

Вступление	2
Установка	3
Запуск скрипта установки через контекстное меню.....	3
Запуск скрипта установки через командную строку	3
Процедура запуска парсинга	6
Правила заполнения семантики парсинга	6
Теория json - структура в виде дерева	6
Типы данных симантики	10
Alias функции	13
Табличные alias функции	13
Json alias функции	16
Анонимные аргументы для alias функций.....	18
Настройка конфига парсинга	21
Настройка конфига для Excel	21
Настройка конфига для GoogleSheets	22

Вступление

ExcelToJsonUniversalParser – является универсальным парсером таблиц в формат данных json. Цель парсера иметь возможность получать json объекты любой сложности и вложенности, без понимания языков программирования. Парсер написан на языке программирования Python, для удобного запуска и установки используется командная строка Windows PowerShell. Для работы парсера необходимо задавать правильную семантику в таблицах источника данных.

Инструкция разделена на разделы по установке, правилам написания семантики, настройки конфигов парсинга. Разделы инструкции тесно связаны друг с другом. Для изучения работы парсера рекомендуется запускать существующий пример (папка Example), приложенный к репозиторию.

Установка

Выкачать репозиторий с <https://github.com/saigor33/ExcelToJsonUniversalParser>.

В выкаченном репозитории найти файл **Install.ps1** и запустить через PowerShell. Исполнение скрипта можно запустить через контекстное меню, либо через командную строку.

Запуск скрипта установки через контекстное меню

Запустить можно через контекстное меню (Правой кнопкой мыши → Run with PowerShell, либо правой кнопкой мыши → Open with → PowerShell).

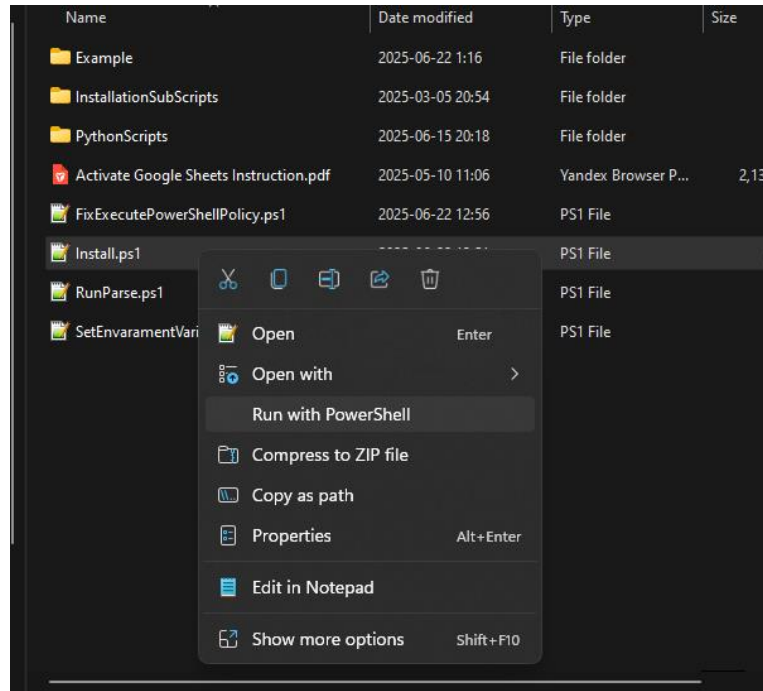


Рис. 1 Запуск скрипта установки через контекстное меню

Запуск скрипта установки через командную строку

Для этого открыть Windows PowerShell (Рис. 2)

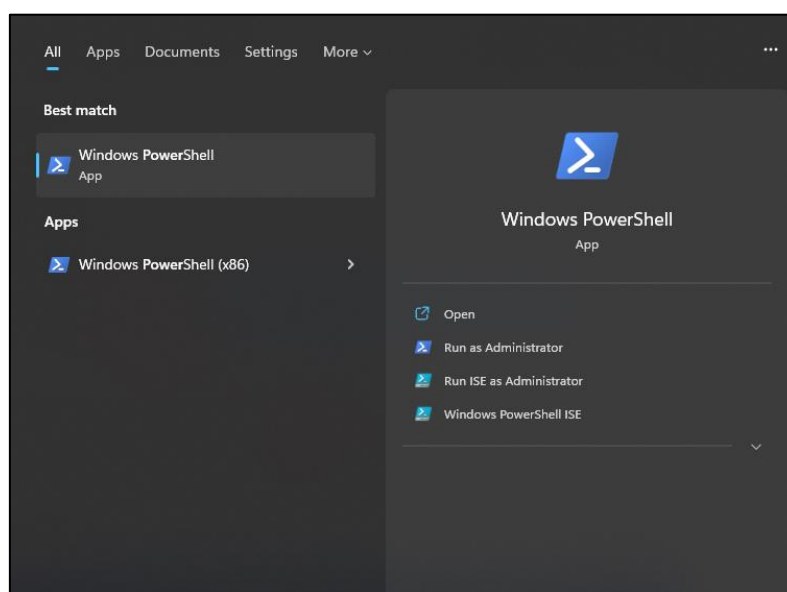
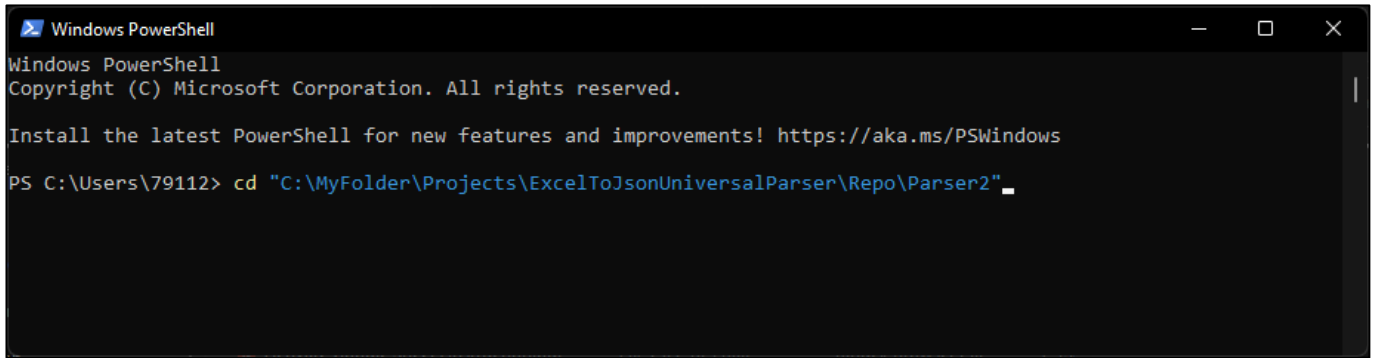


Рис. 2 Запуск PowerShell

В окне командной строки выполнить команду **cd** “<путь к папке>” перехода в папку, где расположен скрипт **Install.ps1**.



