
# PeopleKeys 3 API (rev 20160104)

## Contents

**PeopleKeys 3 API (rev 20160104)** ..... 1
- Log of Changes ..... 2
- Accounts ..... 3
- Subaccounts ..... 6
- Users ..... 7
- Invites ..... 10
- Assessments ..... 11
- Client-Side Assessments ..... 24
- Benchmarking ..... 35
- Error Handling ..... 43

## Log of Changes

Benchmarking

- GET /app/api/benchmark/list. Updated (new assessment types).
- GET /app/api/benchmark/calculated/results/:config_id. Updated (new assessment types).
- GET /app/api/benchmark/apply/:benchmark_id/:result_id. Updated (new assessment types).
- GET /app/api/ benchmark/apply/client_side/:benchmark_id/:result_id. Updated (new assessment types).

## Accounts

**Method**: GET
**Description**: Returns information about the API caller account.
**Route**: /app/api/account/
**Response:** JSON array

```
{
  account_id,
  name,
  desc,
  type,
  owner_id,
  created_date,
  last_updated,
  region
}
```

*account_id* – account ID.
*name* – account name.
*desc* – account description.
*type* – account type [public | private].
*owner_id* – ID of the account owner.
*created_date* – [yyyy-mm-dd hh:mm:ss] account creation timestamp.
*last_updated* – [yyyy-mm-dd hh:mm:ss] account data last update timestamp.
*region* – account default geographic region.
**Method**: PUT
**Description**: Updates information on the API caller account.
**Route**: /app/api/account/
**HTTP Form Parameters:**
*account_id* – API caller account ID.
*name* – account name.
*desc* – account description.
*type* – account type [public | private].
*region* – account default geographic region.

**Method**: POST
**Description**: Creates a metadata entry for the API caller account or one of its subaccounts.
**Route**: /app/api/account/meta/
**HTTP Form Parameters:**
*account_id* – account ID.
*key* – the meta-entry name.
*value* – the meta-entry value.

**Method**: GET
**Description**: Retrieves information about existing account tags.
**Route**: /app/api/account/:account_id/tags
**HTTP URL Parameters:**
*account_id* – ID of the account.
**Response:** JSON array

```
[
  {
    tag,
    count
  },
  …
]
```

*tag* – tag name.
*count* – number of PK3 users and/or external users associated with the tag.

**Method**: POST
**Description**: Creates new tags to use in the account.
**Route**: /app/api/account/:account_id/tags
**HTTP URL Parameters:**
*account_id* – ID of the account.
**HTTP Form Parameters:**
*tags* – JSON array

```
{
  tag_list = […]
}
```

*tag_list* – list of tags to be created in the account.
**Response:** JSON array

```
{
  tags_affected
}
```

*tags_affected* – number of listed tags created in the account.

**Method**: PUT
**Description**: Updates existing account tags.
**Route**: /app/api/account/:account_id/tags
**HTTP URL Parameters:**
*account_id* – ID of the account.
**HTTP Form Parameters:**
*tags* – JSON array

```
{
  tag_list = [
    {
      old_name,
      new_name
    },
    …
  ]
}
```

*tag_list* – list of tags to be updated (renamed).
*old_name* – old tag name.
*new_name* – new tag name.
**Response:** JSON array

```
{
  tags_affected
}
```

*tags_affected* – number of listed tags renamed.

**Method**: DELETE
**Description**: Deletes account tags.
**Route**: /app/api/account/:account_id/tags?tags=:tags
**HTTP URL Parameters:**
*account_id* – ID of the account.
**HTTP GET Parameters:**
*tags* – JSON array

```
{
  tag_list = […]
}
```

*tag_list* – list of tags to be deleted from the account.
**Response:** JSON array

```
{
  tags_affected
}
```

*tags_affected* – number of listed tags deleted from the account.

## Subaccounts

**Method**: GET
**Description**: Returns information about subaccounts of the API caller account.
**Route**: /app/api/subaccounts/
**Response:** JSON array

```
[
  {
    account_id,
    name,
    desc,
    type,
    owner_id,
    created_date,
    last_updated,
    region
  },
  …
]
```

*account_id* – account ID.
*name* – account name.
*desc* – account description.
*type* – account type [public | private].
*owner_id* – ID of the account owner.
*created_date* – [yyyy-mm-dd hh:mm:ss] account creation timestamp.
*last_updated* – [yyyy-mm-dd hh:mm:ss] account data last update timestamp.
*region* – account default geographic region.

**Method**: POST
**Description**: Creates a new subaccount for the API caller account or one of its subaccounts.
**Route**: /app/api/subaccounts/
**HTTP Form Parameters:**
*account_id* – [optional] parent account ID. Default value: API caller account ID.
*name* – account name.
*desc* – account description.
*type* – account type [public | private].
*region* – account default geographic region.

**Method**: PUT
**Description**: Updates information on the subaccount of the API caller account.
**Route**: /app/api/subaccounts/
**HTTP Form Parameters:**
*account_id* – ID of the account to update.
*name* – account name.
*desc* – account description.
*type* – account type [public | private].
*region* – account default geographic region.

**Method**: DELETE
**Description**: Deletes the subaccount of the API caller account.
**Route**: /app/api/subaccounts/:account_id/
**HTTP URL Parameters:**
*account_id* – ID of the account to delete.

## Users

**Method**: GET
**Description**: Returns information about users in the API caller account or one of its subaccounts.
**Route**: /app/api/account/:account_id/users/
**Response:** JSON array

```
[
  {
    user_id,
    name,
    email,
    role,
    category,
    stub,
    created_date,
    last_visited,
    region
  },
  …
]
```

*user_id* – user ID.
*name* – user name (nick).
*email* – user email (login).
*role* – user role in the account [owner | manager | member].
*category* – account membership category.
*stub* – custom ‘stub’ part of the user invite URL.
*created_date* – [yyyy-mm-dd hh:mm:ss] user creation timestamp.
*last_visited* – [yyyy-mm-dd hh:mm:ss] last time the user visited the PeopleKeys application.
*region* – user geographic region.

**Method**: POST
**Description**: Creates new user and adds it to the API caller account or one of its subaccounts.
**Route**: /app/api/account/users/
**HTTP Form Parameters:**
*account_id* – ID of the account to add the new user to.
*role* – user role in the account [owner | manager | member].
*category* – [optional] account membership category.
*name* – new user name (nick).
*email* – new user email (login).
*password* – new user password as plain-text string.

**Method**: PUT
**Description**: Updates information about existing membership of the user in the API caller account or one of its subaccounts.
**Route**: /app/api/account/users/
**HTTP Form Parameters:**
*account_id* – ID of the account.
*user_id* – ID of the user.
*role* – user role in the account [owner | manager | member].
*category* – account membership category.

**Method**: DELETE
**Description**: Removes the user from the API caller account or one of its subaccounts.
**Route**: /app/api/account/:account_id/users/:user_id
**HTTP URL Parameters:**
*account_id* – ID of the account to delete.
*user_id* – ID of the user.

**Method**: GET
**Description**: Retrieves list of tags associated with the existing user within the context of given account.
**Route**: /app/api/user/tags/:account_id/:user_id
**HTTP URL Parameters:**
*account_id* – ID of the account. Tags are looked up within the context of this account.
*user_id* – ID of the user.
**Response:** JSON array

```
{
  tag_list = […]
}
```

*tag_list* – list of tags associated with the user record within given account.

**Method**: POST
**Description**: Associates tags with the existing user. New account tags are created if needed.
**Route**: /app/api/user/tags/:account_id/:user_id
**HTTP URL Parameters:**
*account_id* – ID of the account. Tags are looked up within the context of this account.
*user_id* – ID of the user.
**HTTP Form Parameters:**
*tags* – JSON array

```
{
  tag_list = […]
}
```

*tag_lists* – list of tags to be associated with the user record.
**Response:** JSON array

```
{
  tags_affected
}
```

*tags_affected* – number of listed tags associated with the user.

**Method**: DELETE
**Description**: Removes association of given tags with the existing user.
**Route**: /app/api/user/tags/:account_id/:user_id?tags=:tags
**HTTP URL Parameters:**
*account_id* – ID of the account. Tags are looked up within the context of this account.
*user_id* – ID of the user.
**HTTP GET Parameters:**
*tags* – JSON array

```
{
  tag_list = […]
}
```

*tag_list* – list of tags to be removed from association with the user record.
**Response:** JSON array

```
{
  tags_affected
}
```

*tags_affected* – number of listed tags removed from association with the user.

## Invites

**Method**: POST

**Description**: Generates an invite URL on behalf of API caller account or one of its subaccounts. The customer that wants to invite new user to take specific assessment test on PK3 site, sends the request and passes the parameters as specified in the doc (the user identity, the assessment ID and so on). The PK3 Web service executes the request, generates the invite URL and returns it to the caller as the 'link' parameter of the response. Next, the user can copy the URL in the address field of his/her Internet browser and press 'Enter'. Alternatively, a client Web application can automatically open the invite URL in the new browser window or in the iFrame of the existing browser window. If the 'participant' flag is set, the user will be automatically logged into PK site and redirected to the beginning of the assessment test. If the same invite URL for the 'participant' user is used more than once, the user is redirected to the page of the assessment test where the test was dropped (or to the final page of the assessment, if the test was completed).

**Route**: /app/api/account/invite/

**HTTP Form Parameters:**

*account_id* – ID of the account.

*email* – user email (login).

*expire date* – [optional] the date invite URL expires on. Invite URL never expires, if not specified.

*category* – [optional] account membership category.

