АБИС РУСЛАН НЕО

ПОДСИСТЕМА ФОРМИРОВАНИЯ ОТЧЕТОВ 1.3

РУКОВОДСТВО АДМИНИСТРАТОРА

СОДЕРЖАНИЕ

СПИСОК УСЛОВНЫХ СОКРАЩЕНИЙ И ТЕРМИНОВ

UTF-8	–	(от англ. Unicode Transformation Format, 8-bit – «формат преобразования Юникода, 8-битный») – одна из общепринятых и стандартизированных кодировок текста, которая позволяет хранить символы Unicode
АБИС	–	автоматизированная библиотечно-информационная система
АРМ	–	(автоматизированное рабочее место) – программно-технический комплекс, предназначенный для автоматизации деятельности определенного вида

ВВЕДЕНИЕ

В данном руководстве содержатся основные сведения о подсистеме формирования отчетов, а также информация о настройке и администрировании. Подсистема формирования отчетов используется в АРМе Книговыдачи и АРМе Комплектования/Каталогизации.

1. АРХИТЕКТУРА ПОДСИСТЕМЫ

1.1. Общие сведения

Подсистема формирования отчетов используется в АРМе Книговыдачи и в АРМе Комплектования/Каталогизации для построения различных отчетов.

Далее приведена схема работы подсистемы при формировании отчета.

1.2. Обработка данных для построения отчета

Процесс обработки данных для построения отчета показан на рисунке ниже.

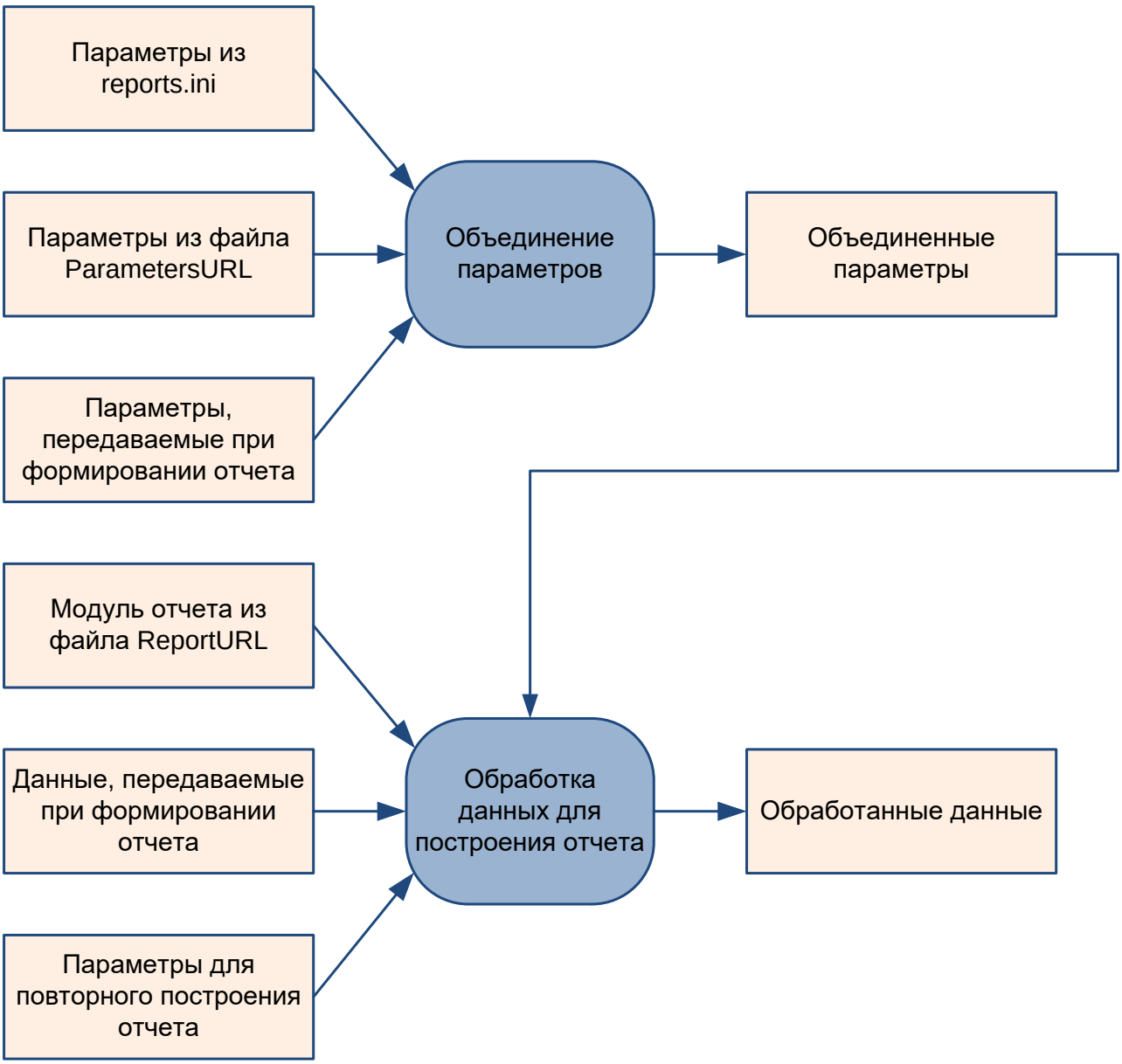


Рис. 1. Обработка данных для построения отчета

На первом этапе происходит подготовка параметров для отчета. При этом учитываются параметры, указанные в конфигурационном файле reports.ini (Parameter.*), параметры, загружаемые из файла, указанного в параметре ParametersURL, а также параметры, передаваемые из АРМа, запускающего процесс формирования отчета.

