

1.1 Описание заданий и задач на языке JSON

Задания и задачи, направляемые в ГридННС, должны быть описаны на языке JSON (<http://json.org/json-ru.html>). Отметим, что во многих случаях пользователь может обойтись даже без знания того как конкретно это делается благодаря наличию интуитивно понятного графического веб-интерфейса ГридННС. Однако, для полноценного использования ресурсов ГридННС весьма желательно понимания и умения самостоятельно создавать описания заданий и задач.

В этом разделе после очень краткого определения собственно языка JSON, мы даем неформальное и простое введение в язык описания заданий/задач, направляемых в ГридННС.

1.1.1 Язык JSON

JSON (англ. JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript и обычно используемый именно с этим языком. Как и многие другие текстовые форматы, JSON легко читается людьми.

Для понимания дальнейшего достаточно знать, что JSON строится на двух структурах:

1. Набор пар имя/значение.

- В JSON такой набор называется объектом. Точнее объект это неупорядоченное множество пар имя/значение, заключенное в фигурные скобки { }. Между именем и значением стоит символ «:», а пары имя/значение разделяются запятыми.
- Заметим, что аналогичную структуру данных содержит практически любой язык программирования, хотя она может называться иначе,

например: запись, структура, словарь, хэш-таблица, список с ключом или ассоциативный массив.

2. Пронумерованный набор значений.

- Массив (одномерный) — это множество значений, имеющих порядковые номера (индексы). Массив заключается в квадратные скобки []. Значения отделяются запятыми.
- Во многих языках это реализовано как массив, вектор, список или последовательность.

Значение может быть:

- строкой (упорядоченное множество из нуля или более символов юникода, заключенное в двойные кавычки, с использованием escape-последовательностей начинающихся с обратной косой черты (backslash)),
- числом (похоже на C или Java-число, за исключением того, что восьмеричные и шестнадцатеричные форматы не используются),
- булевым значением (true, false)
- null,
- объектом,
- массивом.

Эти структуры могут быть вложены друг в друга.

Следующий пример показывает JSON-представление объекта, описывающего человека. В объекте есть строковые поля имени и фамилии, объект, описывающий адрес, и массив, содержащий список телефонов.

```
{
```

```
"firstName": "Иван",
"lastName": "Иванов",
"address": {
  "streetAddress": "Московское ш., 101, кв.101",
  "city": "Ленинград",
  "postalCode": 101101
},
"phoneNumbers": [
  "812 123-1234",
  "916 123-4567"
]
}
```

1.1.2 Описание заданий на языке JSON

Формальное и строгое определение формата описания заданий и задач на языке JSON представлено с помощью языка JSON Schema (<http://www.json-schema.org>, <http://json-schema.org/schema>), см. Приложение ?. Как любой строгий и формальный язык, JSON Schema требует некоторых усилий и времени для того, чтобы изучить и привыкнуть к нему. Поэтому здесь мы даем простое, менее формальное введение в язык описания заданий и задач, которое тем не менее вполне достаточно для их подготовки и запуска в ГридННС.

Прежде всего, описание задания это объект в терминологии JSON. Поэтому, в соответствии с предыдущим разделом все описание должно быть заключено фигурные скобки:

```
{
...
}
```

В нашем неформальном введении в язык описания многоточие обозначает любой набор элементов описания (в выше приведенном выражении это все описание задания).

В объекте описания задания (то есть, в наборе пар имя/значение, описывающих задание) есть всего два **обязательных** элемента:

- пара с именем "version" и целочисленным значением, равным номеру используемой версии схемы описания задания (это нужно для ППО, которое обрабатывает описание задания в процессе его запуска в грид);
- пара с именем "tasks" и массивом в качестве значения (набор элементов в квадратных скобках); каждый элемент этого массива представляет собой описание одной из задач, входящих в состав задания.