*role* – [optional] user role in the account [owner | manager | member]. User is assigned a 'member' role, if not specified.

*name* – new user name (nick).

*allow_view* – [optional] the flag indicates the user can see assessment results [0|1]. Flag is not set by default.

*exit_url* – [optional] custom URL the user is redirected to after completion of the assessment.

*allow_email* – [optional] the flag indicates the system sends email notification to account managers on completion of the assessment [0|1]. Flag is not set by default.

*participant* – [optional] the flag indicates the new user is automatically signed on into the system when it enters the site via this invite URL [0|1]. Flag is not set by default.

*assessment_id* – ID of the assessment the new user is granted rights to pass.

*tags* – [optional] JSON array

```
{
  tag_list = […]
}
```

*tag_lists* – list of tags to be associated with the user record. If the user already exists, the list of tags will be merged with the list of existing tags for this user. New account tags are created if needed.

**Response:** JSON array

```
{
  link
}
```

*link* – invite URL.

## Assessments

**Method**: GET
**Description**: Returns list of assessments available for the API caller account or one of its subaccounts.
**Route**: /app/api/assessments/:account_id
**HTTP URL Parameters:**
*account_id* – ID of the account.
**Response:** JSON array

```
[
  {
    assessment_id,
    owner_id,
    name,
    quantity,
    infinite
  },
  …
]
```

*assessment_id* – assessment ID.
*owner_id* – user ID of the record owner.
*name* – assessment name.
*quantity* – number of available assessments.
*infinite* – the flag indicates the account has infinite number of available assessments [0|1].

**Method**: PUT
**Description**: Transfer the rights to pass assessments between the API caller account and its subaccounts.
**Route**: /app/api/assessment/transfer
**HTTP Form Parameters:**
*assessment_id* – assessment ID.
*sender_account_id* – ID of the account to transfer the assessment rights from.
*recipient_account_id* – ID of the account to transfer the assessment rights to.
*quantity* – number of available assessment passes to transfer.

**Method**: GET
**Description**: Returns detailed information about given assessment.
**Route**: /app/api/assessment/:assessment_id/
**HTTP URL Parameters:**
*assessment_id* – assessment ID.
**Response:** JSON array

```
{
  assessment_id,
  name,
  description,
  environment
}
```

*assessment_id* – assessment ID.
*name* – assessment name.
*description* – assessment description.
*environment* – assessment 'environment' tag.

**[The method is DEPRECATED. Current version of API rejects requests to delete the assessment]**
**Method**: DELETE
**Description**: Deletes the assessment.
**Route**: /app/api/assessment/:assessment_id/
**HTTP URL Parameters:**
*assessment_id* – ID of the assessment to delete.

**Method**: GET
**Description**: Returns a list of core assessment IDs for given assessment.
**Route**: /app/api/assessment/core/:assessment_id/
**HTTP URL Parameters:**
*assessment_id* – assessment ID.
**Response:** JSON array

```
{
  core_assessments
}
```

*core_assessments* – JSON array containing list of core assessment IDs.

**Method**: GET
**Description**: Returns information about pages for given assessment.
**Route**: /app/api/assessment/pages/:assessment_id/
**HTTP URL Parameters:**
*assessment_id* – assessment ID.
**Response:** JSON array

```
[
  {
    page_id,
    sort_order,
    type,
    content = {
      Type,
      Header,
      Sub-Header,
      Instructions
    }
  },
  …
]
```

*page_id* – assessment page ID.
*sort_order* – page order in the assessment.
*type* – page UI type.
*content* – page content:
- *Type* – page type.
- *Header* – page header.
- *Sub*-Header – page sub-header.
- *Instructions* – user instructions for assessment passage.

**Method**: GET
**Description**: Returns information about assessment page items.
**Route**: /app/api/assessment/items/:page_id/
**HTTP URL Parameters:**
*page_id* – assessment page ID.
**Response:** JSON array

```
[
  {
    item_id,
    sort_order,
    content = {
      Type,
      Question,
      Options = [
        { Value, Text },…
      ]
    }
  },
  …
]
```

*item_id* – page item ID.
*sort_order* – page item order on the page.
*content* – page item content:
- *Type* – page item type.
- *Question* – the question to answer.
- *Value* – answer value code.
- *Text* – answer text.

**[The method is DEPRECATED. Use new version of this method instead]**

**Method**: GET

**Description**: Returns information about assessment results available for the API caller account or one of its subaccounts. Result status records are sorted by assessment start timestamp, the latest record is first.

**Route**: /app/api/assessment/results/:account_id[?filterBy=:filterBy]

**HTTP URL Parameters:**

*account_id* – account ID.

**HTTP GET Parameters:**

*filterBy* – [optional] user email (login). Filters assessment results, returns the results available for user identified by the email (login).

**Response:** JSON array

```
[
  {
    id,
    user_id,
    assessment_id
    date_completed,
    date_started,
    status
  },
  …
]
```

*id* – assessment result ID.

*user_id* – ID of the user that took the assessment.

*assessment_id* – assessment ID.

*date_completed* – [mm/dd/yy hh:mm[AM|PM]] assessment completion timestamp.

*date_started* – [mm/dd/yy hh:mm[AM|PM]] assessment start timestamp.

*status* – assessment completion status [Completed|In Progress].

**Method**: GET

**Description**: Returns information about assessment results available for the API caller account or one of its subaccounts. Result status records are sorted by assessment start timestamp, the latest record is first.

**Route**: /app/api/assessment/results/:account_id?extended[&email=:email][&max_records=:max_records][&start_record=:start_record][&date_since=:date_since][&tags=:tags]

**HTTP URL Parameters:**

*account_id* – account ID.

**HTTP GET Parameters:**

*email* – [optional] user email (login). Filters assessment results, returns the results available for user identified by the email (login).

*max_records* – [optional] the maximum number of result status records returned per request. By default, the maximum number of records is set to 10000. You cannot return more than 10000 result status records per request.

*start_record* – [optional] index of the first record. By default, index of the first record is 0.

*date_since* – [optional] [yyyy-mm-dd] filters out result records older than specified date.

*tags* – [optional] JSON array

```
{
  tag_list = […]
}
```

*tag_lists* – list of tags to apply as a filter. Filters assessment results, returns the results for users associated with at least one of the listed tags.

**Response:** JSON array

```
{
  total_record_number,
  results = [
    {
      id,
      user_id,
      assessment_id
      date_completed,
      date_started,
      status
    },
    …
  ]
}
```

*total_record_number* – total number of result status records found that match the filters.

*results* – JSON array of result status records:

- *id* – assessment result ID.
- *user_id* – ID of the user that took the assessment.
- *assessment_id* – assessment ID.
- *date_completed* – [yyyy-mm-dd hh:mm:ss] assessment completion timestamp.
- *date_started* – [yyyy-mm-dd hh:mm:ss] assessment start timestamp.
- *status* – assessment completion status [Completed|In Progress].

**[The method is DEPRECATED. Use GET /app/api/assessment/results/calculate/:result_id instead]**

**Method**: POST

**Description**: Obtains calculated raw scores for given assessment result.

**Route**: /app/api/assessment/calculate/

**HTTP Form Parameters:**

*result_id* – assessment result ID.

**Response:** JSON array

```
{
  assessment_id,
  type,
  graph1_D,
  graph1_I,
  graph1_S,
  graph1_C,
  graph2_D,
  graph2_I,
  graph2_S,
  graph2_C,
  graph3_D,
  graph3_I,
  graph3_S,
  graph3_C,
  intensity,
  style
}
```

*assessment_id* – assessment result ID.

*type* – "DISC 24".

*graph1_D* – graph #1 "D" score.

*graph1_I* – graph #1 "I" score.

*graph1_S* – graph #1 "S" score.

*graph1_C* – graph #1 "C" score.

*graph2_D* – graph #2 "D" score.

*graph2_I* – graph #2 "I" score.

*graph2_S* – graph #2 "S" score.

*graph2_C* – graph #2 "C" score.

*graph3_D* – graph #3 "D" score.

*graph3_I* – graph #3 "I" score.

*graph3_S* – graph #3 "S" score.

*graph3_C* – graph #3 "C" score.

*intensity* – DISC keyword.

*style* – calculated personality style.

**Method**: GET
**Description**: Obtains calculated raw scores for given assessment result.
**Route**: /app/api/assessment/results/calculate/:result_id
**HTTP URL Parameters:**
*result_id* – assessment result ID.
**Response:** JSON array

```
{
  result_id,
  type = […],
  bai = {
    score_I,
    score_S,
    score_P,
    score_E,
    score_A,
    score_K,
    primary_keyword,
    secondary_keyword
  },
  childrens = {
    score_D,
    score_I,
    score_S,
    score_C,
    intensity,
    style
  },
  cognitive = {
    score_L,
    score_I,
    score_T,
    score_E,
    primary_keyword,
    secondary_keyword
  },
  disc15 = {
    score_D,
    score_I,
    score_S,
    score_C,
    intensity,
    style
  },
  disc24 = {
    graph1_D,
    graph1_I,
    graph1_S,
    graph1_C,
    graph2_D,
    graph2_I,
    graph2_S,
    graph2_C,
```

```
    graph3_D,
    graph3_I,
    graph3_S,
    graph3_C,
    intensity,
    style
  },
  perceptual = {
    score_A,
    score_V,
    score_K,
    primary_keyword,
    secondary_keyword
  },
  sgi = {
    score_Administration,
    score_Apostle,
    score_Craftsman,
    score_Discernment,
    score_Encouragement,
    score_Evangelist,
    score_Faith,
    score_Giving,
    score_Healing,
    score_Helps,
    score_Intercession,
    score_Knowledge,
    score_Leadership,
    score_Mercy,
    score_Music,
    score_Pastor,
    score_Prophet,
    score_Serving,
    score_Teacher,
    score_Wisdom,
    absent_gifts = […],
    present_gifts = […]
  },
  teams = {
    score_T,
    score_E,
    score_A,
    score_M,
    score_S,
    primary_keyword,
    secondary_keyword
  },
  values = {
    score_L,
    score_E,
    score_P,
    score_J,
    primary_keyword,
    secondary_keyword
```

}
}

*result_id* – assessment result ID.

