logged in user Filter actions logged by this application id. Optional Filter actions logged after time in ISO 8601:2004 format. Optional. Filter actions logged before time in ISO 8601:2004 format. Optional/ Filter actions containing tags. Optional. (Predifined tags (LOGIN,LOGOUT) or not) Filter actions related to features. Optional Filter actions related to specific resources. Optional WORD,SENTENCE,TEXT,MULTIMEDIA

	....] }	
Success Response Code	200	
Sucess Response Data	{ "_start" : [int], "_limit" : [int], "_size" : [int], [{ "logid": [string]. "applicaitonid": [string], "time_start": [string], "time_end": [string], "tags": [[string],...], "features": [{ "model_id": [int], "feature_id" : [int], "resources" : [{ "id":[string], "type": [string], "result": [string], "content" : [string] }, ...]	Results starting at this position. Results count. Total resultset size.

	}, ...], "data": { ... } }, ...] }	
Error Response Code	403 or 500	
Error Response Data	{ error : [string] }	

8.3. Log application messages

Description	Log an applicaiton message	
URL	/log/application	
Method	PUT	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "applicationid": [string], "timestamp": [string], "tags": [[string],...], "data": { ... } }	The application id. Time in ISO 8601:2004 format. searchable tags for this action Extra non searchable application specific data
Success Response Code	200	
Sucess Response Data	{ "logid" : [string] }	
Error Response Code	403 or 500	
Error Response Data	{ error : [string] }	
Comments	This API is used by applications as a logging facility for any logs not related to a user's actions and progress on a profile, e.g. login failures, alerts, etc.	

8.4. Search application logs

Description	Get logged application messages	
URL	/log/application	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "_start" : [int], "_limit" : [int], "applicationid": [string], "timestamp": [string], "tags": [[string],...] }	Fetch results starting at this position. Optional. Default 0. Max results. Optional. Default 100 The application id. Time in ISO 8601:2004 format. Optional searchable tags for this log. Optional
Success Response Code	200	
Sucess Response Data	{ "_start" : [int], "_limit" : [int], "_size" : [int], [{ "logid" : [string], "applicationid": [string], "timestamp": [string], "tags": [[string],...], "data": { ... } }, ...	Results starting at this position. Results count. Total resultset size.

	] }	
Error Response Code	403 or 500	
Error Response Data	{ error : [string] }	

8.5. Get last resources used by the user

Description	Get user's last material	
URL	/log/lastMaterial	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "_start" : [int], "_limit" : [int], "uid" : [string], "applicationid": [string], "time_start": [string], "time_end": [string], "tags": [[string],...], "features": [{ "model_id": [int], "feature_id" : [int], }, ...], "resources_type" : [string], "resource_result": [string] }	 Fetch results starting at this position. Optional. Default 0. Max results. Optional. Default 100. The user's id. If missing, the logged in user Filter resources logged by this application id. Optional Filter resources logged after time in ISO 8601:2004 format. Optional. Filter resources logged before time in ISO 8601:2004 format. Optional. Filter resources containing tags. Optional. Filter resources related to features. Optional Filter resource on type: WORD,SENTENCE,TEXT,MULTIMEDIA. Optional Filter resource on result. Optional.
Success Response Code	200	
Sucess Response Data	{	

	"_start" : [int], "_limit" : [int], "_size" : [int], [{ "logid": [string]. "applicaitonid": [string], "time_start": [string], "time_end": [string], "tags": [[string],...], "resource_id" : [string], "resource_type": [string], "resource_result" [string], "resource_content": [string] "features": [{ "model_id": [int], "feature_id" : [int], }, ...] }, ...] }	Results starting at this position. Results count. Total resultset size.
Error Response Code	403 or 500	
Error Response Data	{ error : [string] }	
Comments	This API is used to fetch the last resources used by a user in his previously logged actions. The same information can be retrieved be the /log/action search api, but this api it is focused in the resources used rather than the actions. Resources	

	can be filtered by the attributes of the logged information.