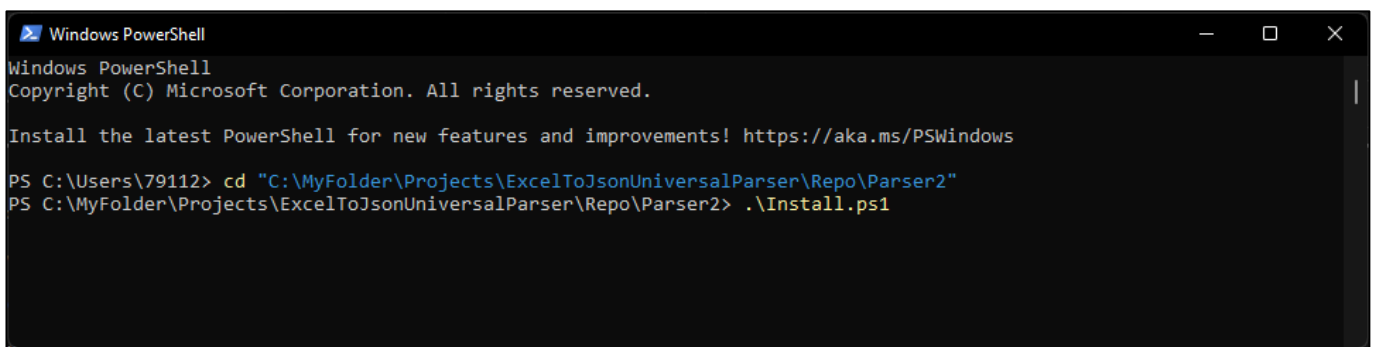
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\79112> cd "C:\MyFolder\Projects\ExcelToJsonUniversalParser\Repo\Parser2"
```

Рис. 3 Выполнение команды перехода в папке репозитория

Выполнить установку командой «**.\Install.ps1**» (Рис. 4).



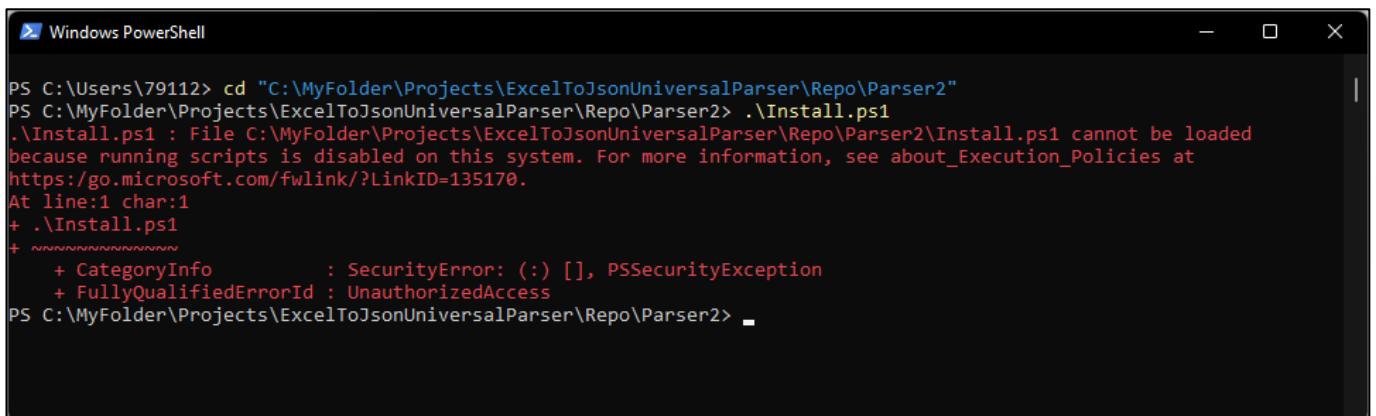
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\79112> cd "C:\MyFolder\Projects\ExcelToJsonUniversalParser\Repo\Parser2"
PS C:\MyFolder\Projects\ExcelToJsonUniversalParser\Repo\Parser2> .\Install.ps1
```

Рис. 4 Выполнение команды установки через PowerShell

Если появилась ошибка «... **.ps1 cannot be loaded because running scripts is disabled on this system.**» (Рис. 5), то необходимо выполнить команду «**Set-ExecutionPolicy Unrestricted -Scope Process**», которая установит политику разрешающую запускать скрипты PowerShell. Данная команда будет действовать в рамках текущей сессии, чтобы выдать разрешение распространяющийся за текущую сессию можно выполнить команду «**Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope CurrentUser**», которая установит политику разрешающую запускать скрипты PowerShell для текущего пользователя.



```
Windows PowerShell

PS C:\Users\79112> cd "C:\MyFolder\Projects\ExcelToJsonUniversalParser\Repo\Parser2"
PS C:\MyFolder\Projects\ExcelToJsonUniversalParser\Repo\Parser2> .\Install.ps1
.\Install.ps1 : File C:\MyFolder\Projects\ExcelToJsonUniversalParser\Repo\Parser2\Install.ps1 cannot be loaded
because running scripts is disabled on this system. For more information, see about_Execution_Policies at
https://go.microsoft.com/fwlink/?LinkID=135170.
At line:1 char:1
+ .\Install.ps1
+ ~~~~~
+ CategoryInfo          : SecurityError: (:) [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
PS C:\MyFolder\Projects\ExcelToJsonUniversalParser\Repo\Parser2> 
```

Рис. 5 Ошибка доступа запуска скрипта PowerShell

```
Windows PowerShell
.\Install.ps1 : File C:\MyFolder\Projects\ExcelToJsonUniversalParser\Repo\Parser2\Install.ps1 cannot be loaded
because running scripts is disabled on this system. For more information, see about_Execution_Policies at
https://go.microsoft.com/fwlink/?LinkID=135170.
At line:1 char:1
+ .\Install.ps1
+ ~~~~~
+ CategoryInfo          : SecurityError: (:) [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
PS C:\MyFolder\Projects\ExcelToJsonUniversalParser\Repo\Parser2> Set-ExecutionPolicy Unrestricted -Scope Process

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic at
https://go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): Y
PS C:\MyFolder\Projects\ExcelToJsonUniversalParser\Repo\Parser2>
```

Рис. 6 Исправление ошибки доступа запуска скриптов PowerShell

Если была получена ошибка, то необходимо повторно запустить скрипт установки «.\Install.ps1». После установки должна появиться надпись «Install finished» (Рис. 7).

```
Windows PowerShell
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\79112\pyenv\pyenv-win\versions\3.9.11\lib\site-pack
ages (from google-auth!=2.24.0,!>=2.25.0,<3.0.0,>=1.32.0->google-api-python-client==2.166.0) (5.5.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\79112\pyenv\pyenv-win\versions\3.9.11\lib\site-packa
ges (from google-auth!=2.24.0,!>=2.25.0,<3.0.0,>=1.32.0->google-api-python-client==2.166.0) (0.4.2)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\79112\pyenv\pyenv-win\versions\3.9.11\lib\site-packages (fro
m google-auth!=2.24.0,!>=2.25.0,<3.0.0,>=1.32.0->google-api-python-client==2.166.0) (4.9)
Requirement already satisfied: pyparsing!=3.0.0,!>=3.0.1,!>=3.0.2,!>=3.0.3,<4,>=2.4.2 in c:\users\79112\pyenv\pyenv-win\
versions\3.9.11\lib\site-packages (from httplib2<1.0.0,>=0.19.0->google-api-python-client==2.166.0) (3.2.3)
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in c:\users\79112\pyenv\pyenv-win\versions\3.9.11\lib\site-packag
es (from pyasn1-modules>=0.2.1->google-auth!=2.24.0,!>=2.25.0,<3.0.0,>=1.32.0->google-api-python-client==2.166.0) (0.6.
1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\79112\pyenv\pyenv-win\versions\3.9.11\lib\site-pa
ckages (from requests<3.0.0,>=2.18.0->google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0,>=1.31.5->google-api-pytho
n-client==2.166.0) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\79112\pyenv\pyenv-win\versions\3.9.11\lib\site-packages (from
requests<3.0.0,>=2.18.0->google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0,>=1.31.5->google-api-python-client==2.
166.0) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\79112\pyenv\pyenv-win\versions\3.9.11\lib\site-packages
 (from requests<3.0.0,>=2.18.0->google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0,>=1.31.5->google-api-python-clie
nt==2.166.0) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\79112\pyenv\pyenv-win\versions\3.9.11\lib\site-packages
 (from requests<3.0.0,>=2.18.0->google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0,>=1.31.5->google-api-python-clie
nt==2.166.0) (2025.1.31)

[notice] A new release of pip is available: 24.0 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
Python module "google-api-python-client" installed

Step 11: Install python module "numpy" v.1.20.3
Requirement already satisfied: numpy==1.20.3 in c:\users\79112\pyenv\pyenv-win\versions\3.9.11\lib\site-packages (1.2
0.3)

[notice] A new release of pip is available: 24.0 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
Python module "numpy" installed

Step 12: Install python module "openpyxl" v.3.1.3
Requirement already satisfied: openpyxl==3.1.3 in c:\users\79112\pyenv\pyenv-win\versions\3.9.11\lib\site-packages (3
.1.3)
Requirement already satisfied: et-xmlfile in c:\users\79112\pyenv\pyenv-win\versions\3.9.11\lib\site-packages (from o
penpyxl==3.1.3) (2.0.0)

[notice] A new release of pip is available: 24.0 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
Python module "openpyxl" installed

3.9.11 (set by C:\Users\79112\pyenv\pyenv-win\version)

Install finished
Press Enter to continue...
```

Рис. 7 Результат выполнения скрипта установки

Установку достаточно выполнить один раз, но стоит следить за обновлениями в репозитории, для перехода на новую версию может потребоваться повторная установка.

