ActivityInfo

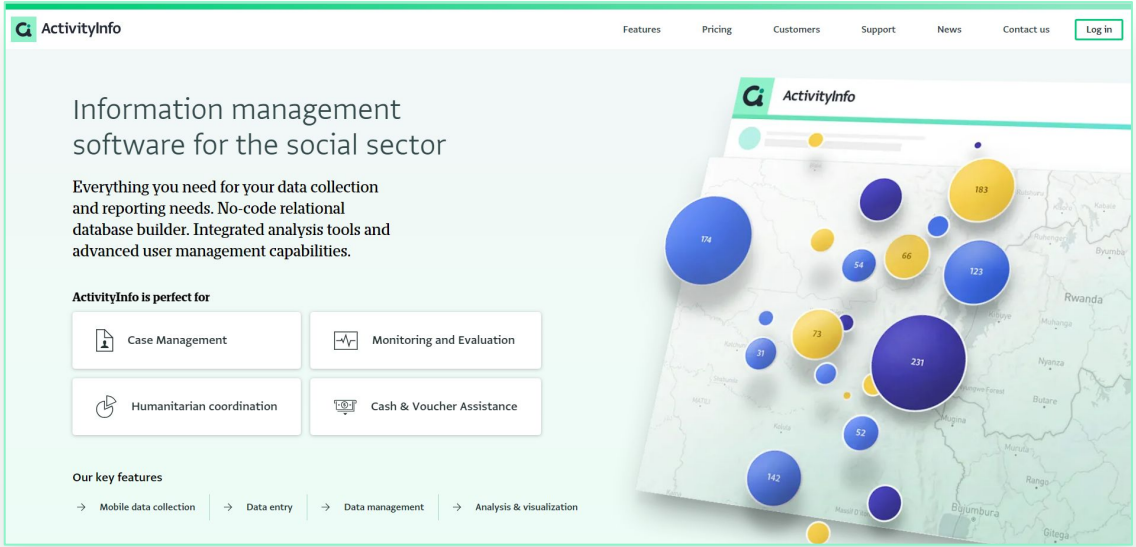
ActivityInfo R package updates

Starting shortly, please wait!

Presented by the ActivityInfo Team

All in one information management software for humanitarian and development operations.

- Track activities, outcomes
- Beneficiary management
- Surveys
- Work offline/online



Outline

- What changed in the last year?
- Form manipulation and data download
- Grant-based roles
- Bulk update/deactivate/delete users

What changed in the last year?

What changed in the last year?

- Grant-based role support has been added
- New tutorials:
  - Working with grant-based roles
  - Advanced user management: bulk add and delete users
  - Advanced use-cases with roles

What changed in the last year?

- getRecords() is more robust
  - Column de-duplication
  - Handles cyclic references and has a maxDepth parameter
- New billing account functions to manage databases
- Improved credentials management: API tokens are now stored per ActivityInfo server

Up next in 2025

- Uploading attachments
- Support for ActivityInfo formulas in filter() and mutate() for prepared views on the server
- Column auto-completion and expansion into parent forms, sub-forms, reference forms, etc.
  records %>% mutate(child_name = records$child$childName)

Installation and Authentication

Loading ActivityInfo and tidyverse

```
library(activityinfo)
library(tidyverse)

# Or one can explicitly include specific tidyverse packages:
library(dplyr)
library(tidyr)
library(purrr)
```

Working with grant-based roles tutorial

Create a database

```
library(activityinfo)

# We can use these options to turn on and off debugging messages
# Useful for logging on servers
options(activityinfo.verbose.requests = FALSE) # http requests
options(activityinfo.verbose.tasks = FALSE) # responses to different tasks

newDb <- addDatabase(
    label =
      sprintf(
        "Demo database %s",
        as.POSIXlt(Sys.time(), "UTC", "%Y-%m-%dT%H:%M")
        )
  )

databaseId <- newDb$databaseId
```

Create a form fields

```
formElements = list(
    textFieldSchema(
      label = "What is your name?",
      code = "NAME",
      description = "Please provide your full name",
      required = TRUE
    ),
    singleSelectFieldSchema(
      label = "What is your sex?",
      code = "SEX",
      options = c("Female", "Male", "Prefer not to answer"),
      required = TRUE
    ),
    singleSelectFieldSchema(
      label = "Are you pregnant",
      relevanceRule = "SEX != 'Male'",
      options = c("Yes", "No"),
      required = TRUE
    )
  )
```