Пример указания параметров в reports.ini:

```
[Report.Default]
…
Parameter.libraryName=Новая библиотека
Parameter.librarySite=library_domain.ru
```

Далее выполняется загрузка модуля отчета, указанного в параметре ReportURL. Загруженный модуль выполняется:

```
report_module(options, callback)
```

В качестве параметра options передается объект, содержащий следующие элементы:

- baseUrl – базовый URL, используемый при загрузке шаблона отчета. Нужен при формировании URL с дополнительными файлами, требуемыми для модуля формирования отчета;
- data – объект с входными данными для формирования отчета;
- parameters – объединенные параметры;
- converter – объект Converter, используемый для различных преобразований;
- proxy – объект, используемый для для поиска записей в каталоге библиотеки;
- subreports – список дочерних отчетов;
- reportParameters – объект с дополнительными параметрами, используемый при повторном формировании отчета;
- subreportChangeHandler (function()) – обработчик события смены активного дочернего отчета;
- createReport (function(newReportParameters)) – функция запуска повторного создания отчета с дополнительными параметрами.

Параметр callback (function (res)) – обработчик завершения процесса формирования отчета. Если все в порядке, то в качестве параметра res передается объект, содержащий обработанные данные. При возникновении ошибки передается объект, содержащий элемент error:

```
{error: "При формировании отчета произошла ошибка"}
```

Далее объект с обработанными данными используется при формировании HTML-документа и при экспорте в другие форматы.

1.3. Формирование HTML-документа

В текущей версии подсистемы формирования отчетов поддерживание только один способ формирования HTML-документа – использование шаблонизатора JavaScript PURE. Описание PURE можно посмотреть на сайте проекта: https://beebole.com/pure/.

Процесс формирования HTML-документа показан на рисунке ниже.

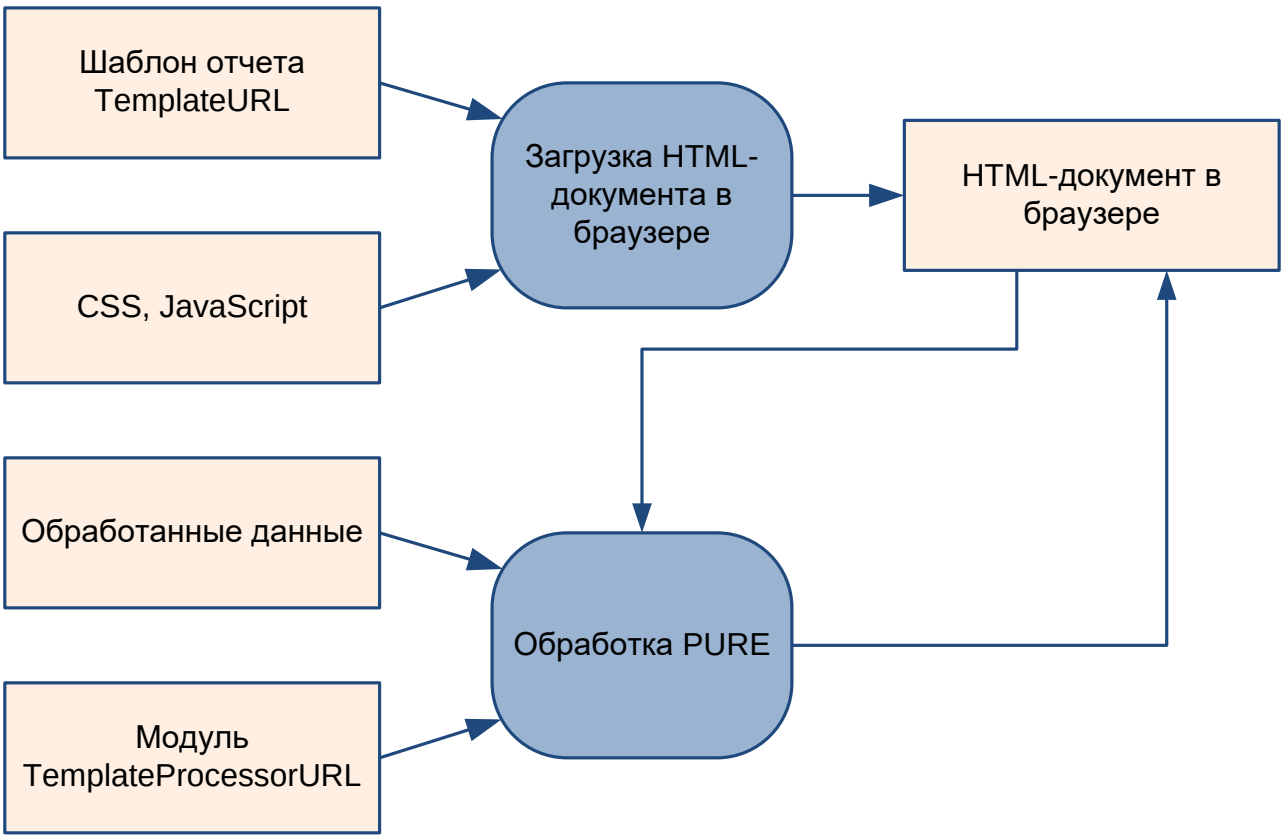


Рис. 2. Формирование документа HTML

Сначала шаблон отчета, указанный в параметре TemplateURL, загружается в iframe. При этом браузер также выполняет загрузку всех дополнительных файлов, включая CSS-стили и скрипты JavaScript.