Процедура запуска парсинга

Запуск процедуры парсинга осуществляется посредством запуска скрипта «.\RunParse.ps1» (Рис. 8). Путь файла настройки парсинга по умолчанию зашит внутри файла «.\RunParse.ps1», при необходимости путь к этому файлу можно изменить на свой с помощью блокнота. Файлы с результатом парсинга по умолчанию добавляются в папку «../Example/Temp/Output»

```
Windows PowerShell
Press Enter to continue...:
PS C:\MyFolder\Projects\ExcelToJsonUniversalParser\Repo\Parser2> .\RunParse.ps1
Start fix execute PowerShell policy
Fix execute PowerShell policy succeded
Add Envarament variables process
1."C:\Users\79112\.pyenv\pyenv-win\bin;"
2."C:\Users\79112\.pyenv\pyenv-win\shims;"
Envarament variables addedPython version 3.9.11 set

config_file_path= C:\MyFolder\Projects\ExcelToJsonUniversalParser\Repo\Parser2\Example\Instruction\Config.json

Done!
Press Enter to continue...:
```

Рис. 8 Результат запуска парсинга

Правила заполнения семантики парсинга

Для того чтобы скрипт успешно выполнил свою работу необходимо корректно заполнить источник данных со всеми правилами и ограничениями.

Теория json - структура в виде дерева

Json-файл можно представить в виде дерева. На каждом уровне вложенности LayerN могут быть любые поля.

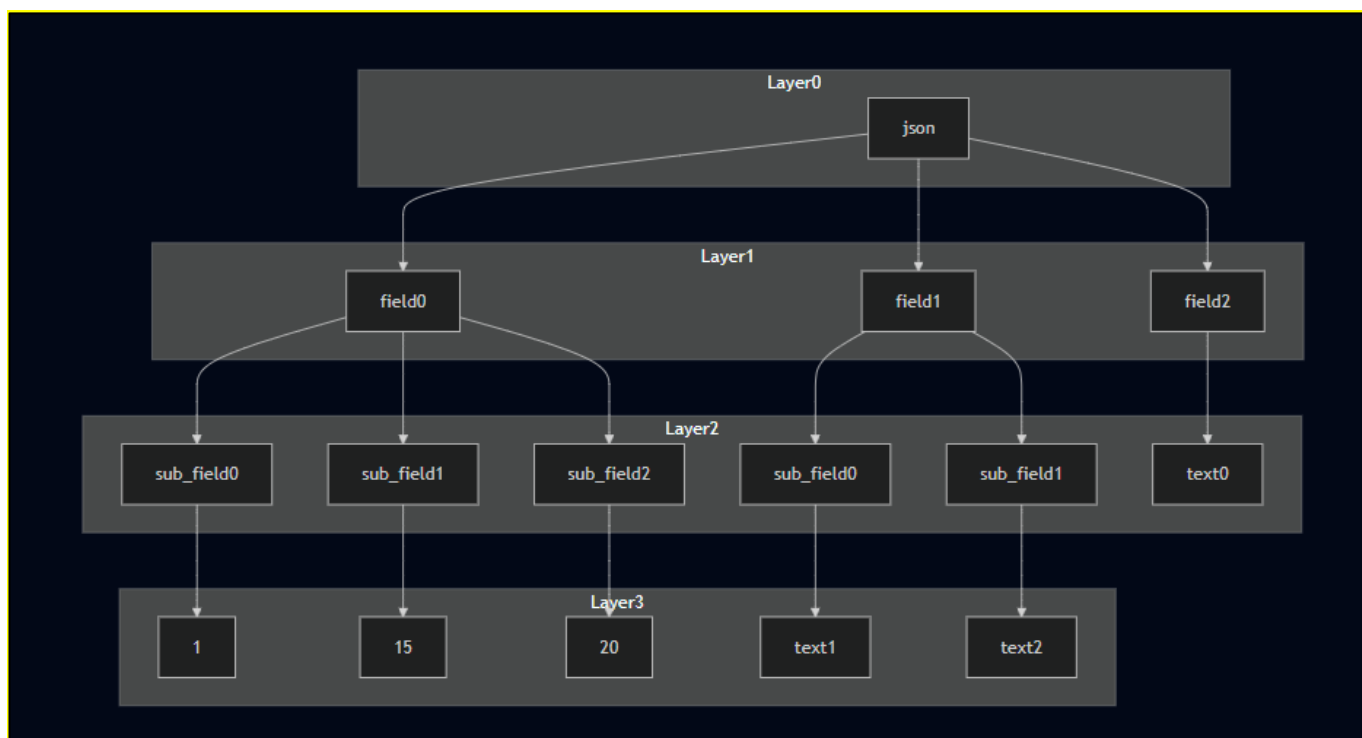
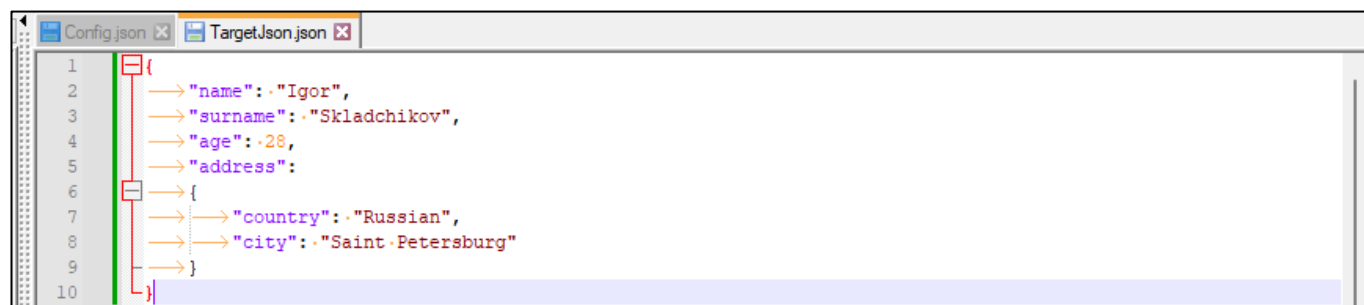


Рис. 9 Json-файл в виде дерева

Для преобразования данных в json необходимо перебрать все ветви деревьев, где в итоге будет получен полный путь с учётом вложенности и значениями.

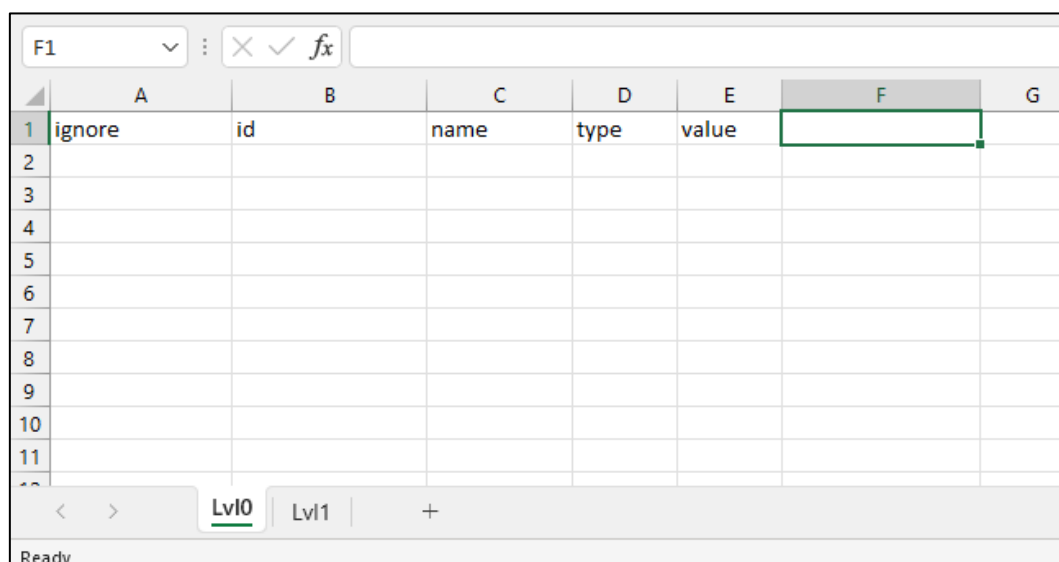
Для примера осуществим парсинг excel таблицы, чтобы получить следующий json файл (Рис. 10).



```
1 {
2   "name": "Igor",
3   "surname": "Skladchikov",
4   "age": 28,
5   "address":
6     {
7       "country": "Russian",
8       "city": "Saint Petersburg"
9     }
10 }
```

Рис. 10 Целевой json файл

В таблице источника данных необходимо задать колонки из которых будет осуществлён парсинг (Рис. 11).