Таким образом, общий вид описания задания является таким:

```
{  
  "version": N,  
  "tasks": [...]  
  ...  
}
```

В реальном описании вместо N надо подставить номер версии описания, например, 1, 2, 3, В дальнейшем мы будем использовать N=2 (текущая версия на момент написания данного Руководства). Многоточия здесь обозначают элементы массива описаний задач и необязательные элементы описания задания. Задание должно включать в себя хотя бы одну задачу, поэтому массив описания задач должен содержать один или более элементов.

Описание задачи (элемент массива с именем "tasks") является объектом (набором пар имя/значение) и содержит по крайней мере две пары:

- пару: имя - "id", значение - строка, задающая идентификатор задачи (уникальный в рамках данного задания);
- вторая пара должна быть одной из следующих:
 - либо пара: имя - "definition", значение - опять объект, который

представляет собой собственно описание задачи и структура которого определяется отдельной схемой (см. следующий раздел Руководства);

- либо пара: имя - "filename", значение - строка, которая является именем отдельного файла с описанием задачи (см. следующий раздел Руководства).

Таким образом обязательная структура описания задания имеет следующий общий вид:

```
{
  "version": 2,
  "tasks": [
    {"id": "a",
     "definition": {...},
     ...},
    {"id": "b",
     "filename": "b.js",
     ...},
    {...}
  ]
  ...
}
```

В этом примере строки "a" и "b" в паре с именем "id" могут быть заменены на любые - по выбору пользователя. Они нужны для того, чтобы можно было ссылаться на данную задачу внутри задания. В задаче с идентификатором "a" используется вариант, когда описание задачи приводится непосредственно в файле описания задания (пара с именем "definition"). В задаче с идентификатором "b" используется вариант, когда описание задачи приводится в отдельном файле (пара с именем "filename", название файла может быть любым).

Остальные элементы описания вообще говоря не являются обязательными, но могут быть важными и даже необходимыми для описания того или иного задания.

Они перечислены ниже.

- Пара с именем "description" и строковым значением. Фактически это возможность включить произвольный комментарий в описание задания и задач. Выглядит это так:

```
{  
  "version": 2,  
  "description": "произвольный комментарий к описанию задания",  
  "tasks": [  
    {"id": "a",  
     "description": "произвольный комментарий к описанию задачи a",  
     "definition": {...},  
     ...},  
    {"id": "b",  
     "description": "произвольный комментарий к описанию задачи b",  
     "filename": "b.js",  
     ...},  
    {...}  
  ]  
  ...  
}
```

- Пара с именем "default_storage_base" и строковым значением. Она определяет базовую часть URL для файлов с данными, которые будут использоваться при выполнении задач задания. Благодаря этому, в описании задач ссылки на файлы с данными могут быть короткими (если файлы для задания лежат в одном хранилище - с указанным базовым URL). Теперь описание выглядит так:

```
{  
  "version": 2,  
  "description": "произвольный комментарий к описанию задания",  
  "default_storage_base": "gsiftp://tb01.ngrid.ru/home/ivanov/job1/",
```

```

"tasks": [
  {"id": "a",
   "description": "произвольный комментарий к описанию задачи a",
   "definition": {...},
   ...},
  {"id": "b",
   "description": "произвольный комментарий к описанию задачи b",
   "filename": "b.js",
   ...},
  {...}
]
...
}

```

- Пара с именем "requirements" и массивом в качестве значения. Каждый элемент этого массива является объектом. Вообще говоря, эти объекты должны выражать требования всего задания к ресурсам, на которых могут выполняться его задачи. Однако в настоящее время эти требования ограничены единственным типом: пользователь может явно перечислить URL вычислительных элементов ГридННС, на которые пользователь разрешает посылать задачи задания. При наличии этого элемента описание выглядит следующим образом:

```

{
  "version": 2,
  "description": "произвольный комментарий к описанию задания",
  "default_storage_base": "gsiftp://tb01.ngrid.ru/home/ivanov/job1/",
  "tasks": [
    {"id": "a",
     "description": "произвольный комментарий к описанию задачи a",
     "definition": {...},
     ...},
    {"id": "b",
     "description": "произвольный комментарий к описанию задачи b",

```

```

        "filename": "b.js",
        ...},
        {...}
    ],
    "requirements": [
        { "hostname": ["tb01.ngrid.ru", "tb10.ngrid.ru"] }
    ]
}

```

Как видно, пара с именем "requirements" имеет в качестве значения массив с одним элементом-объектом, который, в свою очередь, содержит одну пару с именем "hostname" и массивом в качестве значения. Этот массив может содержать один или более строковых элементов, каждый из которых является URL-ом вычислительного элемента ГридННС. Если пара "requirements" отсутствует, описание задания допускает любые ресурсы.