Затем выполняется загрузка модуля, указанного в параметре TemplateProcessorURL. Загруженный модуль может содержать следующие элементы, все они не являются обязательными:

- directive – директива, используемая при вызове обработчика PURE.
- postProcessor (function(reportWindow, data)) – дополнительный обработчик, вызываемый после обработчика PURE.

После окончания загрузки вызывается обработчик PURE:

```
$p.render(data, directive)
```

Если в модуле TemplateProcessorURL нет элемента data, то обработчик PURE вызывается так:

```
$p.autoRender(data)
```

В качестве параметра data передается объект, содержащий обработанные данные.

А в качестве параметра directive передается объект из модуля TemplateProcessorURL.

После обработчика PURE опционально запускается обработчик postProcessor из модуля TemplateProcessorURL.

1.4. Экспорт отчетов

В текущей версии подсистемы формирования отчетов поддерживается три вида экспорта отчетов:

- экспорт по умолчанию;
- экспорт в DOCX на основе шаблона;
- экспорт в текстовый файл;
- экспорт в XLSX на основе шаблона.

Экспорт по умолчанию

Для экспорта отчета по умолчанию используется библиотека html-docx-js (https://github.com/evidenceprime/html-docx-js). Процесс показан на рисунке ниже.

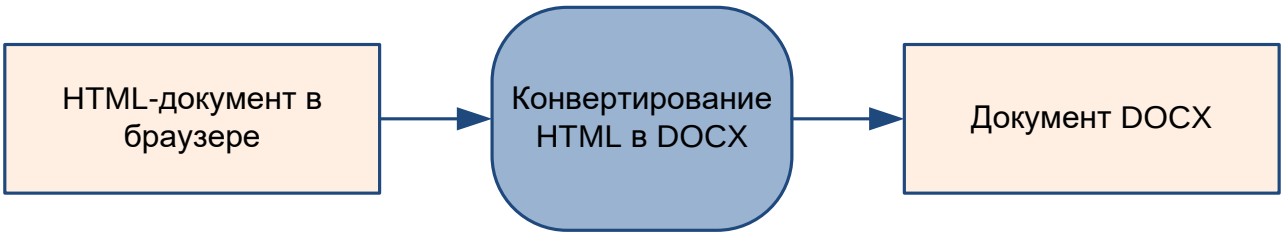


Рис. 3. Экспорт отчета по умолчанию

Экспорт по умолчанию включается или отключается в параметрах отчета:

```
[Report.Default]
DefaultExportEnabled=1
```

При экспорте на основе HTML-страницы с отчетом создается DOCX-документ. Из преимуществ такого подхода можно отметить то, что он доступен для любых отчетов. А главный недостаток – в документ DOCX практически не переносится информация о форматировании текста.

Экспорт в DOCX на основе шаблона

Для экспорта в DOCX используется библиотека docxtemplater (https://github.com/open-xml-templating/docxtemplater). Процесс экспорта показан на рисунке ниже.

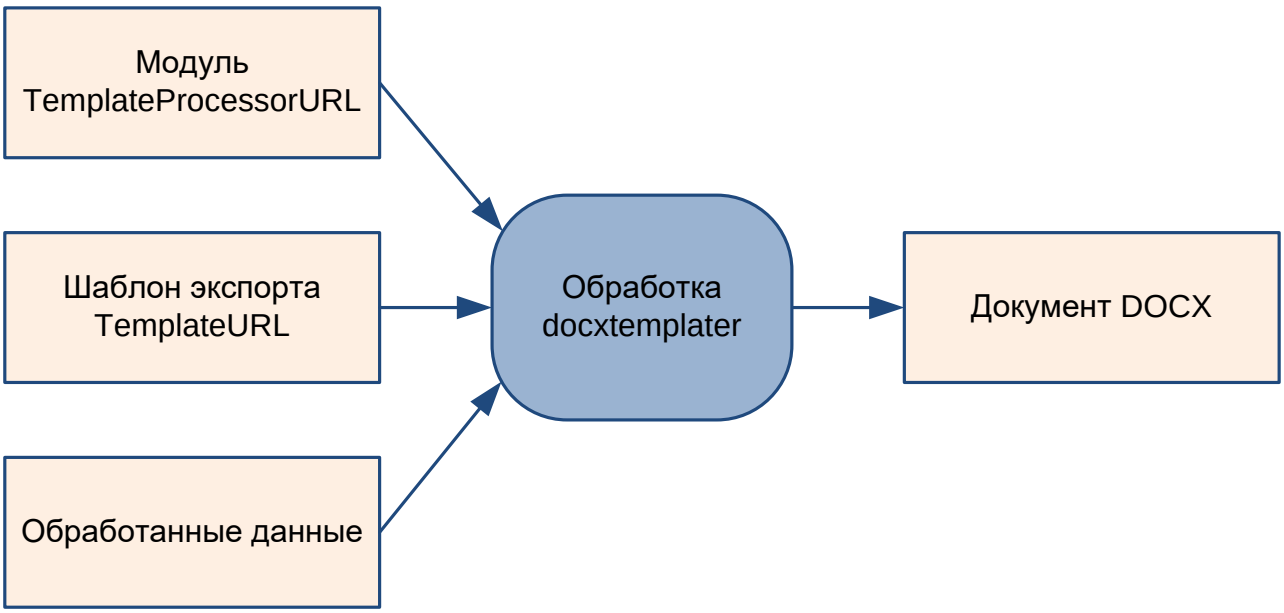


Рис. 4. Экспорт отчета с использованием docxtemplater

Экспорт отчета с использованием docxtemplater можно настроить в параметрах отчета:

```
[Report.BNP]
…
Export.1.Name=Экспорт в DOCX (ИБК СПбПУ)
Export.1.TemplateType=DOCX
Export.1.TemplateURL=export/bnp-spbpu.docx
Export.1.TemplateProcessorURL=export/bnp-spbpu-docx.js
```

При экспорте сначала загружается шаблон экспорта, указанный в параметре TemplateURL. Шаблон экспорта – это обычный документ DOCX, который можно создать с помощью Microsoft Word или любого другого редактора DOCX.