Add the form to the database

```
surveySchema <- formSchema(
  databaseId = databaseId,
  label = "My new survey",
  element <- formElements
)

surveyForm <- addForm(surveySchema)
```

Create form with chaining |>

```
# Or we can split it up and use chaining to build and upload our form
optionalForm <-
  formSchema(databaseId = newDb$databaseId, label = "An optional form") |>
  addFormField(feedbacktextFieldSchema(label = "Anonymous feedback", code =
"feedback")) |>
  addForm()
```

Fetch database tree and system roles

The database metadata

```
dbTree <- getDatabaseTree(databaseId = newDb$databaseId)
as_tibble(dbTree$ownerRef)
```

The roles as a table

```
roles <- getDatabaseRoles(dbTree)
roles
```

Expand administrative permissions on the database

```
roles |>
  tidyr::unnest_longer(permissions) |>
  tidyr::unnest_wider(permissions) |>
  select(id, label, operation)
```

List role grants

A grant can be for a resource:

- Database,
- Folder, or
- Form.

The resourceId in these default roles is the databaseId.

```
roles |>
  select(id, label, grants) |>
  tidyr::unnest_longer(grants) |>
  tidyr::unnest_wider(grants)
```

Expand grants and operations

```
roles |>
  select(id, label, grants) |>
  tidyr::unnest_longer(grants) |>
  tidyr::unnest_wider(grants) |>
  tidyr::unnest_longer(operations) |>
  tidyr::unnest_wider(operations) |>
  select(id, label, resourceId, operation)
```

Retrieve a single role

```
readOnlyRole <- Filter(function(x) x$id == "readonly", dbTree$roles)
str(readOnlyRole)
```

Add users in bulk

Default role and load user data

```
defaultRoleId = "readonly" # Default role for all new users
users <- data.frame(
  name = paste0("Person ", 1:10),
  email = paste0("dickinson+person", 1:10, "@washnote.com"),
  stringsAsFactors = FALSE
)
```

Adding

```
for (i in seq_len(nrow(users))) {
  addDatabaseUser(databaseId = databaseId,
          email = users[i,"email"],
          name = users[i,"name"],
          roleId = defaultRoleId)
}
```

ActivityInfo

Inspect role assignments

```
dbUserRoles <- getDatabaseUsers(dbTree$databaseId) |> unnest_wider(role, names_sep = "_")
dbUserRoles
```

Create Role 1: Deny permission to delete

This is a resource level permission that we will apply to our survey form:

```
dataEntryFormId <- surveySchema$id
noDeletePermissions = resourcePermissions(
        view = TRUE,
        add_record = TRUE,
        edit_record = TRUE,
        delete_record = FALSE, # this prevents deletion
        export_records = TRUE),
      optional = FALSE
    )
```

Create role 1: Create and update role

```
dataEntryNoDeleteRole <- role(
  id = "entrynodelete",
  label = "Data entry without delete",
  grants = list(
    grant(
      resourceId = dataEntryFormId,
      permissions = noDeletePermissions # we created this
  ))
updateRole(dbTree$databaseId, dataEntryNoDeleteRole)
```

Create role 2: Admin without automation

These permissions affect the whole database, not just resources.

```
dbPermissionWithoutAutomation <- databasePermissions(
   manage_automations = FALSE,manage_users = TRUE, manage_roles = TRUE)
adminRoleNoAutomation <- role(
  id = "adminnoautomation", label = "Admin without automation",
  permissions = dbPermissionWithoutAutomation, grants = dbAdminGrants)
addRole(dbTree$databaseId, adminRoleNoAutomation)
```

Update Role with optional form access

```
optionalFormId <- optionalForm$id
optionalFormGrant <- grant(
      resourceId = optionalFormId,
      permissions = resourcePermissions(view = TRUE, add_record = TRUE, edit_record =
FALSE, delete_record = FALSE, export_records = FALSE),
      optional = TRUE # this makes the grant optional)
optionalAccessRole <- role(id = "optional", label = "Optional access to feedback from only",
  grants = list(optionalFormGrant)
)
updateRole(dbTree$databaseId, optionalAccessRole)
```

Create partner and reporting forms and roles

Partner form

```
reportingForm <- formSchema(
  databaseId = dbTree$databaseId,
  label = "Partner reports") |>
 addFormField(referenceFieldSchema(referencedFormId = partnerForm$id, code = "rp",
label = "Partner", required = TRUE)) |>
  addFormField(textFieldSchema(label = "Report", required = TRUE))
addForm(reportingForm)
```

