acm.cms

(версия 5.0 / 678)

Оглавление

Введение

ACM.CMS это многофункциональная система управления присутствием в сети "Интернет".

Данная система использует высокотехнологичные общепризнанные стандарты, такие как: JavaTM и XML.

Система ACM.CMS обладает высокой производительностью и проверена на стабильную работу в таких системах как Sun Solaris, WindowsNT, Windows2000, WindowsXP, Debian Linux, Red-Hat Linux, FreeBSD, NetBSD и с такими СУБД как MySQL, MS SQL 7.0, MS SQL 2000, Oracle 8i, Oracle 9i...

Система ACM.CMS чрезвычайно гибка и масштабируема, она изначально рассчитана на применение при создании и поддержке сайтов и группировок различных уровней сложности: от простейших одноязычных сайтов без персонализации вывода информации, до сложных многоязычных сайтов с закрытыми персонализированными разделами, сложной системой ролей, многовариантным форматированием отображаемых данных и с уникальной логикой интерфейса и внутренней обработки данных подключаемой как программный модуль.

О документе

Данный документ содержит описание текущей версии системы и несет миссию быть единым источником информации по сборке сайтов, по администрированию, по пониманию устройства системы, по решению возникающих проблем.

Текушая версия этого документа может быть найдена в дистрибутиве системы. По мере выпуска версий системы документ обновляется в соответствиями с изменениями в этой версии и, также, по возможности, он дополняется отсутствующей информацией.

Красным цветом выделено то, что не соответствует действительности, но уже запланировано на реализацию в ближайших версиях системы.

Пурпурным цветом выделено то, что будет изменено или удалено в ближайших версиях системы.

Разумеется, документ не полон. Система развита и имеет длинную историю своего развития, а этот документ относительно молод. Если у вас есть замечания по поводу написанного в этом документа, поправки, дополнения или вопросы – направляйте их по адресу **myx@myx.ru**. Нам было бы очень важно узнать ваше мнение об этом документе, и, если у вас есть немного времени – мы будем рады увидеть от вас сообщение отвечающее на два вопроса: 1) что бы вы хотели увидеть в этом документе из того, чего в нем нет и 2) что бы вы хотели увидеть описанным более подробно из того, что в нем уже есть. Спасибо.

Сборка

Работающий сервер ACM.CMS представляет собой набор описанных типов объектов, определенную наполненную иерархию объектов, реализованный(е) интерфейсы взаимодействия пользователей с этими объектами и одну или более точек публичного доступа. Весь процесс называется «сборка».

Подразумевается, что сборщик имеет неплохое представление об XML и JavaScript. Поскольку это стандартные технологии, они не будут отдельно рассмотрены в данной документе.

Для того, чтобы приступить к сборке сайта требуется зарегистрировать сервер в системе (это рассматривается в разделе «Администрирование»). После регистрации сервера мы получаем каталог файлов для нового сервера (включающий в себя файл конфигурации сервера «**`config.xml`**», подкаталог «**`skin`**» для скинов сайта и подкаталог «**`types`**» для типов).

Скриптовые языки и форматы описания

Система имеет множество мест в которых может происходить выполнение кода, написанного сборщиком сайта, например: шаблоны страниц, события и методы объектов, выражения определяющие доступность тех или иных команд в интерфейсе управления, подготовка полей для форм и т.п. Для каждой из задач один из языков может оказаться удобным, другой не очень. Система поддерживает определенное число языков, часть их которых являются в буквальном смысле скриптовыми языками, часть шаблонными и часть псевдо-языками – определенной функцией над проходящими данными, вот список:

Имя	**Скриптование**	**Шаблонный**	**Другое**
ACM.JSCRIPT ACM.JAVASCRIPT ACM.ECMA	Да	Да	Скриптовый язык с поддержкой шаблонов
ACM.TPL	Да	Да	Шаблонный скриптовый язык
ACM.MAP	Не предназначен	Не совсем	Создает набор данных по XML описанию
ACM.CALC ACM.EVALUATE	Не предназначен	Нет	Выполнение выражения
ACM.RETRIEVE	Не предназначен	Нет	Прогоняет набор входящих данных производя преобразование полей
ACM.FIELDSET	Нет	Нет	Создает набор полей по XML описанию
ACM.XSL ACM.XSLT	Нет	Да	Выполняет XSL преобразование
ACM.TXT ACM.TEXT	Нет	Нет	Возвращает константный текст
ACM.NULL	Нет	Нет	Возвращяет NULL

Скриптование подразумевает возможность описания алгоритма, описания функции производимой над проходящими данными, описания последовательности процедур. Всё это не имеет смысла без каких-либо доступных системных функций и объетов в контексте выполнения – для информации по набору системных функций и объектов смотрите раздел «Системные API».

Язык JSCRIPT (ACM.JSCRIPT, ACM.JAVASCRIPT, ACM.ECMA)

Данный язык является реализацией стандарта ECMA-262. Этот стандарт используется повсеместно и известен также под именами ActiveScript, JavaScript, JScript. Реализации этого языка можно найти в виде скриптов на страницах различных сайтов в сети интернет (он является единственным языком стандартно поддерживающимся во всех браузерах поддерживающих скрипты в принципе), на этом языке программируются мултимедиа-ролики, игры и баннеры в Macromedia Flash и на этом же языке можно писать исполняемые скрипты в системе Microsoft Windows. Другими словами – это наиболее стандартный и распространенный язык среди поддерживаемых системой ACM.CMS языков.

Чтобы указать в качестве скриптового языка jscript требуется указать тип скрипта ACM.JSCRIPT или ACM.ECMA. Если в config.xml для сайта указан плагин ACMMOD:ACM.ECMA – данный язык, будет считаться языком по умолчанию и будет использован в качестве скриптового языка везде, где язык указан не верно или не указан вовсе.

Дополнительные возможности

Операция '=' без аргумента слева производит вывод в поток вывода. Пример:

```
= f(); = «123»; = (5 + 6);
= «<tr>\n<td>\n» + i + «\n</td>\n</tr>»;
```

Для удобства, операция типа `= «abcdef»;` может быть заменена на конструкцию вида `%>abcdef<%`. При этом не требуется производить эскейпинг спец символов в поток вывода попадают все символы после открывающего `%>` и до первого `<%`. Пример:

```
for(var i = 0; i < 10; i ++){
	%><tr>
		<td align=left>
			Counter is: <%= i %>
		</td>
	</tr><%
}
```

При помощи специальной директивы `$output` можно перенаправлять поток вывода в переменную или в «никуда» указав в качестве имени переменной `null`. Пример:

```
$output( text ){
	for(var i = 0; i < 10; i++){
		%>c cc<% = i; %> da ds d a<%
	}
	fn(12);
}
```

Для возможности подключения любых Java-библиотек используется специальная директива `import`, которая подгружает в контекст java-класс (для этого надо указать полное имя класса) или java-enum, после чего ими можно оперировать как значениями или создавать экземпляры класса. Пример:

```
import java.lang.System;
System.out.println('hello world!');

import java.util.Date;
var d = new Date();
```

Текущие несоответствия

Оператор **`for( in )`** работает не в соответствии со спецификацией, а именно: при итерации по коллекциям и массивам, в значение итератора попадают не ключи свойств, а их значения, как в операторе языка C# **`foreach`**.

Пример кода в ACM:

```
var array = [1,2,3,9,11,17];
var sum = 0;
for( var element in array ){
      sum += element;
}
```

Пример корректного аналогичного кода в «правильном» ECMA:

```
var array = [1,2,3,9,11,17];
var sum = 0;
for( var index in array ){
      sum += array[index];
}
```

Справедливости ради, нади заметить, что данная форма итерации по массиву в любом случае не особо приветствуется при программировании на языке JavaScript, так как при это пролистываются все свойства объекта, но и листаемые свойства данного массива, Array.prototype и/или Object.prototype которые будут приносить в итерацию «неожиданные» элементы. Для итерации по массивам рекомендуется использовать либо «классические циклы»:

```
for( var index = array.length - 1; index >=0; --index ){
      sum += array[index];
}
```

, либо метод Array.prototype.forEach с передаваемым callback:

```
array.forEach( function(element){
      sum += element;
} );
```

Для облегчения жизни в систему добавлено две вариации данного оператора: 'for each', 'for keys' и 'for v677' они позволяют явно указывать, что требуется в каждом конкретном случае и позволяет, в последствии, заменить на «правильный» код глобальной заменой, не разбираясь в каждом конкретном случае.

Вариация v677 отражает устаревшую реализацию итерации в системе, принятую до того, как она стала совместима со стандартом Ecma-262, а именно: аналогично 'each' при итерации по массиву и аналогично 'keys' при итерации по мэпу.

Примеры кода в ACM (в результате выполнения каждого примера, значение sum становится равно нулю):

```
var array = [1,2,3,9,11,17];
var sum = 43 // sum of values;
for each( var element in array ){
      sum -= element;
}
```

```
var array = [1,2,3,9,11,17];
var sum = 43 // sum of values;
array.forEach( function(element){
      sum -= element;
});


var array = [1,2,3,9,11,17];
var sum = 21 // sum of keys;
for keys( var index in array ){
      sum -= index;
}


var array = [1,2,3,9,11,17];
var sum = 43 // sum of values;
for v677( var element in array ){
      sum -= element;
}


var object = {'1':true, '2':true, '3':true, '9':true, '11':true, '17':true};
var sum = 21 // sum of keys;
for v677( var key in object ){
      sum -= +key; // really simple conversion of string to number
}
```

Язык TPL (ACM.TPL)

ACM.TPL это очередной скриптовый шаблонный язык предназначенный для работы с символьными последовательностями любого рода. Он позволяет формировать различные типы данных: от динамических строк до сложных html, xml, pdf или других документов. Сама аббревиатура TPL означает **T**emplate **P**rocessing **L**anguage.

ACM.TPL обрабатывает шаблоны. Шаблон представляет собой последовательность символов или байтов, в которую могут быть внедрены директивы управления ACM.TPL, называемые тэгами.

Каждый тэг отвечает за определенную операцию, которая доступна при программировании шаблонов на ACM.TPL.

В части тэгов используются вычисляемые выражения (expressions) в которых могут фигурировать вызовы функций и обращения к переменным.

Чтобы указать в качестве скриптового языка tpl требуется указать тип скрипта ACM.TPL. Если в config.xml для сайта указан плагин ACMMOD:ACM.TPL – данный язык, будет считаться языком по умолчанию и будет использован в качестве скриптового языка везде, где язык указан не верно или не указан вовсе.

Тэг AUDIT

```
<%AUDIT: expression1, expression2 %>
```

Производит запись в системный лог аудита, при этом тип события рассчитывается преобразованием в строку выражения expression1, а текст сообщения – преобразованием в строку выражения expression2.

Тэг BREAK

```
<%BREAK%>
```

Используется для выхода из циклов, поддерживается в **FOR**, **ITERATE**, **SQL** и **WHILE**.

Тэг CALL

Устаревший

Тэг CATCH

```
<%TRY%>
      somecode1
<%CATCH%>
      somecode2
<%/TRY%>
```

Используется для разграничения кода предназначенного для обработки ошибки произошедшей в основном блоке тега **TRY**. Ошибка доступна в переменное exception. В данном примере код обработки ошибки обозначен somecode2.

Тэг CHOOSE

```
<%CHOOSE: expression %>
      <%MATCH: value1 %>
            <…>
      <%MATCH: value21, value22, … value2n %>
            <…>
      <%MATCH%>
            <…>
<%/CHOOSE%>
```

Осуществляет выбор выполняемого кода в соответствии со значением выражения (**expression**) являющегося аргументом тэга **CHOOSE**. Позволяет указывать любое количество значение в тэге

MATCH. При соответствии значения выражения значению указанному в **MATCH** происходит выполнение кода ограниченного соответствующим **MATCH**.

Тэг CODE

```
<%CODE: language %>
      Character sequence.
<%/CODE%>
```

Выполняет набор символов, размещенный внутри данного тега, как код\скрипт на языке соответствующем параметру **language**. Указанный язык должен быть зарегистрирован в системе. Список доступных языков можно просмотреть в интерфейсе управления «настройки сайта / плагины / визуализаторы».

Выполнение происходит непосредственно в контексте скрипта TPL, другими словами, например, **return** из скрипта вложенного в этот тэг будет обработан также, как, если бы он был прямо в коде TPL.

COMMENT

```
<%// EXEC: statement %>
<% //EXEC: statement %>
<%//EXEC: statement %>
<% // Big multiline comment
      Second line of big multiline comment
%>
```

Любой тэг начинающийся с последовательности символом **//** игнорируется до признака закрытия тэга (последовательности символов **%>**).

CONST (текст между тегами)

```
%>some characters<%
```

Символы вписанные между тэгами выводятся в поток вывода. В явном виде данный тэг не указывается. Препроцессор скрипта заменяет на этот тег теги **EXPR** если выражение в **EXPR** является константным.

Тэг CONTINUE

```
<%CONTINUE%>
```

Используется для пропуска текущей итерации цикла, подерживается в **FOR**, **ITERATE**, **SQL** и **WHILE**.

Тэг DEEPER

```
<%DEEPER: param1 = expression1, param2 = expression2, … paramN = expression %>
```

Данный тег обеспечивает рекурсивное углубление в блок кода, описанный тегом **RECURSION**. При углублении в рекурсию параметры, указанные аргументами тега **DEEPER** сохраняются и передаются в рекурсию, где они доступны в виде переменных. При выходе из рекурсии значения перечисленных параметров рекурсии будут восстановлены.

Тэг ELSE

```
<%IF: expression %>
      Some code
<%ELSE%>
      Some code
<%/ELSE%>
<%/IF%>
```

Используется для выполнения кода при невыполнении условия **IF**. Является опциональным. Среди особенностей данного тэга, хотелось бы отметить, что тэг **ELSE** может находиться в любой части тэга **IF**.

Тэг EXEC

```
<%EXEC: statement %>
<%EXEC: statement1; statement2; statement3 %>
```

Производит выполнение **statement**. Также позволяет выполнение произвольного числа операций, разделенных символом «;».

Тэг EXPR

```
<%= expression %>
```

Выводит результат выполнения выражения **expression** в поток вывода. Если выражение представляет собой константу, данный тэг будет автоматически заменен на **CONST**.

Тэг FINAL

```
<%FINAL: expression %>
      somecode
<%/FINAL%>
```

Прекращает выполнение скрипта возвращая сообщение (объект типа Message) состоящее из содержимого потока вывода, той части скрипта, что расположена внутри этого тега и имеющее тип данных (Content-Type) равный результату выполнения выражения expression, являющегося аргументом.

Тэг FOR

```
<%FOR: statementInit; expressionCondition; statementLoop %>
      somecode
<%/FOR%>
```

Выполняет **statementInit**, производит циклическое выполнение **somecode** до тех пор, пока **expressionCondition** вычисляется как **TRUE**. В конце каждого цикла производится выполнение **statementLoop**. Одним из вариантов преждевременного выхода из цикла является тег **BREAK**.

Тэг FORMAT

```
<%FORMAT: formatConstant %>
      somecode
<%/FORMAT%>
```

Задает режим интерпретации пустых символов (whitespace) при компиляции шаблона. Константным выражением **formatConstant** задается один из режимов генерации потока вывода: **'default'**, **'no_ident'**, **'no_tags'**, **'js'**, **'xml'**. Режимом по умолчанию является режим **default**.

В формате **default** все whitespace символы попадают в поток вывода. В формате **no_ident** игнорируется все whitespace символы в начале строки. В формате **no_tags** любая последовательность whitespace символов расположенная между текущим выводом и соседними тэгами (директивами) языка TPL заменяется на один символ пробела. В формате **js** удаляются все пробелы в начале и конце строк. В формате **xml** любая последовательность whitespace символов заменяется на один символ пробела.

Тэг IF

```
<%IF: expression %>
      somecode1
<%ELSE%>
      somecode2
<%/ELSE%>
<%/IF%>
```

При выполнении условия expression являющегося аргументом тэга **IF** будет выполнен код расположенный в части «**somecode1**». В противном случае «**somecode2**». Тэг **ELSE** является опциональным.

Тэг IGNORE

```
<%IGNORE%>
      anything
<%/IGNORE%>
```

Игнорировать любой набор данных и кода внутри данного тэга. Может использоваться для написания многострочных комментариев или для отключения части шаблона или скрипта.

Тэг INCLUDE

```
<%INCLUDE: filename %>
```

Подключение файла из той-же директории, где расположен текущий скрипт. Таким образом, данный тэг возможен только в тех шаблонах, которые описаны в виде набора файлов в папке, но никак не для отдельно описанных шаблонов.

Тэг ITERATE

```
<%ITERATE: variable : expression %>
      somecode
<%/ITERATE%>
```

Производит циклическое выполнение **somecode** последовательно устанавливая указанную переменную в соответствующий элемент массива или коллекции или ключа объектов типа **Map** или **Lookup**. При этом выражение **expression** должно возвращать значение одного из следующих типов: **null**, массив, коллекция, **Map** или **Lookup**. Если значение **expression** null, выполнение цикла не произойдет ни одного раза. Одним из вариантов преждевременного выхода из цикла является тег **BREAK**.

По завершению цикла, переменная **variable**, указанная как счетчик цикла будет восстановлена в то же состояние, в котором она была до начала цикла.

Тэг LOG

```
<%LOG: expression1, expression2 %>
```

Производит запись в системный лог, при этом тип события рассчитывается преобразованием в строку выражения **expression1**, а текст сообщения – преобразованием в строку выражения **expression2**.

Тэг MATCH

```
<%MATCH: value %>
      somecode
<%MATCH: value1, value2, … valueN %>
      somecode
<%MATCH%>
      somecode
```

Используется в тэге **CHOOSE** для описания действий под определенное значение выражения, являющегося аргументом тега **CHOOSE**. Имеется возможность указывать несколько значений. Если значение не указано работает «для всех остальных вариантов», в таком случае этот тег должен быть последним среди тегов **MATCH** в данном **CHOOSE**.

Тэг OUTAPPEND

```
<%OUTAPPEND: variable %>
      somecode
<%/OUTAPPEND%>
```

Определяет текущим потоком вывода, символьный поток, который будет добавляться к преобразованной к строковому значению переменной **variable** указанной аргументом к данному тэгу.

Тэг OUTPUT

```
<%OUTPUT: variable %>
      somecode
<%/OUTPUT%>
```

Определяет текущим потоком вывода, символьный поток, который будет сохранен в виде строки в переменную **variable** указанную аргументом к данному тэгу. Если в качестве переменной указано **null**, поток вывода будет подавлен.

Тэг PROCESS

Устаревший

Тэг RECURSION

```
<%RECUSRION: param1 = expr1, param2 = expr2, … paramN = exprN %>
      somecode
<%/RECURSION%>
```

Данный тег обеспечивает возможность создания обособленных блоков кода. Параметры, указанные аргументами данного тега сохраняются и передаются в блок кода, где они доступны в виде переменных. При выходе из блока значения перечисленных переменных блока будут восстановлены.

Данный тег позволяет производить одновременно описание и вызов анонимного метода.

```
<%RECUSRION: param1 = expr1, param2 = expr2, … paramN = exprN %>
      somecode
      <%DEEPER: param1 = expr1, param2 = expr2, … paramN = exprN %>
      somecode
<%/RECURSION%>
```

В паре с тэгом **DEEPER** реализуется возможность «рекурсии на лету», определяется рекурсивный блок и набор параметров рекурсии, которые доступны в виде переменных и имеют независимые значения при каждом вхождении в рекурсию. Для углубления рекурсии используется тег **DEEPER**. При углублении в рекурсию параметры, указанные аргументами тега **DEEPER** сохраняются и передаются в рекурсию, где они доступны в виде переменных. При выходе из рекурсии значения перечисленных параметров рекурсии будут восстановлены.

Тэг REDIRECT

```
<%REDIRECT: expression[, expressionMoved] %>
```

Прекращает выполнение скрипта возвращая сообщение (объект типа Message) содержащее команду перенаправления на адрес вычисленный выражением expression, являющегося аргументом данного тега. Весь поток вывода данного шаблона обнуляется. Необязательный аргумент **expressionMoved** определяет является ли данное перенаправление постоянным. Если он опущен или вычисляется в значение FALSE, перенаправление будет временным.

Тэг RETURN

```
<%RETURN: expression %>
```

Прекращает выполнение скрипта и возвращает значение – результат выполнения выражения expression, являющегося аргументом данного тега. Весь поток вывода данного шаблона обнуляется.

Тэг SET

```
<%SET: variable %>
      Character sequence.
<%/SET%>
```

Присваивает в указанную переменную набор символов, размещенный внутри данного тега. Внимание: набор символов внутри тега не является шаблоном и не обрабатывается.

Тэг SQL

```
<%SQL: expressionConnection, expressionQuery %>
      somecode
<%/SQL%>
```

Выполняет запрос SQL на выборку из базы данных и для каждой строки результата производит выполнение вложенной части шаблона, передавая в него соответствующую запись результата в переменной Record в виде Map. Одним из вариантов преждевременного выхода из цикла является тег **BREAK**. Имя коннекта вычисляется результатом выполнения выражения **expressionConnection**, запрос вычисляется результатом выполнения выражения **expressionQuery**. Данный тег предназначен ТОЛЬКО для выполнения запросов на выборку, возвращающих resultset, для запросов не возвращающих resultset, например INSERT, UPDATE, DELETE используется тег **SQLEXEC**. В целях защиты от SQL-injection в качестве запроса допускается только единичная команда SQL.

Тэг SQLBATCH

```
<%SQLBATCH: expressionConnection, expressionQueryArray %>
```

Выполняет группу запросов к базе данных. Имя коннекта вычисляется результатом выполнения выражения **expressionConnection**, запросы вычисляется результатом выполнения выражения **expressionQueryArray**, которое должно возвращать массив запросов или один отдельный запрос. Данный тэг предназначен для запросов на изменение БД, например: UPDATE, INSERT или DELETE. В целях защиты от SQL-injection в качестве каждого отдельного запроса допускается только единичная команда SQL.

Тэг SQLEXEC

```
<%SQLEXEC: expressionConnection, expressionQuery[, parameters] %>
```

Выполняет запрос к базе данных. Имя коннекта вычисляется результатом выполнения выражения **expressionConnection**, запрос вычисляется результатом выполнения выражения **expressionQuery**. Данный тэг предназначен для запросов на изменение БД, например: UPDATE, INSERT или DELETE. В целях защиты от SQL-injection в качестве запроса допускается только единичная команда SQL. Если указан аргумент parameters то будет подготовлен запрос и параметры будут использованы в нем (как это по русски написать?)

Тэг SQLUSE

```
<%SQLUSE: expressionConnection %>
      somecode
<%/SQLUSE%>
<%SQLUSE: expressionConnection1, … expressionConnectionN %>
      somecode
<%/SQLUSE%>
```

Резервирует подключение к БД, чье имя является результатом вычисления выражения **expressionConnection**. Далее, при выполнении somecode, если в данном коде встречаются обращения к тому-же коннекту БД (при помощи тэгов **SQL**, **SQLBATCH** или **SQLEXEC**) будет использован зарезервированный коннект, что значительно ускорит время организации соединения с СУБД. Допускается одновременное резервирование нескольких соединений к разным базам данных.

Тэг TRY

```
<%TRY%>
      somecode1
<%CATCH%>
      somecode2
<%/TRY%>
```

Используется для перехвата ошибок в коде. Если происходит ошибка в коде **somecode1**, управление передается в **somecode2**, а сама ошибка помещается в переменную **exception**.

Тэг WHILE

```
<%WHILE: expression %>
      somecode
<%/WHILE%>
```

Производит циклическое выполнение **somecode** до тех пор, пока **expression** вычисляется как TRUE. Одним из вариантов преждевременного выхода из цикла является тег **BREAK**.

Пустой тэг

<% любое число пробелов, переводов строк, табуляции %>

Пустой тэг игнорируется препроцессором TPL. Может быть использован для сохранения логического форматирования кода при отсутствии выдачи пустых символов в поток вывода, в этом случае код будет выглядеть следующим образом:

```
<%
      …
      %><%ITERATE: element : array %><%          // iterate trough array
            %><%IF: element > 2 %><%
                  %><%EXEC: counter ++ %><%      // increment counter
                  %><td valign=«top»><%
                  %><%= element %><%
                  %></td><%
            %><%/IF%><%
      %><%/ITERATE%><%
      …
%>
```

Отдельно хочется заметить, что когда система находится в режиме отладки (DEBUG), а скрипт использует **FORMAT: 'default'** вывод пустых тэгов будет попадать в поток вывода, помогая структурно разобраться с отладочными задачами на клиенте (например в браузере пользователя).

Формат MAP (ACM.MAP)

Данный формат позволяет описывать структурированные данные в виде XML.

Такой формат используется системой для хранения и передачи данных, как основной формат настроечных файлов, как базовый формат описания типов и т.п. Уникальность данного формата в сравнении с другими языками поддерживаемыми в системе заключается в том, что он является двусторонним и обратимым – то есть, он может быть использован для любого числа преобразований из вида XML во внутреннее представление и обратно. Также, имеются доступные в скриптовых языках функции для преобразования структурированных данных из внутреннего представления в этот формат и обратно.

В определенных ситуациях этот формат может быть удобен для описания структурированных данных при сборке сайта в описаниях типов.

Структурированные данные в системе представлены в виде объекта с набором свойств, именем каждого свойства является строка, а в качестве значения могут быть как такие же объекты с наборами свойств, так и массивы любых объектов или объекты любых других типов. Такие объекты называются MAP – они очень похожи на понятие Object в JavaScript. В качестве результата преобразования из XML мы не можем получить число или строку – мы можем получить только объект типа map – однако, в его свойствах могут быть и числа и строки и логические значения и даты и другие объекты типа map и вообще любые объекты.

Формат предусматривает различные варианты описания данных создавая определенную гибкость дающую возможность использовать функции для преобразования из XML в MAP для адекватного разбора практически любых XML данных.

Для преобразования из map в xml в системе предусмотрено два режима:

1) «читаемый» - предназначен для более легкого и удобного чтения глазами и разбора внешними средствами, результатирующий XML содержит выравнивание и отступы, содержит минимальное количество служебных атрибутов
2) «нечитаемый» - предназначен для более быстрого разбора и компактного храниения данных, в этом варианте формата система удаляет все лишние пробелы, переводы строк, табуляцию и добавляет служебные атрибуты для быстрого чтения без применения анализа содержания.

Простые примеры

Перед тем как перейти к описанию подробностей, хотелось бы привести несколько простейших примеров, которые должны помочь сориентироваться в том, о чём идет речь и продемонстрировать гибкость формата.

Четыре следующих примера XML будут поняты как Map с двумя свойствами: свойство **`name`** со значением **`Alexander`** и свойство **`surmane`** со строковым значением **`Kharitchev`**, что в виде Map может быть представлено как **`{ name : «Alexander», surname : «Kharitchev» }`**. Они показывают разные способы передачи имени и значения свойства, эти способы предназначены для удобства описания настроек, для гибкого разбора внешних XML данных и для поддержки предыдущих версий данного формата.