Опционально может быть указан модуль обработки TemplateProcessorURL. Если он указан, то загружается этот модуль, в котором содержится следующий элемент:

- processor (function(options, callback)) – обработчик, вызываемый при экспорте отчета в DOCX.

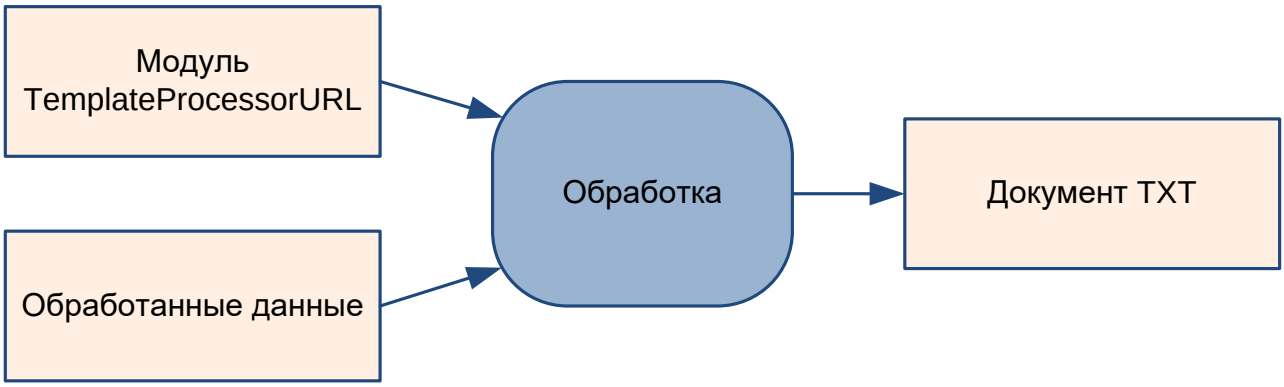
Затем вызывается обработчик docxtemplater, которому в качестве параметров передаются шаблон экспорта и обработанные данные:

```
var doc = new Docxtemplater(template);
doc.setData(data);
doc.render();
```

В результате создается DOCX-документ.

Экспорт в текстовый файл

Процесс экспорта в текстовый файл показан на рисунке ниже.



Экспорт отчета в TXT можно настроить в параметрах отчета:

```
[Settings.Report-List]
…
Export.TXT.Name=Экспорт в TXT
Export.TXT.TemplateType=TXT
Export.TXT.TemplateProcessorURL=export/list-txt.js
Export.TXT.Parameter.Delimiter=\t
```

При экспорте загружается модуль обработки TemplateProcessorURL, в котором содержится следующий элемент:

- processor (function(options, callback)) – обработчик, вызываемый при экспорте отчета в TXT.

В результате создается файл TXT.

Экспорт в XLSX на основе шаблона

Для экспорта в XLSX используется библиотека xlsx-populate (https://www.npmjs.com/package/xlsx-populate). Процесс экспорта показан на рисунке ниже.

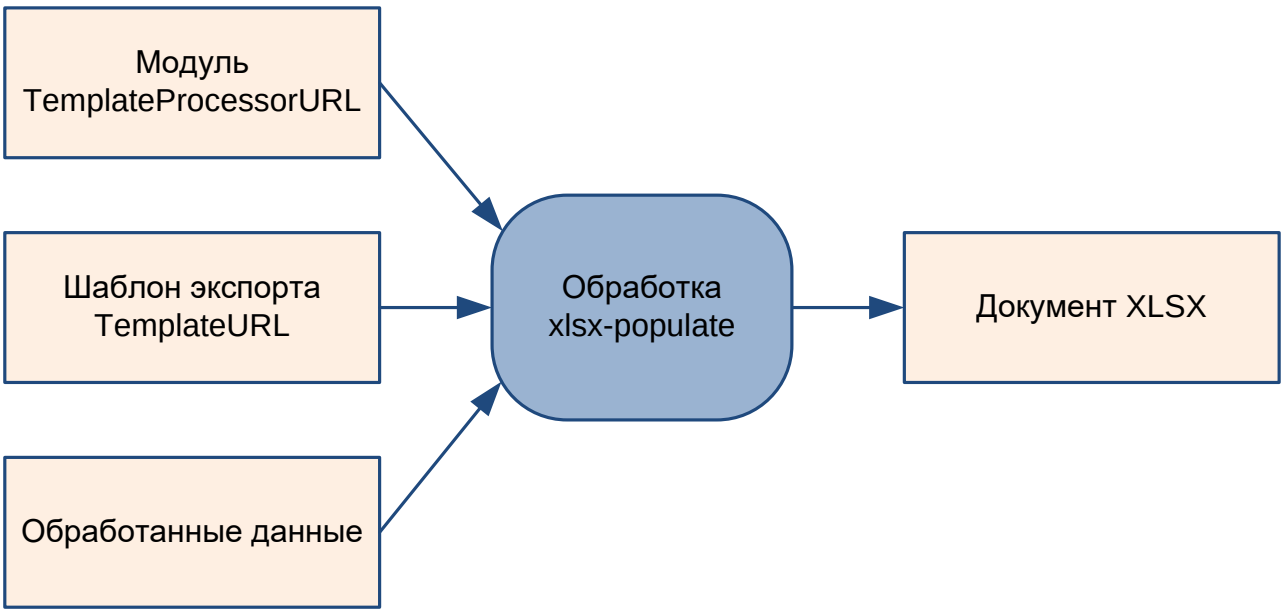


Рис. 5. Экспорт отчета с использованием xlsx-populate

Экспорт отчета с использованием xlsx-populate можно настроить в параметрах отчета:

```
[Report.Act]
…
Export.1.Name=Экспорт в XLSX №1 (ИБК СПбПУ)
Export.1.TemplateType=XLSX
Export.1.TemplateURL=export/act-spbpu1.xlsx
Export.1.TemplateProcessorURL=export/act-spbpu1-xlsx.js
```