Reporting form

```
partnerForm <- formSchema(
  databaseId = dbTree$databaseId,
  label = "Reporting Partners") |>
  addFormField(textFieldSchema(code = "name", label = "Partner name", required =
TRUE))
addForm(partnerForm)
partnerTbl <- tibble(name = c("Partner A", "Partner B", "Partner C"))
importRecords(partnerForm$id, data = partnerTbl)
```

Partner form

```
partnerTbl <- getRecords(partnerForm) |> collect()
# Generate reports and import records using the partner ID to link to partners
partnerReports <- paste0("This is a report from ", partnerTbl[["Partner name"]], ".")
reportingTbl <- tibble(Partner = partnerTbl[["_id"]], Report = partnerReports)
importRecords(reportingForm$id, data = reportingTbl)
```

Create a parameter and grant

We need a partner parameter so we know the user's organization to create a grant that limits access to only reports from one's own organization.

```
partnerParameter <- parameter(id = "partner", label = "Partner", range = partnerForm$id)
reportGrant <- grant(resourceId = reportingForm$id, permissions = resourcePermissions(
    view = sprintf("%s == @user.partner", partnerForm$id),
    edit_record = sprintf("%s == @user.partner", partnerForm$id), discover =
TRUE,export_records = TRUE))
```

Define reporting partner role

```
reportingPartnerRole <- role(id = "rp", label = "Reporting Partner",
  parameters = list(partnerParameter),
  grants = list(reportGrant,
    grant(resourceId = dbTree$databaseId, permissions = resourcePermissions(view =
TRUE))))
addRole(dbTree$databaseId, reportingPartnerRole)
```

Add users with roles

```
partnerAId <- partnerTbl |> filter(`Partner name` == "Partner A") |> pull(`_id`)
addDatabaseUser(
  databaseId = dbTree$databaseId,
  email = "user.a@example.com",
  name = "User A",
  roleId = "rp",
  roleParameters = list(partner = partnerAId)
)
```

Getting records

Getting records

getRecords()

getRecords() is a user-friendly and tidyverse compatible replacement for queryTable()

Use collect() to download to a data frame.

```
# the base pipe |> is available from R4.1.
# Otherwise use the maggitr pipe %>%.
records_df <-
  getRecords("ceam1x8kq6ikcujg") |>
  select(ends_with("Name")) |>
  collect()

records_df
```

```
# ActivityInfo tibble: Remote form: Projects (ceam1x8kq6ikcujg)
# A tibble:           134 x 5
   `Organization Name`                `Sector Name` `Sub-sector Name`                        `Admin 1 Name` `Admin 2 Name`
   <chr>                              <chr>         <chr>                                    <chr>          <chr>
 1 The Fred Hollows Foundation        Health        Basic Health Care                        Shan (South)   Pa-O Self-Adminis~
 2 United Nations Childrens Fund      Education     Quality Basic Education/Formal Education Chin           Falam
 3 American Refugee Committee        Health        Malaria Programme                        Tanintharyi    Dawei
 4 Pact Global Microfinance Fund      Livelihoods   Microfinance                             Yangon         Yangon (South)
 5 Pact Global Microfinance Fund      Livelihoods   Microfinance                             Sagaing        Shwebo
 6 Karuna Mission Social Solidarity   Education     Quality Basic Education/Formal Education Kayah          Loikaw
 7 Proximity Designs                  Agriculture   Irrigation Water Resources               Chin           Hakha
 8 Pact Global Microfinance Fund      Livelihoods   Microfinance                             Ayeyarwady     Hinthada
 9 World Concern Myanmar              Nutrition     Monitoring Breast Milk Substitutes       Kayin          Hpa-An
10 Pact Global Microfinance Fund      Livelihoods   Microfinance                             Rakhine        Thandwe
# i 124 more rows
# i Use `print(n = ...)` to see more rows
```

Columns as requested

ActivityInfo

Getting records

Manipulate the data frame as usual after collect()

```
records_df |>
  filter(`Sub-sector
Name`=="Nutrition") |>
  arrange(`Organization Name`,
`Admin 1 Name`, `Admin 2 Name`)
|>
  slice_head(n=2)
```

```
# ActivityInfo tibble: Remote form: Projects (ceam1x8kq6ikcujg)
# A tibble:            2 x 5
  `Organization Name`          `Sector Name` `Sub-sector Name`    `Admin 1 Name` `Admin 2 Name`
  <chr>                        <chr>         <chr>                <chr>          <chr>
1 Save the Children in Myanmar Nutrition     IEC on Infant and ~ Rakhine        Sittwe
2 World Concern Myanmar        Nutrition     Monitoring Breast ~ Kayin          Hpa-An
```