Пример №1 – использование элементов для хранения значений – «родной формат» –в таком виде система сохранит данный map при сохранении в «читаемом» виде:

```
<xml>
	<name>Alexander</name>
	<surname>Kharitchev</surname>
</xml>
```

Пример №2 – использование атрибутов элемента для хранения значений:

```
<aaa name=«Alexander» surname=«Kharitchev»/>
```

Пример №3 – использование атрибута value для хранения значения:

```
<xxx>
	<name value=«Alexander»/>
	<surname value=«Kharitchev»/>
</xxx>
```

Пример №4 – использование атрибута для указания имени свойства:

```
<zzz>
	<param key=«name» value=«Alexander»/>
	<param key=«surname»>Kharitchev</param>
</zzz>
```

Не любая последовательность символов может стать валидным именем элемента XML – для передачи таких элементов в формате ACM.MAP предусмотрено следующее правило: если имя элемента **`param`** и он имеет атрибут **`key`** – то именем свойства считается значение атрибута **`key`**. Использование этой возможности продемонстрировано в примере №4. Также это может быть удобным если XML строится на основании внешних данных и значения имен свойств являются динамическими.

Типы значений свойств

В приведенных выше примерах рассматривался простой случай – значения свойств имели строковый тип. Для указания указания типа свойства система использует атрибут **`class`**, если этот атрибут не указан действуют следующие правила:

1) если элемент содержит дочерние элементы – его тип map;
2) если элемент пуст и содержит атрибуты – его тип map;
3) в другом случае – его тип string.

При сохранении в «читаемом» виде система не указывает типа значения у коренного элемента и у простых строковых полей. Однако, при сохранении в «нечитаемом» виде система всегда указывает типы атрибутов, поэтому пример, рассмотренный в предыдущей главе, при сохранении в «нечитаемом» виде имел бы следующий вид:

```
<data class="map"><name class="string" type="inner">Alexander</name><surname class="string" type="inner">Kharitchev</surname></data>
```

В «нечитаемом» режиме, наряду с атрибутом **`class`** система всегда сохраняет дополнительные параметры, свойственные каждому типу данных – это позволяет без ухищренных алгоритмов и лишних проверок производить быстрый разбор данных. В этом примере – атрибут **`type="inner"`** является дополнительным параметром для типа string, он описывает как именно сохранена строка.

Разделим все поддерживаемые типы значений на следующие группы:

1) базовые типы – простейшие примитивные типы имеющие широкое применение, универсальные, используются для описания значений;
2) специальные – типы описывающие специальные объекты системы ACM.CMS, используются при описании типов или полей;
3) системные – внутрисистемные типы, используются системой для внутреннего хранения данных и доступные только из системных вызовов.

Базовые типы

Простейшие примитивные типы имеющие широкое применение, универсальные, используются для описания значений.

Тип boolean

Тип значения – логический. В качестве самого значения может быть указана TRUE или FALSE, система не чувствительна к заглавным или прописным буквам. А также использует внутренний метод по разбору логических значений, который в состоянии понимать и значения YES/NO и значения 1/0… В принципе – значению TRUE соответствуют только: сам TRUE, YES и любое не нулевое числовое значение.

Пример:

```
<xml>
	<registered class=«boolean»>true</registered>
	<active class=«boolean»>false</active>
</xml>
```

Тип date

Тип значения – дата. Значение передается в атрибуте **`value`** и содержит число миллисекунд прошедших с 1 января 1970 года (разумеется, если дата до 1970 года – значение отрицательное). Поддерживается специальное значение **`NOW`**, в этом случае дата равна текущей дате.

Пример:

```
<xml>
	<birthday class=«date» value=«12431311234»/>
	<modified class=«date» value=«NOW»/>
	<date class=«date» value=«12431311234»>16 march 1997</date>
</xml>
```

Тип double

Тип значения – натуральное число в двойной точностью (стандартное число с плавающей точкой).

Пример:

```
<xml>
	<number1 class=«double»>-14.5</number1>
	<pi class=«double»>3.14</pi>
</xml>
```

Тип float

Тип значения – натуральное число в одинарной точностью (половина точности от стандартного числа с плавающей точкой).

Пример:

```
<xml>
	<number1 class=«double»>-14.5</number1>
	<pi class=«double»>3.14</pi>
</xml>
```

Тип integer

Тип значения – 32-битное знаковое целое число.

Пример:

```
<xml>
	<number1 class=«integer»>-14</number1>
	<number2 class=«integer»>3</number2>
</xml>
```

Тип long

Тип значения – 64-битное знаковое целое число.

Пример:

```
<xml>
	<number1 class=«long»>-14</number1>
	<number2 class=«long»>3</number2>
</xml>
```

Тип map

Тип значения – map – структурированные данные. Свойства могут быть описаны как вложенными элементами, так и атрибутами элемента.

Пример:

```
<xml class=«map»>
	<myx class=«map»>
		<id class=«string»>f3c56w35366d982sgf756dd463h0</id>
		<rank class=«integer»>1</rank>
		<married class=«boolean»>yes</boolean>
		<contact class=«map» name=«myx»>
			<icq class=«string»>2206241</icq>
			<web class=«string»>http://myx.ru</web>
		</contact>
	</myx>
</xml>
```

Системе не обязательно наличие указания на class map. Однако он помогает не тратить время на определение реального типа посредством поиска вложенных элементов. В сокращенном, но, тем не менее, идентичном виде данный нобор данных может выглядеть следующим образом:

```
<xml>
	<myx>
		<id>f3c56w35366d982sgf756dd463h0</id>
		<rank class=«integer»>1</rank>
		<married class=«boolean»>yes</boolean>
		<contact name=«myx»>
			<icq>2206241</icq>
			<web>http://myx.ru</web>
		</contact>
```

```
	</myx>
</xml>
```

Тип null

Тип значения – null.

Пример:

```
<xml>
	<null1 class=«null»/>
	<null2 class=«null»/>
</xml>
```

Тип number

Тип значения – числовой. Числовое значение может быть как целым, так и плавающим, за конкретный тип значения отвечает атрибут **`type`**, принимающие одно из 4ех значений: **`double`**, **`float`**, **`long`**, **`integer`**. Значением атрибута **`type`** по умолчанию (если не указан или указан неверно) является **`double`**.

```
<xml>
	<number1 class=«number» type=«double»>-14</number1>
	<number2 class=«number» type=«float»>3.14</number2>
	<number3 class=«number» type=«long»>-1234523564244</number3>
	<number4 class=«number» type=«integer»>2423</number4>
	<number5 class=«number»>2206241</number5>
</xml>
```

Тип string

Тип значения – строка. Дополнительный атрибут **`type`** определяет метод описания строкового значения. Атрибут **`type`** принимает следующие значения:

1) **`empty`** – пустая строка (строка нулевой длинны)
2) **`inner`** – строка сохранена в теле элемента
3) **`cdata`** – строка сохранена в теле элемента в виде секции CDATA

На самом деле, использование атрибута type не обязательно и не влияет на способность системы разобрать значение поля. Сама система использует значение empty для более быстрого разбора прустой строки, а на остальные значения этого атрибута или на его отсутствие внимания не обращает. При сохранении в формат XML система выставляет эти атрибута, для других систем, которым они могут быть необходимы для разбора.

Пример:

```
<xml>
	<stringEmpty1 class=«string» type=«empty»/>
	<stringEmpty2 class=«string»/>
	<stringEmpty3 class=«string»></stringEmpty3>
	<stringInline1 class=«string»>строка!</stringInline1>
	<stringInline2 class=«string» type="inline">строка!!!!!</stringInline2>
	<string3 class=«string»><![CDATA[&copy; 2005 ООО <b>"ZyShop"</b>.]]></string3>
	<string4 class=«string» type="cdata"><![CDATA[ <><><><<<>>><><><> ]]></string4>
</xml>
```

Специальные типы

Типы описывающие специальные объекты системы ACM.CMS, используются при описании типов или полей.

Тип buffer

Тип значения – buffer. Дополнительный атрибут **`type`** определяет метод описания значения. Атрибут **`type`** принимает следующие значения:

1) **`empty`** – пустой буфер (нулевой длинны)
2) **`base64`** – буфер сохранен в теле элемента в формате base64
3) **`reference`** – буфер сохранен отдельно во внешнем источнике, идентификатор данных указывается в атрибуте **`reference`**.

Тип bytes

Тип значения – массив байтов. Дополнительный атрибут **`type`** определяет метод описания значения. Атрибут **`type`** принимает следующие значения:

1) **`empty`** – пустой массив (нулевой длинны)
2) **`base64`** – массив сохранен в теле элемента в формате base64
3) **`reference`** – массив сохранен отдельно во внешнем источнике, идентификатор данных указывается в атрибуте **`reference`**.

Тип copier

Тип значения – copier. Дополнительный атрибут **`type`** определяет метод описания значения. Атрибут **`type`** принимает следующие значения:

1) **`empty`** – пустой copier (нулевой длинны)
2) **`base64`** – copier сохранен в теле элемента в формате base64
3) **`reference`** – copier сохранен отдельно во внешнем источнике, идентификатор данных указывается в атрибуте **`reference`**.

Тип fieldset

Тип значения – набор полей.

Тип message

Тип значения – message. Дополнительный атрибут **`message_type`** определяет метод хранения тела сообщения. Атрибут **`message_type`** может принимать следующие значения:

1) **`empty`** – сообщение не имеет тела
2) **`base64`** – тело сообщения лежит закодированным в формате base64 в тексте элемента
3) **`text`** – тело сообщения лежит в тексте элемента
4) **`sequence`** – тело сообщения составное, состоит из нескольких сообщений

Тип script

Тип значения – скрипт.

Системные типы

Внутрисистемные типы, используются системой для внутреннего хранения данных и доступные только из системных вызовов.

Тип extra

Тип restorable

Тип serialized

Выражения ACM.EVALUATE / ACM.CALC

Осуществляет «вычисление выражений». Выражения воспринимаемые и выполняемые ACM.EVALUATE абсолютно идентичны expressions в ACM.JSCRIPT.

Пример:

Выражение ACM.EVALUATE `this.state == 2` эквивалентно соответствующему коду на ACM.JSCRIPT: `return this.state == 2;`.

Формат fieldset (ACM.FIELDSET и ACM.RETRIEVE)

Fieldset – это описание набора полей. Оно может быть использовано для формирования форм, для валидации данных, для преобразования map в map содержащий нужные поля, нужных типов и т.п. Данный формат используется в различных местах, например, для описания аргументов функции в описании типа в виде XML или для описания полей форм, также, его можно использовать как скриптовый язык, для этого в качестве скриптового языка требуется указать ACM.FIELDSET или ACM.RETRIEVE в соответствующем параметре описания свойств скрипта.

Отличие ACM.FIELDSET от ACM.RETRIEVE заключается в том, что первый создает набор полей (fieldset) и этот набор полей является результатом выполнения, а второй создает набор полей (fieldset) для того, чтобы прогнать через него входящие данные, а преобразовынные данные уже будут являться результатом выполнения.

Fieldset описывается в виде XML и имеет следующий вид:

```
<fieldset class=”fieldset”>
      <field атрибуты-поля />
      …
</fieldset>
```

Пример описания:

```
<fields class="fieldset">
      <field
            id="text"
            class="string"
            title="Текст"
            type="text"
            variant="html"
            max="128k"/>
      <field
            id="locals"
            class="set"
            title="Сайты"
            variant="multiselect"
            lookup_expression="NewsItem.getLocals()"/>
      <field
            id="image"
            class="binary"
            title="Картинка"
            max="2m"/>
      <field
            id="keywords"
            class="string"
            title="Ключевые слова"
            type="text"
            max="64k"
            hint="тучу слов через запятую..."/>
      <field
            id="anounce"
            class="string"
            type="text"
            title="Анонс"
            hint="Можно не заполнять - сделается само из текста тогда"
            max="4k"/>
</fields>
```

В тех случаях, когда fieldset описывается для переопределения другого fieldset также можно указать атрибут **merge**, который определяет расположение нового поля, заменяющего старое поле в случае конфликта имен, а также позволяет указать что новое определение полей полностью заменяет старое, он может принимать значения: **default** (значение по умолчанию, при конфликте имен старое поле заменяется новым, которое будет расположено на месте старого поля), **append** (при конфликте имен старое поле заменяется новым, которое будет расположено на месте нового поля) или **replace** (новое описание целиком заменяет старое). Пример описания

```
<fieldset class="fieldset" merge="append">
      <field атрибуты-поля />
      …
</fieldset>
```

Обязательным атрибутом любого поля является атрибут **id**, определяющий имя описываемого поля. Каждое поле в наборе имеет своё имя. Имена полей в одном fieldset уникальны – не может быть двух полей в одном fieldset с одинаковыми именами.

Также, обязательным атрибутом поля является атрибут **class** – он определяет тип данных для поля. Когда fieldset используется для преобразования map – все поля осуществляют преобразование данных в типы, соответствующие атрибуту **class**. Если указано значение по умолчанию, а в преобразуемом map нет данных, соответствующих данному полю – в результатирующий map будет сохранено именно это значение.

Возможны следующие значения атрибута **class**:

boolean	Булевое поле, способно принимать значение **true** или **false**. Значение по умолчанию указывается атрибутом «default», если оно не указано, считается равным false. Пример: `<field class="boolean" id="aaa" default="false"/>`
int int32	Поле, хранящее целочисленное значение в виде 32-битного int. Значение по умолчанию указывается атрибутом «default», если оно не указано, считается равным 0. Пример: `<field class="int32" id="aaa" default="5" min="1" max="9"/>`
integer long	Поле, хранящее целочисленное значение в виде 64-битного long. Значение по умолчанию указывается атрибутом «default», если оно не указано, считается равным 0. Пример: `<field class="integer" id="aaa" default="5" min="1" max="9"/>`
floating	Поле, хранящее значение в виде числа в плавающей точкой в виде double. Значение по умолчанию указывается атрибутом «default», если оно не указано, считается равным 0.0. Пример: `<field class="floatin" id="aaa" default="5.1" min="1" max="9"/>`
string	Поле, хранящее строковое значение. Значение по умолчанию указывается атрибутом «default». Пример: `<field class="string" id="aaa" default="n/a"/>`

date	Поле, хранящее дату. Значение по умолчанию указывается атрибутом «default». Имеется возможность в качестве значения по умолчанию указать строку NOW в таком случае будет подставлена текущая дата. Если значение по умолчанию описано в виде строки, необходимо использовать атрибут format для описания формата данной строки. Пример: `<field class="date" id="aaa" default="NOW"/>`
set	Поле, хранящее неупорядоченную коллекцию уникальных значений
list	Поле, хранящее упорядоченную коллекцию любых значений

map	Поле, хранящее объект типа map – ассоциативный массив.
	Понимает атрибуты «default» и «fieldset», которык могут быть указаны в форматах ACM.MAP и ACM.FIELDSET соответственно.
	Если не указаны и «default» и «fieldset», то значение retrieve будет равно null если входящий объект не определен, что позволяет осуществлять такие проверки как: !value.
	Примеры:
	`<field class="map" id="aaa"/>`
	`<field class="map" id="bbb">`
	`<default/>`
	`</field>`
	`<field class="map" id="ccc">`
	`<default field1=”asd” field2=”cxvxc” field3=”sdsd”/>`
	`</field>`
	`<field class="map" id="ddd">`
	`<default>`
	`<field1>value1</field1>`
	`<field2 value=”value2”/>`
	`<field3 class=”boolean">true</field3>`
	`</default>`
	`</field>`
	`<field class="map" id="eee">`
	`<fieldset class=”fieldset”>`
	`<field id=”aaa” class=”boolean” default=”true”/>`
	`</fieldset>`
	`</field>`

object	Поле, хранящее любые данные.
evaluate	Значение данного поля не хранится, а вычисляется при обращении, выражение, для вычисления результата хранится в атрибуте «expression» Пример: `<field class="evaluate" id="aaa" expression="Library.getSomething()"/>`
binary	Поле, хранящее бинарные данные (например файл, message или буфер)
guid	
owner	
money	

Поскольку XML является легко расширяемым форматом, в каждом конкретном случае использования fieldset могут быть также определены дополнительные атрибуты к полям и к всему набору полей в целом.

В случае, когда fieldset используется для описания формы имеется возможность указать название (заголовок) поля и варианты редактора подходящие для этого поля. Заголовок поля указывается атрибутом **`title`**. Основной тип редактора поля указывается атрибутом **`type`** который, по умолчанию, равен значению указанному в атрибуте **`class`**. Вариант редактора указывается в атрибуте **`variant`**. Если поле формы предназначено только для чтения – указывается атрибут **`constant`** равный **`true`**.

Ниже приведена таблица нестандартных (в качестве стандартных редакторов доступны редакторы для всех возможных значений атрибута **`class`**) редакторов поля доступных в интерфейсе управления:

Тип (type)	**Вариант (variant)**	**Описание**
text		Редактирование строкового значения в виде многострочного текста.
text	html	Редактирование строкового значения в визуальном HTML редакторе

Преобразование XSL (ACM.XSL / ACM.XSLT)

Если в качестве типа скрипта указано ACM.XSL или ACM.XSLT – код скрипта компилируется в XSL трансформатор, при каждом выполнении он будет осуществлять преобразование входных данных. Для XSL входные данные будут выглядеть как map преобразованный в XML по правилам ACM.MAP. Результатом выполнения скрипта будет результат наложения шаблона XSL на входные данные.

Псевдоязыки ACM.TXT / ACM.TEXT / ACM.NULL

Данный типы скрипта является псевдоязыками, так как они не производит совершенно никаких действий.

Результатом выполнения скрипта типа ACM.TXT / ACM.TEXT является исходный код этого скрипта, возвращаемый в виде строки.

Результатом выполнения скрипта типа ACM.NULL является значение null независимо от исходного кода этого скрипта.

Системные API

Скриптовые языки позволяют описывать алгоритмы, но они не имели бы практически никакого смысла без объектов и методов, предоставленных системой в контекст выполнения программы. Функционал системы предоставляется в виде набора объектов, каждый из которых предоставляет определенный набор методов – API.

Особенно хочется обратить внимание, что возможности скриптов не ограничиваются системными API, т.к. система позволяет использовать в процедурных скриптовых языках не только эти API, но и любые библиотеки java, в том числе и не входящие в состав системы. Однако, несмотря на такие возможности, в системе выделен набор API которые: дают стандартный и простой доступ к основным системным функциям, призваны решать типовые задачи, возникающие при сборке сайтов или просто являются достоянием прошлого.

Application API

Версия последнего изменения данного API: 655

Данный API предоставляет возможность из любого скрипта, выполняющегося в любом контексте иметь доступ к общим параметрам виртуального домена (сервера). При этом через этот API имеется не только возможность работать с параметрами домена, но и обмениваться произвольными параметрами между различными скриптами домена на усмотрение сборщика сайта.

Доступ к методам API осуществляется через объект с именем **`Application`**, который регистрируется в глобальном контексте домена.

Application

```
Object get(
              String name)


int    getInt(
              String name)


int    getInt(
              String name,
              int default)


long   getLong(
              String name)


long   getLong(
              String name,
              long default)


double getDouble(
              String name)


double getDouble(
              String name,
              double default)
```

```
String getString(
              String name)


String getString(
              String name,
              String default)


Object put(
              String name,
              Object value)


Map    getData()


Entry  getFinderResultEntry(
              String request)


String getFinderResultLocation(
              String request)


String getSystemUserName()


String getSystemCountryCode()


String getSystemLanguage()


Share[]       getDomainSharePoints(
              boolean all)


Node   getControlNodeForShareName(
              String shareName)
```

Share

Объекты типа **`Share`** описывают точки доступа определенные в виртуальном домене.

```
String getPath()
```

Возвращает путь в системном дереве на который указывает данная точка доступа.

```
String getAlias()
String getKey()      // same as getAlias()
```

Возвращает имя точки доступа. Это имя представляет собой доменное имя (например: www.myx.ru) при обращении к которому должна срабатывать данная точка доступа.

```
String getTitle()
```

```
int    getAuthType()

String getSkinner()

int    getLanguageMode()

String getLanguageName()

boolean       getCommandMode()
```

Node

Объекты типа **`Node`** представляют собой узла дерева системной иерархии.

```
String getTitle()

String getIcon()

Command[]     getForms()

Command[]     getCommands()

Node[] getChildren()

Node[] getChildrenExternal()

Node[] getContents()
       Что угодно
```

Admin

```
String getCommonActor(
              controlPath)

Command[]     filterAccessible(
              controlPath,
              commands)

Node[] filterAccessible(
              controlPath,
              nodes)

Node   childForPath(
              node,
              path)
```

Calendar API

Версия последнего изменения данного API: 673

Данный API предоставляет набор различных функций для работы с келендарем, датой и временем.

Доступ к методам API осуществляется через объект с именем **`Calendar`**, который регистрируется в глобальном контексте домена.

В методах getXXX без параметра date или с параметром date равным null, за дату будет принят текущий момент времени. В методах с параметром date типа Number, переданное число должно представлять собой количество миллисекунд с 1 января 1970 года по UTC.

Calendar

```
int    getWeekOfYear()
int    getWeekOfYear(Date date)
int    getWeekOfYear(Number date)


int    getWeekOfMonth()
int    getWeekOfMonth(Date date)
int    getWeekOfMonth(Number date)


int    getDayOfYear()
int    getDayOfYear(Date date)
int    getDayOfYear(Number date)


int    getDayOfMonth()
int    getDayOfMonth(Date date)
int    getDayOfMonth(Number date)


int    getDayOfWeek()
int    getDayOfWeek(Date date)
int    getDayOfWeek(Number date)


int    getYear()
int    getYear(Date date)
int    getYear(Number date)


int    getMonth()
int    getMonth(Date date)
int    getMonth(Number date)


int    getHour()
int    getHour(Date date)
int    getHour(Number date)
```

Значения от 0 до 12, где полночь и полдень представлены значением 0 (не 12).

```
int    getHourOfDay()
int    getHourOfDay(Date date)
int    getHourOfDay(Number date)
```

Значения от 0 до 23.

```
int    getMinute()
int    getMinute(Date date)
int    getMinute(Number date)
```

Значения от 0 до 59.

```
int    getSecond()
int    getSecond(Date date)
int    getSecond(Number date)
```

Значения от 0 до 59.

```
int    getPm()
int    getPm(Date date)
int    getPm(Number date)
```

0 – AM, 1 – PM.

```
Map    get()
Map    get(Date date)
Map    get(Number date)
```

Возвращает набор всех календарных полей со значениями отражающими указанную аргументом **`date`** дату. Вариант без аргумента **`date`** работает с текущим моментом времени.

```
long   build(
              Map data)
```

Создает дату по переданному набору полей, неуказанные в переданном наборе поля заполняются значениями на основе ткущего момента времени.

Пример:
`Calendar.build( YEAR = 2003, MONTH = 5, DAY_OF_MONTH = 3, HOUR_OF_DAY = 0, MINUTE = 0, SECOND = 0 )`

```
long   roundTime(
              Date date,
              long round)
long   roundTime(
              long date,
              long round)
```

Округляет дате, переданную аргументом **`date`** до точности в миллисекундах указанную аргументом round. Например: **`date = Calendar.roundDate( date, 6000 )`** округлит время в переменной **`date`** до точности в одну минуту.

```
Object[]      ARRAY_MONTHNAMES3
```

```
Object[]      ARRAY_WEEKDAYS2
```

```
Date   NOW
```

Специальная константа всегда соответствующая текущему времени, особенно актуальна при сохранении и передаче данных.

```
Date   UNDEFINED
```

Константа указывающая неопределенную дату.

Control API

Версия последнего изменения данного API: 635

Доступ к методам API осуществляется через объект с именем **`Control`**, который регистрируется в глобальном контексте домена.

Control

```
Command      createCommand(
             String name,
             Object title)


Command      createCommandSplitter()


Options      createOptions()


Field  createField(
             String type,
             Map attributes)


Field  createFieldBoolean(
             String id,
             Object title,
             boolean default)


Field  createFieldBinary(
             String id,
             Object title,
             int lengthLimit)


Field  createFieldGuid(
             String id,
             Object title)


Field  createFieldOwner(
             String id,
             Object title)


Field  createFieldDate(
             String id,
             Object title,
             Date defaultValue)
```

```
Field  createFieldDate(
             String id,
             Object title,
             long defaultValue)


Field  createFieldEvaluate(
             String id,
             Object expression)


Field  createFieldTemplate(
             String id,
             Object title,
             String defaultValue)


Field  createFieldString(
             String id,
             Object title,
             Object defaultValue)


Field  createFieldString(
             String id,
             Object title,
             Object defaultValue,
             int minLength,
             int maxLength)


Field  createFieldInteger(
             String id,
             Object title,
             int defaultValue)


Field  createFieldInteger(
             String id,
             Object title,
             int defaultValue,
             int minValue,
             int maxValue)


Field  createFieldFloating(
             String id,
             Object title,
             double defaultValue)


Field  createFieldFloating(
             String id,
             Object title,
             double defaultValue,
             double minValue,
             double maxValue)


Field  createFieldSet(
             String id,
             Object title,
             Set defaultValue)
```

```
Field  createFieldList(
             String id,
             Object title,
             List defaultValue)


Field  createFieldMap(
             String id,
             Object title,
             Map defaultValue)


Field  createFieldObject(
             String id,
             Object title,
             Object defaultValue)


Basic  createBasic(
             String key,
             Object title,
             Map data)


Node   relativeNode(
             Node root,
             String path)


Fieldset     createFieldset()


Fieldset     createFieldset(
             String id)


Fieldset     createFieldset(
             Fieldset base,
             Fieldset tail)


Fieldset     createFieldsetConstant(
             Fieldset fieldset)


String serializeFieldset(
             Fieldset fieldset,
             boolean readable)


Fieldset     materializeFieldset(
             String definition)


Form   createSimpleForm(
             Object title,
             Map target,
             Fieldset fieldset,
             Object result)
```

Create API

Версия последнего изменения данного API: 671

Доступ к методам API осуществляется через объект с именем **`Create`**, который регистрируется в глобальном контексте домена.

Create

Доступ к методам Create API осуществляется через объект **`Create`** в системном контексте. Например: **`var guid = Create.guid();`**

```
String formattedRandom(
             String format)
```

Создает строку со случайными элементами, построенную по указанному аргементом **`format`** шаблону. Шаблон указывает каким образом и в какое место строки должны попасть случайные числа. Формат состоит из «специальных» символов и «других» символов: первые определяют вывод случайного значения, вторые просто выводятся в результатирующую строку как есть. Специальные символы: «**`D`**» – десятичный разряд случайного числа, «**`H`**» и «**`h`**» – шестнадцатиричный разрад случайного числа (буквы выводятся заглавными и прописными символами соответственно), «**`Z`**» и «**`z`**» – 36-тиричный разряд случайного числа (буквы выводятся заглавными и прописными символами соответственно), «\» (обратная косая черта) – следующий символ выводить в результатирующую строку как есть. Все остальные символы просто выводятся в строку.