При экспорте сначала загружается шаблон экспорта, указанный в параметре TemplateURL. Шаблон экспорта – это обычный документ XLSX, который можно создать с помощью Microsoft Excel или любого другого редактора XLSX.

Далее загружается модуль обработки TemplateProcessorURL, в котором содержится следующий элемент:

- processor (function(options, callback)) – обработчик, вызываемый при экспорте отчета в XLSX.

В результате создается XLSX-документ.

2. КОНФИГУРАЦИОННЫЕ ФАЙЛЫ

2.1. Структура конфигурационных файлов

Конфигурационные файлы подсистемы формирования отчетов загружаются из файла reports.ini. По умолчанию он находится в каталоге «/reports-config». В АРМе Книговыдачи при выборе рабочего места также дополнительно загружаются файлы reports.ini, специфичные для конкретного рабочего места, которые затем объединяются.

Далее приведено описание всех секций файла reports.ini и используемых параметров.

Обозначение «*» в названии раздела или параметра означает, что раздел или параметр могут повторяться несколько раз. Например, для описания различных типов отчетов используются секции:

```
[Report.RegCard]
…

[Report.UserCircList]
…
```

Также может использовать секция с кодом «Default» вместо «*» для задания параметров по умолчанию:

```
[ReportType.Default]
ShowMode=2
```

Примечание. Для типа параметра «Список строк» значения отделяются с помощью «,», например:

```
Parameter.departments=ИМОП,ОНЛ,ОУЛ,ОЧЗ
```

2.2. Секция «ReportType.*»

№	Параметр	Тип	Комментарии
1	DefaultReport	строка	Код отчета по умолчанию

2	ShowMode	0/1/**2**	Режим отображения формы для выбора отчета: • 0 – не отображать форму выбора отчета • 1 – показывать только форму выбора отчета • 2 – показывать форму для просмотра отчета

2.3. Секция «Report.*»

№	Параметр	Тип	Комментарии
1	Enabled	0/**1**	Разрешение использования отчета
2	Name	0/**1**	Название отчета
3	HoldingsMode	0/2/3	Режим работы с информацией об экземплярах: • 0 – не указано • 2 – информация об экземплярах содержится в полях 899, 999 библиографической записи • 3 – информация об экземплярах содержится в записях Holdings
4	ReportType	список строк	Тип отчета (основной)
5	ReportType2	список строк	Тип отчета (дополнительный)
6	BaseURL	строка	Базовый URL для всех файлов отчета
7	ReportURL	строка	URL отчета
8	ParametersURL	строка	URL параметров отчета
9	TemplateType	строка	Тип шаблона отчета
10	TemplateURL	строка	URL шаблона отчета
11	TemplateProcessorURL	строка	URL обработчика для шаблона отчета
12	SubreportType	список строк	Тип дочернего отчета (основной)
13	SubreportType2	список строк	Тип дочернего отчета (дополнительный)
14	DisplayOrder	число	Порядок для сортировки отчетов. Чем меньше число, тем раньше в списке будет отчет. По умолчанию – 0
15	DefaultExportEnabled	0/**1**	Признак того, что экспорт по умолчанию разрешен

16	Export.*.Enabled	0/**1**	Признак того, что экспорт разрешен
17	Export.*.Name	строка	Название экспорта
18	Export.*.TemplateType	строка	Тип шаблона экспорта
19	Export.*.TemplateURL	строка	URL шаблона экспорта
20	Export.*. TemplateProcessorURL	строка	URL обработчика для шаблона отчета
21	Export.*.Parameter.*	строка	Пользовательский параметр экспорта
22	Parameter.*	строка	Пользовательский параметр отчета
23	Settings	строка	Код настроек, добавляемых к параметрам из секции

2.4. Секция «Settings.*»

Данные секции содержат произвольный набор параметров, на которые могут присутствовать ссылки из секций «Report.*» или из других секций «Settings.*»:

```
[Settings.Report-List]
Enabled=1
BaseURL=list
ReportURL=list.js

[Settings.Report-DocList]
Settings=Report-List
ReportType=Documents
Parameter.reportTitle=Список документов

[Report.UserList]
Settings=Report-DocList
Name=Список читателей
…
```

Это позволяет использовать общие настройки для нескольких отчетов.

3. ТИПЫ ОТЧЕТОВ

3.1. Общие сведения

Подсистема формирования отчетов содержит отчеты разных типов. Каждый отчет может относиться к одному или нескольким типам.

Список типов отчетов приведен далее в таблицах. Во входных данных параметры, передающиеся обязательно, отмечены подчеркиванием. Остальные параметры являются необязательными.