- Каждый элемент (объект) массива "tasks" может содержать указания на взаимозависимость задач задания. Для этого в эти элементы добавляется пара с именем "children" и массивом в качестве значения, элементами которого являются идентификаторы задач, выполнение которых непосредственно зависит от выполнения данной задачи. Так если выполнение задачи "b" в нашем описании зависит от результатов выполнения задачи "a", то в описании это указывается следующим образом:

```

{
  "version": 2,
  "description": "произвольный комментарий к описанию задания",
  "default_storage_base": "gsiftp://tb01.ngrid.ru/home/ivanov/job1/",
  "tasks": [
    { "id": "a",
      "description": "произвольный комментарий к описанию задачи a",
      "children": ["b"],

```

```

    "definition": {...}
  },
  {"id": "b",
   "description": "произвольный комментарий к описанию задачи b",
   "filename": "b.js"
  }
],
"requirements": [
  { "hostname": ["tb01.ngrid.ru", "tb10.ngrid.ru"] }
]
}

```

Конечно, задач в задании может быть больше двух и зависимости могут быть более сложными. Соответствующие пояснения и примеры приведены в последнем разделе.

В приведенном выше примере осталось одно многоточие - содержание объекта - значение пары "definition", другими словами, описание задачи. Формат этого описания представлен в следующем разделе.

Важным общим замечанием является то, что описание заданий и задач *допускает* включение любых других JSON-структур. Системой управления выполнением заданий (СУВЗ) эти дополнительные структуры будут просто игнорироваться. Однако для некоторых целей они могут быть полезны. Например, при создании описания заданий с помощью графического редактора ВИГ в описание могут быть добавлены координаты узлов НАГ в окне редактора. Благодаря этому позднее по такому описанию может быть восстановлено его графическое представление.

1.1.3 Описание задач на языке JSON

Как указано в предыдущем разделе описание задач дается непосредственно в

описании всего задания (пара с именем "definition") или в отдельном файле. В любом случае описание задачи является объектом.

В объекте описания задачи есть два *обязательных* элемента:

- пара с именем "version" и целочисленным значением, равным номеру используемой версии схемы описания задания (аналогично случаю задания);
- пара с именем "executable" и строковым значением, представляющим собой локальный (относительно директории запуска задачи) или абсолютный путь для выполняемого файла.

Таким образом, общий вид описания задачи является таким:

```
{ "version": 2,  
  "executable": "/bin/ls",  
  ...  
}
```

Конечно, вместо исполнимого файла /bin/ls должен быть указан реальный исполнимый файл задачи.

Остальные элементы описания вообще говоря не являются обязательными, но могут быть важными и даже необходимыми для описания того или иного задания. Они перечислены ниже.

- Пара с именем "description" и строковым значением - произвольным комментарием в описании задачи:

```
{ "version": 2,  
  "description": "произвольный комментарий к описанию задачи",  
  "executable": "/bin/ls",  
  ...  
}
```

- Пара с именем "arguments" и массивом в качестве значения, каждый элемент которого является строкой-аргументом для исполнимого файла. Пример:

```
{ "version": 2,
  "description": "произвольный комментарий к описанию задачи",
  "executable": "/bin/ls",
  "arguments": [ "/home", "/etc" ],
  ...
}
```