Пример:
`code = Create.formattedRandom("DDDD-DDDD-DDDD-DDDD")`
создаст код выглядящий аналогично коду указанному на кредитных картах.

```
String guid()
```

Создает новый GUID (глобально-уникальный идентификатор), который представляет собой строковую последовательность длинной до 36 символов. Все символы этой строки печатные.

```
Map    map()
```

Создает новый пустой объект типа **`map`**.

```
Map    map(
             Map initial)
```

Создает копию переданного **`map`**. При этом, все вложенные объекты переданного **`map`** остаются общими, включая произвольно вложенные объекты типа **`map`** и **`list`**.

```
Map    mapClone(
             Map initial)
```

Создает глубокую копию переданного **`map`**. Поддерживает рекурсивное копирование содержания включая произвольно вложенные объекты типа **`map`** и **`list`**. Результатом выполнения всегда является новая копия объекта типа **`map`**. Если входной параметр был равен **`NULL`**, **`undefined`** или не являлся объектом типа **`map`** – результатом выполнения будет незаполненный объект.

```
Map    mapFilter(
              Map parent)


Map    mapFilter(
              Map primary,
              Map secondary)


Map    mapFor(
              <map-type parameter list>)
```

Создает объект типа **`map`**, содержащий переданные именованные параметры, упорядоченный в порядке следования этих параметров.

Пример: **`Create.mapFor( aa = 5, b = 'asd', z = true)`**, в результате будет создан map, со значениями { aa = 5, b = 'asd', z = true, $ORDER = ['aa', 'b', 'z']}.

```
List   list()
```

Создает новый пустой объект типа **`list`**.

```
List   list(
              Collection initial)
```

Создает новый объект типа **`list`** наполненный элементами коллекции переданной аргументом **`initial`**.

```
List   list(
              Object[] initial)
```

Создает новый объект типа **`list`** наполненный элементами массива переданного аргументом **`initial`**.

```
List   list(
              int size,
              Object fill)
```

Создает новый объект типа **`list`** и заполняет его указанным в аргументе **`size`** числом объектов переданных аргументом **`fill`**.

Эквивалентно:

```
var result = Create.list(); for(var i = size; i > 0; i--) { result.add( fill ) }
```

```
Set    set()


Set    set(
              Collection initial)


Set    set(
              Object[] initial)


Set    set(
              Object element)
```

```
LinkedList    fifo()

LinkedList    fifo(
              int limit)

Object[]      array()

Object[]      array(
              int size)

Counter       counter()

Message       message(
              Object body,
              Map attributes)

Message       message(
              Object body,
              String contentType)
```

Counter

```
void   register(
              double value)

int    getCount()

int    intValue()

long   longValue()

float  floatValue()

double doubleValue()

double getAverage()

double getMaximum()

double getMinimum()
```

Default API

Версия последнего изменения данного API: 677

Данный API предоставляет набор методов доступных в глобальном объекте.

```
oiObject     toPrimitive(
             Object o)


boolean      toBoolean(
             Object o)


Object toNumber(
             Object o)


double toDouble(
             Object o)


long   toInteger(
             Object o)


int    toInt32(
             Object o)


int    toUint32(
             Object o)


short  toUint16(
             Object o)


String toString(
             Object o)


Object toObject(
             Object o)


double NaN
```

Константа определяющая особое значение: «не число».

Внимание: для проверки значения на возможный NaN не используйте операции сравнения **==** и **!=**, а используйте специальный метод isNaN(value). Например: **`toNumber( value ) == NaN`** неверно, а **`isNaN( toNumber( value ) )`** верно.

```
double Infinity
```

Константа определяющая числовое значение «бесконечность».

Внимание: для проверки числового значения на возможное бесконечное значение не используйте операции сравнения **==** и **!=**, а используйте специальный метод isFinite(value).

Например: **`toNumber( value ) != Infinity`** неверно, а **`isFinite( toNumber( value ) )`** верно.

```
Object undefined
```

Константа определяющее особое значение: «значение не определено».

```
Object eval(
              Object x)
```

Вычисляет выражение переданное в виде строки аргументом **`x`** и возвращает его результат. Если в качестве выражения было передано значение типа отличного от строки – это значение будет возвращено в качестве результата без каких либо вычислений.

```
Object parseInt(
              Object x,
              Object radix)


double parseFloat(
              Object x)


boolean       isNaN(
              Object x)


boolean       isFinite(
              Object x)


Map    getData()


String Character(
              int code)


String Character(
              String codeString)


String Character(
              int code,
              String default)


String Character(
              String codeString
              String default)


int    charCode(
              String char)


boolean       Boolean(
              Object value)
```

```
long   Integer(
              Object value)


int    Int(
              Object value)


double Floating(
              Object value)


String Textual(
              Object value)


long   Integer(
              Object value,
              long default)


int    Int(
              Object value,
              int default)


double Floating(
              Object value,
              double default)


String Textual(
              Object value,
              String default)


String toHex(
              Object numericValue)


String toOct(
              Object numericValue)


String toBin(
              Object numericValue)


long   CurrentDate()


String CurrentTime() // HH:mm:ss


String CurrentTime(
              String format)


int    hashCode(
              Object object)
```

Вычисляет контрольную сумму объекта. Работает со всеми примитивными типами (числа, строки…), также должен работать с большинством непримитивных типов. Если переданный параметр равен NULL, результатом выполнения будет 0.

Пример:

```
activationCode += '-' + Math.abs( hashCode( activationCode ) ) % 9000 + 1000
```

В данном примере мы получаем 4ех разрядную десячитную контрольную сумму в строковом виде, которую мы добавляем к строке с активационным кодом, для последующих проверок.

```
int    HashCode(
              String string)
```

Устаревший – используйте метод hashCode()

```
String HexHashCode(
              Object object)
```

```
Object intl(
              <map-like parameter list>)
```

Позволяет определять многоязычные строки, в последствии выведенный в виде строки такой объект будет возвращать значение, связанное с текущим языком. Данным методом возможно описание таких строк непосредственно в коде скрипта, что в зависимости от решаемой задачи может быть как преимуществом, так и недостатком.

Пример:

```
title = intl( en = "Site Tree", ru = "Дерево сайта" )
```

В итоге мы получаем многоязычную строку, которую можем использовать в шаблонах и скриптах рассчитанных на поддержку нескольких языков.

```
String textToIdentifier(
              String text)
```

```
String textToIdentifier7bit(
              String text)
```

```
String transliterate(
              String text)
```

```
int    GetOccuranceCount(
              String text,
              String token)
```

```
int    GetOccuranceCount(
              String text,
              String token,
              int max)
```

```
String LimitString(
              String string,
              int maxlength)
```

```
String LimitString(
             String string,
             int maxlength,
             String suffix)


String trim(
             String string)


String Join(
             Object[] array,
             String token)


String Join(
             Collection collection,
             String token)


List   split(
             Object value,
             String splitToken)


List   splitRegexp(
             Object value,
             String splitRegexp)


List   splitCSV(
             String text)


List   splitCSV(
             String text,
             String splitter)


List   splitLines(
             Object value)


List   splitLinesIgnoreQuotes(
             Object value)


String substring(
             String string,
             int start)


String substring(
             String string,
             int start,
             int end)


String substr(
             String string,
             int start)
```

```
String substr(
              String string,
              int start,
              int endOrTail)


String afterPoint(
              String string)


boolean       StartsWith(
              String string,
              String prefix)


boolean       EndsWith(
              String string,
              String suffix)


int    strlen(
              String string)


String Capitalize(
              String string)


String UpperCase(
              String string)


String LowerCase(
              String string)


String replace(
              String string,
              String what,
              String with)


String replaceRegex(
              String string,
              String what,
              String with)


int    indexOf(
              String string,
              String token)


int    indexOf(
              String string,
              String token,
              int start)
```

```
int    lastIndexOf(
              String string,
              String token)


int    lastIndexOf(
              String string,
              String token,
              int start)


boolean       ArrayValid(
              Object anything)


boolean       ArrayFilled(
              Object anything)


List   Array(
              Object anything)
```

Преобразование объекта к массиву. Если исходный объект является массивом, он будет возвращен как есть, если переданный объект является объектом **undefined** будет возвращен пустой массив, в другом случае будет создан массив, единственным элементом которого является переданный объект.

```
Object ArrayGet(
              Object anything,
              int index)


Object ArraySet(
              Object anything,
              int index,
              Object value)


Object ArrayRemove(  !!! UNUSABLE
              Object anything,
              int index)


int    ArrayLength(
              Object anything)


void   ArrayAppend(  -> list.add( value )
              List list,
              Object value)


boolean       IsInArray(
              Object anything,
              Object value)


Object ArraySwap(
              Object anything,
              int index1,
              int index2)
```

```
List<List>    ArraySplit(
              List list,
              int page)


Object[][]    ArraySplit(
              Object[] array,
              int page)


Object[]      ArraySplit(
              String commaText,
              int page)


Object ArrayMin(
              Object anything)


Object ArrayMax(
              Object anything)


boolean       HashValid(
              Object anything)


boolean       HashFilled(
              Object anything)


Object HashApply(    -> map1.putAll( map2 )
              Map map1,
              Map map2)


Object HashRemove(   -> map.remove( key )
              Map map,
              Object key)


int    HashSize(     -> map.size()
              Map map)


boolean       IsInHash(     -> map.containsKey( key )
              Map map,
              Object key)


String GetUserID()


Element       DomParse(
              String xml)


String DomToString(
              Element root)
```

```
String enchanceHtml(
             String text)


String formatByteSize(
             long size)


String formatRoundByteSize(
             long size)


String formatBigDecimal(
             long decimal)


String formatDate(
             Object anything,
             String format)


Date   parseDate(
             String date,
             String format)


String formatFraction(
             Object anythingAsDouble,
             int digits)


String formatPeriod(
             long period)


long   parsePeriod(
             Object anything)


String formatSqlStringParameter(
             String string)
```

Смотри: Format.sqlStringFragment или Format.sqlString.

.

```
String StringToWml(
             Object anything)


String StringToBase64(
             Object anything)


String StringToUrl(
             Object anything)


String StringToUrlHard(
             Object anything)
```

```
String StringToXml(
             Object anything)


String StringToHtmlInput(
             Object anything)


String StringToHtml(
             Object anything)


String StringToSafeHtml(
             Object anything)


String Base64ToString(
             Object anything)


String UrlToString(
             Object anything)


String XmlToString(
             Object anything)


String HtmlToString(
             Object anything)


String uniqueWords(
             String text,
             String delimiter)


String sentenceLeft(
             String text,
             int minChars,
             int maxChars)


String sentenceLeft(
             String text,
             int minChars,
             int maxChars,
             String suffix)


String clearHtmlFormatting(
             String html)


String exceptionToString(
             Throwable exception)


long   dateToNumeric(
             Object anything)
```

```
Date   numericToDate(
              Object anything)
```

```
Map    mapMergeRecursive(
              Map target,
              Map source)
```

Осуществляет глубокое рекурсивное копирование/объединение из объекта типа **map** указанного параметром **source** в объект типа **map** указанный параметром **target**. Результатом выполнения является объект типа **map** равный параметру **target** или **null**, если параметр **target** не являлся объектом типа **map**. Если объект переданный параметром **source** не является объектом типа **map** копирования не происходит. При этом – если на одном уровне иерархий **source** и **target** встречаются объекты типа **map** – их данные будут объединены, если эти объекты разного типа – то данные в иерархии **target** будут заменены соответствующими данными из иерархии **source**.

```
Map    mapPutRecursive(
              Map target,
              Map source)
```

Осуществляет глубокое рекурсивное копирование из объекта типа **map** указанного параметром **source** в объект типа **map** указанный параметром **target**. Результатом выполнения является объект типа **map** равный параметру **target** или **null**, если параметр **target** не являлся объектом типа **map**. Если объект переданный параметром **source** не является объектом типа **map** копирования не происходит. При этом, любые данные в иерархии **source**, заменяют соответствующие данные в иерархии **target** (объединения данных не происходит).

```
String mapToXml(
              Map map)
String mapToXml(
              Map map,
              String root)
String mapToXml(
              Map map,
              String root,
              boolean readable)
```

Осуществляет преобразование набора свойств переданного объекта в XML-вид в соответствии с правилами формата ACM.MAP.

Варианты со вторым аргументом **root** позволяют указывать имя коренного элемента (по умолчанию, в варианте без этого аргумента имя коренного элемента будет «data»). Вариант с третьим аргументом **readable** позволяет указать должен ли результатирующий XML быть читаемым (readable) или компактным (compact) – по умолчанию данные методы делают компактный XML.

```
Map    xmlToMap(
              String xml)
Map    xmlToMap(
              String xml
              String namespace)
```

Осуществляет преобразование XML в свойства свежесозданного объекта.

Поддерживает как формат ACM.MAP, так и просто любой XML (в этом случае элемент с атрибуиами или дочерними элементами становится объектом типа Map, а, собственно, атрибуты и дочерние элементы образуют набор его свойств).

Вариант с аргументом namespace позволяет указать имя интересующего нас неймспейса из переданного XML – элементы других неймспейсов будут просто проигнорированы.

```
int    binarySize(
              Object anythingBinary)
```

Возвращает длинну в байтах для переданного бинарного объекта.

```
String binaryDigest(
              Object anythingBinary)
```

Возвращает строку содержащую в шестнадцатеричном виде контрольную сумму MD5 для переданного бинарного объекта. Такая контрольная сумма повсеместно используется для сравнения файлов, особенно расположенных удаленно друг от друга.

```
String binaryToString(
              Object anythingBinary,
              String charset)
```

```
Copier stringToBinary(
              Object anything,
              String charset)
```

```
Map    zipToMap(
              Object anythingBinary)
```

```
Copier mapToZip(
              Map map)
```

```
Copier decompressGzip(
              Object anythingBinary)
```

Производит разжатие потока сжатого методом GZIP и переданного аргументом **`anythingBinary`**. Допустимыми значениями для аргумента является любой объект, который система может интерпретировать как **`binary`**. Если аргумент равен **`null`**, результат выполнения данной функции будет также **`null`**, во всех остальных случаях результатом выполнения является либо объект содержащий разжатый поток данных, либо ошибка формата GZIP, либо ошибка о невозможности интерпретировать аргумент как объект типа **`binary`**.

```
String guessContentTypeFromBinary(
              Object binary,
              String defaultValue)
```

Download API

Версия последнего изменения данного API: 667

Данный API предоставляет доступ к функционалу модуля **ACMMOD:DOWNLOAD** – и появляется в контексте домена только если этот модуль подключен к сайту.

Модуль предназначен для индексации, классификации, хранения дополнительной метаинформации и поиска файлов. Он не предназначен для хранения или отдачи файлов пользователям. Для хранения и выдачи файлов в системе имеется специальный вид сервера (настраивается в servers.xml). Данный модуль (ACMMOD:DOWNLOAD) может быть настроен для подключения к любому числу этих специальных серверов, после чего он начинает собирать и обновлять информацию о находящихся там файлах.

Доступ к методам API осуществляется через объект с именем **Download**, который регистрируется в глобальном контексте домена.

Download

```
RecFile       getItemByGuid(
              String guid)

RecFile[]     searchByName(
              String name)

RecFile[]     searchByMd5(
              String md5)

RecFile[]     searchByNameAndType(
              String name,
              String type)

RecFile[]     searchByFollowLink(
              String followLink)

RecFile[]     searchByAlias(
              String alias)

RecFileGroup[]      searchFileGroupsByFollowLink(
              String followLink)

RecFileGroup[]      searchFileGroupsByFollowLink(
              String followLink,
              boolean visible)

RecFileGroup[]      searchFileGroupsByAlias(
              String alias)

RecFileGroup[]      searchFileVariantsByFollowLink(
              String followLink)

RecFileGroup[]      searchFileVariantsByFollowLink(
              String followLink,
              boolean visible)

RecFileGroup[]      searchFileVariantsByAlias(
              String alias)

RecKnown[]    searchKnown(
              String names)

RecKnown      getKnownByFollowLink(
              String followLink)

RecAlias[]    getAliases()

RecKnown[]    getKnown()

int    getL1KnownCount()

int    getL1KnownAliasCount()

int    getL1UnknownCount()

RecQueued[]   getQueue()

RecQueued     getQueued(
              int luid)
```

```
RecAlias      getAlias(
              String alias)

String getKnown(
              String name)

void   setQueued(
              String name,
              String text)

int    getNextQueued()

void   shiftQueue()

void   setItemDescription(
              String md5,
              String description)

RecFile[]     getUndescribedItems()

List<RecFile> search(
              int limit,
              boolean all,
              long timeout,
              String sort,
              long startDate,
              long endDate,
              String filter )
```

В большинстве случаев (в зависимости от условий поиска) поиск будет осуществляться в виде нескольких параллельных и последовательных запросов, если при запуске очередного задания окажется, что указанное параметром **`timeout`** отрезок времени уже истек, поиск будет прерван (не ожидайте от этого аргумента филигранной точности).

```
RecSource[]   getSources(
              boolean all)
```

RecFile

```
String getComment()

String getType()

String getGuid()

int    getFolderLuid()

String getName()

boolean       hasPreview()

String getLevel2Name()

String getLevel3Name()

long   getSize()

long   getDate()

String getPath()

String getPathLocal()

String getMd5()

String getDescription()

boolean       isHidden()

Collection<String>  getAliases()
```

RecFileGroup

```
RecFile       getFirst()
```

```
List<RecFile> getFiles()

List<RecFile> getFiles(
              String sort)  // history, log, alphabet

Int    getSize()

String getComment()

Set<String>   getTypes()

Int    getFolderLuid()

String getName()

String getLevel2Name()

String getLevel3Name()
```

RecQueued

```
int    getLuid()

void   doLock()

void   doRequeue()

void   doConnect(
              Collection<String> followLinks)

void   doAcknowledge(
              String followLink,
              Collection<String> names)

void   doCreateAlias(
              String followLink)

String getName()

String getText()

String getHint()
```

RecKnown

```
String getFollowLink()

String getName()

RecFile[]     getFiles()

RecFileGroup[]      getFileGroups()

RecFileGroup[]      getFileVariants()

RecAlias[]    getAliases()
```

RecAlias

```
Collection<String>  getFollowLinks ()

RecKnown[]    getKnown()

String getName()

RecFile[]     getFiles()

RecFileGroup[]      getFileGroups()

RecFileGroup[]      getFileVariants()

void   doUpdateConnections(
              Collection<String> addFollowLinks,
              Collection<String> removeFollowLinks)
```

RecSource

```
long   getChecked()
```

```
long   getCreated()
String getGuid()
String getHost()
String getPort()
boolean      isActive()
boolean      isIndex()
double getHealth()
double getReady()
String getVersion()
RecFolder    getRootFolder()
```

RecFolder

```
String getName()
RecFolder    getParent()
RecSource    getSource()
String getPath()
String getPathLocal()
Map<String, RecFolder>     getFolders()
Map<String, RecFile>       getFiles()
```

ExcelWorkbook API

Версия последнего изменения данного API: 639

Данный API предоставляет набор функций для чтения, создания, редактирования и прочих операций над книгами формата excel.

Доступ к методам API осуществляется через объект с именем **`ExcelWorkbook`**, который регистрируется в глобальном контексте домена.

ExcelWorkbook

```
Workbook     getWorkbook(
             Buffer buffer)
Workbook     getWorkbook(
             Copier copier)
Workbook     getWorkbook(
             Message message)
Workbook     getWorkbook(
             byte[] bytes)
Workbook     getWorkbook(
             File file)
WorkbookChange      createWorkbook()
```

Workbook

```
int    getNumberOfSheets()
Sheet  getSheet(
             int index)
Sheet[]      getSheets()
void   close()
```

Sheet

```
int    getColumns()

int    getRows()

Cell   getCell(
              int columnIndex,
              int rowIndex)

Cell[] getColumn(
              int columnIndex)

Cell[] getRow(
              int rowIndex)
```

Cell

```
int    getColumn()

int    getRow()

String getContents()
```

WorkbookChange

```
SheetChange  createSheet(
             String title)

CellFormat   createCellFormat()

CellFormat   createCellFormatInteger()

CellFormat   createCellFormatFloating()

CellFormat   createCellFormatText()

CellFormat   createCellFormatDate(
             String format)

Transfer.Copier     buildWorkbook()
```

SheetChange

```
void   setColumnWidth(
              int pixels)

void   mergeCells(
              int x1,
              int y1,
              int x2,
              int y2)

void   setCellLabel(
              int x,
              int y,
              String text,
              CellFormat format)

void   setCellHyperlink(
              int x,
              int y,
              String location,
              String text,
              CellFormat format)

void   setCellInteger(
              int x,
              int y,
              long value,
              CellFormat format)
```

```
void   setCellFloating(
              int x,
              int y,
              double value,
              CellFormat format)

void   setCellDate(
              int x,
              int y,
              long date,
              String format,
              CellFormat format)
```

CellFormat

```
void   setFont(
              String family,
              int size,
              boolean bold,
              boolean italic,
              int underline)

void   setWrap(
              boolean wrap)

void   setBorderNone()

void   setBorderLeft()

void   setBorderRight()

void   setBorderTop()

void   setBorderBottom()

void   setBorderLeftDotted()

void   setBorderRightDotted()

void   setBorderTopDotted()

void   setBorderBottomDotted()

void   setAlignmentGeneral()

void   setAlignmentLeft()

void   setAlignmentCenter()

void   setAlignmentRight()

void   setAlignmentJustify()

void   setAlignmentFill()
```

File API

Версия последнего изменения данного API: 639

Доступ к методам API осуществляется через объект с именем **`File`**, который регистрируется в глобальном контексте домена.

File

```
String clearSlashes(
              String name)
```

Удаляет слэши (символы '/' и '\') из указанного имени файла.

```
String getFileName(
              String url)
```

Возвращает полное имя файла (с расширением).

```
String getFileExtension(
             String url)
```

Возвращает расширение имени файла.

```
String getContentTypeForExtension(
             String extension)
```

Возвращает тип содержания (MIME Content Type) для указанного расширения файла.

```
String getContentTypeForName(
             String url)
```

Возвращает тип содержания (MIME Content Type) для указанного имени файла.

```
String getFileTitle(
             String url)
```

Возвращает имя файла без расширения.

```
String niceNameNotation(
             String name)
```

Преобразовывает имя файла так, чтобы оно не содержало никаких специальных символов и было совместимо с большинством применений в web и одновременно легко читаемо пользователем.

```
String niceNameEncode(
             String name)
```

Преобразовывает имя файла так, чтобы оно не содержало никаких специальных символов и было совместимо с большинством применений в web и могло быть декодировано обратно методом **`niceNameDecode`**.

```
String niceNameDecode(
             String nameEncoded)
```

Раскодирует имя файла, которое было закодировано методом **`niceNameEncode`**.

Format API

Версия последнего изменения данного API: 677

Доступ к методам API осуществляется через объект с именем **`Format`**, который регистрируется в глобальном контексте домена.

Format

```
String jsString(
             String string)
```

Форматирует переданную аргументом **string** строку в формата стандарта ECMA-262 для непосредственной вставки данной строки в исходный код скрипта как строковой константы.

```
var js = "x.a + " + Format.jsString( request.filter );
```

ВАЖНО: позволяет избежать возможности атаки на сервер посредством JSON-Injection при вставке полученных от пользователя данных.

```
String jsStringFragment(
            String string)
```

Форматирует переданную аргументом **string** строку в формата стандарта ECMA-262 для непосредственной вставки данной строки в исходный код скрипта, как часть строковой константы. Также заменяет строку вида </script> на </scr\x69pt>, что равносильно, но не разрывает скрипты вставленные в HTML страницы.

```
var js = "x.a + 'id-" + Format.jsStringFragment( request.filter ) + ".xml'";
```

ВАЖНО: позволяет избежать возможности атаки на сервер посредством JSON-Injection при вставке полученных от пользователя данных.

```
String sqlString(
            String string)
```

Форматирует переданную аргументом **string** строку в формат стандарта SQL-92 для непосредственной вставки данной строки в SQL-запрос.

```
var sql = "SELECT c1, c2 FROM table WHERE c3 = "
                  + Format.sqlString( request.filter )
```

ВАЖНО: позволяет избежать возможности атаки на сервер посредством SQL-Injection при вставке полученных от пользователя данных в SQL запрос для дальнейшего обращения к базе данных.

```
String sqlStringFragment(
            String string)
```

Форматирует переданную аргументом **string** строку в формат стандарта SQL-92 для непосредственной вставки данной строки в стровое значение SQL-запроса.

```
var sql = "SELECT c1, c2 FROM table WHERE c3 = 'xxx-"
                  + Format.sqlStringFragment( request.filter ) + "'"
```

ВАЖНО: позволяет избежать возможности атаки на сервер посредством SQL-Injection при вставке полученных от пользователя данных в SQL запрос для дальнейшего обращения к базе данных.

Imaging API

Версия последнего изменения данного API: 664

Доступ к методам API осуществляется через объект с именем **Imaging**, который регистрируется в глобальном контексте домена.

Imaging

```
ImageSize    getImageDimensions(
             Binary image)
```

```
ImageSize    getImageDimensions(
             Image image)

Image  createImage(
             int width,
             int height)

Image  imageForBinary(
             Binary image)

Image  bitmapResize(
             Image image,
             int width,
             int height)
```

Результатом выполнения является новое растровое изображение представляющее собой переданное в качестве аргумента image изображение растянутое или сжатое до указанного аргументами width и height размера.

```
Image  bitmapResizeCrop(
             Image image,
             int width,
             int height)
```

Результатом выполнения является новое растровое изображение указанного аргументами width и height размера, в которое вписано переданное аргументом image изображение так, что оно полностью заполняет поверхность нового изображения с сохранением пропорций. При этом, если одна из сторон исходного изображения выходит за границы соответствующей стороны результатирующего изображения, она выравнивается по центру, а его края обрезаются.

```
Image  bitmapResizeFit(
             Image image,
             int width,
             int height)
```

Результатом выполнения является новое растровое изображение указанного аргументами width и height размера, в которое вписано переданное аргументом image изображение так, что оно полностью помещается с сохранением пропорций. При этом, если одна из сторон исходного изображения не дотягивает до соответствующей стороны результатирующего изображения, она выравнивается по центру, а свободное место остается прозрачным.

```
Image  bitmapResizeCanvas(
             Image image,
             int canvasColor,
             int width,
             int height)
```

Результатом выполнения является новое растровое изображение указанного аргументами width и height размера, в которое вписано переданное аргументом image изображение так, что оно полностью помещается с сохранением пропорций. При этом, если одна из сторон исходного изображения не дотягивает до соответствующей стороны результатирующего изображения, она выравнивается по центру, а свободное место заполняется указанным аргументом canvasColor цветом.

```
int    colorForComponents(
             int r,
             int g,
             int b)
```

```
void   graphicsSetColor(
              Graphics g,
              int color)

void   graphicsSetXorMode(
              Graphics g,
              int color)

void   graphicsDrawLine(
              Graphics g,
              int x1,
              int y1,
              int x2,
              int y2)

void   graphicsDrawOval(
              Graphics g,
              int x,
              int y,
              int w,
              int h)

void   graphicsFillOval(
              Graphics g,
              int x,
              int y,
              int w,
              int h)

void   graphicsDrawRect(
              Graphics g,
              int x,
              int y,
              int w,
              int h)

void   graphicsFillRect(
              Graphics g,
              int x,
              int y,
              int w,
              int h)

void   graphicsDrawImage(
              Graphics g,
              Image image,
              int x,
              int y)

void   graphicsDrawText(
              Graphics g,
              String text,
              int x,
              int y,
              String face,
              boolean bold,
              boolean italic,
              int size,
              int horizontal,
              int vertical)

Message       encodeGif(
              Collection images,
              int delay,
              boolean loop)

Message       encodeGif(
              Image img)

Message       encodeJpeg(
              Image img)

Message       encodeJpeg(
              Image img
```

```
             boolean progressive
             double quality)

Message      encodePng(
             Image img)

Binary encodeGifBinary(
             Collection images,
             int delay,
             boolean loop)

Binary encodeGifBinary(
             Image img)

Binary encodeJpegBinary(
             Image img)

Binary encodeJpegBinary(
             Image img
             boolean progressive
             double quality)

Binary encodePngBinary(
             Image img)

String getColorCode(
             int rgb)
```

Image

```
int    getWidth()

int    getHeight()

Graphics     getGraphics()
```

Возвращает объект java Graphics2D. Этот представляет собой графический контекст изображения и обеспечивает рисование графических примитивов (предоставляя методы для изменения объекта Image у которого был вызван метод getGraphics). Он используется некоторыми методами ImagingAPI.