Getting records

Filter large data sets *before* downloading

It is possible to use some filters and limit the records before they are downloaded (with collect()).

```
# Taking away collect() results in a reference to the server instead of a data frame. information is displayed about the query being prepared.
getRecords("ceam1x8kq6ikcujg") |>
  select(ends_with("Name")) |>
  arrange(`Organization Name`) |>
  filter(`Sector Name`=="Nutrition") |>
  slice_head(n=2)
```

```
Adding filter: (c3g7i69kq6jst8k3z.caxmhjxkq6jqe373c == "Nutrition")
# Form (id):          Projects (ceam1x8kq6ikcujg)
# Total form records: 134
# Table fields types: c(`Organization Name` = "Text", `Sector Name` = "Text", `Sub-sector Name` = "Text", `Admin 1 Name` = "Text", `Admin 2 Name` = "Text")
# Table filter:       ((c3g7i69kq6jst8k3z.caxmhjxkq6jqe373c == "Nutrition"))
# Table sort:         list(list(dir = "ASC", field = "Organization Name"))
# Table Window:       offSet: 0; Limit: 2
  `Organization Name`           `Sector Name` `Sub-sector Name`               `Admin 1 Name` `Admin 2 Name`
  <chr>                         <chr>         <chr>                          <chr>          <chr>
1 Save the Children in Myanmar  Nutrition     IEC on Infant and Child feeding Rakhine        Sittwe
2 World Concern Myanmar         Nutrition     Nutrition Assessment with W/H   Shan (North)   Lashio
```

ActivityInfo

Limitations before collect()

Only select(), filter(), arrange(), slice_head(), and slice_tail() can be used before fetching records.

You must use the verbs in order: 1. arrange() (limited to a single column) and/or dplyr::filter() in any combination. 2. slice_head(), slice_tail() or adjustWindow(x, offSet = 0L, limit) in any combination 3. **Always end with collect()**

More columns/different styles

Columns from getRecords() are as in the web UI by default with the addition of record id columns but can be modified with helper functions and the style argument:

- getRecords(x, style = prettyColumnStyle()) : the default style
- minimalColumnStyle() : removes all ID columns not found in the web UI.

Getting records

Adding reference columns

Using styles, it is possible to include more columns from referenced tables to include regional codes from *Admin 1* and *Admin 2*:

```
getRecords("ceam1x8kq6ikcujg",
           style =
prettyColumnStyle(allReferenceFields
= TRUE)) |>
    select(ends_with("Name"),
ends_with("CODE")) |>
    arrange(`Organization Name`)  |>
    filter(`Sector Name`=="Nutrition") |>
    slice_head(n=2) |> collect()
```

```
# ActivityInfo tibble: Remote form: Projects (ceam1x8kq6ikcujg)
# A tibble:           2 x 7
  `Organization Name`        `Sector Name` `Sub-sector Name` `Admin 1 Name` `Admin 2 Name` `Admin 1 P-CODE` `Admin 2 P-CODE`
  <chr>                      <chr>         <chr>             <chr>          <chr>          <chr>            <chr>
1 Save the Children in Myan~ Nutrition     IEC on Infant an~ Rakhine        Sittwe         MMR012           MMR012D001
2 World Concern Myanmar      Nutrition     Nutrition Assess~ Shan (North)   Lashio         MMR015           MMR015D001
```

Summary of getRecords()

- Use the record id or a form tree to getRecords() and then select() columns to select and rename them. Use column styles for fine control.
- Always end with collect() to continue analysis on a data frame.
- If you are managing very large data sets and want to reduce download time, use filter() before collect().

Manipulating an existing ActivityInfo forms

Manipulate existing elements and upload form

```
fmSchema <- getFormSchema(surveySchema$id)
fmSchema$elements <- fmSchema$elements[c(1,3,2)]
fmSchema |>
  deleteFormField(code = "age")
addForm(fmSchema)
```

Using getRecords() to copy form fields to a new form

```
getRecords(surveyForm) |>
  extractSchemaFromFields(
    databaseId, "Copied form", useColumnNames = TRUE
    ) |>
  addForm()
```

ActivityInfo

Questions?

Follow us:

LinkedIn page: https://www.linkedin.com/showcase/activityinfo/
LinkedIn group: https://www.linkedin.com/groups/5098257/