9. Resources API

9.1. Get Words From a Dictionary

Description	Get words from dictionary	
URL	/resources/dictionary	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "domain_model_id: [int], "child_dictionary: [boolean]" "max_results" : [int], "content" : [string], "number_of_characters": [string]. "phonetic": [string], "grapheme_phoneme" : [[string],...], "number_of_phonemes" : [string] "syllables" : [[string],...], "number_of_syllabes" : [int], "part_of_speech" : [string], "number_of_morphemes" : [int], "cv_form" : [string], "prefix" : [string], "prefix_type" : [string],	domain_model_id is required. All other parameters are optional but at least one must exist. The dictionary's Domain Model Whether the word is contained in the model child dictionary. Default: false Maximum results to return. Default 100 The word to return info for. A range, e.g. "5" or "1-3" substring of phonetic as defined in the dictionary Subset of the array of phonemes as defined in the dictionary, e.g ["p-p","n-n"] A range, e.g. "5" or "1-3" Subset of the array of syllables as defined in the dictionary, e.g ["gram","mar"] A range, e.g. "5" or "1-3" E.g. "noun","verb",.... A range, e.g. "5" or "1-3" substring of cv_form as defined in the dictionary Exact match of prefix as defined in the dictionary Exact match of prefix_type as defined in the dictionary

	"suffix": [string], "suffix_type" : [string], "has_picture" : [boolean], "related_word_difficulty" : [int] "feature_ids" : [[int], ...], }	Exact match of suffix as defined in the dictionary Exact match of suffix_type as defined in the dictionary whether the word has an associated picture A range, e.g. "5" or "1-3" Ids of the features, all must be present in the resource
Success Response Code	200	
Success Response Data	[{ "resource_id" : [string], "child_dictionary: [boolean]" "content" : [string], "number_of_characters": [string]. "phonetic": [string], "grapheme_phoneme" : [[string],...], "number_of_phonemes" : [string] "syllables" : [[string],...], "number_of_syllabes" : [int], "part_of_speech" : [string], "number_of_morphemes" : [int], "cv_form" : [string], "prefix" : [string], "prefix_type" : [string], "suffix": [string], "suffix_type" : [string], "picture_url" : [string], "related_word_difficulty" : [int] "feature_info" : [	

	{ "featureId": [int], "matched" : [{"start": [int], "end" : [int] }, ...] }, ...], ... }, ...]
Error Response Code	403 or 500
Error Response Data	{ error : [string] }

9.2. Get Sentences

Description	Get sentences	
URL	/resources/sentence	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "domain_model_id: [int], "max_results" : [int], "content" : [string], "feature_ids" : [[int], ...], }	domain_model_id is required. All other parameters are optional but at least one must exist. The dictionary's Domain Model Maximum results to return Substring match of the sentence to return info for. Ids of the features, all must be present in the resource
Success Response Code	200	
Success Response Data	[{ "resource_id" : [string], "content" : [string], "syntax_tree": { ... } . "feature_info" : [{ "featureId": [int], "matched" : [{"start": [int], "end" : [int] }, ...	

	] }, ...], ... }, ...]
Error Response Code	403 or 500
Error Response Data	{ error : [string] }

9.3. Upload Text Resources

Description	Upload text resources (Books etc)	
URL	/resources/texts	
Method	PUT	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "domain_model_id: [int], "name" : [string], “fileId” : [string], “content”:[string], “content_url”: [string], “description”: [string], "feature_info" : [{ "featureId": [int], "matched" : [{"start": [int], "end" : [int] }, ...] }, ...] }	 The resource’s Domain Model The text name. The id of an uploaded to iRead file with the resource’s content. Optional. The content of the resource. Optional. The url to the resource’s content. Optional. A description of this resource. Ids of the features, all must be present in the resource
Success Response Code	200	
Success Response Data	{ "resource_id" : [string] }	
Error Response Code	403 or 500	

Error Response Data	{ error : [string] }
Comments	A text resource can be defined in iRead using this API. A text resource can be either - a file uploaded in iRead using the respective API (see below). - a small text. - a url to an external file. In each of the above cases one the respective parameter must be set: - fileID: the id as returned by iRead's API - content: The actual text - content_url: The url containing the file.