*type* – a JSON array, list of assessment types [bai|childrens|cognitive|disc15|disc24|perceptual|sgi|teams|values].

*bai* – section contains 'BAI' assessment results:

- *score_I* – the "Inner Awareness" score;
- *score_S* – the "Social/Humanitarian" score;
- *score_P* – the "Political/EmPowerment" score;
- *score_E* – the "Economic" score;
- *score_A* – the "Artistic" score;
- *score_K* – the "Knowledge" score;
- *primary_keyword* – primary 'BAI' style;
- *secondary_keyword* – secondary 'BAI' style.

*childrens* – section contains 'CHILDREN'S' assessment results:

- *score_D* – the "D" score;
- *score_I* – the "I" score;
- *score_S* – the "S" score;
- *score_C* – the "C" score;
- *intensity* – DISC code;
- *style* – calculated personality style.

*cognitive* – section contains 'COGNITIVE' assessment results:

- *score_L* – the "Literal" score;
- *score_I* – the "Intuitive" score;
- *score_T* – the "Theoretical" score;
- *score_E* – the "Experimental" score;
- *primary_keyword* – primary 'COGNITIVE' style;
- *secondary_keyword* – secondary 'COGNITIVE' style.

*disc15* – section contains 'DISC 15' assessment results:

- *score_D* – the "D" score;
- *score_I* – the "I" score;
- *score_S* – the "S" score;
- *score_C* – the "C" score;
- *intensity* – DISC code;
- *style* – calculated personality style.

*disc24* – section contains 'DISC 24' assessment results:

- *graph1_D* – graph #1 "D" score;
- *graph1_I* – graph #1 "I" score;
- *graph1_S* – graph #1 "S" score;
- *graph1_C* – graph #1 "C" score;
- *graph2_D* – graph #2 "D" score;
- *graph2_I* – graph #2 "I" score;
- *graph2_S* – graph #2 "S" score;
- *graph2_C* – graph #2 "C" score;
- *graph3_D* – graph #3 "D" score;
- *graph3_I* – graph #3 "I" score;

*graph3_S* – graph #3 “S” score;
*graph3_C* – graph #3 “C” score;
*intensity* – DISC code;
*style* – calculated personality style.

*perceptual* – section contains ‘PERCEPTUAL’ assessment results:
*score_A* – the “Auditory” score;
*score_V* – the “Visual” score;
*score_K* – the “Kinesthetic” score;
*primary_keyword* – primary ‘PERCEPTUAL’ style;
*secondary_keyword* – secondary ‘PERCEPTUAL’ style.

*sgi* – section contains ‘SGI’ assessment results:
*score_Administration* – the “Administration” score;
*score_Apostle* – the “Apostle” score;
*score_Craftsman* – the “Craftsman” score;
*score_Discernment* – the “Discernment” score;
*score_Encouragement* – the “Encouragement” score;
*score_Evangelist* – the “Evangelist” score;
*score_Faith* – the “Faith” score;
*score_Giving* – the “Giving” score;
*score_Healing* – the “Healing” score;
*score_Helps* – the “Helps” score;
*score_Intercession* – the “Intercession” score;
*score_Knowledge* – the “Knowledge” score;
*score_Leadership* – the “Leadership” score;
*score_Mercy* – the “Mercy” score;
*score_Music* – the “Music” score;
*score_Pastor* – the “Pastor” score;
*score_Prophet* – the “Prophet” score;
*score_Serving* – the “Serving” score;
*score_Teacher* – the “Teacher” score;
*score_Wisdom* – the “Wisdom” score;
*absent_gifts* – an array, list of absent gifts;
*present_gifts* – an array, list of present gifts.

*teams* – section contains ‘TEAMS’ assessment results:
*score_T* – the “Theorist” score;
*score_E* – the “Executor” score;
*score_A* – the “Analyst” score;
*score_M* – the “Manager” score;
*score_S* – the “Strategist” score;
*primary_keyword* – primary ‘TEAMS’ style;
*secondary_keyword* – secondary ‘TEAMS’ style.

*values* – section contains ‘VALUES’ assessment results:
*score_L* – the “Loyalty” score;
*score_E* – the “Equality” score;
*score_P* – the “Personal Freedom” score;

*score_J* – the “Justice” score;
*primary_keyword* – primary ‘VALUES’ style;
*secondary_keyword* – secondary ‘VALUES’ style.

**[The method is DEPRECATED. Use GET /app/api/assessment/results/download/:result_id instead]**
**Method**: GET
**Description**: Downloads full PDF report for given assessment result.
**Route**: /app/api/assessment/:result_id/download
**HTTP URL Parameters:**
*result_id* – assessment result ID.
**Response:** Binary stream for PDF report. Content-Type: application/pdf. Content-Disposition: attachment.

**Method**: GET
**Description**: Downloads full PDF report for given assessment result.
**Route**: /app/api/assessment/results/download/:result_id
**HTTP URL Parameters:**
*result_id* – assessment result ID.
**Response:** Binary stream for PDF report. Content-Type: application/pdf. Content-Disposition: attachment.

**Method**: GET
**Description**: Downloads PDF report (graphs only) for given assessment result.
**Route**: /app/api/assessment/results/download_graph/:result_id
**HTTP URL Parameters:**
*result_id* – assessment result ID.
**Response:** Binary stream for PDF report. Content-Type: application/pdf. Content-Disposition: attachment.

**Method**: GET
**Description**: Returns HTML nodes of "summary" report for given assessment result.
**Route**: /app/api/assessment/results/report_summary/:result_id
**HTTP URL Parameters:**
*result_id* – assessment result ID.
**Response:** JSON array

```
{
  result_id,
  type = […],
  report_nodes = [
    {
      node_type,
      node_html
    },
    …
  ]
}
```

*result_id* – assessment result ID.
*type* – a JSON array, list of assessment types [bai|childrens|cognitive|disc15|disc24|perceptual|sgi|teams|values].
*report_nodes* – the list of summary report HTML nodes:
*node_type* – node type [bai|childrens|cognitive|disc15|disc24|perceptual|sgi|teams|values];
*node_html* – HTML fragment that corresponds to the node.

## Client-Side Assessments

**Method**: POST

**Description**: Initializes assessment passage by external user. This method creates new record for assessment test taken on client side. The method does not cause creation of fully-fledged PK3 user that can access PK3 site. Instead, it creates special record that holds information about client-side authenticated user.

**Route**: /app/api/assessment/client_side/initialize

**HTTP Form Parameters:**

*user_external_id* – client-side user identifier (email, GUID, etc.). Must be unique within the PK3 account the user is bound to.

*user_display_name* – [optional] the user display name to be used on report generation. By default, user external ID is used as display name.

*account_id* – [optional] the PK3 account the user is bound to. By default, the user is bound to API caller account.

*assessment_id* – assessment ID.

*tags* – [optional] JSON array

```
{
  tag_list = […]
}
```

*tag_lists* – list of tags to be associated with the user record. If the user already exists, the list of tags will be merged with the list of existing tags for this user. New account tags are created if needed.

**Response:** JSON array

```
{
  result_id
}
```

*result_id* – assessment result ID (assessment passage tracking ID).

**Method**: PUT

**Description**: Saves user responses to questions on the assessment test page for the test taken on client side. The assessment passage must be initialized beforehand via *POST /app/api/assessment/client_side/initialize* call.

**Route**: /app/api/assessment/client_side/response

**HTTP Form Parameters:**

*result _id* – assessment result ID (assessment passage tracking ID).

*page_id* – assessment page ID.

*response* – JSON array

```
[
  {
    item_id,
    options = [
      { value, order },…
    ]
  },
  …
]
```

*item_id* – page item ID.

*options* – For BAI, COGNITIVE, DISC 15, DISC 24, TEAMS, VALUES assessments – the answer options as ordered by the user:
*value* – answer value code;
*order* – 1-based index of answers as ordered by the user (for BAI assessments valid order values are in the range [1; 6], for TEAMS assessments valid order values are in the range [1; 5], for COGNITIVE, DISC 15, DISC 24 and VALUES assessments valid order values are in the range [1; 4]).
For CHILDREN'S, PERCEPTUAL and SGI assessments – the answer option chosen by the user:
*value* – answer value code;
*order* – must be set to 1.

**Method**: GET

**Description**: Returns information about statuses of client-side assessment tests available for the API caller account or one of its subaccounts. Assessment test status records are sorted by assessment start timestamp, the latest record is first.

**Route**: /app/api/assessment/client_side/results/:account_id? [user_external_id=:user_external_id] [&max_records=:max_records] [&start_record=:start_record] [&date_since=:date_since][&tags=:tags]

**HTTP URL Parameters:**

*account_id* – account ID.

**HTTP GET Parameters:**

*user_external_id* – [optional] client-side user identifier. Filters assessment results, returns the results available for given user.

*max_records* – [optional] the maximum number of status records returned per request. By default, the maximum number of records is set to 10000. You cannot return more than 10000 status records per request.