	A	B	C	D	E	F	G
1	ignore	id	name	type	value		
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							

Рис. 11 Настройка колонок парсинга в источнике данных

Описание колонок:

- ignore – используется как флаг для необходимости игнорировать строку во время парсинга.
- Id – используется для связывания вложенных уровней.
- Name – используется для описания поля в json файле
- Type – указывает какой тип данных будет использован в поле json
- Value – значение поля json

Выбранный пример (Рис. 10) имеет 2 уровня Layer1 и Layer2 вложенности (Рис. 12). Каждый уровень вложенности должен быть расположен на отдельном листе источника данных. Для этого будут использоваться листы Lv10 и Lv11 в источнике данных.

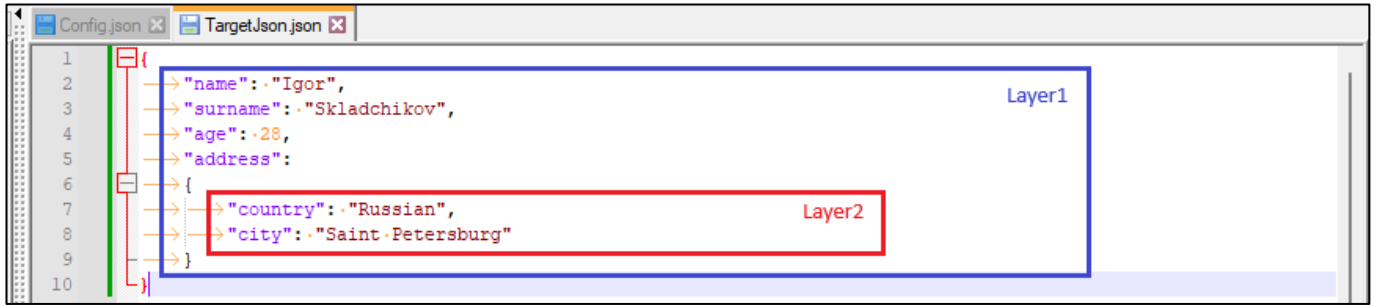


Рис. 12 Уровни целевого json файла

На рисунках 13 и 14 показан пример заполнения семантики для целевого примера.

K21						
	A	B	C	D	E	F
1	ignore	id	name	type	value	
2						
3						
4						
5	TRUE	This row start parsing				
6						
7	TRUE	Example1. Parsing target json				
8		example1	name	str	Igor	
9			surname	str	Skladchikov	
10			age	num	28	
11			address	ref	example1_adress	
12						
13						
14						
15						
16						
17						
<div> <div><</div> <div>></div> <div>Lvl0</div> <div>Lvl1</div> <div>+</div> <div>⋮</div> </div>						

Рис. 13 Example1. Parsing target json. Уровень вложенности 1

L18						
	A	B	C	D	E	F
1	ignore	id	name	type	value	
2						
3						
4						
5	TRUE	This row start parsing				
6						
7	TRUE	Example1. Parsing target json				
8		example1_adress	country	str	Russian	
9			city	str	Saint Petersburg	
10						
11						
12						
13						
14						
15						
16						
17						
<div> <div><</div> <div>></div> <div>Lvl0</div> <div>Lvl1</div> <div>+</div> <div>⋮</div> </div>						

Рис. 14 Example1. Parsing target json. Уровень вложенности 2

Для работы скриптов парсинга необходимо правильно заполнить конфиг парсинга. На рисунке 15 показан пример заполнения конфига для источника данных Excel. Специфичные поля для каждого источника данных будут описаны в разделах «**Настройка конфига для Excel**» и «**Настройка конфига для GoogleSheets**» соответственно. В данном разделе описываются общие настройки для всех источников данных.

В конфиге необходимо настроить **features** и в разделе выбранного источника данных **featuresParsing**.

В featuresParsing перечисляются настройки парсинга:

- **startParsingRowIndex** – строка с которой будет начато считывания данных
- **ignoreColumnName** – колонка с флагом игнорирования строки. Значение в источнике данных должно быть TRUE (игнорировать строку) или FALSE (не игнорировать строку), по умолчанию используется значение false.
- **linkIdColumnName** – колонка в которой указывается название фичи парсинга (если это страница начала парсинга), либо указывается уникальный id являющийся ссылкой на следующий уровень вложенности (пример поле **address**).
- **fieldNameColumnName** – колонка из которой будет браться имя для поля json (пример **name, surname**).
- **fieldValueTypeColumnName** – колонка в значении которой указывается тип данных поля json (**str, num, bool, null, arr, имя alias функции**). Подробнее о типах данных читайте в «Типы данных симантики»
- **fieldValueColumnName** – колонка в значении которой указывается значение для поля в json.
- **orderedByLevelSheetNames** – список листов источника данных, указанный в порядке вложенности уровней json. Кол-во вложенности листов определяется кол-вом уровней итогового json файла. (**Lvl0 → Lvl1 → Lvl2 → LvlN**). Например, поля **address** (Рис. 14) на листе **Lvl0** будет искать значение своего поля на странице **Lvl1** по id «example1_adress».

P.S. Все название колонок **ignore, id, name, type** и т.д. можно переименовать по собственному усмотрению, главное чтобы значение в источнике данных и конфиге совпадало.

В features необходимо перечислить весь список фичей, которые необходимо спарсить. Поле представляет из себя массив, где для каждого элемента необходимо заполнить:

- **featureName** – название фичи по которому в колонке **linkIdColumnName** на первой страницы **Lvl0** будет осуществлён парсинг.
- **outputDirectory** – указание директории куда разместить полученный json файл с именем **outputFileName**. Можно указывать абсолютный или относительный путь.
- **outputFileName** – имя получаемого json файла

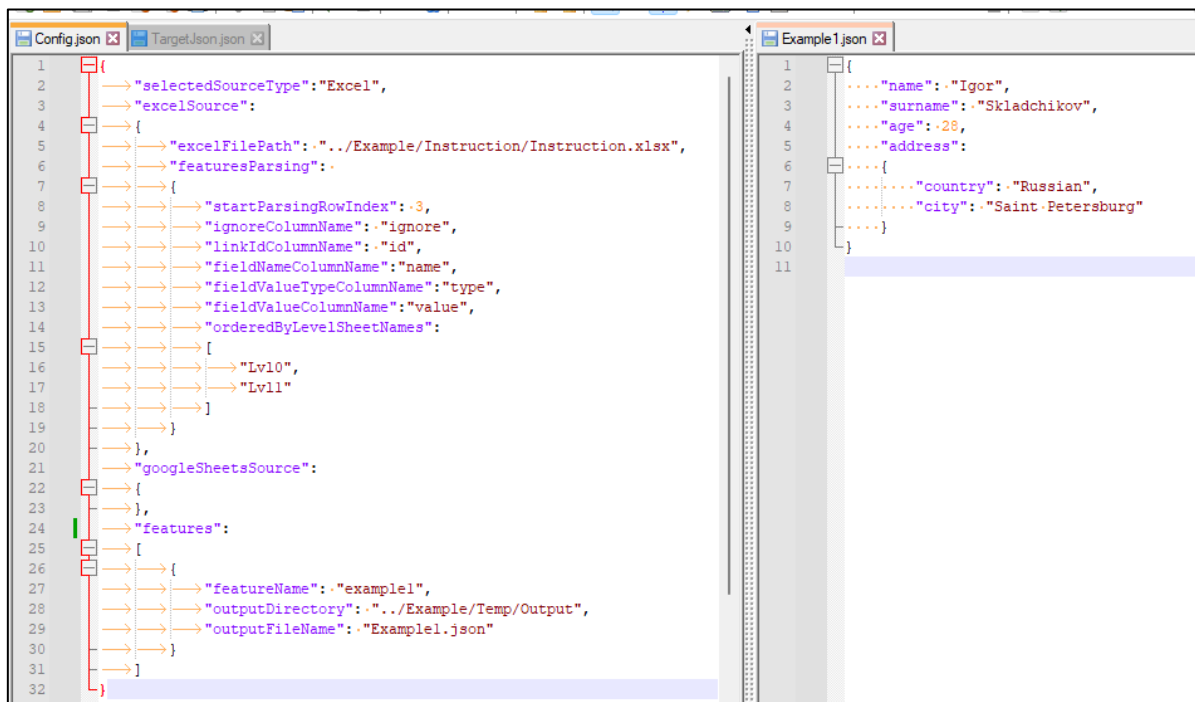


Рис. 15 Example 1. Конфиг парсинга и результат парсинга

Типы данных симантики

Парсер поддерживает следующий типы данных:

- **str** – текстовое поле
- **num** – числовое поле
- **bool** – логическое поле
- **null** – поле со значением null
- **object** – вложенный json объект
- **arr** - массив
- **alias функции** – аббревиатура для повторяющихся элементов, принимающая изменяющиеся данные как параметр (подробнее смотрите раздел «**Alias функции**»).

Тип данных определяет в каком формате данные будут записаны в json файл. Например, строке необходимо добавить кавычки (см поле **stringField** на рис. 16), в отличие от поля с числовым значением (см поле **numberField** на рис 16)

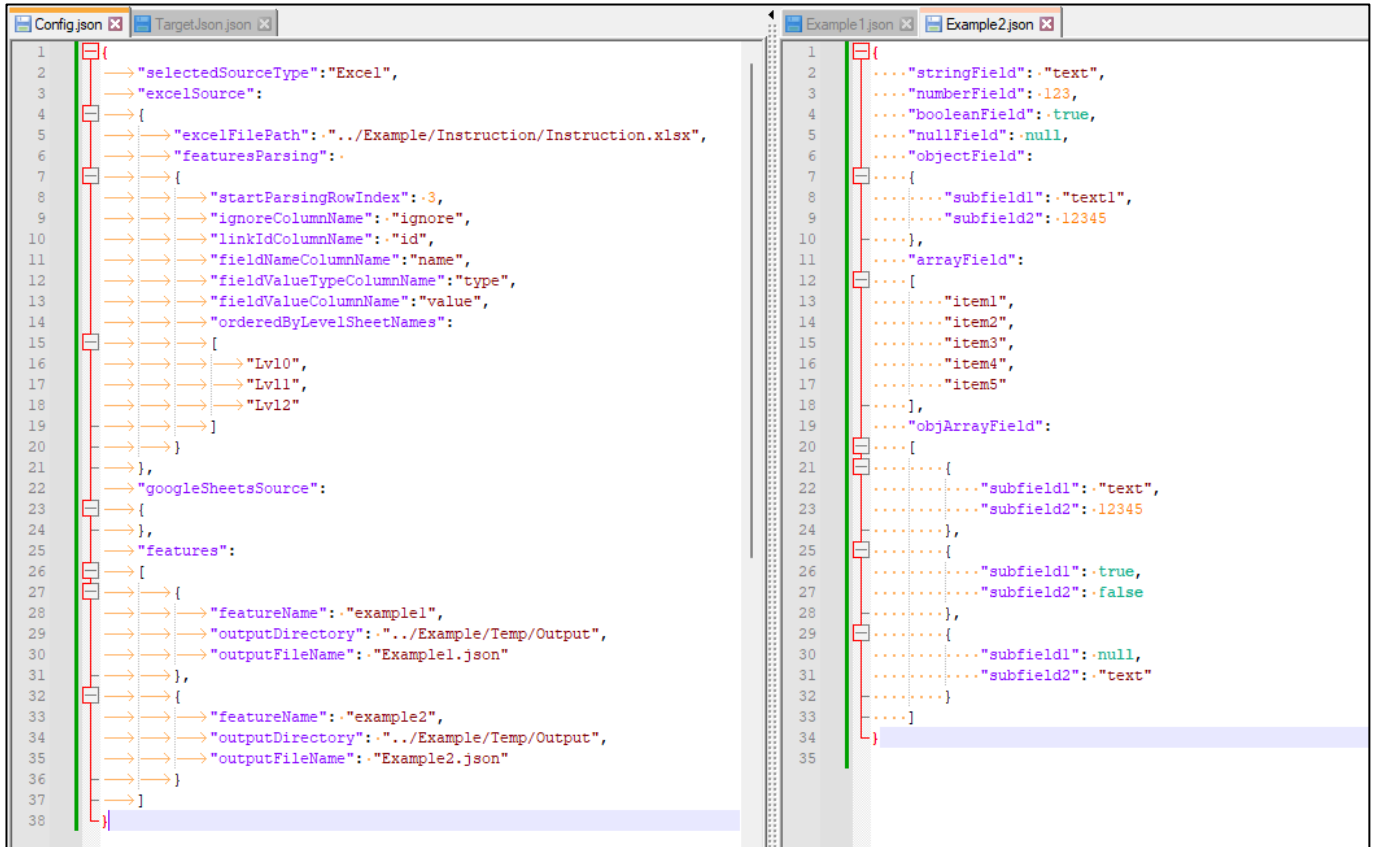


Рис. 16 Example 2. Types variation. Результат парсинга

На рисунке 16 изображены примеры результата парсинга типов данных в различных вариациях. Для примера был добавлен лист **Lvl2** т.к. поле **objArrayField** имеет три уровня вложенности. На рисунка 17-19 показана семантика парсинга для примера 2.

G27		A	B	C	D	E	F
1	2	ignore	id	name	type	value	
2							
3							
4							
5		TRUE	This row start parsing				
6							
7		TRUE	Example1. Parsing target json				
13							
14		TRUE	Example2. Types variation				
15			example2	stringField	str	text	
16				numberField	num	123	
17				booleanField	bool	TRUE	
18				nullField	null		
19				objectField	ref	example2_objField	
20				arrayField	arr	example2_arrField	
21				objArrayField	arr	example2_objArrayField	
22							
23							
24							

Рис. 17 Example 2. Types variation. Семантика Lvl0

A27							
1	2	A	B	C	D	E	F
1		ignore	id	name	type	value	
4							
5		TRUE	This row start parsing				
6							
7		TRUE	Example1. Parsing target json				
11							
12		TRUE	Example2. Types variation				
13			example2_objField	subfield1	str	text1	
14				subfield2	num	12345	
15							
16			example2_arrField		str	item1	
17					str	item2	
18					str	item3	
19					str	item4	
20					str	item5	
21			example2_objArrayField		ref	example2_objArrayField_item1	
22					ref	example2_objArrayField_item2	
23					ref	example2_objArrayField_item3	
24							
25							
26							
27							

Рис. 18 Example 2. Types variation. Семантика Lvl1

C3							
1	2	A	B	C	D	E	F
1		ignore	id	name	type	value	
5		TRUE	This row start parsing				
6							
7		TRUE	Example2. Types variation				
8			example2_objArrayField_item1	subfield1	str	text	
9				subfield2	num	12345	
10							
11			example2_objArrayField_item2	subfield1	bool	TRUE	
12				subfield2	bool	FALSE	
13							
14			example2_objArrayField_item3	subfield1	null		
15				subfield2	str	text	
16							
17							
18							

Рис. 19 Example 2. Types variation. Семантика Lvl2

Alias функции

Alias функции позволяют создать короткую запись для часто повторяемых элементов json.

Предположим нам нужно получить json вида указанного на рисунке 20.