9.4. Get Text Resources

Description	Get large texts	
URL	/resources/texts	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "domain_model_id: [int], "max_results" : [int], "name" : [string], “id”: [string], “fileId” : [string], “content_url”: [string], "feature_ids" : [[int], ...], }	domain_model_id is required. All other parameters are optional but at least one must exist. The texts’s Domain Model Maximum results to return The text name. The resource id. A file id. A url. Ids of the features, all must be present in the resource
Success Response Code	200	
Success Response Data	[{ "resource_id" : [string], "name" : [string], “fileId” : [string], “content”:[string], “content_url”: [string], “description”: [string], "feature_info" : [{	The text resource's content will be returned as a text or as an iRead fileId or as a url to the text file (pdf,epub). One of fileId,content, content_url will exist in the results.

	"featureId": [int], "matched" : [{"start": [int], "end" : [int] }, ...] }, ...], ... }, ...]	
Error Response Code	403 or 500	
Error Response Data	{ error : [string] }	

9.5. Upload Multimedia Resources

Description	Upload mutlimedia resources (Videos/Audio etc.)	
URL	/resources/mutlimedia	
Method	PUT	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "domain_model_id: [int], "name" : [string], “fileId” : [string], “content_url”: [string], "content-type" : [string], “description”: [string], "feature_info" : [{ "featureId": [int], "matched" : [{"start": [int], "end" : [int] }, ...] }, ...] }	 The resource’s Domain Model The resource’s name. The id of an uploaded to iRead file with the resource’s content. Optional. The url to the resource’s content. Optional. The resource type (video/audio) A description of this resource. Ids of the features, all must be present in the resource
Success Response Code	200	
Success Response Data	{ "resource_id" : [string] }	
Error Response Code	403 or 500	

Error Response Data	{ error : [string] }
Comments	A mutlimedia resource can be defined in iRead using this API. A text resource can be either - a file uploaded in iRead using the respective API (see below). - a url to an external file. In each of the above cases one the respective parameter must be set: - fileID: the id as returned by iRead's API - content_url: The url containing the file.

9.6. Get Multimedia Resources

Description	Get multimedia	
URL	/resources/multimedia	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Data	{ "domain_model_id: [int], "max_results" : [int], "name" : [string], “fileId” : [string], “content_url”: [string], "content-type" : [string], "feature_ids" : [[int], ...], }	domain_model_id is required. All other parameters are optional but at least one must exist. The file's Domain Model Maximum results to return The file's name as defined in iRead infrastructure. Ids of the features, all must be present in the resource
Success Response Code	200	
Success Response Data	[{ "resource_id" : [string], "content_url": [string], “fileId” : [string], "content-type" : [string], "feature_info" : [{ "featureId": [int], "matched" : [	The multimedia resource's content will be returned as a text or as an iRead fileId or as a url to the file. One of fileId, content_url will exist in the results.

	{"start": [int], "end" : [int] }, ...] }, ...], ... }, ...]	
Error Response Code	403 or 500	
Error Response Data	{ error : [string] }	

9.7. Upload File

Description	Upload a file for a text or multimedia resource	
URL	/resources/files	
Method	POST	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN	
Body Type	multipart/form-data	
Form Data	Attachment =<file>	The file contents to upload
Success Response Code	200	
Success Response Data	{ "file_id" : [string] }	The id of the uploaded file
Error Response Code	401 or 403 or 500	
Error Response Data	{ error : [string] }	

9.8. Download File

Description	Download a file for a text or multimedia resource	
URL	/resources/files	
Method	GET	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN FileId: [string]	
Form Data	-	
Success Response Code	200	
Success Response Data		The file contents
Error Response Code	401 or 403 or 500	
Error Response Data		

9.9. Delete File

Description	Delete a file	
URL	/resources/files	
Method	DELETE	
URL Params	-	
HTTP Headers	Authorization: Bearer ACCESS_TOKEN FileId: [string]	
Form Data	-	
Success Response Code	200	
Success Response Data		
Error Response Code	401 or 403 or 500	
Error Response Data		