Package 'acumos'

October 15, 2020

Title 'Acumos' R Interface

Version 0.4-1

Description Create, upload and run 'Acumos' R models.
'Acumos' (<https://www.acumos.org>) is a platform and open source framework intended to make it easy to build,
share, and deploy AI apps. 'Acumos' is part of the 'LF AI Foundation', an umbrella organization within The Linux Foundation. With this package, user can create a component, and push it to an 'Acumos' platform.

License Apache License (== 2.0)

Imports httr, jsonlite, RProtoBuf, Rserve, RestRserve, yaml

Suggests randomForest, testthat, callr

BugReports https://jira.acumos.org/projects/ACUMOS/issues

URL https://www.acumos.org (Acumos project website)

https://gerrit.acumos.org/r/gitweb?p=acumos-r-client.git (code repository) https://github.com/acumos/acumos-r-client (mirror repository)

https://docs.acumos.org/en/latest/submodules/acumos-r-client/docs (docs)

RoxygenNote 7.0.2

NeedsCompilation no

Author Simon Urbanek [aut],
Alassane Samba [aut, cre],
AT&T [cph],
Tech Mahindra [cph],
Orange [cph]

Maintainer Alassane Samba <alassane.samba@orange.com>

Repository CRAN

Date/Publication 2020-10-15 12:00:02 UTC

R topics documented:

`compose` *Compose a Acumos microservice*

Description

`compose` generates everything necessary to create a Acumos microservice.

Usage

```
compose(predict, transform, fit, generate, service, initialize,
        aux = list(), name = "R Component", componentVersion="unknown version",
        file = "component.amc")
```

Arguments

`predict`	predict function (optional)
`transform`	transform function (optional)
`fit`	fit function (optional)
`generate`	generate function (optional)
`service`	function handling additional non-Acumos requests (optional)
`initialize`	function for any one-shot initializations of the environment
`aux`	list of any auxiliary objects that are to be passed to the global workspace of the component.
`name`	string, name of the component
`componentVersion`	string, version of the component
`file`	string, filename for the resulting component file

Details

A regular component will have at least one of the three functions `predict`, `transform` or `fit` set in which case those functions are exposed via the Acumos API.

A special component can instead provide the `generate` function only in which case the `generate` function is called upon instantiation instead of serving the Acumos API. This is useful when adapting from other inputs than Acumos since the `generate` function can use arbitrary input methods and

then use Acumos API to push to other Acumos components. Similarly, non-Acumos requests can be served using the `service` call-back function with the signature `function(path,query,body,headers)` where `body` is `NULL` or a raw vector. This interface is experimental and currently not part of the official API.

The functions can have two special arguments `inputs` and `output` which are used to define the types of the input and output data. They have to be named string vectors where the names will match formats of the functions and the string specifies the input type (class). At this point only `"character"`, `"integer"`, `"numeric"` and `"raw"` are supported. If those arguments are not present, they default to `c(x="character")`. If the result of the function is a list, it is assumed that the list holds the outputs, otherwise only one output is used.

The `compose()` function is called mainly for its side-effect of creating the Acumos API component file, at this point it is a (ZIP) bundle of `meta.json` (metadata), `component.bin` (serialized R functions and code) and `component.proto` (I/O definition)

Value

Structure describing the component (parsed content of the JSON description).

Author(s)

Simon Urbanek

See Also

`run`

Examples

```
## pass-through component
compose(transform=identity, name="identity")

## simple addition
compose(transform=function(a, b, inputs=c(a="numeric", b="numeric"),
        outputs=c(x="numeric")) a + b, name="Addition")

## silly RF trained on Iris
library(randomForest)
compose(predict=function(..., inputs=lapply(iris[-5], class)){
                    as.character(predict(rf, as.data.frame(list(...))))
                },
        aux = list(rf = randomForest(Species ~ ., data=iris)),
        name="Random Forest")
file.remove("component.amc")
```

composeFromSource *Compose a Acumos microservice from its source file*

Description

composeFromSource generates everything necessary to create a Acumos microservice directly from a specifically written R file, representing the component source.

Usage

```
composeFromSource(
  file = "acumos.R",
  name = "R Component",
  componentVersion = "unknown version",
  outputfile = "component.zip",
  addSource = TRUE
)
```

Arguments

file	string, name of component source file (an R script)
name	string, name of the component
componentVersion	string, version of the component
outputfile	string, filename for the resulting component file
addSource	boolean, to add source file to the (ZIP) bundle or not

Value

Structure describing the component (parsed content of the JSON description).

Note

A regular component source file is an R script in which at least one of the three following functions are defined: acumos_predict, acumos_transform or acumos_fit. They respectively correspond to the functions predict, transform and fit of compose. In that script, if the functions acumos_generate, acumos_service or acumos_initialize are defined, they will also correspond to the other function type arguments of compose, namely generate, service and initialize.

All the R objects defined in that script are included as auxiliary objects that are to be passed to the global workspace of the component. They will fill the aux argument of compose.