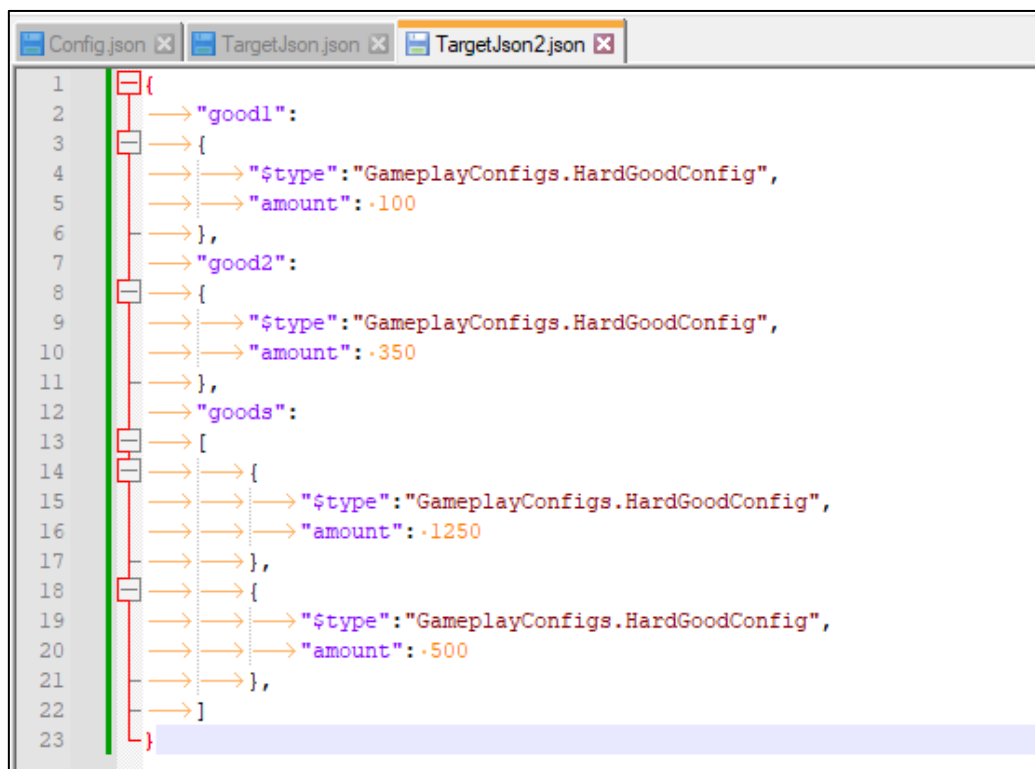


Рис. 20 Example3.1 Alias function. Semantics without alias func. Целевой json

Можно заметить что следующая часть повторяется в четырёх элементах, меняется только значение в поле amount.

```
{  
    "$type": "GameplayConfigs.HardGoodConfig",  
    "amount":  
}
```

Alias функции делятся на табличные alias функции и json alias функции.

Табличные alias функции

Вместо того, чтобы писать полную семантику парсинга как показано на рисунке 21, можно использовать alias функции и сократить запись как показано на 22.

1	2	A	B	C	D	E	F
1		ignore	id	name	type	value	
23							
24		TRUE	Example3 Alias function				
25		TRUE	Example3.1 Alias function. Semantics without alias func				
26							
27			example3.1	good1	ref	example3.1_good1	
28				good2	ref	example3.1_good2	
29				goods	arr	example3.1_goods	
30							
31							

1	2	A	B	C	D	E	F
1		ignore	id	name	type	value	
25							
26		TRUE	Example3 Alias function				
27		TRUE	Example3.1 Alias function. Semantics without alias func				
28							
29			example3.1_good1	\$type	str	GameplayConfigs.HardGoodConfig	
30				amount	num	100	
31			example3.1_good2	\$type	str	GameplayConfigs.HardGoodConfig	
32				amount	num	350	
33			example3.1_goods		ref	example3.1_goods_item1	
34					ref	example3.1_goods_item2	
35							

1	2	A	B	C	D	E	F
1		ignore	id	name	type	value	
17							
18		TRUE	Example3 Alias function				
19		TRUE	Example3.1 Alias function. Semantics without alias func				
20							
21			example3.1_goods_item1	\$type	str	GameplayConfigs.HardGoodConfig	
22				amount	num	1250	
23			example3.1_goods_item2	\$type	str	GameplayConfigs.HardGoodConfig	
24				amount	num	500	
25							
26							
27							
28							

Рис. 21 Example3.1 Alias function. Semantics without alias func.

1	2	A	B	C	D	E	F	G
1	ignore	id	name	type	value	aliasFuncArg		
23								
24	TRUE	Example3 Alias function						
25	TRUE	Example3.1 Alias function. Semantics without alias func						
26								
27		example3.1	good1	ref	example3.1_good1			
28			good2	ref	example3.1_good2			
29			goods	arr	example3.1_goods			
30								
31								
32	TRUE	Example3.2 Alias function. Semantics with alias func						
33								
34		example3.2	good1	HardGood	amount		100	
35			good2	HardGood	amount		350	
36			goods	arr	example3.2_goods			
37								
38								
39								

1	2	A	B	C	D	E	F	G
1	ignore	id	name	type	value	aliasFuncArg		
25								
26	TRUE	Example3 Alias function						
27	TRUE	Example3.1 Alias function. Semantics without alias func						
28								
29		example3.1_good1	\$type	str	GameplayConfigs.HardGoodConfig			
30			amount	num		100		
31		example3.1_good2	\$type	str	GameplayConfigs.HardGoodConfig			
32			amount	num		350		
33		example3.1_goods		ref	example3.1_goods_item1			
34				ref	example3.1_goods_item2			
35								
36	TRUE	Example3.2 Alias function. Semantics with alias func						
37		example3.2_goods		HardGood	amount		1250	
38				HardGood	amount		500	
39								
40								
41								

1	2	A	B	C	D	E	F
1	ignore	funcid	name	type	value		
2							
3							
4							
5	TRUE	This row start parsing					
6		HardGood	\$type	str	GameplayConfigs.HardGoodConfig		
7			amount	num	%amount%		
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							

Рис. 22 Example3.2 Alias function. Semantics with alias func

Для использования alias функций необходимо:

- настроить конфиг парсинга (Рис. 23) alias функций. Добавляется новый раздел aliasFuncsParsing, который оформляется по тем же правилам, что и featuresParsing. В разделе featuresParsing добавляется поле aliasFuncArgValueColumnName с названием колонки откуда брать параметры для alias функции.
- добавить колонку параметров для alias функций (Рис. 22)
- написать функцию на отдельном листе. (Рис. 22). Имена параметров оборачиваются в знак % (например, %amount%).

P.S. Alias функции имеет такие же возможности, что и обычная семантика. Внутри alias функций можно использовать другие alias функции. В alias функциях можно использовать неограниченное кол-во параметров.

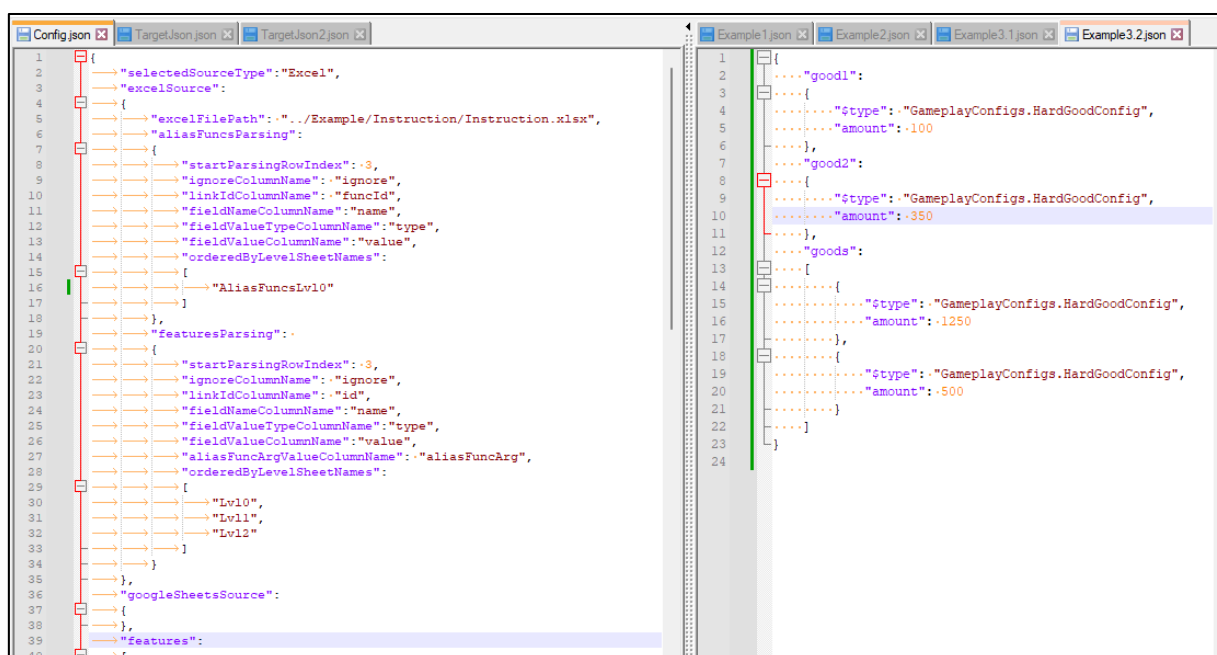


Рис. 23 Конфиг парсинга с настройкой alias функций

Json alias функции

Json alias функции являются вторым вариантом использования alias функции. Отличие json alias функций от табличных alias функций в том, что семантика пишется в json файле (Рис. 24). Alias функции можно комбинировать, одновременно работать с табличными и json alias функциями.