*start_record* – [optional] index of the first record. By default, index of the first record is 0.

*date_since* – [optional] [yyyy-mm-dd] filters out status records older than specified date.

*tags* – [optional] JSON array

```
{
  tag_list = […]
}
```

*tag_lists* – list of tags to apply as a filter. Filters assessment results, returns the results for users associated with at least one of the listed tags.

**Response:** JSON array

```
{
  total_record_number,
  results = [
    {
      id,
      user_external_id,
      assessment_id
      date_completed,
      date_started,
      status,
      next_page_id,
      pages_completed = [
        { page_id, completed_on }, …
      ]
    },
    …
  ]
}
```

*total_record_number* – total number of assessment test results found that match the filters.

*results* – JSON array of assessment test status records:

- *id* – assessment result ID.
- *user_external_id* – client-side identifier of the user that took the assessment.
- *assessment_id* – assessment ID.
- *date_completed* – [yyyy-mm-dd hh:mm:ss] assessment completion timestamp.
- *date_started* – [yyyy-mm-dd hh:mm:ss] assessment start timestamp.
- *status* – assessment completion status [Completed | In Progress].
- *next_page_id* – page ID of the next assessment test page to respond.
- *page_id* – page ID of the completed test page (test pages with user responses).
- *completed_on* – [yyyy-mm-dd hh:mm:ss] page completion timestamp.

**Method**: GET
**Description**: Obtains calculated raw scores for given client-side assessment test.
**Route**: /app/api/assessment/client_side/results/calculate/:result_id
**HTTP URL Parameters:**
*result_id* – assessment result ID.
**Response:** JSON array

```
{
  result_id,
  type = […],
  bai = {
    score_I,
    score_S,
    score_P,
    score_E,
    score_A,
    score_K,
    primary_keyword,
    secondary_keyword
  },
  childrens = {
    score_D,
    score_I,
    score_S,
    score_C,
    intensity,
    style
  },
  cognitive = {
    score_L,
    score_I,
    score_T,
    score_E,
    primary_keyword,
    secondary_keyword
  },
  disc15 = {
    score_D,
    score_I,
    score_S,
    score_C,
    intensity,
    style
  },
  disc24 = {
    graph1_D,
    graph1_I,
    graph1_S,
    graph1_C,
    graph2_D,
    graph2_I,
    graph2_S,
    graph2_C,
```

```
    graph3_D,
    graph3_I,
    graph3_S,
    graph3_C,
    intensity,
    style
  },
  perceptual = {
    score_A,
    score_V,
    score_K,
    primary_keyword,
    secondary_keyword
  },
  sgi = {
    score_Administration,
    score_Apostle,
    score_Craftsman,
    score_Discernment,
    score_Encouragement,
    score_Evangelist,
    score_Faith,
    score_Giving,
    score_Healing,
    score_Helps,
    score_Intercession,
    score_Knowledge,
    score_Leadership,
    score_Mercy,
    score_Music,
    score_Pastor,
    score_Prophet,
    score_Serving,
    score_Teacher,
    score_Wisdom,
    absent_gifts = […],
    present_gifts = […]
  },
  teams = {
    score_T,
    score_E,
    score_A,
    score_M,
    score_S,
    primary_keyword,
    secondary_keyword
  },
  values = {
    score_L,
    score_E,
    score_P,
    score_J,
    primary_keyword,
    secondary_keyword
```

```
    }
}
```

*result_id* – assessment result ID.

*type* – a JSON array, list of assessment types [bai|childrens|cognitive|disc15|disc24|perceptual|sgi|teams|values].

*bai* – section contains 'BAI' assessment results:

- *score_I* – the "Inner Awareness" score;
- *score_S* – the "Social/Humanitarian" score;
- *score_P* – the "Political/EmPowerment" score;
- *score_E* – the "Economic" score;
- *score_A* – the "Artistic" score;
- *score_K* – the "Knowledge" score;
- *primary_keyword* – primary 'BAI' style;
- *secondary_keyword* – secondary 'BAI' style.

*childrens* – section contains 'CHILDREN'S' assessment results:

- *score_D* – the "D" score;
- *score_I* – the "I" score;
- *score_S* – the "S" score;
- *score_C* – the "C" score;
- *intensity* – DISC code;
- *style* – calculated personality style.

*cognitive* – section contains 'COGNITIVE' assessment results:

- *score_L* – the "Literal" score;
- *score_I* – the "Intuitive" score;
- *score_T* – the "Theoretical" score;
- *score_E* – the "Experimental" score;
- *primary_keyword* – primary 'COGNITIVE' style;
- *secondary_keyword* – secondary 'COGNITIVE' style.

*disc15* – section contains 'DISC 15' assessment results:

- *score_D* – the "D" score;
- *score_I* – the "I" score;
- *score_S* – the "S" score;
- *score_C* – the "C" score;
- *intensity* – DISC code;
- *style* – calculated personality style.

*disc24* – section contains 'DISC 24' assessment results:

- *graph1_D* – graph #1 "D" score;
- *graph1_I* – graph #1 "I" score;
- *graph1_S* – graph #1 "S" score;
- *graph1_C* – graph #1 "C" score;
- *graph2_D* – graph #2 "D" score;
- *graph2_I* – graph #2 "I" score;
- *graph2_S* – graph #2 "S" score;
- *graph2_C* – graph #2 "C" score;
- *graph3_D* – graph #3 "D" score;
- *graph3_I* – graph #3 "I" score;

*graph3_S* – graph #3 “S” score;
*graph3_C* – graph #3 “C” score;
*intensity* – DISC code;
*style* – calculated personality style.

*perceptual* – section contains ‘PERCEPTUAL’ assessment results:
*score_A* – the “Auditory” score;
*score_V* – the “Visual” score;
*score_K* – the “Kinesthetic” score;
*primary_keyword* – primary ‘PERCEPTUAL’ style;
*secondary_keyword* – secondary ‘PERCEPTUAL’ style.

*sgi* – section contains ‘SGI’ assessment results:
*score_Administration* – the “Administration” score;
*score_Apostle* – the “Apostle” score;
*score_Craftsman* – the “Craftsman” score;
*score_Discernment* – the “Discernment” score;
*score_Encouragement* – the “Encouragement” score;
*score_Evangelist* – the “Evangelist” score;
*score_Faith* – the “Faith” score;
*score_Giving* – the “Giving” score;
*score_Healing* – the “Healing” score;
*score_Helps* – the “Helps” score;
*score_Intercession* – the “Intercession” score;
*score_Knowledge* – the “Knowledge” score;
*score_Leadership* – the “Leadership” score;
*score_Mercy* – the “Mercy” score;
*score_Music* – the “Music” score;
*score_Pastor* – the “Pastor” score;
*score_Prophet* – the “Prophet” score;
*score_Serving* – the “Serving” score;
*score_Teacher* – the “Teacher” score;
*score_Wisdom* – the “Wisdom” score;
*absent_gifts* – an array, list of absent gifts;
*present_gifts* – an array, list of present gifts.

*teams* – section contains ‘TEAMS’ assessment results:
*score_T* – the “Theorist” score;
*score_E* – the “Executor” score;
*score_A* – the “Analyst” score;
*score_M* – the “Manager” score;
*score_S* – the “Strategist” score;
*primary_keyword* – primary ‘TEAMS’ style;
*secondary_keyword* – secondary ‘TEAMS’ style.

*values* – section contains ‘VALUES’ assessment results:
*score_L* – the “Loyalty” score;
*score_E* – the “Equality” score;
*score_P* – the “Personal Freedom” score;

*score_J* – the "Justice" score;
*primary_keyword* – primary 'VALUES' style;
*secondary_keyword* – secondary 'VALUES' style.

**Method**: GET
**Description**: Downloads full PDF report for completed client-side assessment test.
**Route**: /app/api/assessment/client_side/results/download/:result_id
**HTTP URL Parameters:**
*result_id* – assessment result ID.
**Response:** Binary stream for PDF report. Content-Type: application/pdf. Content-Disposition: attachment.

**Method**: GET
**Description**: Downloads PDF report (graphs only) for completed client-side assessment test.
**Route**: /app/api/assessment/client_side/results/download_graph/:result_id
**HTTP URL Parameters:**
*result_id* – assessment result ID.
**Response:** Binary stream for PDF report. Content-Type: application/pdf. Content-Disposition: attachment.

**Method**: GET
**Description**: Returns HTML nodes of "summary" report for completed client-side assessment test.
**Route**: /app/api/assessment/client_side/results/report_summary/:result_id
**HTTP URL Parameters:**
*result_id* – assessment result ID.
**Response:** JSON array

```
{
  result_id,
  type = […],
  report_nodes = [
    {
      node_type,
      node_html
    },
    …
  ]
}
```

*result_id* – assessment result ID.
*type* – a JSON array, list of assessment types [bai|childrens|cognitive|disc15|disc24|perceptual|sgi|teams|values].
*report_nodes* – the list of summary report HTML nodes:
*node_type* – node type [disc24|values|teams];
*node_html* – HTML fragment that corresponds to the node.

**Method**: GET
**Description**: Retrieves list of tags associated with existing external user.
**Route**: /app/api/assessment/client_side/user/tags?user_external_id=:user_external_id [&account_id=:account_id]
**HTTP GET Parameters:**
*user_external_id* – client-side identifier of the user.
*account_id* – [optional] the PK3 account the user is bound to. If not specified, the user is looked up in the API caller account.
**Response:** JSON array

```
{
  tag_list = […]
}
```

*tag_list* – list of tags associated with the user record within given account.