3.2. Отчеты для АРМа Книговыдачи

№	Код	Назначение	Входные данные
1	Circs	Список записей на выданные документы	• objects – список служебных записей на выданные документы
2	Circs2	Список записей на выданные документы и запись на читателя	• objects – список служебных записей на выданные документы • reader – служебная запись на читателя
3	Event	Информация о мероприятии	• event – служебная запись на мероприятие
4	Event2	Расширенная информация о мероприятии	• event – служебная запись на мероприятие • visits – список служебных записей на посещения мероприятия
5	Events	Список мероприятий	• objects – список служебных записей на мероприятия
6	Extd	Информация об электронном заказе	• extd – служебная запись на электронный заказ
7	Extds	Список электронных заказов	• objects – список служебных записей электронные заказы
8	Holdings	Список записей на экземпляры документов	• objects – список служебных записей на экземпляры документы
9	Holdings2	Список записей на экземпляры документов и запись на читателя	• objects – список служебных записей на экземпляры документы • reader – служебная запись на читателя
10	InhouseVisits	Список записей на стационарные обслуживания	• objects – список служебных записей на стационарные обслуживания

11	LibServices	Список записей на библиотечно-информационные услуги	• objects – список служебных записей на библиотечно-информационные услуги
12	Queues	Список очередей	• objects – список служебных записей на очереди
13	Reader	Информация о читателе	• reader – служебная запись на читателя
14	Reader2	Расширенная информация о читателе	• reader – служебная запись на читателя • circs – список служебных записей на выданные документы • extds – список служебных записей на электронные заказы • queues – список служебных записей на очереди
15	Readers	Список читателей	• objects – список служебных записей на читателей
16	Receipt	Квитанция	• reader – служебная запись на читателя • receipt – служебная запись на квитанцию • receiptDescriptionValueMap – словарь, в котором задается соответствие кодов поля «Назначение платежа» («description») и их названий, заданных в конфигурационном файле АРМа Книговыдачи
17	Receipts	Список квитанций	• reader – служебная запись на читателя • objects – список служебных записей на квитанции • receiptDescriptionValueMap – словарь, в котором задается соответствие кодов поля «Назначение платежа» («description») и их названий, заданных в конфигурационном файле АРМа Книговыдачи
18	RemoteCircs	Список удаленных возвратов документов	• objects – список служебных записей на удаленные возвраты документов

3.1. Отчеты для АРМа Комплектования/Каталогизации

№	Код	Назначение	Входные данные
1	Act	Информация об акте	• act – служебная запись на акт • documents – список библиографических записей на документы, относящихся к акту • holdings – список записей на экземпляры, относящиеся к библиографическим записям
2	Acts	Список актов	• objects – список служебных записей на акты
3	Acquisition	Поступления в фонд библиотеки	• bill – служебная запись на счет • bills – список служебных записей на счета • departments – список подразделений библиотеки • documents – список библиографических записей на документы • holdings – список записей на экземпляры, относящиеся к библиографическим записям
4	Bill	Информация о счете	• bill – служебная запись на счет • documents – список библиографических записей на документы, относящихся к акту • holdings – список записей на экземпляры, относящиеся к библиографическим записям
5	Bills	Список счетов	• objects – список служебных записей на счета
6	BNP	Список новых поступлений	• bills – список служебных записей на счета • departments – список подразделений библиотеки • dates ({from, to}) – объект, содержащий начальную и конечную дату • documents – список библиографических записей на документы • inventoryNumbers – список инвентарных номеров
7	Document	Информация о документе	• record – библиографическая запись на документ
8	Documents	Список документов	• objects – список библиографических записей на документы

9	Inventory-Numbers	Инвентарная книга	• documents – список библиографических записей на документы • bill – служебная запись на счет (можно указать, если bills = null) • bills – список служебных записей на счета • inventoryNumbers – список инвентарных номеров • dates ({from, to}) – объект, содержащий начальную и конечную дату • checkBillsDates (false/true) – признак того, что надо проверять даты открытия счетов на соответствие диапазону дат • departments
10	Inventory-Process	Инвентаризация	• allHoldings – общий список, содержащий полочные и отсканированные экземпляры • arrangementType – тип расстановки («A» – алфавитная, «I» – инвентарная, «FI» – форматно-инвентарная, «S» – систематическая) • departments – список подразделений библиотеки • documents – список библиографических записей на документы • inventoryResult – результат инвентаризации (словарь, содержащий списки по разным категориям в соответствии с результатами обработки найденных записей) • scannedHoldings – список записей на экземпляры, найденных по указанным идентификаторам экземпляров • scannedHoldingsItemIds – список введенных идентификаторов экземпляров • shelfHoldings – список записей на экземпляры, относящихся полке в соответствии с выбранным диапазоном

11	Move	Движения в фонде библиотеки (поступления и списания)	• documents – список библиографических записей на документы • holdings – список записей на экземпляры, относящиеся к библиографическим записям • bills – список служебных записей на счета • bill – служебная запись на счет (можно укзаать, если bills = null) • acts – список служебных записей на акты • act – служебная запись на акт (можно укзаать, если acts = null) • departments – список подразделений библиотеки
12	Periodical	Карточка периодического издания	• record – библиографическая запись на периодическое издание
13	Periodicals	Список периодических изданий	• objects – список библиографических записей на периодические издания
14	Subscriptions	Заявки на подписку	• objects – список служебных записей на заявки на подписку
15	WriteOff	Списания из фонда библиотеки	• documents – список библиографических записей на документы • holdings – список записей на экземпляры, относящиеся к библиографическим записям • acts – список служебных записей на акты • act – служебная запись на акт (можно укзаать, если acts = null) • departments – список подразделений библиотеки
16	Universal	Универсальные отчеты	• databases – список библиографических баз данных • billDatabases – список служебных баз данных, в которых хранятся записи на счета • actDatabases – список служебных баз данных, в которых хранятся записи на акты • departments – список подразделений библиотеки

4. ПРОГРАММНАЯ МОДЕЛЬ

4.1. Служебная запись

Далее в примерах record – служебная запись на читателя.