- Пара с именем "environment" и объектом в качестве значения. Этот элемент описания предназначен для установки дополнительных переменных среды окружения, которое будет выполнено перед запуском задачи. Названиями пар объекта-значения являются требуемые переменных окружения (они будут переведены в верхний регистр), строковые значения задают значения переменных. Пример:

```
{ "version": 2,
  "description": "произвольный комментарий к описанию задачи",
  "executable": "/bin/ls",
  "arguments": [ "/home", "/etc" ],
  "environment": { "path": "/scratch/bin/",
                  "soft_location": "/opt/soft/"
                  },
  ...
}
```

- Пара с именем "count" и целочисленным значением. Значение этого атрибута определяет количество процессоров, необходимых для выполнения задачи, причем если это значение более 1, подразумевается, что задача является MPI-задачей. Любые другие значения или отсутствие атрибута означают, что задача не является MPI-задачей. Пример:

```
{ "version": 2,
  "description": "произвольный комментарий к описанию задачи",
  "executable": "/bin/ls",
```

```

"arguments": [ "/home", "/etc" ],
"environment": { "path": "/scratch/bin/",
                 "soft_location": "/opt/soft/"
               },
"count": 8,
...
}

```

- Пара с именем "default_storage_base" и строковым значением. Как и в случае описания задания (см. предыдущий раздел), она определяет базовую часть URL для файлов с данными, которые будут использоваться при выполнении данной задачи. Если значения этого атрибута в описании задания и задачи различаются, то при выполнении данной задачи используется значение атрибута задачи. Если в описании задачи этого атрибута нет, используется значение из описания задания.

```

{ "version": 2,
  "description": "произвольный комментарий к описанию задачи",
  "executable": "/bin/ls",
  "arguments": [ "/home", "/etc" ],
  "environment": { "path": "/scratch/bin/",
                  "soft_location": "/opt/soft/"
                },
  "count": 8,
  "default_storage_base": "gsiftp://tb01.ngrid.ru/home/ivanov/job1/",
  ...
}

```

- Следующие два необязательных атрибута являются однотипными и мы их опишем одновременно. Эти атрибуты определяют входные и выходные файлы задачи. Соответственно имена пар - "input_files" и "output_files".

Значениями являются объекты, в которых имена пар - названия входных/выходных файлов, а значения задают относительные пути или URLы файлов. Путь указывается относительно параметра `default_storage_base`. Если данный параметр отсутствует в описании задачи, то будет взято значение параметра `default_storage_base` задания. Если данный параметр отсутствует в описании задания, то все файлы для которых указаны относительные пути будут проигнорированы.

```
{ "version": 2,
  "description": "произвольный комментарий к описанию задачи",
  "executable": "/bin/ls",
  "arguments": [ "/home", "/etc" ],
  "environment": { "path": "/scratch/bin/",
                  "soft_location": "/opt/soft/"
                },
  "count": 8,
  "default_storage_base": "gsiftp://tb01.ngrid.ru/home/ivanov/job1/",
  "input_files": {
    "input_file1": "datafile1",
    "input_file2": "gsiftp://tb02.ngrid.ru/home/ivanov/data/datafile2"
  },
  "output_files": {
    "output_file1": "result1",
    "output_file2": "gsiftp://tb02.ngrid.ru/home/ivanov/data/result2"
  },
  ...
}
```

- Следующие три необязательных атрибута также являются однотипными и мы их опишем одновременно. Эти атрибуты определяют файлы для `stdin`, `stdout` и `stderr` задачи. Соответственно имена пар - "`stdin`", "`stdout`" и "`stderr`", а строковые задают относительные пути или URLы файлов по тем же

правилам, что и в случае input/output-файлов.

```
{ "version": 2,
  "description": "произвольный комментарий к описанию задачи",
  "executable": "/bin/ls",
  "arguments": [ "/home", "/etc" ],
  "environment": { "path": "/scratch/bin/",
                   "soft_location": "/opt/soft/"
                 },
  "count": 8,
  "default_storage_base": "gsiftp://tb01.ngrid.ru/home/ivanov/job1/",
  "input_files": {
    "input_file1": "datafile1",
    "input_file2": "gsiftp://tb02.ngrid.ru/home/ivanov/data/datafile2"
  },
  "output_files": {
    "output_file1": "result1",
    "output_file2": "gsiftp://tb02.ngrid.ru/home/ivanov/data/result2"
  },
  "stdin": "task1_stdin",
  "stdout": "gsiftp://tb02.ngrid.ru/home/ivanov/data/task1_stdout",
  "stderr": "gsiftp://tb02.ngrid.ru/home/ivanov/data/task1_stderr",
  ...
}
```