**Method**: POST
**Description**: Associates tags with the existing external user. New account tags are created if needed.
**Route**: /app/api/assessment/client_side/user/tags
**HTTP Form Parameters:**
*user_external_id* – client-side identifier of the user.
*account_id* – [optional] the PK3 account the user is bound to. If not specified, the user is looked up in the API caller account.
*tags* – JSON array

```
{
  tag_list = […]
}
```

*tag_lists* – list of tags to be associated with the user record.
**Response:** JSON array

```
{
  tags_affected
}
```

*tags_affected* – number of listed tags associated with the user.

**Method**: DELETE
**Description**: Removes association of given tags with the existing external user.
**Route**: /app/api/assessment/client_side/user/tags?user_external_id=:user_external_id [&account_id=:account_id]&tags=:tags

**HTTP GET Parameters:**

*user_external_id* – client-side identifier of the user.

*account_id* – [optional] the PK3 account the user is bound to. If not specified, the user is looked up in the API caller account.

*tags* – JSON array

```
{
  tag_list = […]
}
```

*tag_list* – list of tags to be removed from association with the user record.

**Response:** JSON array

```
{
  tags_affected
}
```

*tags_affected* – number of listed tags removed from association with the user.

## Benchmarking

**Method**: GET
**Description**: Returns list of benchmark subscriptions available for the API caller account or one of its subaccounts.
**Route**: /app/api/benchmark/list?[account_id=:account_id] [&assessment_types=:assessment_types][&active_only=:active_only]
**HTTP GET Parameters:**
*account_id* – [optional] the PK3 account ID. If not specified, the method lists benchmark subscriptions available for the API caller account.
*assessment_types* – [optional] JSON array

```
{
  assessment_type_list = […]
}
```

*assessment_type_list* – list of assessment types. If specified, the method returns only benchmarks compatible with one of the listed assessment types. Supported assessment types are "BAI", "CHILDRENS", "COGNITIVE", "DISC 15", "DISC 24", "PERCEPTUAL", "TEAMS" and "VALUES".
*active_only* – [optional] the flag tells the server to return active benchmark subscriptions only. Any non-zero flag value is interpreted as "true". By default, the method lists active benchmark subscriptions and subscriptions for future periods.
**Response:** JSON array

```
[
  {
    benchmark_id,
    name,
    description,
    types = […],
    starts_on,
    expires_on,
    owner_account_id
  },
  …
]
```

*benchmark_id* – benchmark ID.
*name* – benchmark name.
*description* – benchmark description.
*types* – list of assessment types this benchmark is compatible with.
*starts_on* – [yyyy-mm-dd] the day benchmark subscription starts on.
*expires_on* – [yyyy-mm-dd] the day benchmark subscription expires (left empty, if subscription lasts indefinitely).
*owner_account_id* – ID of the account that owns this benchmark (for account-specific benchmarks).

**Method**: GET
**Description**: Returns information about dataset of assessment test results compatible by type with given benchmark. The results returned are grouped by the assessment.
**Route**: /app/api/benchmark/available_assessment_tests/:benchmark_id?[account_id=:account_id][&tags=:tags]

**HTTP GET Parameters:**

*benchmark_id* – the benchmark ID.
*account_id* – [optional] the PK3 account ID. If not specified, the API caller account is assumed.
*tags* – [optional] JSON array

```
{
  tag_list = […]
}
```

*tag_list* – list of tags to filter assessment test results dataset by.
**Response:** JSON array

```
[
  {
    assessment_id,
    types = […],
    amount
  },
  …
]
```

*assessment_id* – assessment ID.
*types* – list of assessment types.
*amount* – number of assessment test results compatible by type with given benchmark.

**Method**: POST
**Description**: Pre-calculates benchmark for given dataset of assessment test results.
**Route**: /app/api/benchmark/calculate

**HTTP Form Parameters:**

*account_id* – [optional] the PK3 account ID. If not specified, the API caller account is assumed.
*benchmark_id* – ID of the benchmark to apply.
*assessment_id* – assessment ID. The benchmark is applied to test results of the given assessment.
*tags* – [optional] JSON array

```
{
  tag_list = […]
}
```

*tag_lists* – list of tags to filter assessment test results dataset by.
**Response:** JSON array

```
{
  config_id
}
```

*config_id* – configuration ID of pre-calculated benchmark (tracking ID for pre-calculated benchmark).

**Method**: GET

**Description**: Returns list of pre-calculated benchmarks for given account. List of pre-calculated benchmarks is sorted by date and time of calculation, the latest records first.

**Route**: GET /app/api/benchmark/calculated/list?[account_id=:account_id] [&max_records=:max_records][&start_record=:start_record]

**HTTP GET Parameters:**

*account_id* – [optional] the PK3 account ID. If not specified, the API caller account is assumed.

*max_records* – [optional] the maximum number of records returned per request. By default, the maximum number of records is set to 100. You cannot return more than 1000 records per request.

*start_record* – [optional] index of the first record. By default, index of the first record is 0.

**Response:** JSON array

```
{
  total_record_number,
  results = [
    {
      config_id,
      benchmark_id,
      assessment_id,
      tag_list = […],
      created_on,
      created_by
    },
    …
  ]
}
```

*total_record_number* – total number of pre-calculated benchmark records found that match the filters.

*results* – JSON array of pre-calculated benchmark records:

- *config_id* – configuration ID of pre-calculated benchmark.
- *benchmark_id* – benchmark ID.
- *assessment_id* – assessment ID. The benchmark was applied to test results of the given assessment.
- *tag_list* – list of tags that was used to filter assessment test results dataset.
- *created_on* – [yyyy-mm-dd hh:mm:ss] date and time when the calculation was carried out.
- *created_by* – ID of the user that initiated the calculation ('API' for benchmarks pre-calculated via API call).

**Method**: GET

**Description**: Returns user results for given pre-calculated benchmark. User benchmark result records are sorted by benchmark criteria total match count, the records with highest match count first.

**Route**: /app/api/benchmark/calculated/results/:config_id?[max_records=:max_records][&start_record=:start_record][&tags=:tags]

**HTTP URL Parameters:**

*config_id* – configuration ID of pre-calculated benchmark (pre-calculated benchmark tracking ID).

**HTTP GET Parameters:**

*max_records* – [optional] the maximum number of user benchmark result records returned per request. By default, the maximum number of records is set to 100. You cannot return more than 1000 status records per request.

*start_record* – [optional] index of the first record. By default, index of the first record is 0.

*tags* – [optional] JSON array

```
{
  tag_list = […]
}
```

*tag_lists* – list of tags to apply as a filter. Filters user benchmark results, returns the results for users associated with at least one of the listed tags.

**Response:** JSON array

```
{
  total_record_number,
  results = [
    {
      id,
      user_id,
      user_external_id,
      user_name,
      tag_list = […],
      bai_match,
      childrens_match,
      cognitive_match,
      disc15_match,
      disc24_match,
      perceptual_match,
      teams_match,
      values_match,
      total_match,
      date_completed
    },
    …
  ]
}
```

*total_record_number* – total number of user benchmark result records found that match the filters.

*results* – JSON array of benchmark result records:

- *id* – user benchmark result ID.
- *user_ id* – PK3 identifier of the user that took the assessment (left empty for external users).
- *user_external_id* – for external users, client-side identifier of the user that took the assessment (left empty for PK3 users).
- *user_name* – name of the user that took the assessment.
- *tag_list* – list of tags associated with the user.
- *bai_match* – "BAI" benchmark criteria match count.
- *childrens_match* – "CHILDRENS" benchmark criteria match count.
- *cognitive_match* – "COGNITIVE" benchmark criteria match count.
- *disc15_match* – "DISC 15" benchmark criteria match count.
- *disc24_match* – "DISC 24" benchmark criteria match count.
- *perceptual_match* – "PERCEPTUAL" benchmark criteria match count.
- *teams_match* – "TEAMS" benchmark criteria match count.
- *values_match* – "VALUES" benchmark criteria match count.
- *total_match* – benchmark criteria total match count.
- *date_completed* – [yyyy-mm-dd hh:mm:ss] assessment completion timestamp.

**Method**: GET
**Description**: Downloads simple PDF report on user benchmark result for pre-calculated benchmark.
**Route**: /app/api/benchmark/calculated/report/simple/:benchmark_result_id
**HTTP URL Parameters:**
*benchmark_result_id* – user benchmark result ID.
**Response:** Binary stream for PDF report. Content-Type: application/pdf. Content-Disposition: attachment.

**Method**: GET
**Description**: Downloads full PDF report on user benchmark result for pre-calculated benchmark.
**Route**: /app/api/benchmark/calculated/report/full/:benchmark_result_id
**HTTP URL Parameters:**
*benchmark_result_id* – user benchmark result ID.
**Response:** Binary stream for PDF report. Content-Type: application/pdf. Content-Disposition: attachment.

**Method**: GET
**Description**: Returns benchmark match scores for given benchmark applied to specified completed assessment test.
**Route**: /app/api/benchmark/apply/:benchmark_id/:result_id
**HTTP URL Parameters:**
*benchmark_id* – benchmark ID.
*result_id* – assessment result ID.
**Response:** JSON array

```
{
  bai_match,
  childrens_match,
  cognitive_match,
  disc15_match,
  disc24_match,
  perceptual_match,
  teams_match,
  values_match,
  total_match
}
```

*bai_match* – "BAI" benchmark criteria match count.
*childrens_match* – "CHILDRENS" benchmark criteria match count.
*cognitive_match* – "COGNITIVE" benchmark criteria match count.
*disc15_match* – "DISC 15" benchmark criteria match count.
*disc24_match* – "DISC 24" benchmark criteria match count.
*perceptual_match* – "PERCEPTUAL" benchmark criteria match count.
*teams_match* – "TEAMS" benchmark criteria match count.
*values_match* – "VALUES" benchmark criteria match count.
*total_match* – benchmark criteria total match count.