Для самостоятельного использования объекта Graphics2D советую ознакомится с его документацией, расположенной по адресу:
http://java.sun.com/j2se/1.5.0/docs/api/java/awt/Graphics2D.html

```
int    getRGB(
             int x,
             int y)

void   setRGB(
             int x,
             int y,
             int rgb)
```

ImageSize

```
int    width

int    height
```

Mail API

Версия последнего изменения данного API: 640

Доступ к методам API осуществляется через объект с именем **`Mail`**, который регистрируется в глобальном контексте домена.

Mail

```
PopSession    createPopSession(
              String host,
              String user,
              String password )

PopSession    createPopSession(
              String host,
              int port,
              String user,
              String password )

String generateReplySubject(
              String subject )
```

PopSession

```
void   quit()

void   reset()

MessageMark[] list()
```

MessageMark

```
int    getSequenceNumber()

int    getMessageSize()

Message       get()

void   delete()
```

Message

```
MailAddress  getOriginator()

List<Recipient>     getRecipients()

String getSubject()

String getMessageText()

Map<String, Object> getAttachments()

MessageBody  getMessageBody()
```

MailAddress

```
String getName()

String getAddress()
```

Recipient

```
boolean      isTo()

boolean      isCc()

boolean      isBcc()

MailAddress  getAddress()
```

MessageBody

```
String getBody()
```

Math API

Версия последнего изменения данного API: 633

Доступ к методам API осуществляется через объект с именем **`Math`**, который регистрируется в глобальном контексте домена.

Math

```
double PI

double E

double rand()

double abs(
             double x)

double acos(
             double x)

double asin(
             double x)

double atan(
             double x)

double ceil(
             double x)

double cos(
             double x)

double exp(
             double x)

double floor(
             double x)

double log(
             double x)

int    max(
             int a,
             int b)

int    min(
             int a,
             int b)

long   max(
             long a,
             long b)

long   min(
             long a,
             long b)

double max(
             double a,
             double b)

double min(
             double a,
             double b)

double pow(
             double a,
             double b)

double rint(
             double x)
```

```
long   round(
               double x)

double sin(
               double x)

double sqrt(
               double x)

double tan(
               double x)

double toDegrees(
               double x)

double toRadians(
               double x)
```

Random API

Версия последнего изменения данного API: 639

Доступ к методам API осуществляется через объект с именем **`Random`**, который регистрируется в глобальном контексте домена.

Random

```
int    integer(
               int limit)

Object fromList(
               List list)

Object fromList(
               Object[] array)

List   subList(
               List list,
               int limit)

Map    subMap(
               Map map,
               int limit)
```

Request API

Версия последнего изменения данного API: 635

Данный API предоставляет методы для доступа к текущему запросу, в рамках которого происходит выполнение скрипта.

Доступ к методам API осуществляется через объект с именем **`Request`**, который регистрируется в глобальном контексте домена.

Request

```
Request      getCurrentRequest()

Map    getAttributes()

long   getDate()

String getLanguage()

String getOwner()

Map    getParameters()     // getData()

String getParameterString()

String getProtocolName()
```

```
String getProtocolVariant()

String getResourceIdentifier()

String getResourcePrefix()

String getSessionID()

String getUserID()

String getSourceAddress()

String getSubject()

String getVerb()

String getTarget()

String getTargetExact()

String getTitle()

String getUrl()

String getUrlBase()

Object getSharedObject()

Object getAttachment()

Map    getSettings()

String modifyQueryStringParameter(
              String url,
              String name )

String modifyQueryStringParameter(
              String url,
              String name,
              String value )

Message       replyBinary(
              Object binary,
              Map attributes)

Message       replyRedirect(
              String url,
              Boolean moved)
```

Request type

```
long   getDate()

String getOwner()

String getProtocolVariant()

String getProtocolName()

String getTitle()

boolean       hasSubject()

String getSubject()

String getTarget()

Request       setTarget(
              String target)

String getTargetExact()

Request       setTargetExact(
              String targetExact)

String getUrl()

Request       setUrl(
              String url)

String getUrlBase()
```

```
Request       setUrlBase(
              String urlBase)

long   getIfModifiedSince()

String getSessionID()

String getUserID()

String getVerb()

Request       setVerb(
              String verb)

String getVerbOriginal()

String getResourceIdentifier()

Request       setResourceIdentifier(
              String resourceIdentifier)

String getResourcePrefix()

Request       setResourcePrefix(
              String resourcePrefix)

String getContentType()

Request       setContentType(
              String contentType)

String getContentName()

Request       setContentName(
              String contentName)

String getContentID()

Request       setContentID(
              String contentId)

long   getLastModified()

Request       setLastModified(
              long lastModified)

String getEncoding()

Request       setEncoding(
              String charset)

boolean       hasAttributes()

Map    getAttributes()

Request       setAttribute(
              String key,
              Object value)

Request       addAttribute(
              String key,
              Object value)

Request       setAttributes(
              Map attributes)

Request       useAttributes(
              Map attributes)

String getParameterString()

boolean       hasParameters()

Map    getParameters()

Request       setParameter(
              String key,
              Object value)

Request       addParameter(
              String key,
              Object value)
```

```
Request       setParameters(
              Map attributes)

Request       useParameters(
              Map attributes)

boolean       hasArguments()

String[]      getArguments()

Request       setArguments(
              String[] arguments)

String getSourceAddressExact()

String getSourceAddress()

Request       setSourceAddress(
              String address)

String getLanguage()

Request       setLanguage(
              String language)

Object getAttachment()

Request       setAttachment(
              Object attachment)

Message       toCharacterMessage()

Message       toBinaryMessage()

boolean       isEmpty()

boolean       isBinary()

Buffer getBinary()

boolean       isSource()

Source getSource()

boolean       isFile()

File   getFile()

boolean       isTextual()

String getText()

boolean       isObject()

Object getObject()

Class  getObjectClass()

boolean       isSequence()

Message[]     getSequence()
```

Runtime API

Версия последнего изменения данного API: 675

Доступ к методам API осуществляется через объект с именем **`Runtime`**, который регистрируется в глобальном контексте домена.

Runtime

```
String getSystemName()

String getRuntimeName()

String getRuntimeVersion()

String getRuntimeBuild()
```

```
String getMainEntranceUrl()

String log(
              String owner,
              String type,
              String text)

String audit(
              String owner,
              String type,
              String text)

String getUserID()

String getLanguage()

void   setLanguage(
              String language)

String getDefaultCharacterEncoding()

?String       ListLanguages()

?String       listLookup(
              Lookup lookup,
              String keyName,
              String valueName)

Message       UrlAsMessage(
              String url)

Buffer UrlAsBuffer(
              String url)

String UrlAsString(
              String url,
              String charset)

String GetLoginUrl()

String GetLogoutUrl()

String GetLoginUrl(
              String back)

String GetLogoutUrl(
              String back)

void   Return(
              Object anything,
              Map attributes)

boolean       SendMail(
              Map mail)
```

Ставит письмо в очередь на отправку. В Map указываются следующие параметры:

From	Строковое. Указывает отправителя сообщения.
Reply	Строковое. Опциональное. Указывает адресата для ответов на это сообщение. По умолчанию равно значению параметра **From**.
To	Строковое или массив строк. Указывает адресатов сообщения. Позволяет указывать несколько адресов, как элементами массива, так и разделительными символами «,» или «;» в каждом строковом значении. При отправке сообщения на несколько адресов, для каждого уникального адреса из набора произойдет отправка одного сообщения, в котором адресатом будет указан только этот адрес. Позволяет работать со списками рассылок. Для этого в качестве адресата (или одного из адресатов) надо указать **#<имя_списка>**. При нахождении такого адресата система осуществит вызов: **Recipients.getGroupRecipients(key = '<имя_списка>')**

	Вызов данного метода должен вернуть массив адресов списка рассылки, по которым требуется разослать данное сообщение. В качестве исключения, если адресат указан как **#email:<почтовый_адрес>**, обращения за списком рассылки не произойдет, а сообщение будет отправлено на соответствующий адрес.
CC	Строковое. Опциональное. «Carbon Copy». Указывает адресатов для видимой копии письма. Позволяет указывать несколько адресов, разделенных символами «,» или «;».
BCC	Строковое. Опциональное. «Blind Carbon Copy». Указывает адресатов для невидимой копии письма. Позволяет указывать несколько адресов, разделенных символами «,» или «;».
Encoding	Строковое. Опциональное. Указывает кодировку сообщения, по умолчанию используется кодировка koi8-r.
Subject	Строковое. Содержит тему сообщения.
Body	Строковое. Содержит тело сообщения.
Format	Строковое. Принимает значения «**plain**» или «**html**». Отвечает за формат тела сообщения. Форматирование параметра **Body** должно соответствовать значению указанному в параметре **Format**.
Attachments	Map. Определяет вложения к письму. В качестве ключей мэпа выступают имена файлов, в качестве значений, текстовые или бинарные данные.

Возвращает true если сообщение было поставлено в очередь на отправку. По поводу возможных форматов почтовых адресов смотрите RFC 822.

При желании использовать вложенные файлы, как изображения в HTML письме, требуется в атрибуте **SRC** тэга **IMG** указать ссылку вида **cid:<имя_вложения>**, например:

```
<img src="cid:picture.jpg" width="24" height="67">
```

обратите внимание: кавычки обязательны. Система автоматически произведет необходимые действия по обеспечению соответствия данного письма стандарту, описанному в RFC 2111.

```
String searchPrepareField(
             String fieldName,
             String phrase)

boolean      evaluateBoolean(
             String expression)
```

Вычисляет переданное выражение и возвращает преобразовынный к типу **boolean** результат. Смотри также метод **eval** из **Default API**.

```
Object evaluateObject(
             String expression)
```

Вычисляет переданное выражение и возвращает результат выполнения этого выражения. Вместо этого метода рекомендуется использовать стандартный метод **eval** из **Default API**.

```
void   evaluateVoid(
             String expression)
```

Вычисляет переданное выражение игнорируя результат его выполнения. Смотри также метод **eval** из **Default API**.

```
Node   getLinkageRootNode()
```

Вернет виртуальный узел иерархии содержащий в себе все точки публичного доступа и их иерархию. Может быть использовано для построения дерева доступных извне адресов.

```
String[]      getLinkageRootPoints()

String getLinkagePublicUrl(
               String controlLocation)

String getLinkagePrivatePath(
               String publicUrl)
```

Session API

Версия последнего изменения данного API: 633

Данный API предоставляет доступ к текущей сессии, в рамках которой происходит выполнение скрипта.

Доступ к методам API осуществляется через объект с именем **`Session`**, который регистрируется в глобальном контексте домена.

Session

```
Map    getData()

Map    getParameters()

String SID()
```

Sort API

Версия последнего изменения данного API: 677

Доступ к методам API осуществляется через объект с именем **`Sort`**, который регистрируется в глобальном контексте домена.

Sort

```
Comparator   ASCENDING
```

Упорядочивает элементы в порядке возрастания значений.

```
Comparator   DESCENDING
```

Упорядочивает элементы в порядке убывания значений.

```
Comparator   NUMERIC
```

Упорядочивает элементы в порядке возрастания значений полученных преобразованием значения в числовые значения.

```
Comparator   NUMERIC_DESC
```

Упорядочивает элементы в порядке убывания значений полученных преобразованием значения в числовые значения.

`Comparator TEXT`

Упорядочивает элементы в порядке возрастания значений полученных преобразованием значения в строковые значения.

`Comparator TEXT_DESC`

Упорядочивает элементы в порядке убывания значений полученных преобразованием значения в строковые значения.

`Comparator ENTRY_CREATED`

Упорядочивает объекты хранилища в порядке возрастания даты создания объекта.

`Comparator ENTRY_CREATED_DESC`

Упорядочивает объекты хранилища в порядке убывания даты создания объекта.

`Comparator ENTRY_TITLE`

Упорядочивает объекты хранилища в порядке возрастания строкового представления заголовка объекта.

`Comparator ENTRY_TITLE_DESC`

Упорядочивает объекты хранилища в порядке убывания строкового представления заголовка объекта.

`Comparator MAP_KEY`

Упорядочивает элементы объекта типа **`map`** в порядке возрастания ключа из каждой пары ключ+значение.

`Comparator MAP_KEY_DESC`

Упорядочивает элементы объекта типа **`map`** в порядке убывания ключа из каждой пары ключ+значение.

`Comparator MAP_KEY_NUMERIC`

Упорядочивает элементы объекта типа **`map`** в порядке возрастания численного представления ключа из каждой пары ключ+значение.

`Comparator MAP_KEY_NUMERIC_DESC`

Упорядочивает элементы объекта типа **`map`** в порядке убывания численного представления ключа из каждой пары ключ+значение.

`Comparator MAP_KEY_TEXT`

Упорядочивает элементы объекта типа **`map`** в порядке возрастания строкового представления ключа из каждой пары ключ+значение.

```
Comparator    MAP_KEY_TEXT_DESC
```

Упорядочивает элементы объекта типа **`map`** в порядке убывания строкового представления ключа из каждой пары ключ+значение.

```
Comparator    MAP_VALUE
```

Упорядочивает элементы объекта типа **`map`** в порядке возрастания значения из каждой пары ключ+значение.

```
Comparator    MAP_VALUE_DESC
```

Упорядочивает элементы объекта типа **`map`** в порядке убывания значения из каждой пары ключ+значение.

```
Comparator    MAP_VALUE_NUMERIC
```

Упорядочивает элементы объекта типа **`map`** в порядке возрастания численного представления значения из каждой пары ключ+значение.

```
Comparator    MAP_VALUE_NUMERIC_DESC
```

Упорядочивает элементы объекта типа **`map`** в порядке убывания численного представления значения из каждой пары ключ+значение.

```
Comparator    MAP_VALUE_TEXT
```

Упорядочивает элементы объекта типа **`map`** в порядке возрастания строкового представления значения из каждой пары ключ+значение.

```
Comparator    MAP_VALUE_TEXT_DESC
```

Упорядочивает элементы объекта типа **`map`** в порядке убывания строкового представления значения из каждой пары ключ+значение.

```
Object[]      array(
              Object[] array,
              Comparator comparator)
```

Возвращает массив переданный аргументом **`array`**, упорядоченный в соответствии в переданным в аргументе **`comparator`**. Если **`comparator`** не указан или равен **`null`** – сравнение элементов массива будет производиться в соответствии со стандартной Ecma сортировкой, что эквивалентно **`Sort.ASCENDING`**. Если массив переданный в аргументе **`array`** равен **`null`** – результатом выполнения метода будет **`null`**.

```
Object[]      array(
              Collection collection,
              Comparator comparator)
```

Возвращает вновь созданный массив элементов из коллекции переданной аргументом **collection**, упорядоченый в соответствии в переданным в аргументе **comparator**. Если **comparator** не указан или равен **null** – сравнение элементов массива будет производиться в соответствии со стандартной Ecma сортировкой, что эквивалентно **Sort.ASCENDING**. Если коллекция переданная в аргументе **collection** равна **null** – результатом выполнения метода будет **null**.

```
Collection   collection(
             Object[] array,
             Comparator comparator)
```

Возвращает вновь созданную коллекцию без возможности изменения из массива, переданного аргументом **array**, упорядоченый в соответствии в переданным в аргументе **comparator**. Если **comparator** не указан или равен **null** – сравнение элементов массива будет производиться в соответствии со стандартной Ecma сортировкой, что эквивалентно **Sort.ASCENDING**. Если массив переданный в аргументе **array** равен **null** – результатом выполнения метода будет **null**.

```
Collection   collection(
             Collection collection,
             Comparator comparator)
```

Возвращает вновь созданную коллекцию без возможности изменения из коллекции переданной аргументом **collection**, упорядоченый в соответствии в переданным в аргументе **comparator**. Если **comparator** не указан или равен **null** – сравнение элементов массива будет производиться в соответствии со стандартной Ecma сортировкой, что эквивалентно **Sort.ASCENDING**. Если коллекция переданная в аргументе **collection** равна **null** – результатом выполнения метода будет **null**.

```
Map    map(
             Map map,
             Comparator comparator)
```

Возвращает вновь созданный объект типа map без возможности изменения содержащий пары ключей и значений из объекта типа map переданного аргументом **map**, упорядоченые в соответствии в переданным в аргументе **comparator**. Если **comparator** не указан или равен **null** – сравнение элементов будет производиться как сравнение значений в соответствии со стандартной Ecma сортировкой, что эквивалентно **Sort.MAP_VALUE**. Если объект переданный в аргументе **map** равен **null** – результатом выполнения метода будет **null**.

Storage API

Версия последнего изменения данного API: 677

Доступ к методам API осуществляется через объект с именем **Storage** (имя может быть изменено – она указывается в параметрах модуля хранилища), который регистрируется в глобальном контексте домена.

Каждый из объектов хранилища может находиться в одном их состояний, описанных для его типа. Состояние определяет, подлежит ли данный объект пролистыванию (для построения меню и списков на сайте) или поиску (для поиска объектов на сайте), а также доступен ли он

пользователям сайта без авторизации (более сложные и гибкие права доступа для отдельных пользователей и групп пользователей на любой объект хранилища ве зависимости от его типа также могут быть определены при помощи ACL). Набор статусов по умолчанию следующий:

Состояние	**Смысл**	**Пролистывание**	**Поиск**	**Доступ**
«**draft**»	Черновик	*Нет*	*Нет*	Авторизация
«**ready**»	Готов	*Нет*	*Нет*	Свободный
«**system**»	Системный	**Да**	*Нет*	Свободный
«**published**»	Опубликован	**Да**	**Да**	Свободный
«**archive**»	Архивный	*Нет*	**Да**	Свободный
«**dead**»	Устаревший	*Нет*	*Нет*	Свободный

В методах с аргументом **`filter`**, при помощи данного аргумента указываются критерии отбора элементов для результатирующего списка объектов. В выражении поддерживаются скобки, условия «и» и «или», например: **`$type=Image || ($folder=true && $type=Gallery)`**. Следующие поля могут быть использованы как элементы критерия:

`$folder`	Условие сравнения: «**=**», возможные значения: **`true`** / **`false`**. Отбирает элементы по признаку **`$folder`**.
`$listable`	Условие сравнения: «**=**», возможные значения: **`true`** / **`false`**. Отбирает элементы по признаку **`$listable`**, соответствующему текущему состоянию объекта. Для умолчательного набора статусов: **`($state=system || $state=published)`**.
`$searchable`	Условие сравнения: «**=**», возможные значения: **`true`** / **`false`**. Отбирает элементы по признаку **`$listable`**, соответствующему текущему состоянию объекта. Для умолчательного набора статусов: **`($state=system || $state=published)`**.
`$state`	Условие сравнения: «**=**», возможные значения: **`draft`**/**`ready`**/**`system`**/**`published`**/**`archive`**/**`dead`**.
`$type`	Условие сравнения: «**=**», возможные значения: <имя_типа>.
`$key`	Условие сравнения: «**=**», возможные значения: <имя объекта>
`$title`	Условие сравнения: «**=**», возможные значения: <строка> Поиск по этому полю производится при помощи запроса к СУБД, т.к. значение этого поля не хранится в дисковом кеше.
`$owner`	Условие сравнения: «**=**», возможные значения: <идентификатор_пользователя> Поиск по этому полю производится при помощи запроса к СУБД, т.к. значение этого поля не хранится в дисковом кеше.

Когда в строковом значении требуется использовать символы пробела, операторы и прочии не alpha-numeric их следует эскейпить обратным слэшом \. Учитывая, что фильтр передается в виде строки такой эскейпинг должен выглядеть как последовательность из двух обратных слэшей. Например:

```
var query = "$title=\\#\\ INVALID\\ USER\\ ID\\ \\#"
```

В методах с аргументом **sort**, при помощи данного аргумента указывается порядок следования элементов в списке, данный аргумент может принимать следующие значения:

null, «**history**» или «**$created-**»	По убыванию даты создания объекта
«**alphabet**», «**$title**» или «**$title+**»	В алфавитном порядке заголовка объекта
«**log**», «**$created**» или «**$created+**»	По возрастанию даты создания объекта
«**$title-**»	В обратном алфавитном порядке заголовка объекта
«**changed**» или «**$modified-**»	По убыванию даты изменения объекта
«**$modified+**» или «**$modified**»	По возрастанию даты изменения объекта
«**listing**», «**$key**» или «**$key+**»	По возрастанию имени объекта
«**$key-**»	По убыванию имени объекта

Storage

Объект **Storage** является интерфейсом к методам хранилища.

```
Entry  getRoot()
```

Возвращает коренной объект хранилища.

```
Entry  getByGuid(
             String guid )
```

Возвращает объект хранилища с указанным **guid**. Если такой объект недоступен, результатом выполнения будет **null**.

```
Entry  getByAlias(
             String alias,
             boolean all )

List<Entry>  searchForIdentity(
             String identity,
             boolean all )


boolean      areLinksSupported()

boolean      areSynchronizationsSupported()

boolean      areSchedulesSupported()

boolean      areHistoriesSupported()

boolean      areVersionsSupported()

boolean      areAliasesSupported()


String        getEntryGuid(
                    Entry entry )
```

Возвращает **guid** указанного объекта, если указанный объект **null** – результат выполнения данного метода будет **null**. Требуется для получения истинного **guid** объекта, так как метод **Entry.getGuid()** может быть переопределен. Также, выполняется быстрее, чем

Entry.getGuid(), так как не проверяет переопределение метода и не производит загрузку полей объекта.

```
String         getEntryKey(
                     Entry entry )
```

Возвращает **key** указанного объекта, если указанный объект **null** – результат выполнения данного метода будет **null**. Требуется для получения истинного **key** объекта, так как метод **Entry.getKey()** может быть переопределен. Также, выполняется быстрее, чем **Entry.getKey()**, так как не проверяет переопределение метода и не производит загрузку полей объекта.

Entry

Объекты типа **Entry** представляют собой ссылки на объекты хранилища. Ссылка на объект хранилища представляет собой совокупность данных объекта и его привязки к иерархии дерева хранилища. На каждый реальный объект хранилища может быть заведено несколько ссылок (линков) в различных местах иерархии дерева хранилища, также, они могут отличаться именем и статусом. Нет способа работать напрямую с данными нижестоящего объекта хранилища – вся работа производится через ссылки на объекты.

```
String getGuid()
String getParentGuid()
String getLinkedIdentity()
String getLocationControl()
String getLocationAbsolute()
String getLocation()
String getKey()      // $key
String getTitle()    // $title
int    getState()    // $state
boolean      isFolder()    // $folder
String getTypeName() // $type
long   getCreated()  // $created
long   getModified() // $modified
String getOwner()    // $owner
Map    getData()     // not editable
Map    getDataClean()      // without system fields
String[]     getAliases()

Change createChange()
```

Создает объект представляющий собой транзакцию для внесения изменений в текущий **Entry** и, соответственно, на связанный с ним объект хранилища.

```
Schedule     getSchedule()
```

Возвращает объект содержащий план заданий для текущего **Entry**, при помощи этого объекта можно также вносить изменения в план заданий.

```
Sync   getSynchronization()
```

```
Entry  getHistorySnapshot(
             String historyId )
History[]    getHistory()
```

```
Entry  getVersion(
             String versionId )
Version[]    getVersions()
```

```
Change createChild()
```

Возвращает транзакцию для последующего создания дочернего к текущему **Entry** объекта.

```
Entry  getParent()
```

Возвращает родительский объект для текущего **Entry**. Если текущий объект является коренным объектом хранилища – этот метод вернет **null**.

```
Entry  getChildByName(
             String name )
```

Возвращает дочерний к текущему **Entry** объект с именем указанным аргументом **name**. Если такого объекта не найдено, результатом выполнения данного метода будет **null**.

```
Entry  getChildByPath(
             String path )
List<Entry>  getFiles(
             int limit,
             String sort )
```

Идентично **searchLocal(limit, true, sort, -1L, -1L, "$folder=false")**. Возвращает список дочерних элементов, не являющихся папками. Параметр **limit** равный нулю – означает отсутствие ограничение на количество результатов.

```
List<Entry>  getFilesListable(
             int limit,
             String sort )
```

Идентично **searchLocal(limit, true, sort, -1L, -1L, "$folder=false && $listable=true")**. Возвращает список дочерних элементов, не являющихся папками и состояние которых подразумевает пролистывание. Параметр **limit** равный нулю – означает отсутствие ограничение на количество результатов.

```
List<Entry>  getChildren(
             int limit,
             String sort )
```

Идентично **`searchLocal(limit, true, sort, -1L, -1L, null)`**. Возвращает список дочерних элементов. Параметр **`limit`** равный нулю – означает отсутствие ограничение на количество результатов.

```
List<Entry>  getChildrenListable(
             int limit,
             String sort )
```

Идентично **`searchLocal(limit, true, sort, -1L, -1L, "$listable=true")`**. Возвращает список дочерних элементов, состояние которых подразумевает пролистывание. Параметр **`limit`** равный нулю – означает отсутствие ограничение на количество результатов.

```
List<Entry>  getFolders()
```

Идентично **`searchLocal(0, true, "alphabet", -1L, -1L, "$folder=true")`**. Возвращает список дочерних элементов, являющихся папками и состояние которых подразумевает пролистывание. Параметр **`limit`** равный нулю – означает отсутствие ограничение на количество результатов.

```
List<Entry>  getFoldersListable()
```

Идентично **`searchLocal(0, true, "alphabet", -1L, -1L, "$folder=true && $listable=true")`**. Возвращает список дочерних элементов, являющихся папками и состояние которых подразумевает пролистывание. Параметр **`limit`** равный нулю – означает отсутствие ограничение на количество результатов.

```
List<Entry>  search(
             int limit,
             boolean all,
             long timeout,
             String sort,
             long startDate,
             long endDate,
             String filter )
```

Параметр **`limit`** равный нулю – означает отсутствие ограничения на количество результатов.