Получение количества всех полей

```
record.fields.length
```

Возвращаемое значение – число.

Получение поля по индексу

```
record.fields.getField(index)
```

Возвращаемое значение – поле. Объект содержит элементы:

- attribute – атрибут;
- type – тип значения (1 – бинарное значение, 2 – число, 3 – строка, 10 – пустое значение);
- value – значение;
- number – номер значения.

Получение списка полей по атрибуту

```
record.fields.getFields(attribute)
```

Возвращаемое значение – массив полей.

Получение значения по атрибуту

```
record.fields.getValue(attribute, number)
```

Параметры:

- attribute (число) – атрибут
- number (число) – номер поля с указанным атрибутом. Указывать не обязательно.

Возвращаемое значение – строка, число или null. Бинарное значение возвращается как строка в кодировке Base64.

Получение списка значений по атрибуту

```
record.fields.getValues(attribute)
```

Возвращаемое значение – массив значений.

Получение списка полей

```
record.fields.getList (fieldProcessor)
```

Параметр fieldProcessor – function(field), обработчик для поля из исходной записи. Указывать не обязательно.

Возвращаемое значение – массив полей.

Данная функция позволяет получить список полей, выполнив требуемую обработку.

4.2. Библиографическая запись

4.2.1. R2RusmarcRecord

Далее в примерах record – библиографическая запись.

Проверка существования записи

```
record.exist()
```

Для записи функция возвращает true для нормальной записи, false – при ошибке извлечения при загрузке с сервера.

Получение всех полей по номеру

```
record.fields(tag1, tag2)
```

Параметры tag1, tag2 указывают номер или список номеров, для которых нужно получить список полей.

Возвращаемое значение – R2RusmarcFieldsCollection.

Примеры:

Поля с номером 200	`record.fields(200)`
	`record.fields (null, 200)`
Поля с номером 200, встроенные в поле 461	`record.fields (461, 200)`
Поля с номером 200, встроенные в поле 461, или невстроенные	`record.fields([461, null], 200)`
Поля с номерами 700, 701, 702, встроенные в поле 461, или невстроенные	`record.fields([461, null], [700, 701, 702])`

Получение первого поля по номеру

```
record.field(tag1, tag2)
```

Данная функция аналогична предыдущей, но в результате возвращается коллекция, содержащее только первое поле, удовлетворяющее переданным параметрам.

Фильтрация записи с указанием того, какие поля и подполя следует оставить и исключить

```
record.filter(include, exclude)
```

Параметр «include» – словарь, указывающий, что надо оставить в записи. Если null или пустой, то оставляется всё.

Параметр «exclude» – словарь, указывающий, что надо исключить из записи. Если null или пустой, то ничего не исключается.

Сначала применяется правило «include», потом «exclude».

Возвращаемое значение – R2RusmarcRecord.

Поддерживаемые варианты указания фильтров в параметрах «include» и «exclude»:

Поле целиком	`100`
Группа полей, начинающихся на одну цифру	`3xx`
Поле, встроенное в другое поле	`461%100`
Поле, не встроенное в другое поле	`%100`
Указанное подполе	`200$b`
Указанное подполе в поле, встроенном в другое поле	`461%200$b`
Указанное подполе в поле, не встроенном в другое поле	`%200$b`

Признак записи

```
record.isRecord()
```

Для записи функция всегда возвращает true.

Получение маркера

```
record.leader()
```

Возвращаемое значение – R2RusmarcLeader.

Получение первого подполя по номеру по номеру поля и коду подполя

```
record.subfield(tag1, tag2, tag3)
```

Параметры tag1, tag2, tag3 указывают номер или список номеров полей, а также код или список кодов, для которых нужно получить список подполей. Возвращается только первое подполе, удовлетворяющее переданным параметрам.

Возвращаемое значение – R2RusmarcSubfieldsCollection.

Примеры:

Подполя с кодом 'a' в полях с номером 200	`record.subfields(200, 'a')`
	`record.subfields (null, 200, 'a')`
Подполя с кодом 'a' в полях с номером 200, встроенных в поле 461	`record.subfields (461, 200, 'a')`
Подполя с кодами 'a', 'b' в полях с номером 200, встроенных в поле 461, или невстроенных	`record.subfields([461, null], 200, ['a', 'b'])`
Подполя с кодом 'a' в полях с номерами 700, 701, 702, встроенных в поле 461, или невстроенных	`record.subfields([461, null], [700, 701, 702] , 'a')`

Получение первого подполя по номеру по номеру поля, коду подполя и порядковому номеру подполя

```
record.subfield2(tag1, tag2, position)
```

Параметр tag1 – код поля, tag2 – код подполя, position – порядковый номер подполя (начиная с 0). В результате возвращается коллекция, содержащее только первое подполе, удовлетворяющее переданным параметрам.

Возвращаемое значение – R2RusmarcSubfieldsCollection.

Получение всех подполей по номеру поля и коду подполя

```
record.subfields(tag1, tag2, tag3)
```

Данная функция аналогична функции «subfield», но в результате возвращается коллекция, содержащее все подполя, удовлетворяющие переданным параметрам.

Получение всех значений в виде одной строки

```
record.val(delimiter)
```

Параметр delimiter задает разделитель. Указывать не обязательно.

Функция возвращает значение маркера, всех подполей всех полей в виде строки с использованием указанного разделителя.

Получение всех значений в виде массива

```
record.vals()
```

Функция возвращает значение маркера, всех подполей всех полей в виде массива строк.