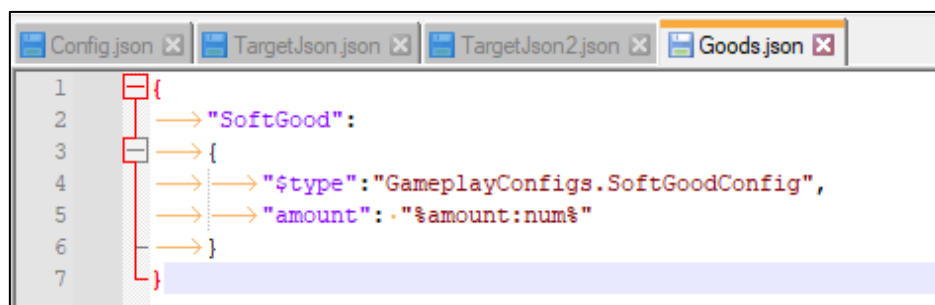


Рис. 24 Alias функция в json файле

Аргументы записываются с использованием символа %. На рисунке 24 приведён пример аргумента amount (%amount:num%), где **amount** название аргумента, а **num** тип данных

аргумента. Поддерживаются следующие типы данных: **str** – строка, **num** – число, **bool** – логическое значение.

P.S. Чтобы сохранить семантику json аргументы оборачиваются в кавычки. Парсер заменит символы "%" и "%" при вставке (Рис. 25).

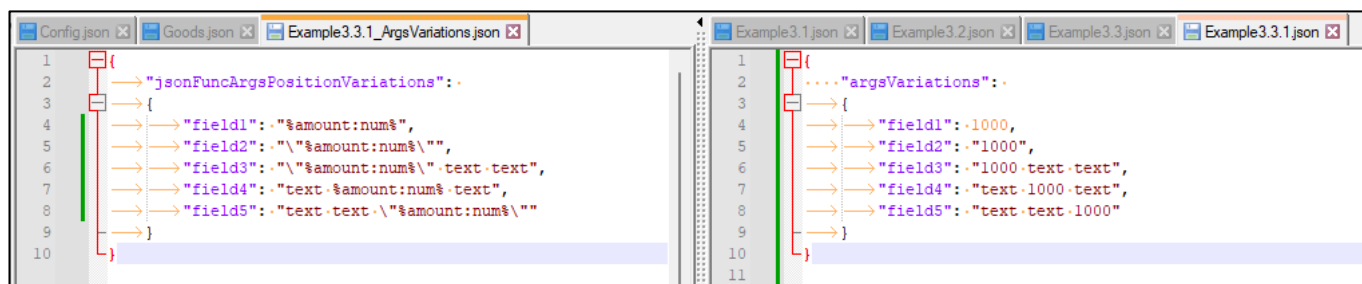


Рис. 25 Пример вариаций использования аргументов в json alias функциях

Для использования json alias функций необходимо создать файл (Рис. 24) и указать путь к нему в конфиге парсинга (Рис. 26) поле **jsonAliasesFilePaths**.

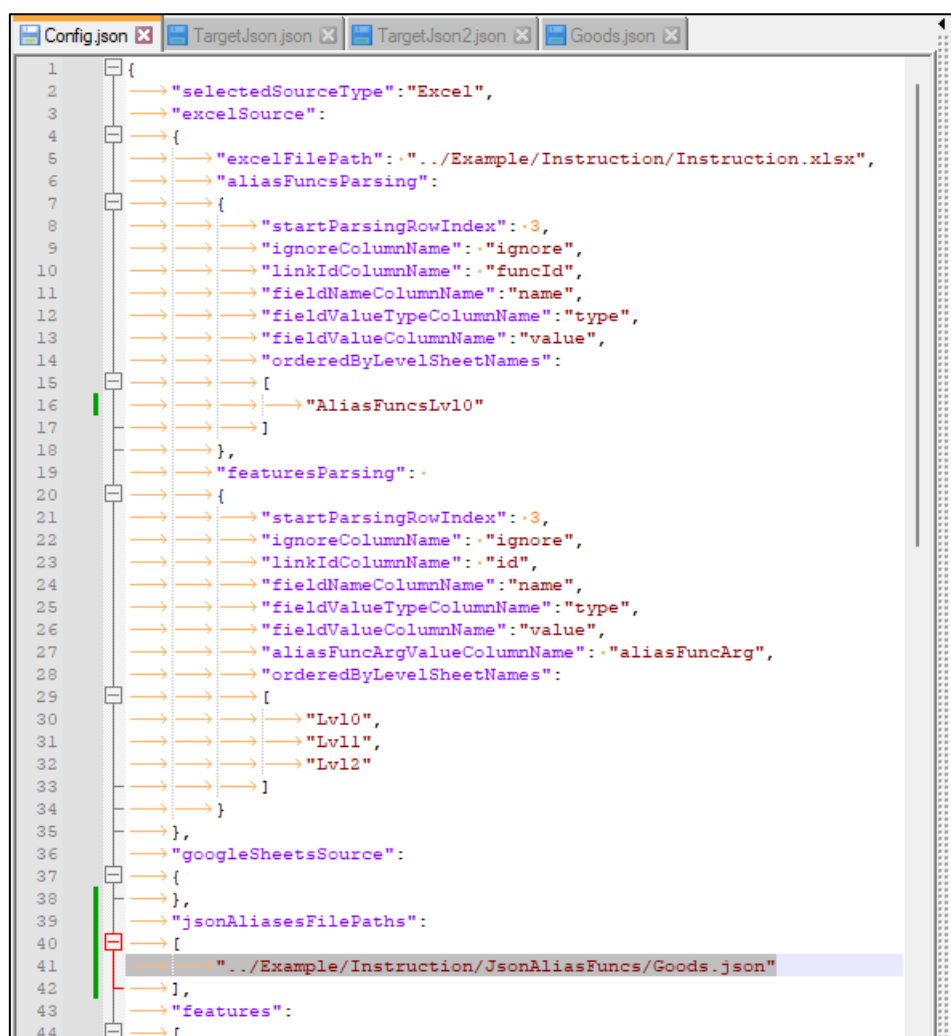


Рис. 26 Добавление пути к файлу с json alias функциями

P.S. В одном файле может быть множество alias функций.

Пример использования табличной alias функции **HardGood** и json alias функция **SoftGood** приведён на рисунках 27 и 28.

	A	B	C	D	E	F	G
1	ignore	id	name	type	value	aliasFuncArg	
31							
32	TRUE	Example3.2 Alias function. Semantics with alias func					
33							
34		example3.2	good1	HardGood	amount	100	
35			good2	HardGood	amount	350	
36			goods	arr	example3.2_goods		
37							
38							
39	TRUE	Example3.3 Alias function. Json and table alias functions					
40							
41		example3.3	tableAliasFuncField	HardGood	amount	1500	
42			jsonAliasFuncField	SoftGood	amount	1500	
43							
44							
45							
46							
47							

Рис. 27 Example 3.3. семантика комбинированного использования alias функций

```

1 {
2   ... "tableAliasFuncField":
3     ... {
4       ... "$type": "GameplayConfigs.HardGoodConfig",
5       ... "amount": 1500
6     },
7   ... "jsonAliasFuncField": { "$type": "GameplayConfigs.SoftGoodConfig", "amount": 1500 }
8 }
9

```

Рис. 28 Результат парсинга примера Example 3.3

P.S. На текущий момент json alias функции не умеют сохранять исходное форматирование (вставляет всё одной строкой)

Анонимные аргументы для alias функций

Для удобства использования формул Excel или GoogleSheets были добавлены анонимные аргументы, которые позволяют задавать параметры в столбик с неограниченным количеством. Для использования необходимо в конфиге парсинга (поле `anonymAliasFuncArgNameByColumnName`, рисунок 29) указать колонку где искать анонимный аргумент и название анонимного аргумента, который будет использоваться в alias функции.