Параметр **`all`** определяет среди какого набора объектов проводится поиск, если этот параметр равен **`false`** поиск будет производиться только по объектам в состоянии помеченном как **`$searchable`**, в любом случае в поиске не участвуют объекты в состоянии **`draft`**/**`черновик`** – так как такие объекта не индексируются.

Если указанный **`timeout`** менее одной секунды (1000L) – он игнорируется и запросы выполняются столько времени, сколько требуется для их полного прохождения.

Параметры **`startDate`** и **`endDate`** при значениях, отличных от **`-1`**, определяют интервал времени для фильтра по полю **`$created`**.

В большинстве случаев (в зависимости от условий поиска) поиск будет осуществляться в виде нескольких параллельных и последовательных запросов, если при запуске очередного задания окажется, что указанное параметром **`timeout`** отрезок времени уже истек, поиск будет прерван (не ожидайте от этого аргумента филигранной точности).

```
Map<Integer, Map<Integer, Map<String, Number>>>
       searchCalendar(
              boolean all,
              long timeout,
              long startDate,
              long endDate,
              String filter )
```

Возвращает map содержащий сгруппированную по годам, месяцам и дням информацию о количестве объектов продходящих под условия поиска.

Параметр **`all`** определяет среди какого набора дочерних объектов проводится поиск, если этот параметр равен **`false`** поиск будет производиться только по объектам в состоянии помеченном как **`$searchable`**, в любом случае в поиске не участвуют объекты в состоянии **`draft`**/**`черновик`** – так как такие объекта не индексируются.

Параметры **`startDate`** и **`endDate`** при значениях, отличных от **`-1`**, определяют интервал времени для фильтра по полю **`$created`**.

В большинстве случаев (в зависимости от условий поиска) поиск будет осуществляться в виде нескольких параллельных и последовательных запросов, если при запуске очередного задания окажется, что указанное параметром **`timeout`** отрезок времени уже истек, поиск будет прерван (не ожидайте от этого аргумента филигранной точности).

```
List<Entry>  searchLocal(
              int limit,
              boolean all,
              String sort,
              long startDate,
              long endDate,
              String filter )
```

Параметр **`limit`** равный нулю – означает отсутствие ограничения на количество результатов.

Параметр **`all`** определяет среди какого набора дочерних объектов проводится поиск, если этот параметр равен **`false`** поиск будет производиться только по объектам в состоянии помеченном как **`$searchable`**.

Параметры **`startDate`** и **`endDate`** при значениях, отличных от **`-1`**, определяют интервал времени для фильтра по полю **`$created`**.

```
Map<String, Number>
       searchLocalAlphabet(
              boolean all,
              Map<String, String> alphabetConversion,
              String defaultLetter,
              String filter)
```

Возвращает map содержащий сгруппированную по буквам информацию о количестве объектов продходящих под условия поиска. Имеются ввиду первые буквы из заголовка объекта.
Параметр **`all`** определяет среди какого набора дочерних объектов проводится поиск, если этот параметр равен **`false`** поиск будет производиться только по объектам в состоянии помеченном как **`$searchable`**. Поскольку алфавит сайта не обязан содержать всех возможных unicode символов, аргументом **`alphabetConversion`** передается таблица соответствия интересующих нас букв, все остальные символы считаются равными аргементу **`defaultLetter`**.

```
List<Entry>  searchLocalAlphabet(
             int limit,
             boolean all,
             String sort,
             Map<String, String> alphabetConversion,
             String defaultLetter,
             String filterLetter,
             String filter)
```

Возвращает список объектов соответствующих условиям поиска и фильтрации по первой букве учитывая таблицу преобразования букв. Имеются ввиду первые буквы из заголовка объекта. Параметр **`all`** определяет среди какого набора дочерних объектов проводится поиск, если этот параметр равен **`false`** поиск будет производиться только по объектам в состоянии помеченном как **`$searchable`**. Поскольку алфавит сайта не обязан содержать всех возможных unicode символов, аргументом **`alphabetConversion`** передается таблица соответствия интересующих нас букв, все остальные символы считаются равными аргементу **`defaultLetter`**.

```
Map<Integer, Map<Integer, Map<String, Number>>>
       searchLocalCalendar(
              all,
              startDate,
              endDate,
              filter )
```

Возвращает map содержащий сгруппированную по годам, месяцам и дням информацию о количестве объектов подходящих под условия поиска.

Параметр **`all`** определяет среди какого набора дочерних объектов проводится поиск, если этот параметр равен **`false`** поиск будет производиться только по объектам в состоянии помеченном как **`$searchable`**.

Параметры **`startDate`** и **`endDate`** при значениях, отличных от **`-1`**, определяют интервал времени для фильтра по полю **`$created`**.

History

```
String getGuid()
long   getDate()
String getTitle()
```

Version

```
String getGuid()
String getParentGuid()
long   getDate()
String getComment()
String getTitle()
String getOwner()
String getTypeName()
```

Change

Объекты типа **Change** представляют собой транзакцию связанную с внесением изменений в структура хранилища. Каждый объект типа **Change** привязан к конкретной ссылке на объект хранилища. Изменения могут быть вложенными и связанными – это позволяет вносить множественные изменения за одну транзакцию.

```
void   setCreateLocal(
              boolean local )     // apply synchronizations or not

void   setCreateLinkedIn(
              Entry folder )

void   setCreateLinkedIn(
              Entry folder,
              String key )

void   setCreateLinkedWith(
              Entry entry ) // Only for newly created changes

void   setCommitLogged()   // Only for existent objects
```

Последующий вызов метода **commit()** произведет запись в историю объекта, что предоставит возможность в течении определенного времени откатить изменения (например, метод **setCommitLogged** вызывается когда пользователь редактирует документ в административном интерфейсе). Также, если для данного объекта включен режим версионности – вызов метода **commit()** создаст новую версию (чтобы новая версия стала активной см. **setCommitActive**).

По умолчанию, без вызова метода **setCommitLogged** независимо от режимов версионности и возможностей хранилища вести историю изменений эти возможности не будут задействованы и будет произведено обновление непосредственно текущий активной версии объекта.

```
void   setCommitActive()   // Only for existent objects
```

Если для данного объекта включен режим версионности – последующий вызов метода **commit** создаст новую активную версию.

Вызов метода **setCommitActive** не окажет никакого влияния, если небыло произведено вызова метода **setCommitLogged** или для данного объекта не был включен режим версионности.

```
String getGuid()

String getParentGuid()

String getLocationControl()

String getLocationAbsolute()

String getLocation()

String getKey()      $key

boolean      isFolder()    $folder

long   getCreated()  $created

void   setParent(
              Entry entry )

void   setParentGuid(
              String guid )

void   setKey(
              String key )  $key

void   setFolder(
              boolean folder )    $folder
```

```
void   setCreated(
              long created )      $created

Sync   getSynchronization()

Schedule      getSchedule()

Change createChild()
```

Создает объект представляющий собой транзакцию для создания нового объекта хранилищща. При этом новая транзакция является вложенной в текущую тразакцию: изменения созданной транзакции будут произведены в рамках выполнения родительской транзакции и либо завершатся успешно, либо откатятся. Разумеется, что для того чтобы транзакция осуществилась требуется вызвать метод commit у вложенной транзакции до того как будет осуществлен вызов метода commit у родительской транзакции.

```
Change createChange(
              Entry entry )
```

Создает объект представляющий собой транзакцию для внесения изменений в объект **`entry`** и, соответственно, на связанный с ним объект хранилища. При этом новая транзакция является вложенной в текущую тразакцию: изменения созданной транзакции будут произведены в рамках выполнения родительской транзакции и либо завершатся успешно, либо откатятся. Разумеется, что для того чтобы транзакция осуществилась требуется вызвать метод commit у вложенной транзакции до того как будет осуществлен вызов метода commit у родительской транзакции.

```
boolean       getVersioning()

void   setVersioning(
              boolean versioning )

void   setVersionComment(
              String comment )

void   setVersionData(
              Map data )

String getVersionId()

Change getVersion(
              String versionId )

Version[]     getVersions()

Change getHistorySnapshot(
              String historyId )

History[]     getHistory()


String getTitle()    $title

int    getState()    $state

String getTypeName() $type

void   setTitle(
              String title )      $title

void   setState(
              int state )   // STATE_DRAFT, STATE_READY, STATE_PUBLISH,

                            // STATE_SYSTEM, STATE_ARCHIVE, STATE_DEAD

void   setTypeName(
              String typeName )   $type


Map    getParentalData()   // not editable
```

```
Map    getData()           // editable


void   aliasAdd(
              String alias )
void   aliasRemove(
              String alias )


void   commit()
```

Сохраняет внесенные изменения. Если реально внесенные изменения не отличаются от уже имеющихся в объекте, дальнейшие действия игнорируется. Для подробностей см. метод **`touch()`**.

```
void   resync()
void   segregate()
```

Отделяет текущий линк на объект от остальных линков на тотже объект. После вызова этого метода данный линк указывает на новый, идентичный предыдущему объект.

```
void   touch()
```

При нормальном вызове **`commit()`** объект может быть не обновлен, если в объект **`change`** небыло внесено никаких реальных изменений. После вызова метода **`touch()`**, объект будет обновлен при выполнении **`commit()`** в любом случае.

Обновление объекта включает в себя: смену владельца объекта, обновление даты изменения объекта (поле **`$modified`**), обновление поисковых индексов объекта, сброс связанных с этим объектов кешей, пересинхронизацию всех линков на объект.

```
void   unlink()
```

Удаляет текущий линк на объект. Когда у объекта не останется больше линков – он будет удален автоматически.

```
void   unlink(
              Boolean soft )      // unlink to recycle bin
void   delete()
```

Удаляет все линки на объект.

```
void   delete(
              Boolean soft )      // delete to recycle bin
```

Schedule

Объект типа **`Schedule`** представляет собой план задач планировщика для конкретного объекта хранилища. Каждая задача имеет имя, назначенную дату, имя метода и набор параметров. В указанное время система осуществит вызов указанного метода и передаст ему параметры.

```
boolean      isEmpty()
```

Возвращает истину если для данного объекта очередь запланированных задач пуста.

```
void   scheduleFill(
              Schedule target,
              boolean replace )
```

Копирует все задачи из текущего плана в план указанный аргументом **target**, аргумент **replace** определяет будет ли производиться замена одноименных задач, которые уже могут быть определены в плане, в который осуществляется копирование.

```
void   schedule(
              String name,
              boolean replace,
              long date,
              String command,
              Map parameters )
```

Устанавливает привязанное к документу задание для планировщика. В дату указанную аргументом **date** у документа должен быть вызван метод с именем указанным параметром **command** и аргументами указанными в **parameters**. Аргумент **name** определяет имя задания. Не может существовать двух заданий с одинаковым именем для одного документа. Если задание с таким именем уже существует, в зависимости от значения аргумента **replace** оно будет заменено новым заданием или новое задание будет проигнорировано.

```
void   scheduleCancel(
              String name )
```

Отменяет задание с именем указанным аргументом name. Если задания с таким именем у документа в данный момент не определено – ничего не происходит.

```
void   clear()
```

Удаляет все задачи планировщика ассоциированные с данным объектом.

```
void   commit()
```

Сохраняет изменения произведенные с задачами планировщика для данного объекта.

Sync

```
boolean      isEmpty()

String[]     getExportSynchronizations()

String[]     getImportSynchronizations()

void   synchronizeExport(
              String guid )

void   synchronizeImport(
              String guid )

void   synchronizeExportCancel(
              String guid )

void   synchronizeImportCancel(
              String guid )

void   clear()

void   commit()
```

UserManager API

Версия последнего изменения данного API: 676

Данный API предоставляет доступ к системе управления пользователями текущего домена и доступ к текущему пользователю, в контексте которого происходит выполнение скрипта.

Доступ к методам API осуществляется через объект с именем **`UserManager`**, и объект с именем **`User`**, которые регистрируется в глобальном контексте домена.

UserManager

```
Group[]       getAllGroups()

User   getUser(
              String userId)      //create == true

User   getUser(
              String userId,
              boolean create )

User   getUserByLogin(
              String login) //create == true

User   getUserByLogin(
              String login,
              boolean create )

User[] searchUsers(
              String login,
              String email,
              long logonStart,
              long logonEnd)

Group  getGroup(
              String groupId,
              boolean create )

Group[]       setGroups(
              User user,
              Group[] groups)

void   updateGroups(
              User user,
              Group[] groupRemove,
              Group[] groupAdd)

String registerUser(
              Map data)     // login, email, password, etc.

String registerUser(
              String guid,  // nulls allowed
              String login,
              String email,
              String password,    // nulls allowed
              Map data)
```

Если аргумент guid равен NULL – система сформирует идентификатор пользователя автоматически. Данный метод возвращает идентификатор созданного пользователя.

```
boolean       checkPassword(
              String password)

String generatePassword()

String generatePassword(
              int length,
              boolean smallLetters,
              boolean bigLetters)
```

```
boolean      deleteUserByID(
             String userId)

boolean      deleteUserByLogin(
             String login)

boolean      deleteUserByMail(
             String email)
```

Удаляет пользователя по идентификатору пользователя, логину пользователя или адресу электронной почты, соответственно.

User

scope methods (current user access)

```
int    US_UNAUTHORIZED

int    US_AUTHORIZED_AUTOMATICALLY

int    US_AUTHORIZED

int    US_AUTHORIZED_HIGH


User   getUser()


String getLanguage()

void   setLanguage(
             String language)

int    getState()

String getUserID()


String requestAuthorization()

String requestHighAuthorization()

String ensureAutomaticAuthorization()


String requestGroup(
             String groupId)
```

User

object methods

```
boolean      isInGroup(
             Group group)

boolean      isInGroup(
             String groupId)

Group[]      getGroups()

String getKey()

String getLogin()

String getEmail()

String getDescription()

int    getType()

String getLanguage()
```

```
Map    getProfile()

Map    getProfile(
              String key,
              boolean create)

long   getCreated()
```

Дата создания пользователя.

```
long   getChanged()
```

Дата последнего входа в систему (login) или последнего изменения настроек пользователя (commit).

```
boolean       isAnonymous()

boolean       isActive()

boolean       isSystem()

boolean       checkPassword(
              String password)

void   setRegistered()
```

При использовании этого метода получается не прошедший валидацию зарегистрированный пользователь.

```
void   setActive()
```

При использовании этого метода получается прошедший валидацию зарегистрированный пользователь.

```
void   setSystem()


void   setLogin(
              String login)


void   setLanguage(
              String language)


void   setEmail(
              String email)


void   setDescription(
              String description)


void   setType(
              int type)


void   setProfile(
              Map profile)
```

```
void   setProfile(
              String key,
              Map profile)


void   groupAdd(
              Group group)


void   groupRemove(
              Group group)


void   setPasswordHigh(
              String password)


void   setPasswordNormal(
              String password)


void   setPassword(
              String password)
```

Устанавливает сразу и HIGHER и NORMAL пароли.

```
void   commit()
```

Сохраняет произведенные изменения.

Group

```
User[] getUsers()

String getKey()

String getTitle()

String getDescription()

int    getAuthLevel()

void   setTitle(
              String title)

void   setDescription(
              String description)

void   setAuthLevel(
              int authLevel)

void   commit()
```

Описание типов

Система оперирует объектами – весь виртуальный домен представляет собой определенную иерархию объектов, каждый из которых представляет собой произвольный набор полей и методов. Объект это минимальная макро-единица данных в системе – любые данные хранятся в виде объекта. Каждый объект является экземпляром определенного типа, тип определяет базовое поведение объекта, поля и методы присущие ему по умолчанию и некоторые прочие параметры. Часть типов являются встроенными в систему, например: права доступа, настройки публичной точки доступа и т.п. Остальные типы сайта описывает сборщик, например: новость, голосование и т.п.

Для описания типов используется scheme (схема). Файлы с расширением .scheme должны быть расположены в каталоге **`%SITE_FOLDER%/types/`** или **`%ACM_PROTECTED%/resources/type/`**.

Схема (scheme)

Версия последнего изменения в данном формате: 673

Схема является XML-описанием типа. Далее будут приводиться строки из абстрактного файла-схемы с комментариями по поводу увиденного (пример рассматриваемого в этой главе файла расположен в **`%ACM_PUBLIC%/resources/doc/example/SchemeDescribedEnglish.scheme`**):

```
<?xml version="1.0" encoding="UTF-8"?>
```

Первая строка схемы, как и любого xml-файла должна описывать кодировку файла. Притом, важно, чтобы указанная здесь кодировка соответствовала реальной кодировке файла – той, какая указана/выбрана в том редакторе, в котором этот файл редактируется. Иначе у системы будут проблемы с загрузкой данной схемы.

```
<type>
```

Коренной элемент XML.

```
    <title>common title</title>
```

Имя типа, под этим именем оно будет значиться в выборе типа объекта в интерфейсе управления.

```
    <visibility>default</visibility>
```

Определяет «видимость» данного типа в интерфейсе управления. Может принимать значения: **`default`** (значение по умолчанию, тип видно в списках выбора типов) и **`hidden`** (тип не показывается в списке выбора, однако, его можно увидеть в качестве уже выбранного типа среди свойств объекта). Не всем типам требуется быть доступными с клиентской стороны, некоторые типы могут быть внутренними для системы типами объектов или наборами статических функций.

```
    <final>false</final>
```

Определяет доступно ли создание типов, наследующих от данного типа. Допустимые значения: **`false`** (данный тип запрещает дальнейшее наследование) или **`true`** (значение по умолчанию, разрешено наследование новых типов от этого типа).

```
    <icon>document</icon>
```

Определяет иконку соответствующую этому типа. Значение по умолчанию: **`document`**.

```
<versioning>false</versioning>
```

Включает или выключает поддержку версионности для объектов этого типа по умолчанию. Пользователь с достаточными правами может явно выбирать режим версионности проходя полный путь мастера создания документа, этот параметр отвечает только за значение по умолчанию. Если этот параметр не указан, режим версионности будет выключен по умолчанию.

```
<state>publish</state>
```

Определяет статус объектов этого типа по умолчанию. У пользователей с достаточными правами есть возможность управлять статусом объекта, этот параметр отвечает только за значение по умолчанию. Среди принятых допустимых состояний объекта значения: **`draft`**, **`ready`**, **`system`**, **`published`**, **`archive`**, **`dead`**. Если этот параметр не указан статусом по умолчанию будет **`draft`**.

```
<statelist>
	<state>ready</state>
	<state>publish</state>
	<state>draft</state>
</statelist>
```

Задает набор валидных для объектов данного типа статусов. Если этот параметр не указан для объектов данного типа будут считаться валидными любые статусы из набора: **`draft`**, **`ready`**, **`system`**, **`publish`**, **`archive`**, **`dead`**.

```
<!-- optional (false by default) default instance type -->
<folder>true</folder>

<!-- one or more optional type replacements -->
<replacement>type_key1</replacement>
<replacement>type_key2</replacement>

<!-- optional ($default used when omitted) parental type key,
	This type will be derived from type specified -->
<extends>type_key</extends>

<!-- optional content listing fields definition -->
<listing>$key, $title, typeName:fieldName</listing>

<children>
	<type>type1</type>
	<type>type2</type>
</children>
```

Задает набор типов, пригодных для создания в качестве детей объектов описываемого типа. Разумеется, унаследованные от перечисленных типы будут также включены в множество потенциальных деток объекта.

Если список пуст – никакие объекты непригодны в качестве детей объектов описываемого типа. Если элемент «**`children`**» не указан вовсе – настройки будут взяты из родительского типа. Если по всей иерархии родителей не найдется явно указанного элемента «**`children`**», то объекты любых типов будут разрешены к созданию в качестве деток.

```
<parents>
	<type>type1</type>
	<type>type2</type>
</parents>
```

Задает набор типов, в которых можно создавать объекты описываемого типа в качестве детей. Разумеется, унаследованные от перечисленных типы будут также включены в множество потенциальных родителей объекта.

Если элемент «**parents**» не указан вовсе – настройки будут взяты из родительского типа. Если по всей иерархии родителей не найдется явно указанного элемента «**parents**», то объекты данного типа разрешены к созданию в качестве детей объектов любых типов.

```
<!-- optional ('modify' used when omitted) object constructor
       definition -->
<create>
       <!-- optional ('modify' form used when omitted) form to
              show in control interface -->
       <form>
              <title>Form title.</title>

              <!-- optional script to execute before form is shown,
                     parameters:
                            'data' - form data -->
              <prepare class="script" type="ACM.TPL"> <![CDATA[
              ]]> </prepare>

              <!-- optional fieldset for a form to show -->
              <fields class="fieldset">
              </fields>

              <!-- optional script to execute on submission, parameters:
                            'change' - current change object
                            'data' - form data
                     Default script is equivalent to:
                            change.getData().putAll( data )  -->
              <submit class="script" type="ACM.TPL"> <![CDATA[
                     <%EXEC: change.getData().putAll( data ) %>
              ]]> </submit>

       </form>

       <!-- optional ('modify' script used when omitted) script
                     to execute prior commit, parameters:
                     'this' - dummy entry to make commands accessible
                     'change' - current change object -->
       <trigger class="script" type="ACM.TPL"> <![CDATA[
       ]]> </trigger>
</create>

<!-- property modification definition -->
<modify>
       <!-- form to show in control interface -->
       <form>
              <title>Form title.</title>

              <!-- optional script to execute before form is shown,
                     parameters:
                            'this' - current storage object
                            'data' - form data -->
              <prepare class="script" type="ACM.TPL"> <![CDATA[
              ]]> </prepare>

              <!-- optional fieldset for a form to show -->
              <fields class="fieldset">
              </fields>

              <!-- optional script to execute on submission, parameters:
                            'this' - current storage object
                            'change' - current change object
```

```
                                'data' - form data
                        Default script is equivalent to:
                                change.getData().putAll( data )  -->
                <submit class="script" type="ACM.TPL"> <![CDATA[
                        <%EXEC: change.getData().putAll( data ) %>
                ]]> </submit>
        </form>

        <!-- optional script to execute prior commit, parameters:
                        'change' - current change object
                        'this' - current storage object -->
        <trigger class="script" type="ACM.TPL"> <![CDATA[
        ]]> </trigger>
</modify>

<!-- optional, one or more static type fields initialization.
        NOTE: this section is initialized before type commands, so you cannot use this
type's
                commands within constant expression.
        -->
<static class="fieldset">
</static>

<!-- optional, fieldset to retrieve fields when loading an object instance.
        Use it to specify evaluable, externally-accessible and non-indexable fields.
        Field attribute respond='true' - enable respond handler
                response is disabled for all fields by default.
        Field attribute indexing='false' - disable field indexing
                all fields are indexable by default.
        -->
<load class="fieldset">
</load>

<!-- optional (no response when omitted) object
                responding definition -->
<respond>

        <!-- optional: default behavior for response -->
        <behavior>
                <!-- what requests to handle:
                                'this' - only own requests (default)
                                'parent' - all own requests are redirected to parent
                                'any' - any requests passing through this
                                                object and not pointing to existing
                                                child objects
                                'all' - all requests passing through this
                                                object -->
                <handle>this</handle>

                <!-- is authorization required to access this kind of objects? default
                                value is 'true'. This feature doesn't specify any
                                security permissions, but implies GUEST permissions
                                indirectly since there are no way for real GUEST to
                                pass authorization. -->
                <anonymous>true</anonymous>

                <!-- public or private response, default value is
                                'true' but response still may be private
                                depending on access rights and so on.
                                The main idea is that public responses
                                may be safely cached by caches\proxies between
                                server and client for public access acceleration
                                in contrast with private responses with some
                                user-specific data. -->
                <public>true</public>

                <!-- client cache expiration time (2h by default), time to keep the
                        response in the user-side caches. The actual time the response
                        is kept may vary depending on user settings and browser's
                        caching algorithm. -->
```

```
                <ttl>2h</ttl>

                <!-- server cache expiration time (2h by default), time to keep the
                        response in the server-side caches. The actual time the response
                        is kept may vary depending on available memory amount and system
                        load. -->
                <cache>2h</cache>
        </behavior>

        <!-- filter to finish content result map generated by a
                child's respond script. All filters are applied in
                order from just responded entry's parent till
                share root. Content, or any other response should be returned
                in the same way as in respond scripts. Parameters:
                'this' - current storage object
                'Request' - current request object
                'content' - upper content to filter -->
        <filter class="script" type="ACM.JAVASCRIPT"> <![CDATA[
                // if( !HashValid( content ) ) return content;
                return content;
        ]]> </filter>

        <!-- script to render, parameters:
                'this' - current storage object
                'Request' - current request object  -->
        <script class="script" type="ACM.TPL"> <![CDATA[
        ]]> </script>

</respond>

<!-- optional object deletion definition -->
<delete>
        <!-- form to show in control interface if required -->
        <form>
                <title>Form title.</title>

                <!-- optional script to execute before form is shown,
                        parameters:
                                'this' - current storage object
                                'data' - form data -->
                <prepare class="script" type="ACM.TPL"> <![CDATA[
                ]]> </prepare>

                <!-- optional fieldset for a form to show -->
                <fields class="fieldset">
                </fields>

                <!-- optional script to execute on submission, parameters:
                                'this' - current storage object
                                'change' - current change object
                                'data' - form data  -->
                <submit class="script" type="ACM.TPL"> <![CDATA[
                ]]> </submit>
        </form>

        <!-- script to execute prior deletion, parameters:
                        'this' - current storage object -->
        <trigger class="script" type="ACM.TPL"> <![CDATA[
        ]]> </trigger>
</delete>

<!-- one or more type commands -->
<command>
        <!-- command type for access checks, one of:
                'view' - command is intended to view some query result.
                'execute' - command is intended to update an object.
                'modify' - command is intended to modify an object.
                'publish' - command is intended to change object state.
                'hidden' - command is hidden.
                -->
```

```
<type>view</type>

<!-- command key
       Initial command key, used to add commands to a type and not
       required to reflect actual property name from where this command
       is referenced at the moment.
       Usage example:
              var key = o.method.key;
       -->
<key class="string">cmd_key</key>

<!-- optional (key used when omitted) command title
       Usage example:
              var title = obj.method.title || "command-run";
       -->
<title class="string">Command title</title>

<!-- optional ('command-run' used when omitted) command icon
       Usage example:
              var icon = obj.method.icon || "command-run";
       -->
<icon class="string">command-run</icon>

<!-- command availability expression, 'true' by default.
       Serves as a hint for user interfaces to hide commands that will
       make no sense or produce errors in current state.

       Current storage object is passed as 'this' parameter.

       Usage example:
              var show = !obj.method.enable || obj.method.enable(obj);
              var hide = obj.method.enable && !obj.method.enable(obj);
       Implementation example:
              <enable class="expression">!this.image</enable>
        -->
<enable class="expression">true</enable>

<!-- command belonging, one of:
       'false' - command belongs to an instance (default), instance
                     is being passed as 'this' in context of execution.
       'true' - command belongs to a type, no 'this' reference
                     accessible in context, slightly faster.
       -->
<static>false</static>

<!-- command export, one of:
       'false' - normal command (default).
       'true' - command will be applied to all types in given TypeRegistry.
       -->
<export>false</export>

<!-- command result holdability and execution type, one of:
       'always' - command execution occurs every time command called (default)
       'buffered' - executed asynchronously, immediately returns null, data
                           passed as an array of maps.
       'deferred' - executed asynchronously, immediately returns null, called
                           serially thereafter.
       'once' - command result may be constantly cached till next type reload.
                           Such commands should not use any arguments.
       'cache' - command results are cached for current parameters.
       'auto' - called automatically in a loop, no explicit calls possible. No
                           'data' passed to this method and this method cannot
                           be an instance method (static only).
                           Such commands should not use any arguments.
       -->
<execute>always</execute>

<!-- cache expiration, when execution type is 'cache', 15m is default -->
<expire>15m</expire>
```

```
            <!-- first call in sequence delay, when execution type is 'buffered' or
                  'auto', 0 is default - starts immediately -->
            <delay>0</delay>

            <!-- next call in sequence delay, when execution type is 'buffered' or
                  'auto', 0 is default for 'buffered', 10m is default for 'auto',
                  when this parameter is 0 for an 'auto' type - only first call
                  will be performed. -->
            <period>10m</period>

            <!-- form to show in control interface if required -->
            <form>
                  <!-- optional (command title user when omitted) title
                              for a form -->
                  <title>Form title.</title>

                  <!-- optional script to execute before form is shown,
                        parameters:
                              'this' - current storage object
                              'data' - form data -->
                  <prepare class="script" type="ACM.TPL"> <![CDATA[
                  ]]> </prepare>

                  <!-- optional fieldset for a form to show -->
                  <fields class="fieldset">
                  </fields>

                  <!-- optional script to execute on submission, parameters:
                              'this' - current storage object
                              'data' - form data -->
                  <submit class="script" type="ACM.TPL"> <![CDATA[
                  ]]> </submit>
            </form>

            <!-- command result type:
                        Void
                        Object
                        Boolean
                        Integer
                        Number
                        String... -->
            <result>Object</result>

            <!-- command arguments fieldset, identifies type and default values
                  for every command parameter to be accessibe like ordinary
                  local variables.
                  Arguments feature is not applicable to 'once' or 'auto' method
                  execution type. There no arguments for 'once' and 'auto'
                  execution types at all.
                  For 'buffered' methods an array of maps with arguments will
                  be accessible via 'arguments' local variable in the same
                  manner as passed parameters accessible via 'data' variable, i.e.
                  in the form of array of maps.
                  -->
            <arguments class="fieldset">
            </arguments>

            <!-- script to execute, parameters:
                        'this' - current storage object
                        'arguments' - parameters or form data if arguments are described
                        'data' - parameters or form data if arguments are not described
                        every parameter described in arguments as a variable.
                  -->
            <script class="script" type="ACM.TPL"> <![CDATA[
            ]]> </script>
      </command>
</type>
```