- Следующие два необязательных атрибута определяют название локального менеджера ресурсов (ЛМР), на который выбранный вычислительный элемент будет направлять задачу и название очереди для данного ЛМР. названия атрибутов - "scheduler" и "queue", строковые значения соответствуют названиям ЛМР и очереди:

```
{ "version": 2,
  "description": "произвольный комментарий к описанию задачи",
  "executable": "/bin/ls",
```

```

"arguments": [ "/home", "/etc" ],
"environment": { "path": "/scratch/bin/",
                  "soft_location": "/opt/soft/"
                },
"count": 8,
"default_storage_base": "gsiftp://tb01.ngrid.ru/home/ivanov/job1/",
"input_files": {
  "input_file1": "datafile1",
  "input_file2": "gsiftp://tb02.ngrid.ru/home/ivanov/data/datafile2"
},
"output_files": {
  "output_file1": "result1",
  "output_file2": "gsiftp://tb02.ngrid.ru/home/ivanov/data/result2"
},
"stdin": "task1_stdin",
"stdout": "gsiftp://tb02.ngrid.ru/home/ivanov/data/task1_stdout",
"stderr": "gsiftp://tb02.ngrid.ru/home/ivanov/data/task1_stderr",
"scheduler": "PBS",
"queue": "main",
...
}

```

- Наконец последним необязательным атрибутом является пара с именем "requirements" и массивом в качестве значения, который полностью аналогичен соответствующему атрибуту в описании задания. Каждый элемент этого массива является объектом. Как и в случае всего задания в настоящее время требования ограничены единственным типом: пользователь может явно перечислить URL вычислительных элементов ГридННС, на которые пользователь разрешает посылать данную задачу. Требования задания и требования задачи объединяются операцией логическое И. Другими словами берется пересечение множеств ресурсов определенных в

атрибутов "requirements" задания и задачи. Например, если в "requirements" задания допускаются ресурсы ["a", "b", "c"], а в задаче - ["c", "d", "e"], то для данная задача может быть направлена только на ресурс "c". Отсутствие требований (в задании или задаче) эквивалентно требованию «подходит любой ресурс». Теперь описание со всеми возможными атрибутами выглядит так:

```
{ "version": 2,
  "description": "произвольный комментарий к описанию задачи",
  "executable": "/bin/ls",
  "arguments": [ "/home", "/etc" ],
  "environment": { "path": "/scratch/bin/",
                  "soft_location": "/opt/soft/"
                },
  "count": 8,
  "default_storage_base": "gsiftp://tb01.ngrid.ru/home/ivanov/job1/",
  "input_files": {
    "input_file1": "datafile1",
    "input_file2": "gsiftp://tb02.ngrid.ru/home/ivanov/data/datafile2"
  },
  "output_files": {
    "output_file1": "result1",
    "output_file2": "gsiftp://tb02.ngrid.ru/home/ivanov/data/result2"
  },
  "stdin": "task1_stdin",
  "stdout": "gsiftp://tb02.ngrid.ru/home/ivanov/data/task1_stdout",
  "stderr": "gsiftp://tb02.ngrid.ru/home/ivanov/data/task1_stderr",
  "scheduler": "PBS",
  "queue": "main",
  "requirements": [
    { "hostname": ["tb10.ngrid.ru", "tb12.ngrid.ru"] }
  ]
}
```

Как и в случае заданий, описание задач допускает включение любых других JSON-структур, которые при обработке системой управления выполнением заданий игнорируются (см. замечание в конце предыдущего раздела).

1.1.4 Примеры описаний заданий и задач для выполнения в ГридННС

- Пример 1. Простейшая тестовая задача.