**Method**: GET
**Description**: Downloads simple PDF report on given benchmark applied to specified completed assessment test.
**Route**: /app/api/benchmark/apply/report/simple/:benchmark_id/:result_id
**HTTP URL Parameters:**
*benchmark_id* – benchmark ID.
*result_id* – assessment result ID.
**Response:** Binary stream for PDF report. Content-Type: application/pdf. Content-Disposition: attachment.

**Method**: GET
**Description**: Downloads full PDF report on given benchmark applied to specified completed assessment test.
**Route**: /app/api/benchmark/apply/report/full/:benchmark_id/:result_id
**HTTP URL Parameters:**
*benchmark_id* – benchmark ID.
*result_id* – assessment result ID.
**Response:** Binary stream for PDF report. Content-Type: application/pdf. Content-Disposition: attachment.

**Method**: GET
**Description**: Returns benchmark match scores for given benchmark applied to specified completed client-side assessment test.
**Route**: /app/api/benchmark/apply/client_side/:benchmark_id/:result_id
**HTTP URL Parameters:**
*benchmark_id* – benchmark ID.
*result_id* – assessment result ID.
**Response:** JSON array

```
{
  bai_match,
  childrens_match,
  cognitive_match,
  disc15_match,
  disc24_match,
  perceptual_match,
  teams_match,
  values_match,
  total_match
}
```

*bai_match* – "BAI" benchmark criteria match count.
*childrens_match* – "CHILDRENS" benchmark criteria match count.
*cognitive_match* – "COGNITIVE" benchmark criteria match count.
*disc15_match* – "DISC 15" benchmark criteria match count.
*disc24_match* – "DISC 24" benchmark criteria match count.
*perceptual_match* – "PERCEPTUAL" benchmark criteria match count.
*teams_match* – "TEAMS" benchmark criteria match count.
*values_match* – "VALUES" benchmark criteria match count.
*total_match* – benchmark criteria total match count.

**Method**: GET
**Description**: Downloads simple PDF report on given benchmark applied to specified completed client-side assessment test.
**Route**: /app/api/benchmark/apply/client_side/report/simple/:benchmark_id/:result_id
**HTTP URL Parameters:**
*benchmark_id* – benchmark ID.
*result_id* – assessment result ID.
**Response:** Binary stream for PDF report. Content-Type: application/pdf. Content-Disposition: attachment.

**Method**: GET
**Description**: Downloads full PDF report on given benchmark applied to specified completed client-side assessment test.
**Route**: /app/api/benchmark/apply/client_side/report/full/:benchmark_id/:result_id
**HTTP URL Parameters:**
*benchmark_id* – benchmark ID.
*result_id* – assessment result ID.
**Response:** Binary stream for PDF report. Content-Type: application/pdf. Content-Disposition: attachment.

## Error Handling

Response HTTP status code 200 tells the client requested operation is completed successfully. HTTP status code 4XX or 5XX tells the client requested operation has failed. Error message is sent as part of server response.

Generic error messages are sent when server cannot proceed with request authentication.

**Status Code**: 400
**Message**: Invalid request: bad request timestamp
**Description**: 'X-Time' header not found in the request or bad request timestamp provided.

**Status Code**: 400
**Message**: Invalid request: public key not specified
**Description**: 'X-Public' header not found in the request.

**Status Code**: 401
**Message**: Invalid request: public key unknown
**Description**: The public key provided in 'X-Public' header is unknown.

**Status Code**: 400
**Message**: Invalid request: bad request hash
**Description**: 'X-Hash' header not found in the request or the hash value provided does not match the request body.

**Status Code**: 500
**Message**: Server error: cannot proceed with request authentication
**Description**: Internal server error during request authentication.

Beyond the generic errors above, the operation specific error codes 4XX, 5XX and messages can be returned.

### PUT /app/api/account

**Status Code**: 400
**Message**: Account data not valid
**Description**: Unable to update account attributes with values provided in the request.

**Status Code**: 403
**Message**: Not allowed to update specified account
**Description**: The caller does not have rights to update specified account (the account specified in the request is not a caller's account or subaccount).

### POST /app/api/account/meta

**Status Code**: 403
**Message**: Meta data key already exists or is reserved for system use
**Description**: Meta data key specified in the request already exists or is reserved for system use.

**Status Code**: 403
**Message**: Not allowed to add meta data to the account specified in the request
**Description**: The caller does not have rights to add meta data to the specified account (the account specified in the request is not a caller's account or subaccount).

### GET /app/api/account/:account_id/tags

**Status Code**: 403
**Message**: Not allowed to retrieve information about the account specified in the request
**Description**: The caller does not have rights to retrieve information about specified account (the account specified in the request is not a caller's account or subaccount).

### POST /app/api/account/:account_id/tags

**Status Code**: 400
**Message**: Tags not specified
**Description**: The *tags* parameter not found, not a JSON array or contains an empty list of tags.

**Status Code**: 403
**Message**: Not allowed to create tags for the account specified in the request
**Description**: The caller does not have rights to create tags for the specified account (the account specified in the request is not a caller's account or subaccount).

### PUT /app/api/account/:account_id/tags

**Status Code**: 400
**Message**: Tags not specified
**Description**: The *tags* parameter not found, not a JSON array or contains an empty list of tags.

**Status Code**: 403
**Message**: Not allowed to edit tags for the account specified in the request
**Description**: The caller does not have rights to edit tags for the specified account (the account specified in the request is not a caller's account or subaccount).

### DELETE /app/api/account/:account_id/tags

**Status Code**: 400
**Message**: Tags not specified
**Description**: The *tags* parameter not found, not a JSON array or contains an empty list of tags.

**Status Code**: 403
**Message**: Not allowed to delete tags from the account specified in the request
**Description**: The caller does not have rights to delete tags from the specified account (the account specified in the request is not a caller's account or subaccount).

### POST /app/api/subaccounts

**Status Code**: 400
**Message**: Sub-account data not valid
**Description**: Unable to create subaccount with attributes provided in the request.

**Status Code**: 403
**Message**: Not allowed to add sub-account to the account specified in the request
**Description**: The caller does not have rights to create subaccount in the specified account (the account specified in the request is not a caller's account or subaccount).

### PUT /app/api/subaccounts

**Status Code**: 400
**Message**: Sub-account data not valid
**Description**: Unable to update subaccount with attributes provided in the request.

**Status Code**: 403
**Message**: Not allowed to update sub-account specified in the request
**Description**: The caller does not have rights to update specified subaccount (the account specified in the request is not a caller's subaccount).

### DELETE /app/api/subaccounts/:account_id/

**Status Code**: 400
**Message**: Unable to delete sub-account (sub-account does not exist)
**Description**: Subaccount specified in the request does not exist or cannot be deleted (constrained by the existing data linked to the account in the database).

**Status Code**: 403
**Message**: Not allowed to delete sub-account specified in the request
**Description**: The caller does not have rights to delete specified subaccount (the account specified in the request is not a caller's subaccount).

GET /app/api/account/:account_id/users/

**Status Code**: 403
**Message**: Not allowed to get the list of users for the account specified in the request
**Description**: The caller does not have rights to list users in the account (the account specified in the request is not a caller's account or subaccount).

POST /app/api/account/users/

**Status Code**: 400
**Message**: User data not valid
**Description**: Unable to create new user with attributes provided in the request (user with given email already exists).

**Status Code**: 403
**Message**: Not allowed to add new user to the account specified in the request
**Description**: The caller does not have rights to add users to the specified account (the account specified in the request is not a caller's account or subaccount).

PUT /app/api/account/users/

**Status Code**: 400
**Message**: User data not valid or the user is not a member of the account specified
**Description**: User membership attributes are not valid or the user is not a member of the account specified.

**Status Code**: 403
**Message**: Not allowed to modify user in the account specified in the request
**Description**: The caller does not have rights to modify membership data for the account (the account specified in the request is not a caller's account or subaccount).

DELETE /app/api/account/:account_id/users/:user_id

**Status Code**: 400
**Message**: User does not exist or is not a member of the account specified
**Description**: User with given ID does not exist or is not a member of the account specified.

**Status Code**: 403
**Message**: Not allowed to remove users from the account specified in the request
**Description**: The caller does not have rights to modify membership of the account (the account specified in the request is not a caller's account or subaccount).

### GET /app/api/user/tags/:account_id/:user_id

**Status Code**: 400
**Message**: User does not exist or is not a member of the account specified
**Description**: User with given ID does not exist or is not a member of the account specified.

**Status Code**: 403
**Message**: Not allowed to list tags for the user specified in the request
**Description**: The caller does not have rights to list tags for users of the given account (the account specified in the request is not a caller's account or subaccount).

### POST /app/api/user/tags/:account_id/:user_id

**Status Code**: 400
**Message**: Tags not specified
**Description**: The *tags* parameter not found, not a JSON array or contains an empty list of tags.

**Status Code**: 400
**Message**: User does not exist or is not a member of the account specified
**Description**: User with given ID does not exist or is not a member of the account specified.

**Status Code**: 403
**Message**: Not allowed to add tags to the user specified in the request
**Description**: The caller does not have rights to modify tags for users of the given account (the account specified in the request is not a caller's account or subaccount).