Встроенные обработчики событий

В типе можно описать некоторые методы – обработчики событий, они будут автоматически вызываться при определенных событиях:

- onBeforeCreate –вызывается непосредственно перед коммитом транзакции в которой будет произведена попытка сохранить созданный объект.
- onAfterCreate – вызывается непосредственно после коммита транзакции в которой был создан объект.
- onBeforeLink – вызывается непосредственно перед коммитом транзакции в которой будет произведена попытка создать линк на объект.
- onAfterLink – вызывается непосредственно после коммита транзакции в которой был создан линк на объект.
- onBeforeModify – вызывается непосредственно перед коммитом транзакции в которой будет произведена попытка изменить содержимое объекта.
- onAfterModify – вызывается непосредственно после коммита транзакции в которой было изменено содержимое объекта.
- onBeforeUnlink – вызывается непосредственно перед коммитом транзакции в которой будет произведена попытка удалить ссылку на объект.
- onAfterUnlink – вызывается непосредственно после коммита транзакции в которой было произведено удаление ссылки на объект.
- onBeforeDelete – вызывается непосредственно перед коммитом транзакции в которой будет произведена попытка удалить объект.
- onAfterDelete – вызывается непосредственно после выполнения транзакции в которой был удален объект.
- onBeforeRequestPassThrough – вызывается при прохождении запроса «вглубь» насквозь данного объекта.
- onDirectRequest – вызывается при запросе к данному объекту
- onAfterRequestPassThrough – вызывается при прохождении ответа на запрос обратно через данный объект.

Авторизация формой на сайте

При выбранной авторизации «форма на сайте» процесс происходит следующим образом: при необходимости произвести авторизацию система вызавает скин со следующими параметрами {template: 401, error: <ошибка>, private: true}. Поле error содержит значение с текстом ошибки произошедшей при попытке авторизации или null если ошибки еще не произошло.

Скины (Skins)

Доступ пользователей (публичных и внутренних) к сервисам осуществляется через точки доступа – к каждой точке доступа привязывается скин. Скин определяет поведение каждой конкретной точки доступа. Скины можно разбить на две основные группы: системные встроенные утилитарные скины и скины созданные сборщиком сайта отвечающие за отображение информации сайта и реализацию взаимодействия пользователей с объектами сайта.

Скины могут быть расположены в четырех местах (в случае конфликта/совпадении имени скина, приоритет убывает по мере роста порядкового номера в следующем списке):

1) **`%SITE_FOLDER%/skin/`** - эти скины будут доступны только для данного сайта.
2) **`%ACM_PRIVATE%/skin/`** - эти скины будут доступны только данному инстансу системы.
3) **`%ACM_PROTECTED%/skin/`** - эти скины доступны всему кластеру ACM.
4) **`%ACM_PUBLIC%/resources/skin/`** - в этом каталоге лежат системные скины поставляемые с системой и обновляемые при обновлении системы.

Скин представляет собой zip-архив или папку. Имя архива или папки являются именем/идентификатором скина. В случае конфликта/совпадении имени архива и папки приоритетным будет являться папка. Скин состоит из конфигурационного файла (skin.settings.xml) и, собственно, файлов скина – тип файлов и их значение определяется настройками скина.

Файл skin.settings.xml

Данный файл содержит настройки скина, а также описывает его тип и параметры доступные в процессе обработке любого запроса на точке доступа в которой выбран данный скин. При отсутствии этого файла в скине – данный скин будет проигнорирован системой.

Основным параметром является параметр **`type`**, он определяет типа скина и значение всех остальных файлов и параметров скина. Данный параметр может принимать одно из перечисленных значений:

1) HIERARCHY – является типом по умолчанию.
2) PLAIN
3) EXACT – набор статических файлов.

Для любого скина может быть указан параметр **`prototype`**. Если скин должен переопределять или расширять функционал другого скина, то идентификатор скина, являющегося прототипом создаваемого должен быть указан в параметре **`prototype`**. Значение по умолчанию: «skin-standard».

Пример:

```
<prototype>ctrl-simple</prototype>
```

Также для любого скина может быть указано любое число параметров **`import`**. Их следует использовать когда требуется подключение функционала другого скина для последующего использования.

Примеры:

```
<import package=«ctrl-cimple» namespace=«Admin» />
<import package=«generic-forum» namespace=«Forum» />
```

При помощи атрибута **abstract** можно скрыть скин из списков выбора скина. Значение по умалчанию **false**, такие скины будут показаны в списках выбора.

Пример:

```
<!-- Hide from users -->
<abstract>true</abstract>
```

Скин типа HIERARCHY

Структура файлов такова: в коренной папке лежат шаблоны скина и файлы **skin.settings.xml** и **skin.fieldset.xml**, в папке **$files/** лежат файлы, которые отдаются пользователю немедленно при запросе к **$files/** (обычно это картинки дизайна скина, таблицы стилей и клиентские скрипты), при этом – файлы в этой папке рассматриваются как двоичные и никаких преобразований содержимого и кодировки не производится. Для ситуаций, когда требуется обработка файлов или преобразование кодировки файла в соответствии с запросом браузера пользователя имеется папка **$build/ -** расположенные в ней файлы воспринимаются как скрипты, выполняются единожды, а результат их выполнения выдается пользователям также немедленно при запросе к **$build/** (обычно это таблицы стилей или скрипты сайта, требующие динамического построения или преобразования кодировок). К файлам в **$build/** применяются те же настройки входной кодировки и скриптового языка как и к основным файлам скина.

При прохождении запроса через этот скин, для каждого элемента URL запроса производится прохождение по иерархии дерева объектов сайта. Коренной точкой этого процесса является объект, на котором назначена точка доступа. В процессе прохождения отрабатываются ACL доступа, при этом, для команды GET проверяется право доступа read, а для команды POST право доступа execute.

При достижении объекта отвечающего на данный запрос (являющегося ближайшим по иерархии к месту, куда указывает URL запроса) у него вызывается метод onDirectRequest. Если результат выполнения данного метода не был FINAL – то для каждого элемента в пройденной иерархии будет вызван метод onAfterRequestPassThrough в обратном порядке. Если при прохождении данной цепочки объект характеризующий ответ сервера становится FINAL – процесс прерывается и ответ возвращается.

Если после всех этих операций мы имеем не FINAL ответ – то из шаблонов скина вызывается шаблон соответствующий параметру респонза «**template**» или коду ответа, если **template** не указан система производит выбор шаблона по коду ответа (page.200 – ок, page.204 – пустой ответ, page.401 – необходима авторизация, page.403 – доступ запрещен, page.404 – документ не найден и т.д., если шаблон не найден – производится попытка открыть page.default). Шаблоны скина, также, могут перебрасывать ответ другим шаблонам скина указывает параметр **template**.

Для скина данного типа определены следующие параметра **skin.settings.xml**:

1) type – (string), всегда HIERARCHY.
2) charset – (string), кодировка в которой сохранены файлы скина, являющиеся скриптами. Если этот параметр не указан используется кодировка операционной системы или кодировка явно переданная jvm.
3) renderer – (string), идентификатор языка, на котором написаны файлы, являющиеся скриптами. Или (map) с полем type содержащим идентификатор языка и полем suffix содержащим расширение файлов-шаблонов (включая точку, например: '.js').

4) personalized – (boolean), при значении true будет запрещать кешировать динамические страницы для возможности отображения на этих страницах информации зависящей от текущего пользователя (персонализированной информации). Если этот параметр не указан, его значение считается false.
5) contentType – (string), content-type результатов обработки в скине по умолчанию. Если не указан, его значение считается равным «text/html».
6) любые другие параметры.

Все параметры указанные в **`skin.settings.xml`** будут доступны скриптам через метод Request.getSettings().

Скин типа PLAIN

Структура файлов такова: в **`$files/`** лежат файлы отдаваемые немедленно и без обработки на запрос к ним, в **`icons/`** лежат файлы иконок, в папке **`skin/`** лежат шаблоны скина, в корневой папке лежит файл **`skin.settings.xml`**, остальные файлы являются скриптами скина.

При прохождении запроса через данный скин выбирается файл скина соответствующий URL запроса, если этот файл является скриптом – он выполняется, в другом случае результат немедленно отдается клиенту. Если результат выполнения скрипта является FINAL – он немедленно отдается клиенту.

Если после всех этих операций мы имеем не FINAL ответ – то из шаблонов скина вызывается шаблон соответствующий параметру респонза «**`template`**» или коду ответа, если **`template`** не указан система производит выбор шаблона по коду ответа (200 – ок, 204 – пустой ответ, 401 – необходима авторизация, 403 – доступ запрещен, 404 – документ не найден и т.д.). Шаблоны скина, также, могут перебрасывать ответ другим шаблонам скина указывая параметр **`template`**.

Если относительный путь URL у пришедшего запроса был пуст – путем по умолчанию считается **`/index.htm`**, скин должен содержать скрипт с таким именем для ответа на такой запрос.

Доступ пользователя к папке **`skin/`** и к файлу **`skin.settings.xml`** запрещен.

Отдельно стоит обсудить папку **`icons/`**. В ней лежат файлы иконок, с именем, построенном по шаблону **`<key1>-<key2>-<keyX>.AAA.AAA`**, где **`AAA`** последовательность любых символов, кроме дефиса. При запросе к иконке система ищет подходящий файл итеративно отнимая старший **`<keyX>`** от имени файла. Таким образом, при правильной организации содержимого этого каталога система пытается подобрать иконку от более точно подходящей к более общей. Если в результате этого процесса иконка не найдена – система выдаст прозрачный спейсер.

Для скина данного типа определены следующие параметра **`skin.settings.xml`**:

1) type – (string) всегда PLAIN.
2) charset – (string) кодировка в которой сохранены файлы скина, являющиеся скриптами. Если этот параметр не указан используется кодировка операционной системы или кодировка явно переданная jvm.
3) renderer – (string) идентификатор языка, на котором написаны файлы, являющиеся скриптами. Или (map) с полем type содержащим идентификатор языка и полем suffix содержащим расширение файлов-шаблонов (включая точку, например: '.js').
4) generate – (string) определяет тип произведенного результата, отвечает за возможность кеширования, как на стороне пользователя, так и на сервере. Может

принимать значения: dynamic, private, build. От полностью динамического некешируемого результата, до построенной статики соответственно.

5) secure – (boolean) требуется ли безопасный интерфейс для работы с этим скином. Если этот параметр не указан, его значение считается false.
6) auth – (boolean) требуется ли обязательная авторизация средствами протокола. Если этот параметр не указан, его значение считается false.
7) любые другие параметры.

Все параметры указанные в **`skin.settings.xml`** будут доступны скриптам через метод Request.getSettings().

Скин типа EXACT

Все файлы лежащие в vfs скина отдаются немедленно и без обработки. Исколючение состовляет файл **`skin.settings.xml`**, лежащий в корневой папке, внешний доступ к нему запрещен.

Если относительный путь URL у пришедшего запроса был пуст – путем по умолчанию считается **`/index.htm`**, скин должен содержать файл с таким именем для ответа на такой запрос.

Для скина данного типа определены следующие параметра **`skin.settings.xml`**:

1) type – (string) всегда EXACT.
2) secure – (boolean) требуется ли безопасный интерфейс для работы с этим скином. Если этот параметр не указан, его значение считается false.
3) auth – (boolean) требуется ли обязательная авторизация средствами протокола. Если этот параметр не указан, его значение считается false.
4) любые другие параметры.

Все параметры указанные в **`skin.settings.xml`** будут доступны скриптам через метод Request.getSettings().

DOCTYPE

Я бы посоветовал указывать DOCTYPE так, чтобы он оказывался в первой строке результатирующего HTML. Нет, это не имеет отношения к ACM, зато имеет прямое отношение к сборке сайтов и скинам. Дело в том, что без этого элемента кода страницы браузеры пытаются воспроизводить свои баги и особенности времен «войны браузеров». Если вы уже не живете в 90ых годах двадцатого века, то наверно заинтересованы в том, чтобы страницы показывались максимально похоже в разных браузерах.

В двух словах – при указанном DOCTYPE браузеры хотябы пытаются соответствовать стандартам, что уже само по себе не плохо ☺

Выглядит эта строчка примерно так:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

(любая из двух)… Есть еще куча вариантов, но это уже не предмет для обсуждения в этом документе. Вы уже в курсе того, что надо искать, если действительно интересно.

Встроенные спейсеры

При верстке сайта часто используются спэйсеры и цветные точки, поэтому в системе предусмотрен встроенный источник таких изображений. По относительному url **`/__i/1.gif`** (два символа '_' подряд) можно получить прозрачную точку в формате gif, а по ссылке типа **`/__i/1-`**<номер цвета в 16-ричном формате>**`.gif`** можно получть цветную точку нужного цвета. Размерность полученного изображения будет составлять 1px*1px.

Свои изображения для этих целей использовать не рекомендуется по нескольким причинам:

1) необходимость выполнять лишнюю работу по размещению изображений на сервере;
2) «отдаваться» они будут медленнее, так как происходят проверки прав доступа и обращение к жесткому диску, которого не требуется при загрузке встроенного изображения;

Примеры использования:

<img src=/__i/1.gif width=1 height=1>

<img src=/__i/1-ff0000.gif width=100 height=1>

Встроенные флаги государств

Кроме встроенных спейсеров система предоставляет встроенные флаги государств мира. По относительному url вида /__f/16x16/<ISO3166-1 alpha-2 country code> система отдаст флаг соответствующего государства, вписанный в квадрат 16px*16px.

Примеры использования:

```
<img  src=/__f/16x16/gb width=16 height=16>
<img  src=/__i/16x16/<%= (language.countryCode || "--").toLowerCase() %>
      width=16
      height=16>
```

Правила форматирования / Хорошего тона

Если над сборкой и поддержкой сайта работает несколько человек, одновременно или последовательно, поверьте, у них будет предостаточно проблем поисках ответов на вопросы «что тут и зачем написано», «почему именно так», «каким же образом это всё работает» и т.п. Даже когда тот же самый единственный сборщик возвращается к сайту через пару месяцев занятия чем-либо другим, у него возникают те же проблемы… Что, собственно, свойственно не только сайтам собранным на ACM.CMS, но и любой другой системе, например тоже самое происходит с любой программой и её программистом ☺.

Дабы не усложнять, и без того не легкий, процесс понимания устройства каждого конкретного сайта, сервиса сайта и т.п. достаточно придерживаться некоторых простых правил. Эти правила можно назвать «правила хорошего тона». Они позволят не создавать «ребусы» на пустом месте. Итак:

1) Имена всех типов должны начинаться заглавной буквы, название должно соответствовать смыслу, если состоит из нескольких слов - каждое слово с заглавной буквы.

2) Имена всех методов, аргументов, полей и локальных переменных должны начинаться с прописной (маленькой) буквы, название должно соответствовать по смыслу, если состоит из нескольких слов - каждое слово (кроме первого) с заглавной буквы.

3) Имена всех констант должны быть написаны ЗАГЛАВНЫМИ буквами, название должно соответствовать смыслу, если состоит из нескольких слов - каждое слово отделяется подчеркиванием.

4) Не пишите несколько операций на одной строке – для каждой операции отдельная строка.

5) Если данный скриптовый язык поддерживает объявление переменных, обязательно объявляйте переменную явно в том блоке, в котором она будет использоваться.

6) Внутри скобок управляющих конструкций языка всегда пробел после открывающей скобки и перед закрывающей скобкой. Внутри скобок с аргументами функций и арифметикой дополнительный пробел не нужен. Операторы кроме отделять от скобок и имен пробелом, после запятой пробел:

```
f(34, 46543, "fdfs");
f2(22, ( 23, 453, 54, 46, 234 ));
if( a == 5 ){
	//
}

a = b;
a = b + c * (23 + d)
```

7) Все стандартные блоки всегда писать блоками - при этом открывающая скобка на той-же строке где суть блока, закрывающая на отдельной, else пишется "}else{" на отдельной строке, внутри блока всегда отступ:

```
if( true ){
	return c;
}

for( var entry in children ){
	...
}

if( a < 5 ){
	a++;
}else{
	a = 5;
	b--;
```

```
}
```

8) У всех методов должны быть описаны аргументы – должно быть сразу видно какие параметры и какого типа (если описание типов поддерживается в том языке, в котором производится описание аргументов) принимает данный метод.

9) Там, где нет явной причины использовать специальный скриптовый язык, используйте ACM.ECMA, как самый стандартный и общеизвестный из имеющихся языков.

Система управления пользователями

Группы

Системные группы

Часть групп автоматически добавляется системой. Также как и группы определенные администратом и сборщиками сайта эти группы могут быть настроены.

Идентификатор	Описание
def.guest	Любой пользователь является членом этой группы. Она может быть использована для указания прав для «любых» пользователей, включая тех пользователей, с которыми система не знакома.
def.registered	Данная группа автоматически содержит всех зарегистрированных пользователей. В эту группу не обязательно явно добавлять пользователей и, более того, удаленные из нее пользователи снова попадут в нее автоматически.
def.handmade	Данная группа предназначена для пользователей добавленных администратором системы через интерфейс управления. При добавлении такого пользователя в интерфейсе управления пользователь включается в эту группу.
def.supervisor	Члены этой группы являются администраторами, для них доступны все каталоги и команды виртуального домена.

«Пользователи почти»

В системе управления пользователями отдельно показана папка именуемая «Пользователи почти» - в ней отображаются пользователи зарегистрировавшиеся, но не разу не вошедшие в систему и не подтвердившие свою регистрацию. Такие пользователи не считаются полноценными пользователями, для них в настройках системы управления пользователями может быть отдельно настроен интервал автоматического удаления.

FAQ

Как при наследовании типа скрыть из формы одно из его полей?

Это делается также, как если бы вы хотели просто модифицировать поле, но таким образом, что единственное изменение – это приведение атрибута **type** к значению **hidden**.

Допустим у нас в родительском типе описан следующий fieldset:

```
<fieldset class="fieldset">
	<field id="keywords" class="string" type="text" title="keywords" max="64k"/>
</fieldset>
```

Чтобы скрыть поле **keywords** в унаследованном типе надо описать следующий fieldset:

```
<fieldset class="fieldset">
	<field id="keywords" type="hidden"/>
</fieldset>
```

Если алгоритмы этого типа предусматривают определенные ограничения на значение скрываемого нами поля и не могут работать нормально при отсутствии такового значения – используйте атрибут **default** для указания этого значения, например:

```
<fieldset class="fieldset">
	<field id="keywords" type="hidden" default="news acticle"/>
</fieldset>
```

В данном примере поле будет скрыто, а его значение будет равно «news article».

Как подключить внешнюю ява-библиотеку?

Для этого нужно положить jar-файл данной библиотеки в каталог **%ACM_PROTECTED%/axiom/** (при необходимости изменить имя файла в соответствии с требованиями описанными в разделе «Архитектура/Загрузка»). После перезапуска системы данная библиотека станет доступна скриптам. Для доступа к классам библиотеки необходимо использовать оператор **import**. Для создания экземпляров классов библиотеки использовать оператор **new** (предварительно сделав соответствующий **import**). Методы, поля и константы типов и объектов библиотеки будут доступны также, как и методы, поля и константы любых других объектов. Примеры использования **import** и **new** можно увидеть в разделе «Сборка/Язык JSCRIPT».

Как сгенерировать читаемый код с простой контрольной суммой и как потом проверять вводимые пользователем коды?

Для создания и проверки такого кода можно использовать методы Create.formattedRandom(format) и Default.hashCode(string). Допустим мы хотим видеть код, схожий с кодом кредитных карт (по виду) из 10тичных цифр для удобства ввода. Поскольку поскольку в нашем примере рассматривается случай с необходимостью контрольной суммы – чтобы это было оправдано, будем считать, что для каждого произведенного кода мы храним некое состояние на нашем сервере и хотим избежать постоянного поиска данных при попытках подбора данного кода.

Итак, скрипт для создания наших активационных кодов может выглядеть примерно так:

```
var code = Create.formattedRandom( "DDDD-DDDD-DDDD-DDDD" );
code = Math.abs( hashCode( code ) ) % 9000 + 1000 + '-' + code;
```

Мы получим код к которому можно привязать информацию. Этот код можно раздавать участникам акции. Скрипт для предварительной проверки легальности введенного пользовотелем сайта кода может выглядеть следующим образом:

```
var valid = code.length() > 5
      && Number( code.substring(0, 4) )
            == Math.abs( hashCode( code.substring(5) ) ) % 9000 + 1000;
```

Как привязать к пользователю сайта набор информации, без необходимости регистрации, например, «корзина»

Для этого надо использовать профили пользователя и автоматическую авторизацию. Автоматическая авторизация позволяет идентифицировать пользователя по cookies сохраненным в его браузере. В целом это выглядит как:

```
User.ensureAutomaticAuthorization();
var user = User.getUser();
var profile = user.getProfile("testdrive", true);
… // делаем различные операции с profile.
user.setProfile("testdrive", profile);
```

Что надо сделать, чтобы один файл с типом стал доступен для всех сайтов?

Файлы с описаниями типов сохраненные в каталоге **%ACM_PROTECTED%/resources/type/** доступны для всех виртуальных доменов работающих на данном экземпляре системы ACM.CMS, любой из этих типов работает именно так, как если бы его копия лежала в каталоге **types** с типами сайта. Такие типы называются общими типами. Эти типы имеют более низкий приоритет, чем типы указанные непосредственно в каталоге сайта (доменные типы), поэтому, если среди доменных типов явно определен тип с таким же именем – будет работать именно он.

Системные типы (типы предоставляющиеся вместе с системой) расположены в каталоге **%ACM_PUBLIC%/resources/type/**, туда можно посмотреть, чтобы ознакомится с системными типами и понять представление, о чем идет речь. Приоритет системных типов более низкий, чем приоритет общих типов – поэтому любой системный тип может быть переопределен при помощи общих типов или доменных типов.

Не стоит забывать, что каталог **%ACM_PUBLIC%** не предназначен для внесения изменений – не кладите свои типы в этот каталог и не изменяйте типы описанные в нем. Если требуется отключить один из системных типов – создайте одноименный файл нулевой длинны в каталоге для общих или доменных типов.

Как добавить дополнительное подключение к базе данных?

Для этого надо в config.xml добавить запись вида:

```
<pool id='pool_name' url='jdbc_url' user='login' password='*****'/>
```

Формат параметра jdbc_url определяется драйвером, при помощи которого будет осуществлено подключение. Для регистрации драйвера в системе, требуется положить jar файл с классами драйвера в каталог %ACM_PROTECTED%/axiom/ и прописать его главный класс (он должен быть указан в документации к драйверу) в файле %ACM_PROTECTED%/conf/initialize.xml

Если требуется указание дополнительных параметров для конкретного драйвера, эти параметры можно указывать добавляя соответствующие атрибуты к данной записи.

Как подключить к системе хранилище другого сайта?

Для этого надо в config.xml добавить запись вида:

```
<plugin
      id='name'
      class='ACMMOD:STORAGE3'
      api='api_name'
      connection='pool_name'
      client='boolean_value'
      scheduling='boolean_value'
      />
```

Параметр **id** определяет имя во внутренней иерархии (соответственно и в интерфейсе управления) в котором подключится хранилище. Значение по умолчанию для хранилища указанного в примере 's3', при котором корень хранилища будет располагаться по пути **/s3/**. Этот путь будет использоваться для построения **getLocationControl()**, вычисления точек доступа для обращения к документам данного хранилища и привязки ACL системы безопасности.

Параметр **api** определяет имя свойства глобального скриптового объекта (коренного контекста) через которое скрипты смогут доступаться к API предоставляемому данным хранилищем. Значение по умолчанию для хранилища указанного в примере 's3'. Также, первое зарегистрированное в системе хранилище будет предоставлять свой API под именем 'Storage'.

Параметр **connection** определяет имя соединения, через которое производится подключение хранилища типа STORAGE3 к базе данных. Значение по умолчанию '**default**'.