В данном тестовом задании на удаленном ресурсе выполняется команда `whoami`, которая определяет название локальной учетной записи, на которую отображается прокси-сертификат грид-пользователя и, соответственно, от имени которого выполняется задание. Результат записывается в файл `test.txt` на GridFTP-сервере, указанном в атрибуте `default_storage_base`.

```
{ "version": 2,
  "description": "тестовое задание 1",
  "default_storage_base": "gsiftp://tb01.ngrid.ru/home/ivanov/jt/",
  "tasks": [ { "id": "a",
               "description": "задача #1",
               "definition": { "version": 2,
                               "executable": "/usr/bin/whoami",
                               "stdout": "test.txt"
                             }
             }
  ]
}
```

- Пример 2

В данном примере выполняется практически то же задание, что и в Примере 1, но описания задания и задачи разделены.

Описание задания:

```
{ "version": 2,  
  "description": "тестовое задание 2",  
  "tasks": [ { "id": "a",  
               "description": "task whoami",  
               "filename": "task_whoami.js"  
             }  
            ]  
}
```

Файл описания задачи task_whoami.js:

```
{ "version": 2,  
  "description": "testing task 1",  
  "executable": "/usr/bin/whoami",  
  "stdout": "local_user.txt"  
}
```

- Пример 3

Задание в этом примере состоит из четырех связанных между собой задач (то есть результаты части задач используются как входные данные для других). Направленный ациклический граф, описывающий эти связи (поток данных), представлен на рис. 1.

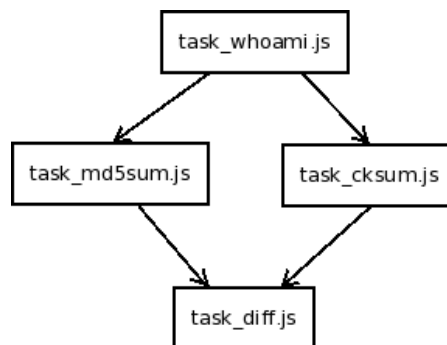


Рис. 1. Направленный ациклический граф, описывающий зависимость задач в задании данного примера

Описание задания:

```
{ "version": 2,
  "default_storage_base": "gsiftp://tb01.ngrid.ru/home/demichev/tst/",
  "tasks": [{
    "id": "n0",
    "description": "a",
    "filename": "task_whoami.js",
    "children": ["n2", "n1"]
  },
  {
    "id": "n1",
    "description": "b",
    "filename": "task_md5sum.js",
    "children": ["n3"]
  },
  {
    "id": "n2",
    "description": "c",
    "filename": "task_cksum.js",
    "children": ["n3"]
  },
  {
    "id": "n3",
    "description": "d",
    "filename": "task_diff.js"
  }
]
}
```

Задача task_whoami.js:

```
{ "version": 2,
  "description": "testing task 1",
  "executable": "/usr/bin/whoami",
  "stdout": "local_user.txt"
}
```

Задача task_md5sum.js

```
{ "version": 2,  
  "description": "md5 sum of the input file",  
  "executable": "/usr/bin/md5sum",  
  "arguments": [ "local_user_b.txt" ],  
  "input_files": { "local_user_b.txt": "local_user.txt"  
                  },  
  "stdout": "md5_result.txt"  
}
```

Задача task_cksum.js:

```
{ "version": 2,  
  "description": "cksum sum of the input file",  
  "executable": "/usr/bin/cksum",  
  "arguments": [ "local_user_c.txt" ],  
  "input_files": { "local_user_c.txt": "local_user.txt"  
                  },  
  "stdout": "cksum_result.txt"  
}
```

Задача task_diff.js:

```
{ "version": 2,  
  "description": "diff of the input files",  
  "executable": "/usr/bin/diff",  
  "arguments": [ "md5_result_d.txt", "cksum_result_d.txt" ],  
  "input_files": { "md5_result_d.txt": "md5_result.txt",  
                  "cksum_result_d.txt": "cksum_result.txt"  
                  },  
  "stdout": "diff_result.txt"  
}
```