Author(s)

Alassane Samba

See Also

compose

Examples

```
# see an example source file in:
print(system.file("examples","example_0/", package = "acumos"))
# compose from acumos.R
example_source<-system.file("examples","example_0","acumos.R", package = "acumos")
composeFromSource(
  file=example_source,
  outputfile = "acumos_bundle_example_0.zip"
)
file.remove("acumos_bundle_example_0.zip")
```

---

push *Push a component into the Acumos repository*

Description

push pushes a component into the Acumos repository.

auth obtains an authentication token to be used with push where required.

Usage

```
push(url, file = "component.amc", token, create=TRUE, license,
     headers, ...)
auth(url, user, password)
```

Arguments

url	URL for the POST request
file	component bundle file as created by compose
token	token obtained from auth (optional)
create	logical, isCreateMicroservice parameter, see Acumos onboarding documentation
license	optional string, name of a file to supply as the license. If not specified push() will also try to locate a license.json file in the component bundle if present.
headers	optional, named list or named character vector of HTTP headers that are to be added to the request. NOTE: the meaning of optional headers depends on the onboarding server so consult the documentation of the onboarding server for supported additional headers and their meaning.
user	user name to use for authentication
password	password to use for authentication
...	any additional form entries to push as body content. If the entry is to be passed as a file upload, use upload_file(<file>,<mime-type>).

Value

push: invisibly, result of the request (may change in the future)

auth: authentication token

Author(s)

Simon Urbanek

See Also

compose

---

pushFromSource *Push a component into the Acumos repository from its source file*

Description

push pushes a component into the Acumos repository using the component source file (R file).

Usage

```
pushFromSource(
  url,
  file,
  name = "R Component",
  addSource = TRUE,
  token,
  create = TRUE,
  license,
  headers,
  ...
)
```

Arguments

url	URL for the POST request
file	string, name of component source file (an R script)
name	string, name of the component
addSource	boolean, to add source file to the (ZIP) bundle or not
token	token obtained from auth (optional)
create	logical, isCreateMicroservice parameter, see Acumos onboarding documentation
license	optional string, name of a file to supply as the license. If not specified push() will also try to locate a license.json file in the component bundle if present.

headers	optional, named list or named character vector of HTTP headers that are to be added to the request. NOTE: the meaning of optional headers depends on the onboarding server so consult the documentation of the onboarding server for supported additional headers and their meaning.
...	optional, named list or named character vector of HTTP headers that are to be added to the request. NOTE: the meaning of optional headers depends on the onboarding server so consult the documentation of the onboarding server for supported additional headers and their meaning.

Value

invisibly, result of the request (may change in the future)

Note

A regular component source file is an R script in which at least one of the three following functions are defined: `acumos_predict`, `acumos_transform` or `acumos_fit`. They respectively correspond to the functions `predict`, `transform` and `fit` of `compose`. In that script, if the functions `acumos_generate`, `acumos_service` or `acumos_initialize` are defined, they will also correspond to the other function type arguments of `compose`, namely `generate`, `service` and `initialize`.

All the R objects defined in that script are included as auxiliary objects that are to be passed to the global workspace of the component. They will fill the `aux` argument of `compose`.

See Also

`push`

---

run *Run-time tools for Acumos*

Description

The following functions are not exported and not intended to be used by users, but are useful for Acumos developers, testing and used by the platform itself.

Usage

```
run(where = getwd(), file = "component.amc", runtime = "runtime.json",
    init.only = FALSE)
data2msg(data, output)
msg2data(msg, input)
send.msg(url, payload, response=FALSE)
```

Arguments

where	directory in which the component will be run
file	path to the model component file (as created by compose()) or a directory containing the unpacked content of the component.
payload	raw vector, message to send - typically constructed by data2msg
runtime	either path to the runtime JSON file or a structure corresponding to the parsed payload of the file
init.only	logical, if TRUE then the runtime is setup but the actual server/generator is not executed.
data	data to wrap into a message - it is expected to be a list (hence a data frame qualifies)
output	name of the proto message type to use
msg	raw vector containing the message
input	name of the proto message type to use
url	string, URL to send the message to
response	logical, if TRUE then the HTTP response object is returned, otherwise only a logical denoting success or failure.

Details

run loads and runs the component by providing a service endpoint on localhost and post specified by the input_port property in the runtime object.

The component file is expected to be created by the compose function. runtime defines the run-time properties such as input port and outputs.

data2msg performs the conversion of native types to the binary message for communication with other Acumos components.

msg2data converts a Acumos message to data according to the provided schema.

Value

The return value of run is undefined since it is executed for its side-effect of providing the service and may never return.

data2msg returns a raw vector constituting the message

msg2data returns the data represented by the message

send.msg returns TRUE on success, FALSE otherwise unless response is TRUE in which case the full response object is returned.

Note

The .proto file containing the schema definitions must be loaded before data2msg() and msg2data() are used, e.g., by the virtue of being inside the run() function or explicitly using RProtoBuf::readProtoFiles(proto).

The run() function can be used only once in a session, because protobuf definitions are global and cannot be overridden without a conflict.

The internal Acumos communication protocol is subject to change and all of the above functions are hidden.

If the `runtime` list contains an entry `data_response=TRUE` then the component also returns the result in the response body (in addition to any output URL specifications). In all other cases the response is always `"OK"` on success and error string on error.

Author(s)

Simon Urbanek

See Also

`compose`

Index