Параметр **client** позволяет запретить выполнение задач связанных с поддержкой актуальности данных хранилища, таких как индексация, синхронизация версий, обновление версий форматов внутренних данных хранилища, планировщика, сборка мусора и т.п., что может быть полезно при подключении к хранилищу от другого сайта. Значение по умолчанию '**false**'.

Параметр **scheduling** позволяет запретить выполнение заданий планировщика, что может быть полезно при настройке серверов кластера (работающих совместно), если некоторые из серверов кластера должны быть исключены из списка выполняющих задачи планировщика (например рассылка подписок (спама) или расчеты какой-либо статистики и т.п.). Значение по умолчанию '**true**', при включенном параметре **client** задачи планировщика не будут выполняться в любом случае.

Чем же всётаки отличаются методы defered от beffered?

Оба типа методов выполняются асинхронно и не могут возвращать результат в вызывающий скрипт. Однако, метод типа **defered** будет выполнен тоже число раз, сколько раз он был вызван, тогда как метод типа **buffered** копит параметрами и передает наборы параметров в виде массива за один вызов. Дословно первый тип команд переводится на русский как «отложенные», а второй тип как «буферизированные».

Как правильно записать значение поля binary в файл

Не надо делать циклов и создавать буферов – в системе есть метод, который сделает это эффективно. Сигнатура метода:

```
ru.myx.ae3.binary.Transfer.toStream( inputStream, outputStream, true );
```

В качестве первого параметра надо передать `binary.getBinary().toInputStream()`, а в качестве второго параметра объект типа `FileOutputStream`. Третий параметр определяет, следует ли закрывать открытые потоки – врядли вас когданибудь пригодится значение `false`.

Как правильно прочитать значение типа binary из файла

Сигнатура метода:

```
BaseMessage<?>  ru.myx.ae3.flow.Flow.file(
            String owner,
            String title,
            File file,
            Map attributes);
```

В качестве первого параметра надо передать какоенить имя – единственное назначение которого – возможность разобраться какой кусок кода это объект породил. Например если сообщение порождено парзером протокола HTTP, то в качестве аргумента используется «PROTO-HTTP», а в качестве второго параметра тоже какаянить пояснительная простая строка. В качестве атрибутов можно передавать NULL – этот мап определяет дополнительные заголовки, в любом случае система попытается поставить все возможные заголовки на базе имени файла и самого файла, например: его content type или его длинна.

Если я в поле объекта Entry кладу другой Entry, то сохраняется только ссылка на него, а не вся толпа его полей?

Да. При этом сохраняется ссылка на объект хранилища, содержащая в себе информацию о `guid` этого объекта. При последующем доступе к данным эта ссылка будет загружать объект хранилища эквивалентно вызову `Storage.getByGuid( guid )`.

Как защититься от SQL-Injection?

Что такое SQL injection отлично написано тут: http://en.wikipedia.org/wiki/SQL_injection

Несмотря на то, что любой полученный системой параметр проходит через защиту от «SQL injection» при написании скрипта на сайте можно легко допустить оплошность: если при использовании прямой работы с базой данных при построении запросов используются полученные со стороны пользователя параметры – ваш скрипт либо подвержен SQL injection, либо осуществляет следующее:

1) все числовые параметры явно преобразовываются к числу
   (пример: … `age = " + Number( request.age )`)
2) все строковые параметры форматируются в специальный вид в соответствии с правилами оформления строковых параметров в SQL
   (пример: … `name = '" + Format.sqlStringFragment( request.name ) + "'"`)
3) все json- параметры форматируются в специальный вид в соответствии с правилами оформления строковых параметров в ECMA-262
   (пример: … `name = " + Format.jsString( request.name )`)
4) все остальные виды параметров, либо не используются вообще, либо обрабатываются в том-же духе ☺

Как в скрипте определить тип объекта?

Обычно, в скриптовых языках отсутствует строгая типизация данных: объект представляет собой коллекцию полей и методов, набор которых не обязательно определяется их типом. В первую очередь, стоит задуматься «действительно ли требуется определять тип объекта?» - в большинстве случаев достаточно проверить наличие у объекта нужного метода или набора методов, например – чтобы вызвать у объекта хранящегося в переменной **obj** метод **getTitle** и только в том случае если такой метод у этого объекта есть, можно написать:

```
obj.getTitle && obj.getTitle()
```

Если всетаки требуется именно определить тип объекта, то в самом языке ECMA есть оператор **typeof**, который возвращает строковое значение соответствующее любому примитивному типу или значение «object» для остальных объектов, например:

```
typeof obj.title == «function» ? obj.title() : obj.title
```

Этот оператор может возвращать значения: «**undefined**», «**null**», «**boolean**», «**number**», «**string**», «**function**», «**object**». Он может быть полезен, когда требуется проверить является ли объект одним из примитивных типов, например: строкой.

Ну и наконец, в том случае, когда требуется определить является ли объект хранилища экземпляром одного из типов определенных в виртуальном домене системы ACM.CMS, можно воспользоваться методом **isInstance** присутствующем в каждом типе, определенном для данного сайта. Например, чтобы проверить является ли объект хранящийся в переменной obj экземпляром типа **Article**, следует использовать следующую конструкцию:

```
Article.isInstance(obj)
```

Метод **isInstance** учитывает наследование типов, допустим, в системе определен тип **Document** и тип **Article** унаследован от типа **Document**, выражение **Document.isInstance(obj)** вернет истину, если выражение **Article.isInstance(obj)** возвращает истину.

Ошибки

Возвращаемая из обработчика запроса картинка (любой файл) вместо того, чтобы отдаваться пользователю попадает в скиннер и приводит к появлению пустой страницы с дизайном сайта

В типах может быть определен обработчик ответа (response / filter в типах описанных в файлах ***.scheme**). Через них проходят все произведенные ниже по иерархии прохождения запроса ответы пользователю. Когда обработчик запроса отвечает изображением (или любым друшим бинарным объектом) такой ответ проходит через тот же набор фильтров как и любой другой ответ. Скорее всего в обработчике ответа (фильтре) отсутствует проверка на тип ответа и он делаеют и бинарным ответом тоже самое, что он бы сделал с лэйаутом. В первую строчку обработчика требуется добавить строку: `if( !HashValid( content ) ) return content;` При помощи этой строки фильтр перестанет обрабатывать объекты не являющиеся объектами **Map** и станет отдавать их без изменений.

Ошибка об отсутствии метода API описанного в документации, сообщение содержит текст вида No such method: cannot access method (niceNameNotation), class=ru.myx.ae1.types.TypeImplNew!

У вас имеется тип с именем, совпадающим с именем API. Ваши типы имеют больший приоритет, чем системные API, т.к. они определены ближе к выполняемому коду. Требуется переименовать или удалить типа, закрывающий доступ к требуемому API.

Точки доступа

Надо начинать сборку сайта, как настроить точку доступа, не имея свободного имени DNS

Системного администратора нет на месте, как быть с доменом необходимым для настройки точки доступа

На каждом компьютере можно «настроить статический» - обычно это файл, в котором можно описывать соответствие имени хоста его адресу. Разумеется, этот файл влияет только на тот компьютер, на котором он расположен.

Если данная точка доступа нужна для сборки сайта – можно настроить хост в этом файле и начинать сборку. Если впоследствии эта точка доступа будет доступна другим людям или людям из внешнего мира – следует попросить администратора прописать это имя в DNS.

Расположение файла и его формат зависит от операционной системы, например: в 32ух битном WindowsNT этот файл называется **`%WINDOWS%/system32/drivers/etc/hosts`** – в нем и написаны примеры как им пользоваться.

Узнать адрес сервера (ну, вдруг, кто-нибудь не знает) – выполните команду **`ping <имя известной работающей точки доступа на том же сервере>`** - в выводе этой команды будет виден IP-адрес.

Если перед сервером установлен apache (ну или любая другая программа непосредственно принимающая запросы пользователей), скорее всего, он не пустит вас к вашей точке доступа, пока администратор не пропишет этот хост в настройках apache. В этом случае можно пойти к ACM.CMS напрямую, номер порта придется спросить у администратора.

Администрирование

В данном разделе рассмотрены вопросы связанные с администрированием системы.

Системные требования

1) **установленная копия Java SE 5.0 или новее**

система использует технологии Java и, соответственно, для её работы требуется работающая копия Java. Минимально требуется версия 5.0 – все возможности которой задействованы в системе. Отдельно стоит отметить, что системе не требуется Java EE (Enterprise Edition) и она не требует для своей работы установленного servlet container.

2) **возможность процессу системы одновременного открытия не менее 4096 файлов**

система использует локальную файловую систему для хранения кешей и приватных данных, в связи с чем чрезвычайно требовательна к возможностям файловой системы и к возможным искуственным ограничениям, также следует обратить внимание на то, что в некоторых операционных системах ограничения на количество одновременно открытых файлов включают в себя сокеты установленных соединений с пользователями.

3) **возможность процессу, под которым работает система запуска не менее 512 нитей выполнения**

система использует конвейерный подход к обработке запросов, число требуемых для работы ниток не зависит от числа обрабатываемых в данный момент запросов, другими словами системе действительно нужны все нити выполнения, которые она пытает выделить под обработчики конвейеров и другие задачи поддержки системных сервисов.

4) **возможность неограниченного по времени выполнения нитей системы**

так как система использует конвейерный подход – нити обработки заданий каждого конвейера живут значительно дольше времени, необходимого для выполнения запроса, а большинство таких нитей живут (находятся в состоянии ожидания) и в периоды полного отсутствия запросов как таковых.

5) **не менее 384 (256 на 32-битной системе) мегабайт памяти выделенной для процесса Java**

множество технологий использованных в системе для ускорения процессов обработки запросов и уменьшения задержек приводят к тому, что система достаточно требовательна к количеству оперативной памяти.

6) **наличие на 100% JDBC совместимой базы данных и драйвера JDBC для нее**

так как система не завязана на конкретную базу данных, она пользуется стандартом SQL-92 и JDBC для работы с базами данных, в комплект входит драйвер протокола TDS для подключения к MSSQL (стандартный драйвер не совместим со стандартом JDBC), драйвер-фильтр ora-fix для подключения к ORACLE 9i и выше с использованием данного фильтра поверх соединения по стандартному «тонкому» драйверу (который, сам по себе, также не совместим со стандартом JDBC), также проверено, что система прекрасно работает с базами данных PostgreSQL и MySQL через их собственные стандартные драйвера.

Системные требования для production сервера

Ёмкости конвейеров, объемы и количества буферов, число обработчиков дискового кеша и его устройство, таймауты контроллирующих работоспособность системы агентов и т.п. рассчитаны в зависимости от мощности компьютера на котором установлена система. Система пытается использовать все доступные ресурсы для эффективной реакции на запросы Ваших пользователей и не предназначена для установки в укромный уголок свободного места на “пыхтящем” и без нее компьютере. В общем случае, для успешной работы системы требуется

отдельный, выделенный, под нее компьютер или несколько таких компьютеров. Разумеется, с ней могут соседствовать различные задачи, но система рассчитывает, что она на сервере главная, и эти другие задачи (как и их владельцы) должны с этим мириться.

Apache и другие

Система имеет собственный, высокопроизводительный веб-сервер, поддерживающий протоколы HTTP и HTTPS. Таким образом, она не требует ни наличия servlet container, ни сервера apache, ни reverse-proxy squid.

Разумеется, есть ситуации, когда наличие таких дополнительных серверов как apache имеет смысл, например: они могут быть установлены и настроены для разделения одного IP-адреса между различными системами/сервисами установленными на компьютере, однако, являются абсолютно нецелесообразными при наличии выделенного для ACM.CMS сервера, лишь внося задержку в обработку запроса и увеличивая нагрузку, порождая два дополнительных открытых сокета на сервере для каждого соединения и т.п.

Архитектура сетевых интерфейсов ACM.CMS устроена так, что система не боится «медленных» клиентов, поэтому также не имеет смысла установка reverse-proxy с целью быстрого забора ответов несовершенного web-приложения и медленной их отдачи «медленным» клиентам. Установка такого программного обеспечения в общем случае приведет к тому, что в 3 раза возрастет число занятых клиентами сокетов, на диске и в оперативной памяти будут лежать лишние копии ответов, а сама система ACM.CMS не сможет осуществлять шейпинг траффика, применять приоритеты различным ответам сервера, обнаруживать DoS аттаки и противодействовать им.

«Тонкое место»

Отдельно хочется обратить внимание на то, что в web системах массовой обработки особенно тонким местом является производительность дисковой системы, не хочется подробно объяснять почему (искренне надеюсь, что все читающие и так в курсе), но большинство причин сводятся к такому упрощению: когда занят процессор – он хоть что-то делает, притом что-то касающееся задач возложенных на сервер, а когда речь идет о дисках, система просто занята ожиданием, и, поверьте, ожиданием долгим и бесполезным ввиду того, что скорости доступа к данным на дисках на порядки медленнее всего остального, а запросы обычно настолько не конкурентны (в том числе и изза быстрого их выполнения), что ничем полезным заниматься в моменты ожидания диска не представляется возможным. Даже при свободных (в смысле нагрузки) дисках, на практике, быстрее считать zip-архив и распаковать его, чем считывать распакованные данные с диска.

RAID-массив

Ну и для тех, кто совсем плавает в теме настройки web-сервера, хочется также обсудить тему RAID-массивов. Данные отдельной копии ACM.CMS содержат очень малую и редко меняющуюся часть данных, требующих восстановления после их потери, однако RAID-массив предназначен не только для обеспечения целостности данных, они имеют два других важных для production-сервера аспекта:

1) помогают повысить скорость считывания (**`RAID0`** в 2 раза, **`RAID1`** в 2 раза, **`RAID10`** в 4 раза, **`RAID5`** в n/(n-1) раз, где n число дисков в массиве);

2) а также, и самое важное для production-сервера, при отказе диска система будет оставаться online и продолжать обслуживать клиентов (за исключением **`RAID0`**).

При наличии «правильного» RAID-контроллера будет иметься возможность заменить сломанный диск и восстановить нормальное функционирование дисковой системы и без

секунды простоя. Однако даже с самым отстойным RAID-контроллером Вы сможете самостоятельно выбрать время простоя для замены диска.

Относительно RAID0 – данный вид RAID-массива не обеспечивает целостности данных и не позволяет серверу продолжить работу при отказе диска. Однако он представляет определенный интерес т.к. не поглощает дискового пространства под зеркальные копии и контрольные суммы. Использовать его можно только в том случае, если надежность системы в целом решается другими методами и отказ сервера является «штатным» режимом – например при использовании кластера с аппаратным load-balancing с возможностью определения отказавшего сервера и тремя или более серверами отведенными под ACM.CMS работающие в общем кластере.

Возможные конфигурации

Entry level 1	Один выделенный сервер, на котором установлена СУБД, ACM.CMS, почтовый сервер, php и т.п.	Единственная конфигурация, при которой имеет смысл установка сервера apache перед ACM.CMS и то, только в том случае, если имеется нехватка IP адресов, для работы системы ACM.CMS на отдельном, выделенном IP адресе. Даже эта конфигурация, в состоянии отвечать за хостинг массы сайтов при правильной настройки сервера и эффективно собранных сайтах.
Entry level 2	Два выделенных сервера, на одном из них СУБД на другом ACM.CMS	Данная конфигурация не значительно отличается от предыдущей, главными отличиями является то, что СУБД и ACM.CMS используют физически разные диски, являющиемя главным ограничивающим по скорости факторам, а также СУБД имеет свой собственный полный набор ресурсов и может быть использована для решения других задач, другими системами, установленными на других машинах (так как ACM.CMS рассчитывая на возможную работу в кластере не претендует на полное поглощение мощностей СУБД одной копией системы). Справедливости ради, можно заметить, что и в конфигурации entry level можно изобразить подобную схему разнеся СУБД и ACM.CMS на разные дисковые массивы и добавив память.
Deployment	Отдельная СУБД и два сервера ACM.CMS	Рассчитывается, что один из серверов готов подменить второй при необходимости. Переключение происходит вручную. Неактивный сервер используется для проверки новой версии сайтов, после проверки, сервера меняются ролями с возможностью быстрого отката обратно.
High Grade	Отдельная СУБД и несколько серверов ACM.CMS работающие совместно.	В этом варианте ACM.CMS может работать в составе кластера, она рассчитана на такой вариант работы, способна распределять задачи по поддержке и обновлению хранилища между своими копиями, ограничивать число соединений к базе данных, рассылать информацию о своевременном сбросе кеша и запрашивать друг у друга некоторые данные вместо хождения в СУБД. Требуется использование дополнительных средств для распределения нагрузки.

Каталоги

Три основных каталога определены для каждого работающего экземпляра ACM.CMS: public, protected и private.

Каталог public

Public – каталог в котором находятся дистрибутив системы: исполняемые файлы и ресурсы дистрибутива состоящие из библиотек, модулей, встроенных скинов и т.п. Этот каталог не предназначен для изменения пользователями системы. Далее в данном описании этот каталог будет именоваться **`%ACM_PUBLIC%`**. Для нормальной работы системе требуются права на чтение из этого каталога. По умолчанию этим каталогом считается каталог из которого был произведен запуск системы. Для явного указания данного каталога используется переменная окружения JVM **`ru.myx.ae3.properties.path.public`**.

Каталог protected

Protected – каталог в котором хранятся общие файлы сайтов различных экземпляров системы: дополнительные библиотеки, настройки модулей, шаблоны сайтов, скины. Этот каталог может быть общим для нескольких экземпляров системы, работающих на одном компьютере или может свободно копироваться или синхронизироваться между компьютерами (полностью или частично в зависимости от настроек), на которых работают экземпляры системы составляющие кластер. Далее в данном описании этот каталог будет именоваться **`%ACM_PROTECTED%`**. Для нормальной работы системе требуются полные права на этот каталог. Для явного указания данного каталога используется переменная окружения JVM **`ru.myx.ae3.properties.path.protected`**.

Каталог private

Private – каталог для личных данных экземпляра системы, временных файлов, логов, файлового кеша и внутренней базы данных. Этот каталог нельзя копировать между экземплярами системы или делать общим. Далее в данном описании этот каталог будет именоваться **`%ACM_PRIVATE%`**. Для нормальной работы системе требуются эксклюзивные права на этот каталог. Для явного указания данного каталога используется переменная окружения JVM **`ru.myx.ae3.properties.path.private`**.

Параметры запуска

Параметр *(переменная окружения JVM, опция -D)*	**Описание**
ru.myx.ae3.properties.path.public	Путь к каталогу **%ACM_PUBLIC%**. По умолчанию – текущий каталог в момент запуска системы.
ru.myx.ae3.properties.path.protected	Путь к каталогу **%ACM_PROTECTED%**. По умолчанию – <домашний каталог пользователя>/acm.cm5/protected/.
ru.myx.ae3.properties.path.private	Путь к каталогу **%ACM_PRIVATE%**. По умолчанию – <домашний каталог пользователя>/acm.cm5/private/.
ru.myx.ae3.properties.path.cache	Путь к каталогу с файлими кеша. По умолчанию – **%ACM_PRIVATE%/cache**.
ru.myx.ae3.properties.path.logs	Путь к каталогу с лог-файлами. По умолчанию – **%ACM_PRIVATE%/logs**.
ru.myx.ae3.properties.path.temp	Путь к каталогу с временными файлами. По умолчанию – **%ACM_PRIVATE%/temp**.
ru.myx.ae3.properties.cluster.identity	Идентификатор кластера. По умолчанию единожды формируется системой и сохраняется в **%ACM_PROTECTED%/boot.properties**. Не указывайте этот параметр без явной необходимости.
ru.myx.ae3.properties.instance.identity	Идентификатор инстанса системы. По умолчанию единожды формируется системой и сохраняется в **%ACM_PRIVATE%/boot.properties**. Не указывайте этот параметр без явной необходимости.
ru.myx.ae3.properties.session.identity	Идентификатор сессии. По умолчанию формируется системой при каждом запуске. Не указывайте этот параметр без явной необходимости.
ru.myx.ae3.properties.ip.wildcard.host	Указывает адрес, который будет использоваться для запуска интерфейсов с хостом указанным как «*». По умолчанию такие интерфейсы подключаются ко всем доступным адресам.
ru.myx.ae3.properties.ip.shift.port	Указывает целочисленное положительное значение, которое будет прибавлено к номерам портов при инициализации интерфейсов. По умолчанию «0».
ru.myx.ae3.properties.log.level	Определяет уровень подробности логов. Принимает значения из набора: **MINIMAL**, **NORMAL**, **DEBUG**, **DEVEL**. По умолчанию – **NORMAL**.
ru.myx.ae3.properties.report.mailto	Указывает адрес электронной почты для получения системной информации и ошибок. Можно указать несколько адресов.

ru.myx.ae3.properties.hostname	Указывает имя хоста для данного экземпляра системы. Влияет на многое, например: для каждого виртуального домена в автоматически регисстрируется домен, аналогичный записи в файле servers.xml: `aliases=«*.<domain_id>.<hostname>».`
ru.myx.ae1.path.web	Определяет умолчательный коренной каталог для серверов AE1/RT3. Значение по умолчанию **`%ACM_PROTECTED%/web`**

Интерфейсы

Взаимодействие системы ACM.CMS с окружающим миром осуществляется при помощи интерфейсов. Любая активность в системе производится в контексте «текущего запроса», а любому запросу соответствует интерфейс его породивший. Настройка интерфейсов производится в файле %ACM_PROTECTED%/conf/interfaces.xml, примеры настройки интерфейсов приведены в файле %ACM_PROTECTED%/conf/interfaces.xml.sample.

Предопределенные интерфейсы

Определенный набор интерфейсов предопределен, эти интерфейсы запускаются независимо от настроек в файле %ACM_PROTECTED%/conf/interfaces.xml. Эти интерфейсы необходимы для начальной или аварийной настройки системы, для взаимодействия системы со скриптами сервера или для внутреннего функционирования системы.

Список предопределенных интерфейсов:

Имя	**Тип интерфейса**	**Описание**
SSH-ADMIN	SSH	Доступен по TCP: <any-address>:14022
TELNET-ADMIN	TELNET	Доступен по TCP: localhost:14023
HTTP-ADMIN	HTTP	Доступен по TCP: localhost:14080
HTTPS-ADMIN	HTTPS	Доступен по TCP: <any-address>:14433
FILECONTROL	FILECONTROL	Управление файлами. Ограниченные возможности: перезапуск, ожидание. Предназначено для скриптов. Доступен через файловую систему в каталоге: %ACM_PRIVATE%/control/
NULL	NULL	Внутренний интерфейс системы, используется для таких активностей как запуск системы, остановка системы, задач мониторинга состоянии системы и т.п.

Интерфейс FILECONTROL

Назначение:

Управление важными системными функциями через семафорные файлы на сервере.

Метод:

Создание и удаление файлов с определенными именами и содержанием в специально для этого отведенном каталоге.

Возможности:

Простейшее взаимодействие со скриптами операционной системы и прочими несовместимыми с AE2 программными средствами.

Пример:

Временная остановка сервера при выполнении запланированной ранее автоматической проверки целостности базы данных.

Использование:

Система ожидает команды в каталоге %ACM_PRIVATE%/control.

В нормальном состоянии в этом каталоге расположены файлы: execute, restart и suspend.ready. В содержании этих файлов написана краткая инструкция о том как их использовать.

Интерфейс поддерживает следующие команды:

1) **restart** – перезапуск системы. Для выполнения этой команды требуется удалить файл restart. После этого, если система запущена и функционирует, будет произведен перезапуск системы, и, в случае успешного перезапуска файл restart будет автоматически восстановлен. Текущая дата файла restart соответствует последней дате проверки наличия этого файла модулем filecontrol. В нормальном состоянии такая проверка происходит один раз в 5-10 секунд.

2) **suspend** – приостановка работы системы. Для выполнения этой команды требуется создать файл suspend. Для выхода из режима приостановки требуется удалить этот файл. Текущая дата файла suspend.ready соответствует последней дате проверки наличия файла suspend модулем filecontrol. В нормальном состоянии такая проверка происходит один раз в 5-10 секунд. О том что система запущена и приостановлена свидетельствует появление файла suspend.waiting, этот файл автоматически удаляется системой при выходе из режима приостановки.

3) **execute** – выполнение команд операционной оболочки от имени пользователя под которым работает система. Для выполнения требуется сохранить команду в файл execute.command и удалить файл execute. Система произведет попытку выполнения указанной команды потоки вывода будут перенаправлены в файлы execute.out и execute.err, а файл execute будет восстановлен системой. Текущая дата файла execute соответствует последней дате проверки наличия этого файла модулем filecontrol. В нормальном состоянии такая проверка происходит один раз в 5-10 секунд.

Сертификат для SSL (например для интерфейса HTTPS)

В настройках интерфейса работающего по SSL/TLS параметр domain отвечает за CN-записи сертификата. Пример настройки:

```
<filter>
      <factory>TLS</factory>
      <domain>mydomain.net</domain>
      <domain>*.mydomain.net</domain>
</filter>
```

По умолчанию система автоматически формирует самовыданный сертификат (иначе невозможна установка SSL/TLS соединения). Для большинства случаев такого самовыданного сертификата будет достаточно – например для управления содержанием сайта, для работы разработчика сайта с файлами по WebDAV, для работы сотрудников с backoffice интерфейсом сайта и т.п., более того – при правильном указании CN имен, такой сертификат можно добавить в доверенные сертификаты обозревателя интернет и не сталкиваться с дастающими окошками с предупреждениями.

Для того случая, если имеется желание установить другой сертификат, например для использования его на публичном сервисе для защиты передаваемой пользователями

информации: на каждый SSL/TLS интерфейс существует пара файлов certreq* и keystore* в каталоге %ACM_PROTECTED%/settings/flt-tls/. Файл certreq* содержит запрос на выдачу сертификата для указанных имен CN – он требуется для выдачи и передачи сертификата по открытым/незащищенным сетям. После получения сертификата его требуется установить в keystore – для этого используется стандартная утилита входящая в состав java – keytool.

Виртуальные домены

Виртуальный домен в системе ACM.CMS определяется одним общим набором типов, скинов, групп пользователей, самих пользователей, публичных точек доступа и прав доступа. Каждому виртуальному домену отведена, как минимум, одна база данных в СУБД. Настройка виртуальных доменов производится в файле %ACM_PROTECTED%/conf/servers.xml и включает в себя имя/идентификатор виртуального домена, тип виртуального домена, список позволенных DNS-зон и доменов для публичных точек доступа, параметры подключения к базе данных. Примеры настройки виртуальных доменов приведены в файле %ACM_PROTECTED%/conf/servers.xml.sample.

Настройка DNS

Пользователи каждого виртуального домена (одаренные соответствующими правами) могут назначать публичные точки доступа для любых DNS имен, и, если эти имена входят в список разрешенных DNS-доменов для данного виртаульного домена – система в состоянии сразу отвечать на запросы пользователей к этим публичным точкам доступа.

Однако, есть одно «но» - для того чтобы запросы пользователей дошли до системы – требуется направить эти запросы на конкретный сервер при помощи DNS.