### DELETE /app/api/user/tags/:account_id/:user_id

**Status Code**: 400
**Message**: Tags not specified
**Description**: The *tags* parameter not found, not a JSON array or contains an empty list of tags.

**Status Code**: 400
**Message**: User does not exist or is not a member of the account specified
**Description**: User with given ID does not exist or is not a member of the account specified.

**Status Code**: 403
**Message**: Not allowed to drop tags for the user specified in the request
**Description**: The caller does not have rights to modify tags for users of the given account (the account specified in the request is not a caller's account or subaccount).

### POST /app/api/account/invite/

**Status Code**: 400
**Message**: Assessment ID not specified
**Description**: The *participant* flag is set, but the *assessment_id* parameter not found or is set to 0.

**Status Code**: 400
**Message**: Email address not specified
**Description**: The *email* parameter not found or is an empty string.

**Status Code**: 402
**Message**: You have to purchase more assessments
**Description**: The caller does not have available assessment passes for given assessment to transfer to the invitee.

**Status Code**: 403
**Message**: Not allowed to create invite URL for the account specified in the request
**Description**: The caller does not have rights to create invite URL on behalf of the specified account (the account specified in the request is not a caller's account or subaccount).

### GET /app/api/assessments/:account_id

**Status Code**: 403
**Message**: Not allowed to get the list of assessments for the account specified in the request
**Description**: The caller does not have rights to list assessments for the account specified (the account specified in the request is not a caller's account or subaccount).

### PUT /app/api/assessment/transfer

**Status Code**: 400
**Message**: Sender and recipient accounts are the same
**Description**: The sender and recipient accounts specified in the request are the same.

**Status Code**: 400
**Message**: Number of assessments must be a positive value
**Description**: Number of assessments to transfer must be a positive value.

**Status Code**: 403
**Message**: Not allowed to transfer assessments from the account specified
**Description**: The caller does not have rights to make an assessment transfer from the account specified (the sender account is not a caller's account or subaccount).

**Status Code**: 403
**Message**: Not allowed to transfer assessments to the account specified
**Description**: The caller does not have rights to make an assessment transfer to the account specified (the recipient account is not a caller's account or subaccount).

**Status Code**: 403
**Message**: The sender account does not have enough assessments to transfer
**Description**: The sender account does not have enough available assessment passes for given assessment to transfer.

**Status Code**: 500
**Message**: Unable to complete assessments transfer
**Description**: The server has failed to complete assessment transfer transaction.

### GET /app/api/assessment/:assessment_id

**Status Code**: 403
**Message**: Assessment does not exist or not allowed to get details of the assessment specified
**Description**: The caller does not have rights to get detailed information about given assessment (does not have purchase record for the assessment) or the assessment does not exist.

### DELETE /app/api/assessment/:assessment_id

**Status Code**: 403
**Message**: Not allowed to delete the assessment
**Description**: Current version of API rejects requests to delete the assessment.

GET /app/api/assessment/core/:assessment_id

**Status Code**: 403
**Message**: Assessment does not exist
**Description**: Assessment with specified ID does not exist.

GET /app/api/assessment/pages/:assessment_id

**Status Code**: 403
**Message**: Assessment does not exist
**Description**: Assessment with specified ID does not exist.

GET /app/api/assessment/results/:account_id

**Status Code**: 403
**Message**: Not allowed to get results for the account specified
**Description**: The caller does not have rights to list assessment results for the account specified (the account specified in the request is not a caller's account or subaccount).

GET /app/api/assessment/results/calculate/:result_id

**Status Code**: 400
**Message**: Assessment not complete
**Description**: The system cannot calculate results of the unfinished assessment.

**Status Code**: 403
**Message**: Result does not exist or not allowed to calculate scores for the result specified
**Description**: The caller does not have rights to view results of the assessment test (the user who took the test is not a member of the caller's account or subaccount) or the result with specified ID does not exist.

GET /app/api/assessment/results/download/:result_id

**Status Code**: 400
**Message**: Assessment not complete
**Description**: The system cannot calculate results of the unfinished assessment.

**Status Code**: 403
**Message**: Result does not exist or not allowed to get PDF report for the result specified
**Description**: The caller does not have rights to view results of the assessment test (the user who took the test is not a member of the caller's account or subaccount) or the result with specified ID does not exist.

**Status Code**: 500
**Message**: Unable to load result details
**Description**: Unable to load result details – internal server error.

GET /app/api/assessment/results/download_graph/:result_id

**Status Code**: 400
**Message**: Assessment not complete
**Description**: The system cannot calculate results of the unfinished assessment.

**Status Code**: 403
**Message**: Result does not exist or not allowed to get PDF report for the result specified
**Description**: The caller does not have rights to view results of the assessment test (the user who took the test is not a member of the caller's account or subaccount) or the result with specified ID does not exist.

**Status Code**: 500
**Message**: Unable to load result details
**Description**: Unable to load result details – internal server error.

GET /app/api/assessment/results/report_summary/:result_id

**Status Code**: 400
**Message**: Assessment not complete
**Description**: The system cannot calculate results of the unfinished assessment.

**Status Code**: 403
**Message**: Result does not exist or not allowed to get report summary for the result specified
**Description**: The caller does not have rights to view results of the assessment test (the user who took the test is not a member of the caller's account or subaccount) or the result with specified ID does not exist.

POST /app/api/assessment/client_side/initialize

**Status Code**: 400
**Message**: Assessment ID not specified
**Description**: The *assessment_id* parameter not found or is set to 0.

**Status Code**: 400
**Message**: User external ID not specified
**Description**: The *user_external_id* parameter not found or is an empty string.

**Status Code**: 402
**Message**: You have to purchase more assessments
**Description**: The caller's account (or account specified by *account_id* parameter) does not have available passes for given assessment.

**Status Code**: 403
**Message**: Not allowed to initialize assessment passage for the account specified in the request
**Description**: The caller does not have rights to initialize assessment passage on behalf of the specified account (the account specified in the request is not a caller's account or subaccount).

**Status Code**: 500
**Message**: Unable to initialize assessment passage
**Description**: The server has failed to initialize assessment passage (internal server error).

### PUT /app/api/assessment/client_side/response

**Status Code**: 400
**Message**: Invalid assessment page ID
**Description**: The *page_id* parameter not found or page does not belong to the current assessment.

**Status Code**: 400
**Message**: User response is already saved
**Description**: The user response to questions on given assessment test page is already saved.

**Status Code**: 400
**Message**: Unrecognized user response format
**Description**: The *response* parameter not found or server is unable to parse *response* JSON.

**Status Code**: 403
**Message**: Result does not exist or not allowed to save user responses for the result specified
**Description**: The caller does not have rights to access the result specified (the user who is taking the test is not associated with the caller's account or subaccount) or the result with specified ID does not exist.

**Status Code**: 500
**Message**: Unable to save user response
**Description**: The server has failed to save user response to assessment questions (internal server error).

### GET /app/api/assessment/client_side/results/:account_id

**Status Code**: 403
**Message**: Not allowed to get results for the account specified
**Description**: The caller does not have rights to list client-side assessment results for the account specified (the account specified in the request is not a caller's account or subaccount).

### GET /app/api/assessment/client_side/results/calculate/:result_id

**Status Code**: 400
**Message**: Assessment not complete
**Description**: The system cannot calculate results of the unfinished assessment.

**Status Code**: 403
**Message**: Result does not exist or not allowed to calculate scores for the result specified
**Description**: The caller does not have rights to view results of the client-side assessment test (the user who took the test is not associated with the caller's account or subaccount) or the result with specified ID does not exist.

GET /app/api/assessment/client_side/results/download/:result_id

**Status Code**: 400
**Message**: Assessment not complete
**Description**: The system cannot calculate results of the unfinished assessment.

**Status Code**: 403
**Message**: Result does not exist or not allowed to get PDF report for the result specified
**Description**: The caller does not have rights to view results of the client-side assessment test (the user who took the test is not associated with the caller's account or subaccount) or the result with specified ID does not exist.

**Status Code**: 500
**Message**: Unable to load result details
**Description**: Unable to load result details – internal server error.

GET /app/api/assessment/client_side/results/download_graph/:result_id

**Status Code**: 400
**Message**: Assessment not complete
**Description**: The system cannot calculate results of the unfinished assessment.

**Status Code**: 403
**Message**: Result does not exist or not allowed to get PDF report for the result specified
**Description**: The caller does not have rights to view results of the client-side assessment test (the user who took the test is not associated with the caller's account or subaccount) or the result with specified ID does not exist.

**Status Code**: 500
**Message**: Unable to load result details
**Description**: Unable to load result details – internal server error.

GET /app/api/assessment/client_side/results/report_summary/:result_id

**Status Code**: 400
**Message**: Assessment not complete
**Description**: The system cannot calculate results of the unfinished assessment.

**Status Code**: 403
**Message**: Result does not exist or not allowed to get report summary for the result specified
**Description**: The caller does not have rights to view results of the client-side assessment test (the user who took the test is not associated with the caller's account or subaccount) or the result with specified ID does not exist.

### GET /app/api/assessment/client_side/user/tags

**Status Code**: 400
**Message**: User external ID not specified
**Description**: The *user_external_id* parameter not found.

**Status Code**: 400
**Message**: User not found
**Description**: The user with specified external ID does not exist in the given account.

**Status Code**: 403
**Message**: Not allowed to list tags for the user specified in the request
**Description**: The caller does not have rights to list tags for external user of the given account (the account specified in the request is not a caller's account or subaccount).