4.2.2. R2RusmarcLeader

Далее в примерах leader – маркер записи.

Получение всех значений в виде одной строки

```
leader.val(delimiter)
```

Параметр delimiter задает разделитель. Указывать не обязательно. Для маркера записи данный параметр значения не имеет

Функция возвращает значение маркера в виде строки.

Получение всех значений в виде массива

```
record.vals()
```

Функция возвращает значение маркера в виде массива строк, состоящего из одного элемента.

4.2.3. R2RusmarcFieldsCollection

Далее в примерах col – объект R2RusmarcFieldsCollection.

Массив всех полей

```
col.items
```

Количество элементов в коллекции

```
col.length
```

Обработка всех элементов в коллекции

```
col.each(action)
```

Параметры action – function(item), обработчик для каждого поля.

Проверка существования элементов в коллекции

```
col.exist()
```

Функция возвращает true, если есть хотя бы один элемент в коллекции, false – в противном случае.

Получение первого поля по номеру

```
col.field(tag)
```

Параметр tag указывают номер или список номеров, для которых нужно получить список полей. Возвращается только первое поле, удовлетворяющее переданным параметрам.

Возвращаемое значение – R2RusmarcFieldsCollection.

Получение всех полей по номеру

```
col.fields(tag)
```

Данная функция аналогична предыдущей, но в результате возвращается коллекция, содержащее все поля, удовлетворяющее переданным параметрам.

Возвращаемое значение – R2RusmarcFieldsCollection.

Признак записи

```
col.isRecord()
```

Для коллекции полей функция всегда возвращает false.

Получение маркера

```
col.leader()
```

Всегда возвращается объект, содержащий пустой маркер.

Возвращаемое значение – R2RusmarcLeader.

Получение первого индикатора по номеру

```
col.indicator(tag)
```

Параметр tag указывают номер или список номеров, для которых нужно получить индикаторы.

Возвращаемое значение – R2RusmarcIndicatorsCollection.

Получение первого подполя по номеру по номеру поля и коду подполя

```
col.subfield(tag1, tag2)
```

Параметры tag1, tag2 указывают номер или список номеров полей, а также код или список кодов, для которых нужно получить список подполей. Возвращается только первое подполе, удовлетворяющее переданным параметрам.

Возвращаемое значение – R2RusmarcSubfieldsCollection.

Получение первого подполя по номеру по номеру поля, коду подполя и порядковому номеру подполя

```
record.subfield2(tag1, tag2, position)
```

Параметр tag1 – код поля, tag2 – код подполя, position – порядковый номер подполя (начиная с 0). В результате возвращается коллекция, содержащее только первое подполе, удовлетворяющее переданным параметрам.

Возвращаемое значение – R2RusmarcSubfieldsCollection.

Получение всех подполей по номеру поля и коду подполя

```
col.subfields(tag1, tag2)
```

Данная функция аналогична предыдущей, но в результате возвращается коллекция, содержащее все подполя, удовлетворяющее переданным параметрам.

Получение всех значений в виде одной строки

```
col.val(delimiter)
```

Параметр delimiter задает разделитель. Указывать не обязательно.

Функция возвращает значение всех подполей всех полей в виде строки с использованием указанного разделителя.

Получение всех значений в виде одной строки с форматированием

```
col.val2()
```

В функцию передается произвольное число аргументов. Каждый аргумент может указывать на подполе, в таком случае он является строкой из символа '$' и кода подполя. Также аргумент может быть другой строкой, в таком случае он считает-

ся разделителем. И дополнительно аргумент может быть массивом, в таком случае все элементы массива обрабатываются единым блоком.

Например, в поле field есть такие подполя:

```
$a абв $b 123 $b 456.
```

Примеры использования функции val2:

Значения двух подполей, разделенных «: ».	`field.val2('$a', ': ', '$b')`
	
	`абв: 123`

Значения двух подполей, разделенных «: ». Для подполя $b допускается наличие нескольких экземпляров, между которыми также ставится разделитель «:».	`field.val2('$a', ': ', '$b/')`
	
	`абв: 123: 456`

Значения двух подполей, разделенных «: ». Для подполя $b допускается наличие нескольких экземпляров, между которыми ставится разделитель «, ».	`field.val2('$a', ': ', '$b/, ')`
	
	`абв: 123, 456`

Значения двух подполей, разделенных «: ». Подполя $c нет, поэтому в результате возвращается только значение подполя $a.	`field.val2('$a', ': ', '$c')`
	
	`абв`

Значения двух подполей, разделенных «: ». Подполей $c и $d нет, поэтому возвращается пустая строка.	`field.val2('$c', ': ', '$d')` *`пустая строка`*

Значения двух подполей, второе обрамляется круглыми скобками.	`field.val2('$a', [' (', '$b'], ')'])`
	
	`абв (123)`

Значения двух подполей, второе обрамляется круглыми скобками. Подполя $c нет, поэтому в результате возвращается только значение подполя $a.	`field.val2('$a', [' (', '$c'], ')'])`
	
	`абв`

Значения двух подполей, второе обрамляется круглыми скобками. Подполей $c и $d нет, поэтому возвращается пустая строка.	`field.val2('$c', [' (', '$d'], ')'])` *пустая строка*

Получение всех значений в виде массива

```
col.vals()
```

Функция возвращает значение всех подполей всех полей в виде массива строк.

4.2.4. R2RusmarcSubfieldsCollection

Далее в примерах col – объект R2RusmarcSubfieldsCollection.

4.2.5. R2RusmarcField

Далее в примерах field – объект R2RusmarcField.

4.2.6. R2RusmarcSubfield

Далее в примерах field – объект R2RusmarcSubfield.