В примере example 3.4 (Рис. 33) показано как использовать анонимные аргументы. Анонимные аргументы можно комбинировать. Для поля `jsonFuncWithTypedArg` используется анонимные аргументы **arg1**, **arg2** и обычный аргумент **amount**

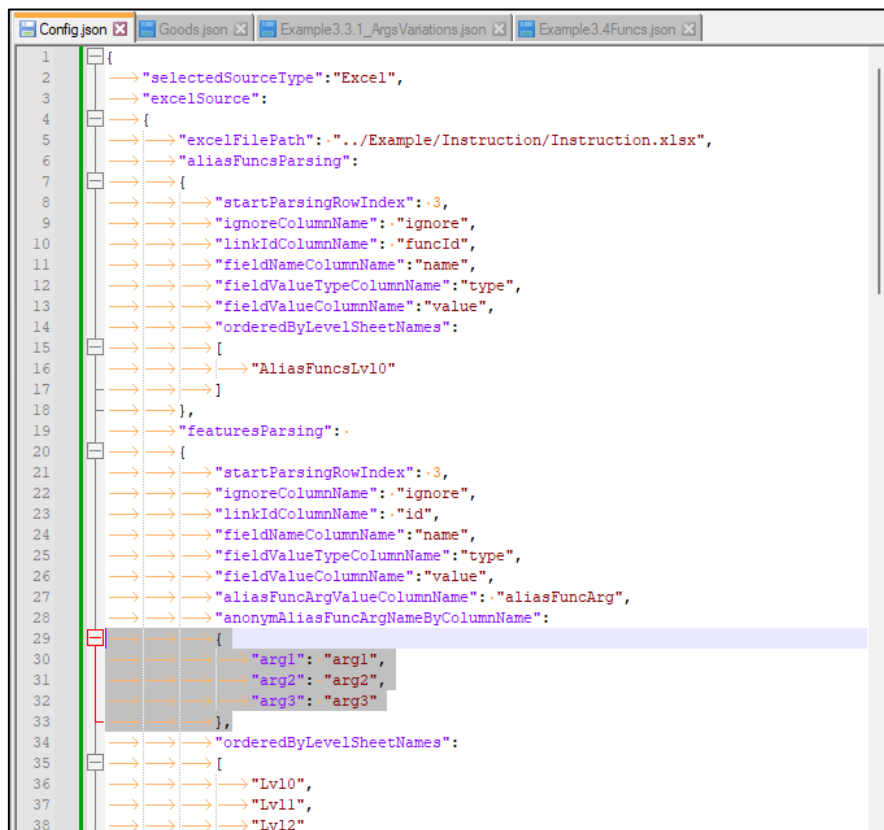


Рис. 29 Добавление анонимных аргументов

A54																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																														
-----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Рис. 30 Example 3.4 Alias function. Anonym args. Семантика парсинга

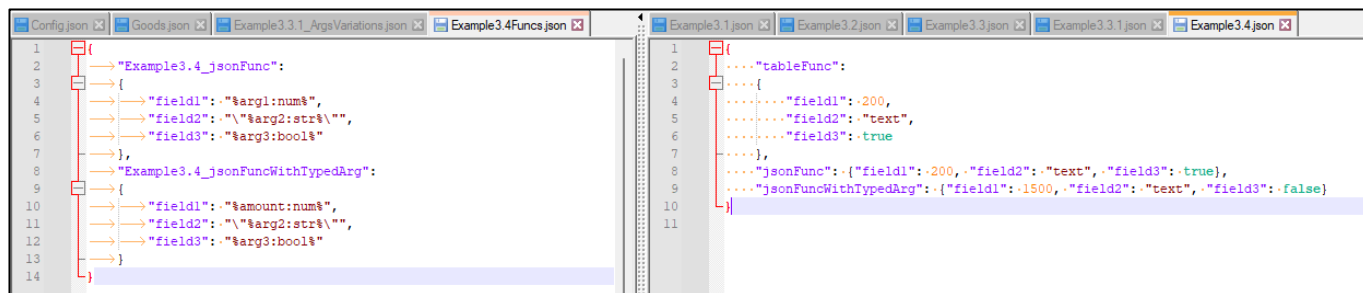


Рис. 31 Example 3.4 Alias function. Anonym args. Результат парсинга

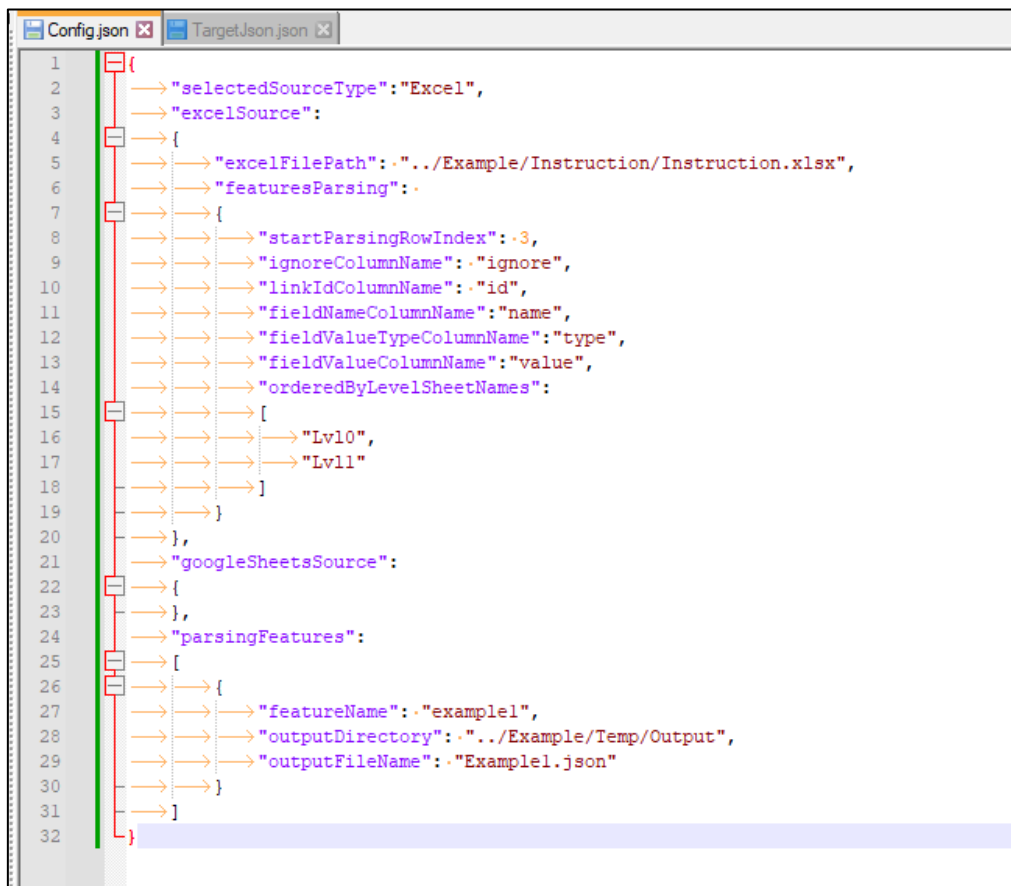


Рис. 33 Минимальная требуемая настройка конфига парсинга

Настройка конфига для GoogleSheets

Читайте файл инструкции с активацией доступа к GoogleSheets «**Activate Google Sheets Instruction.pdf**». Правила семантики заполняются по тем же правилам, что и для excel таблиц.