### POST /app/api/assessment/client_side/user/tags

**Status Code**: 400
**Message**: Tags not specified
**Description**: The *tags* parameter not found, not a JSON array or contains an empty list of tags.

**Status Code**: 400
**Message**: User external ID not specified
**Description**: The *user_external_id* parameter not found.

**Status Code**: 400
**Message**: User not found
**Description**: The user with specified external ID does not exist in the given account.

**Status Code**: 403
**Message**: Not allowed to add tags to the user specified in the request
**Description**: The caller does not have rights to modify tags for external user of the given account (the account specified in the request is not a caller's account or subaccount).

### DELETE /app/api/assessment/client_side/user/tags

**Status Code**: 400
**Message**: Tags not specified
**Description**: The *tags* parameter not found, not a JSON array or contains an empty list of tags.

**Status Code**: 400
**Message**: User external ID not specified
**Description**: The *user_external_id* parameter not found.

**Status Code**: 400
**Message**: User not found
**Description**: The user with specified external ID does not exist in the given account.

**Status Code**: 403
**Message**: Not allowed to drop tags for the user specified in the request
**Description**: The caller does not have rights to modify tags for external user of the given account (the account specified in the request is not a caller's account or subaccount).

### GET /app/api/benchmark/list

**Status Code**: 403
**Message**: Not allowed to retrieve information about the account specified in the request
**Description**: The caller does not have rights to retrieve information about specified account (the account specified in the request is not a caller's account or subaccount).

### GET /app/api/benchmark/available_assessment_tests/:benchmark_id

**Status Code**: 400
**Message**: Benchmark does not exist or not defined properly
**Description**: The benchmark with specified ID does not exist or not defined properly in PK3.

**Status Code**: 403
**Message**: Not allowed to retrieve information about the account specified in the request
**Description**: The caller does not have rights to retrieve information about specified account (the account specified in the request is not a caller's account or subaccount).

**Status Code**: 500
**Message**: Unable to get list of available results
**Description**: Unable to get list of available results – internal server error.

### POST /app/api/benchmark/calculate

**Status Code**: 400
**Message**: Assessment ID not specified
**Description**: The *assessment_id* parameter not found or is set to 0.

**Status Code**: 403
**Message**: Not allowed to calculate benchmark for the account specified in the request
**Description**: The caller does not have rights to use specified account (the account specified in the request is not a caller's account or subaccount).

**Status Code**: 403
**Message**: Benchmark does not exist or not allowed to apply benchmark specified
**Description**: The caller's account (or subaccount specified in the request) does not have active subscription for the benchmark specified in the request or the benchmark with specified ID does not exist.

**Status Code**: 500
**Message**: Unable to calculate benchmark
**Description**: Unable to calculate benchmark – internal server error.

### GET /app/api/benchmark/calculated/list

**Status Code**: 403
**Message**: Not allowed to retrieve information about the account specified in the request
**Description**: The caller does not have rights to retrieve information about specified account (the account specified in the request is not a caller's account or subaccount).

### GET /app/api/benchmark/calculated/results/:config_id

**Status Code**: 403
**Message**: Pre-calculated benchmark does not exist or not allowed to access pre-calculated benchmark
**Description**: The caller does not have rights to retrieve information about specified pre-calculated benchmark (pre-calculated benchmark specified is associated with the account that is not a caller's account or subaccount).

GET /app/api/benchmark/calculated/report/simple/:benchmark_result_id

**Status Code**: 403
**Message**: Result does not exist or not allowed to get PDF report for the result specified
**Description**: The caller does not have rights to retrieve information about specified user benchmark scores (the result is of pre-calculated benchmark associated with the account that is not a caller's account or subaccount) or the user benchmark result with specified ID does not exist.

**Status Code**: 500
**Message**: Unable to load result details
**Description**: Unable to load result details – internal server error.

**Status Code**: 500
**Message**: Unable to generate PDF report
**Description**: Unable to generate PDF report – internal server error.

GET /app/api/benchmark/calculated/report/full/:benchmark_result_id

**Status Code**: 403
**Message**: Result does not exist or not allowed to get PDF report for the result specified
**Description**: The caller does not have rights to retrieve information about specified user benchmark scores (the result is of pre-calculated benchmark associated with the account that is not a caller's account or subaccount) or the user benchmark result with specified ID does not exist.

**Status Code**: 500
**Message**: Unable to load result details
**Description**: Unable to load result details – internal server error.

**Status Code**: 500
**Message**: Unable to generate PDF report
**Description**: Unable to generate PDF report – internal server error.

GET /app/api/benchmark/apply/:benchmark_id/:result_id

**Status Code**: 400
**Message**: Assessment not complete
**Description**: The system cannot apply benchmark to the results of the unfinished assessment.

**Status Code**: 403
**Message**: Benchmark does not exist or not allowed to apply benchmark specified
**Description**: The caller's account does not have active subscription for the benchmark specified or the benchmark with specified ID does not exist.

**Status Code**: 403
**Message**: Result does not exist or not allowed to get benchmark scores for the result specified
**Description**: The caller does not have rights to access results of the assessment test (the user who took the test is not associated with the caller's account or subaccount) or the result with specified ID does not exist.

**Status Code**: 500
**Message**: Unable to load result details
**Description**: Unable to load result details – internal server error.

GET /app/api/benchmark/apply/report/simple/:benchmark_id/:result_id

**Status Code**: 400
**Message**: Assessment not complete
**Description**: The system cannot apply benchmark to the results of the unfinished assessment.

**Status Code**: 403
**Message**: Benchmark does not exist or not allowed to apply benchmark specified
**Description**: The caller's account does not have active subscription for the benchmark specified or the benchmark with specified ID does not exist.

**Status Code**: 403
**Message**: Result does not exist or not allowed to get PDF report for the result specified
**Description**: The caller does not have rights to access results of the assessment test (the user who took the test is not associated with the caller's account or subaccount) or the result with specified ID does not exist.

**Status Code**: 500
**Message**: Unable to load result details
**Description**: Unable to load result details – internal server error.

**Status Code**: 500
**Message**: Unable to generate PDF report
**Description**: Unable to generate PDF report – internal server error.

GET /app/api/benchmark/apply/report/full/:benchmark_id/:result_id

**Status Code**: 400
**Message**: Assessment not complete
**Description**: The system cannot apply benchmark to the results of the unfinished assessment.

**Status Code**: 403
**Message**: Benchmark does not exist or not allowed to apply benchmark specified
**Description**: The caller's account does not have active subscription for the benchmark specified or the benchmark with specified ID does not exist.

**Status Code**: 403
**Message**: Result does not exist or not allowed to get PDF report for the result specified
**Description**: The caller does not have rights to access results of the assessment test (the user who took the test is not associated with the caller's account or subaccount) or the result with specified ID does not exist.

**Status Code**: 500
**Message**: Unable to load result details
**Description**: Unable to load result details – internal server error.

**Status Code**: 500
**Message**: Unable to generate PDF report
**Description**: Unable to generate PDF report – internal server error.

GET /app/api/benchmark/apply/client_side/:benchmark_id/:result_id

**Status Code**: 400
**Message**: Assessment not complete
**Description**: The system cannot apply benchmark to the results of the unfinished client-side assessment.

**Status Code**: 403
**Message**: Benchmark does not exist or not allowed to apply benchmark specified
**Description**: The caller's account does not have active subscription for the benchmark specified or the benchmark with specified ID does not exist.

**Status Code**: 403
**Message**: Result does not exist or not allowed to get benchmark scores for the result specified
**Description**: The caller does not have rights to access results of the client-side assessment test (the user who took the test is not associated with the caller's account or subaccount) or the result with specified ID does not exist.

**Status Code**: 500
**Message**: Unable to load result details
**Description**: Unable to load result details – internal server error.

GET /app/api/benchmark/apply/client_side/report/simple/:benchmark_id/:result_id

**Status Code**: 400
**Message**: Assessment not complete
**Description**: The system cannot apply benchmark to the results of the unfinished client-side assessment.

**Status Code**: 403
**Message**: Benchmark does not exist or not allowed to apply benchmark specified
**Description**: The caller's account does not have active subscription for the benchmark specified or the benchmark with specified ID does not exist.

**Status Code**: 403
**Message**: Result does not exist or not allowed to get PDF report for the result specified
**Description**: The caller does not have rights to access results of the client-side assessment test (the user who took the test is not associated with the caller's account or subaccount) or the result with specified ID does not exist.

**Status Code**: 500
**Message**: Unable to load result details
**Description**: Unable to load result details – internal server error.

**Status Code**: 500
**Message**: Unable to generate PDF report
**Description**: Unable to generate PDF report – internal server error.

GET /app/api/benchmark/apply/client_side/report/full/:benchmark_id/:result_id

**Status Code**: 400
**Message**: Assessment not complete
**Description**: The system cannot apply benchmark to the results of the unfinished client-side assessment.

**Status Code**: 403
**Message**: Benchmark does not exist or not allowed to apply benchmark specified
**Description**: The caller's account does not have active subscription for the benchmark specified or the benchmark with specified ID does not exist.

**Status Code**: 403
**Message**: Result does not exist or not allowed to get PDF report for the result specified
**Description**: The caller does not have rights to access results of the client-side assessment test (the user who took the test is not associated with the caller's account or subaccount) or the result with specified ID does not exist.

**Status Code**: 500
**Message**: Unable to load result details
**Description**: Unable to load result details – internal server error.

**Status Code**: 500
**Message**: Unable to generate PDF report
**Description**: Unable to generate PDF report – internal server error.