Разумеется, у каждого администратора могут быть свои собственные правила и соображения по поводу настройки доменов, находящихся под его контролем. Ну, всетаки, если предположить, что никаких особых соображений нет (например: конкретный домен приобретен специально для сайта), рекомндуется следующий подход: прописать корень зоны и wildcard для зоны на адрес/адреса сервера/серверов ACM.CMS, отдельно прописать все специальные хосты с их адресами, при наличии таковых.

Если для доступа к сайтам используется промежуточное звено (например apache) не забудьте настроить его в том-же духе, что и DNS.

Пример:

Допустим имеется зона supersite.net и 5 серверов с адресами 88.99.77.51 .. 88.99.77.55, первые 4 сервера используются для «определенных» нужд, пятый сервер – веб-сервер с системой ACM.CMS.

Тогда зона может быть настроена следующим образом, показаны только записи типа «A» (address):

Хост	Адрес	Комментарий
	88.99.77.55	Корень зоны, указывает на сервер #5, DNS имя: supersite.net
*	88.99.77.55	Все неопределенные имена в зоне, указывают на сервер #5, любое валидное, не определенное явно имя, например: www.supersite.net
mars	88.99.77.51	Первый сервер, DNS имя: mars.supersite.net
earth	88.99.77.52	Второй сервер, DNS имя: earth.supersite.net
saturn	88.99.77.53	Третий сервер, DNS имя: saturn.supersite.net
uran	88.99.77.54	Четвертый сервер, DNS имя: uran.supersite.net
pluto	88.99.77.55	Пятый сервер, DNS имя: pluto.supersite.net

Коды выхода

Система использует следующие коды возврата при завершении работы процесса.

Код	Причина
0	Штатный перезапуск системы
-1	Проблема запуска – невозможно получить эксклюзивный лок на private данные.
-2	Проблема запуска – система не пришла в заданное состояние за отведенное время.
-3	Аварийный: Система не смогла осуществить штатный выход по коду -2 за 20 секунд.
-13	Watchdog системы обнаружил критический рост числа активных ниток
-14	Аварийный: Система не смогла осуществить штатный выход по коду -13 за 20 секунд.
-15	Watchdog системы обнаружил уверенное отсутствие свободной памяти
-16	Аварийный: Система не смогла осуществить штатный выход по коду -15 за 20 секунд.
-18	OutOfMemory при запуске дополнительного обработчика при переполненной очереди заданий.
-19	Аварийный: Система не смогла осуществить штатный выход по коду -18 за 20 секунд.
-21	Стартер задач модуля NIO наткнулся на OutOfMemory
-22	Аварийный: Система не смогла осуществить штатный выход по коду -21 за 20 секунд.
-23	Конвейер приема соединений NIO словил 500 ошибок подряд.
-24	Аварийный: Система не смогла осуществить штатный выход по коду -23 за 20 секунд.
-25	Основной мультиплексор NIO словил OutOfMemory
-26	Аварийный: Система не смогла осуществить штатный выход по коду -25 за 20 секунд.
-27	Генератор preview словил зависание внешней программы или дополнительной библиотеки
-28	Аварийный: Система не смогла осуществить штатный выход по коду -27 за 20 секунд.
-29	В течении 2ух минут пул запросов отвечал только кодом BUSY (503)
-30	Аварийный: Система не смогла осуществить штатный выход по коду -29 за 20 секунд.
-31	Watchdog системы логгинга решил, что в системе логгинга deadlock и произвел выход jvm.
-32	Аварийный: Система не смогла осуществить штатный выход по коду -31 за 20 секунд.
-102	OutOfMemory при проверке состояния обработчиков заданий и запуске idle заданий
-103	Аварийный: Система не смогла осуществить штатный выход по коду -102 за 20 секунд.
-104	10 подряд неуспешных проверок состояния обработчиков
-105	Аварийный: Система не смогла осуществить штатный выход по коду -104 за 20 секунд.

Установка

Для установки или обновления системы при помощи CVS используйте следующие параметры:

Repository root	:pserver:guest@cvs.myx.ru:2401/var/share
Repository path	export/sys-release для стабильных версий export/sys-current для текущих версий
Tag	HEAD
Password	Guest

Таким образом, вы получите каталог **`%ACM_PUBLIC%`**, содержащий текущую версию системы.

Вызов cvs из командной строки должен выглядеть примерно так:

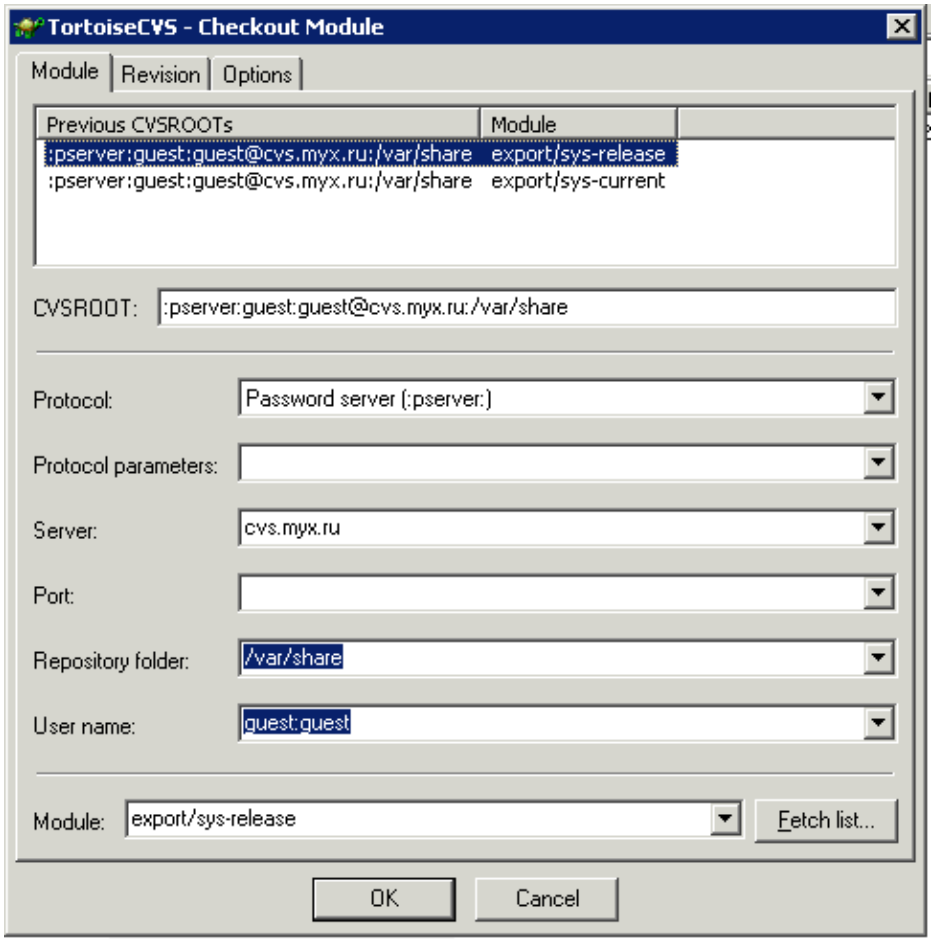
```
cvs -d :pserver:guest:guest@cvs.myx.ru:/var/share -f -z 6 co export/sys-current
```

Скачивает в текущий каталог ветку sys-current

```
cvs -d :pserver:guest:guest@cvs.myx.ru:/var/share -f -z 6 co export/sys-release
```

Скачивает в текущий каталог ветку sys-release

Настройки популярного клиента TortoiseCVS для Windows выглядят слудующим образом:



При разработке и в реальном применении система проверена на устойчивую работу в конфигурациях: FreeBSD / PosgreSQL, Windows 2003 / MSSQL, Linux / MySQL, Solaris / Oracle, Linux / Oracle.

Производительность системы обслуживающих сложные сайты на протяженни длительного времени убывает в соответствии с порядком следования в списке. При этом конфигурация Windows 2003 / MSSQL незначительно уступает FreeBSD / PosgreSQL, однако поддержка первой для начинающих пользователей ощутимо проще. А аутсайдерство конфигураций с СУБД Oracle обусловилось чрезвычайно сложной и комплексной поддержкой СУБД, производительность которой даже при постоянном внимании непрерывно падает день ото дня, пока не пригласишь специалиста, час работы которого превышает стоимость серверной версии операционной системы Windows, при этом такие специалисты требуются не реже, чем раз в неделю.

Также, система будет устойчиво работать в любой операционной системе, для которой имеется стабильная версия JavaSE и с любой базой данных, которая поддерживает SQL-92 и для которой имеются JDBC-совместимые драйвера.

ACM.BSD

Вы можете установить систему бесконечным количеством вариантов на практически любой операционной системе и с использованием практически любой базы, всё это может варьироваться в зависимости от ваших предпочтений, существующей инфраструктуры или требований заказчика, однако, если вы начинаете с нуля и не обременены предрассудками ACM.BSD – решение для вас.

В этом решении в качестве операционной системы выбрана система FreeBSD, в качестве СУБД выбран PosgreSQL, для управления сервером используются простые команды acmbsd позволяющие легко делать все необходимые операции, а для управления дополнительным ПО на сервере используется Webmin.

Основными преимуществами данной конфигурации является её исключительная производительность в сочетании с использованием полностью бесплатного и свободного программного обеспечения.

Поскольку ACM.BSD это попытка сделать полное решение, автоматически производится установка вспомогательных утилит, явы, базы данных, настройка инфраструктуры сервера.

Для установки acmbsd на машину с минимальной установкой freebsd нужно выполнить команды описанные на странице http://acmcms.ru/acmbsd/install/

Версия системы

В каталоге **%ACM_PUBLIC%/version/** расположен файл **version** содержащий подробную версию системы.

Ввиду того, что при обновлении через CVS используется тот-же репозиторий, в который происходит выкладывание всех версий системы, включая, нестабильные версии, релизы и важные обновления релизов – понимание формата версии системы просто необходимо. Если вы используете другой способ обновления системы (не CVS) – вы можете пропустить эту главу ничего не потеряв.

Версия системы записана в виде одной строки фиксированного формата и предназначена для возможности обработки автоматическими скриптами по обновлению системы. Шаблон подробной версии системы следующий **L.LL.VVV/TBBB**, где **L.LL** major-версия системы, **VVV** minor-версия системы, **T** признак типа билда системы, **BBB** билд текущей версии.

Major версия системы отображает поколение системы и не несет никакого системного смысла.

Minor версия системы имеет сквозную (по отношению к major версии) нумерацию, после каждого релиза этот номер увеличивается на единицу и начинается новая бэта-версия.

Признак типа билда принимает одно из следующих значений:

1) **A** – альфа версия – доподлинно известно, что данная версия системы не является полностью работоспособной, производятся работы с кодом существующего функционала системы и эти работы находятся в состоянии когда старое уже не работает, а новое еще не работает как требуется, такая версия выкладывается для обновления текущими участниками процесса изменений в текущей версии;
2) **B** – бэта версия – подразумевается гипотетическая правильность работы системы, однако требуется проверка работоспособности отдельных элементов и всей системы в целом, при этом некоторые работы над новым или измененным функционалом текущей версии могут быть не закончены, данная версия рекомендуется для установки на сервера разработки при наличии интереса к новому функционалу;
3) **C** – кандидат на релиз – работы по новому или измененному функционалу в текущий версии закончены, инструкции по обновлению версии написаны, происходит окончательная проверка системы, выявление подводных камней и возможных ошибок, данная версия рекомендуется для установки на сервера разработки и сервера дублеры при работе в паре, появление нового билда этого типа означает, что в систему внесены очередные поправки по результатам тестирования;
4) **R** – релиз – стабильность работы и правильность описанных для обновления версии действий проверены, все найденные недочеты исправлены, данная версия рекомендуется для установки на продакшен сервера с учетом правил стэйджинга, появление нового билда этого типа означает, что в систему внесены незначительные дополнения, не являющиеся критическими и не влияющие на работу системы в целом;
5) **U** –обновление к релизу – данная версия содержит исправления найденных ошибок или несистемные изменения улучшающие качество соответствующего релиза системы, появление нового билда этого типа означает наличие важных изменений в релизе – исправление ошибки или серьёзные улучшения;

Билд текущей версии системы изменяется для возможности определения наличия изменений в содержимом дистрибутива и сбрасывается в значение **000** с каждой новой версией системы.

Поддержание системы в рабочем состоянии

Поддержка сервера ACM.CMS заключается в штатной поддержке выбранной операционной системы и базы данных, так как любая активность системы ACM.CMS осуществляется с расчетом на то, что вмешательство администратора не требуется. Например:

- автоматическое удаление временных файлов;
- автоматическое удаление старых логов;
- автоматическая поддержка актуального состояния дискового кеша;
- автоматическое удаление лишних объектов из базы данных;
- постоянный мониторинг системы с перезапуском в критических ситуациях.

Отладочные проверки

К отладочным проверкам относятся проверки, предназначенные для анализа возможных проблемных мест. Область их применения ограничена этим направлением, поскольку в нормальном (live/production) режиме работы эти проверки не осуществляются вовсе.

В любом случае система обрабатывает возникающие ошибки, однако, смысл отладочных проверок заключается в том, что они, будучи натыканы в различных опорных точках системы, должны помочь определить неправильный ход выполнения как можно ближе к месту возникновения ошибки, что может существенно помогать в отладке.

Для отладочных проверок используются java-assertions. При разрешенных assertions (опция java –ea) система будет работать выполняя все отладочные проверки. При запрещенных assertions, загружая классы системы, JVM будет сразу исключать код таких проверок, словно его и не было. Такое решение позволяет иметь большое число отладочных проверок с возможностью их полного отключения для увеличения производительности.

Рекомендуется конфигурация, в которой assertions отключены на production сервере и включены на сервере разработки.

FAQ

HTTPS, SSL, TLS

Хочется чтобы встроенные интерфейсы управления не перенаправляли пользователей на безопасный интерфейс

Это происходит изза того, что в настройках встроенных скинов интерфейсов управление указан параметр требующий безопасного интерфейса.

Для того, чтобы позволить пользователям работать с интерфейсом управления по небезопасному интерфейсу – нужно пометить этот небезопасный интерфейс признаком «secure», как это сделать описано в файле-примере interfaces.xml.sample.

ВНИМАНИЕ: после этого перехват логина и пароля пользователя перестанет быть проблематичным. Авторизационная информация легко выделяема из потока данных. При работе по безопасному соединению авторизация является секретом компьютера пользователя и сервера, при работе по открытому интерфейсу данная информация открыта для всех узлов находящихся на пути к серверу и для всех компьютеров локальной сети, а при использовании технологии WiFi при подключении компьютера пользователя это информация будет также доступна всем находящимся в радиусе действия радиосигнала.

Требуется установить сторонний сертификат для одного из защищенных интерфейсов

По умолчанию система автоматически формирует самовыданный сертификат (иначе невозможна установка SSL/TLS соединения). Для того случая, если имеется желание установить другой сертификат на каждый SSL/TLS интерфейс существует пара файлов certreq* и keystore* в каталоге %ACM_PROTECTED%/settings/flt-tls/. Файл certreq* содержит запрос на выдачу сертификата для указанных имен CN – он требуется для выдачи и передачи сертификата по открытым/незащищенным сетям. После получения сертификата его требуется установить в keystore – для этого используется стандартная утилита входящая в состав java – keytool.

Сервер не запускается

В основном логе строка вида: 000030.20: BOOT: WATCHDOG: FATAL: Not initialized in 300 seconds - system exit

На этапе первичного запуска система контролирует работоспособность посредством таймера, если для инициализации встроенных сервисов, подключения локальной базы и считывания настроечных файлов выделенного времени не хватило – будет осуществлен перезапуск. Такое может случиться изза ошибки\deadlock при инициализации системы, изза загруженности файловой системы, компьютера в целом или изза недостаточной производительности компьютера. Если это происходит вследствие загруженности дисковой системы, то после перезапуска часть затронутых данных должна находиться в кеше операционной системы и система рано или поздно запустится. Если цикл перезапусков продолжается бесконечно – следует разобраться в возможной причине снижения производительности системы в целом – скорее всего имеется какая-то пожирающая процессор программа или принципиально не хватает производительности компьютера на котором установлена система.

Сервер необычайно тормозит

Дофига записей в s3ChangeQueue и s3ChangeInfo

Хранилище рассчитано на нормальную работу нескольких серверов с общим набором данных (кластер). Для реализации обмена сообщениями в качестве last resort используются таблицы s3ChangeQueue и s3ChangeInfo – в эти таблицы кладется информация об изменениях данных (сообщения), необходимая для сброса дискового кеша, кеша в оперативной памяти, индексации созданных/инмененных/удаленных объектов. При подключении очередного сервера к хранилищу, он автоматически регистрируется в таблице s3ChangePeer. Регистрация сервера действительна 48 часов для активного сервера и 24 часа для клиентского сервера, каждый сервер в составе кластера обновляет свою регистрацию каждые 15 минут. Все общие сообщения (broadcast) дублируются для всех серверов, зарегистрированных в s3ChangePeer. Каждое сообщение имеет срок годности в пределах одной недели. Если в s3ChangePeer образовалась «подвисшая» запись – количество сообщений в активно изменяющемся хранилище может достичь таких высот, что торможение сервера при любых обновлениях данных станет заметным.

Для решения данной проблемы требуется удалить из s3ChangePeer все записи старее часа (прекратится сохранение лишних сообщений), таблицы s3ChangeInfo и s3ChangeQueue должны очиститься автоматически в течении получаса.

Профайлер базы данных показывает много поисковых запросов

Поисковые запросы – самые сложные запросы. Определить из легко по нахождению в них подстроки `ix.weight`. При наличии в запросе на поиск поискового условия OR (ему может соответствовать как действительно условие OR в строке поиска, так и наличие нескольких словоформ у одного из слов запроса при «неточном» поиске) одному поиску будет соответствовать не один запрос, а целая серия запросов, результаты которых система сложит и отсортирует. Каждое условие OR усложняет поисковый запрос в 2 раза. При большой сложности запроса система автоматически отменяет поиск по словоформам.

Поскольку работа сервера с данными хранилища в базе данных сереализированна (происходит последовательно), такие запросы могут тормозить работу сервера в целом (т.к. другие запросы будут дольше ожидать выполнения в очереди, пока база данных занята поиском). Единственным советом в этой ситуации может быть только произвести ревизию поисков на собранном сайте, упрощение и переформулирование поисковых условий, проверку полей типов обозначенных как «индексируемые» (от общего объема индексов зависит скорость выполнения каждого конкретного поискового запроса).

Также следует избегать использования не только сложных, но и вообще любых поисковых запросов для формирования элементов скина сайта «на лету» - например, можно выполнять такие запросы в определенные интервалы времени и сохранять их результаты для дальнейшего использования в скине.

Зачем WebDAV интерфейс? Почему не использовать FTP или CIFS (SAMBA)?

1) И с ФТП и с САМБОЙ можно легко получить файл нулевой длинны или обрезанный файл (что бывало неоднократно на практике с различными клиентами) – после этого остается только молиться, что файл сохранен в бэкапе или еще гденить. Происходит это изза того, что при попытке сохранения файла по этим протоколам запись начинает производиться сразу.

Встроенный протокол WebDAV осуществляет реальную попытку перезаписи файла только после полного и успешного получения этого файла.

2) При работе с файлами по FTP/CIFS/SSH или любым другим внешним способом требуется вести дополнительную базу пользователей и следить за своевременной и адекватной раздачей прав (в том числе и чтобы сохранить требуемые права доступа к этим файлам для системы). Встроенный протокол WebDAV использует ту же систему авторизации, что и для входа в интерфейс управления сайтом.

3) Также встроенный интерфейс WebDAV консолидирует различные файлы имеющие отношение к сайту – файлы скинов, файлы типов, логи, виртуальные файлы управления модулем FILECONTROL, данная документация, а, если сервер является управляющим сервером – дает доступ ко всем файлам системы и файлам других серверов установленных в этой копии системы ACM.CMS.

4) FTP и CFIS (SAMBA) не являются защищенными протоколами – для безопасной работы с удаленным сервером по этим протоколам требуется установление туннеля. Встроенный протокол WebDAV работает по HTTPS являясь безопасным протоколом.

5) В конце концов – никто не запрещает пользоваться FTP, CIFS, SSH, RDP и прочими способами добраться до файлов.

Как добавить в систему новый content-type или расширения файла?

Для этого надо добавить запись в файл **`%ACM_PUBLIC%/resources/data/mime-types.txt`**.

В системе не хватает нужного языка, как его добавить?

Для этого надо добавить запись в файл **`%ACM_PUBLIC%/resources/data/languages.txt`**.

Как сделать чтобы по любому не разданному имени открывался определенный виртуальный домен?

Для этого надо в файле **`%ACM_PROTECTED%/conf/servers.xml`** указать директиву **`<default>идентификатор сервера</default>`**. Пример данной директивы находится в файле **`%ACM_PROTECTED%/conf/servers.xml.sample`**.

Это даст право данному виртуальному домену получать запросы, которые система не знает к какому виртуальному домену маршрутизировать. Для того, чтобы по конкретному имени открывался определенный сайт с определенными настройками, пользователи этого виртуального домена должны настроить соответствующую точку доступа.

Как можно повлиять на параметры запроса параметрами URL?

Для отладочных работ и т.п. может быть использованы следующие параметры строки запроса:

1) _ic_ - возможность явно указать кодировку (например, если вы отправляете параметры и хотите явно указать кодировку (которая, кстати, по умолчанию UTF-8), вы можете использовать запрос вида http://example.org/?a=b&_ic_=windows-1251&text=абвгд)
2) _cd_ - возможность явно указать команду запроса (например, если вы хотите получть ответ на команду OPTIONS, а ваши средства позволяют сделать только команду GET, вы можете использовать запрос вида http://example.org/?_cd_=OPTIONS)
3) _ht_ - возможность явно указать хост (target) запроса (например, если вы пробросили SSH туннель и доступаетесь к сайту через имя localhost, вы можете использовать запрос вида: http://localhost:8080/?_ht_=example.org

Как добавить пользователя для доступа к админке имея доступ к базе данных и серверу?

Если в системе нет пользователей или если безнадежно потерян пароль с административным доступом для восстановления контроля над виртуальным доменом имея контроль над сервером acm.cms и базой данных к которой подключен сайт нужно:

1) В таблицу **umUserAccounts** добавить запись вида:
   <уникальный идентификатор>, <логин>, null, 0, 0, 'en', 20, null, null
2) В таблицу **umUserGroups** добавить запись вида:
   'def.supervisor', <уникальный идентификатор>, 'manual'
3) Войти в систему используя выбранный логин и пустой пароль
4) Перезапустить acm.cms и войти в систему используя выбранный логин и пустой пароль

Уникальный идентификатор – любая уникальная последовательность цифр и латинских букв. Логин – последовательность символов, которая будет использована для входа в систему.

Как настроить правила отправки почтовых сообщений: задержки, домены и т.п.?

Никак – система не является почтовым сервером, она, как и большинство других программ, рассчитана на работу с MTA (Mail Transfer Agent). В качестве MTA выступает локальный или удаленный почтовый сервер – выбирайте и настраивайте его в соответствии с вашими потребностями. Задача реализации пересылки почтовых сообщений уже давно и многократно решена – существует множество прекрасных почтовых серверов на любой вкус, а принцип работы с MTA и сам протокол SMTP давно является стандартами в интернет-индустрии.

Архитектура

В этом разделе рассматриваются аспекты устройства системы, которые не имеют непосредственного отношения ни к администрированию, ни к сборке сайтов, однако, понимание этих аспектов может оказать определенную помощь при решении проблем или при сравнении данной системы с другими.

Загрузка

Предпочтение отдано собственному загрузчику, т.к. в отличие от встроенных в Java средств, такой загрузчик способен не только производить выбор версий библиотек, но и позволяет значительно ускорить процесс загрузки и экономить занятую оперативную память.

Ускорение процесса загрузки осуществляется за счет того, что библиотеки считываются сразу и классы из каждой из них считываются последовательно, что в случае с zip-архивом дает значительный выигрыш по сравнению с произвольным поиском элементов и произвольным их считыванием, как делает ClassLoader предоставляемый JVM по умолчанию.

Экономия памяти, как это не парадоксально, осуществляется за счет того, что все данные библиотек загружаются сразу – т.к. в ином случае память JVM будет занята огромным числом внутренних данных связанных со структурами открытых zip-файлов. В качестве дополнительных соображений – приложение является сервером, и предназначено для долгой непрерывной работы – поэтому рано или поздно все данные библиотек так или иначе будут загружены в память.

Коренной загрузчик (boot.jar)

Запуск системы осуществляется **%ACM_PUBLIC%/boot.jar** – в этом модуле находится загрузчик системы. Он отвечает за определение **%ACM_PUBLIC%**, **%ACM_PROTECTED%**, **%ACM_PRIVATE%**, за загрузку актуальных версий системных и дополнительных библиотек, определение последней версии ядра, его загрузка и передачу управления ядру.

Для параметра **ru.myx.ae3.properties.path.public**: если он не указан, его значение устанавливается равным пути к каталогу откуда произведен запуск системы.

Для параметра **ru.myx.ae3.properties.path.protected**: если он не указан, его значение устанавливается равным **<user.home>/acm.cm5/protected**.

Затем производится загрузка библиотек. Библиотеки загружаются из каталогов **%ACM_PUBLIC%/axiom** и **%ACM_PROTECTED%/axiom**. Версия библиотеки и её имя содержатся в имени файла, которое построено по следующему принципу:

1) Папка **<library_name>-<version1>.<version2>[.<version3>[.<version4>]]/**
2) Файл **<library_name>-<version1>.<version2>[.<version3>[.<version4>]].jar**
3) Файл **<library_name>-<version1>.<version2>[.<version3>[.<version4>]].zip**

Каждый **versionX** должен содержать целое число от 0 до 127. Последовательность **versionX** определяет версию библиотеки. Загрузчик подключает последнюю версию для каждой библиотеки. Если в имени файла библиотеки не содержится обязательной части шаблона – файл будет проигнорирован загрузчиком.

После этого загружаются классы ядра. Загрузка ядра осуществляется из каталога **%ACM_PUBLIC%/boot**. Версия ядра также содержится в имени файла, построенному по следующему принципу:

4) Папка **<version1>.<version2>[.<version3>[.<version4>]]/**
5) Файл **<version1>.<version2>[.<version3>[.<version4>]].jar**
6) Файл **<version1>.<version2>[.<version3>[.<version4>]].zip**

Каждый **versionX** должен содержать целое число от 0 до 127. Последовательность **versionX** определяет версию ядра. Загрузчик подключает последнюю версию ядра и передает ей управление. Если в имени файла не содержится обязательной части шаблона – файл будет проигнорирован загрузчиком.

Затем если в классах ядра имеется файл properties или .properties – значения указанные в этих файлах используются для устанавливаются системные переменных окружения.

В классах ядра, стандартным способом (META-INF/MANIFEST.MF) определяется главный класс, для запуска этого класса стартует новая нить исполнения.

Загрузчик ядра

Загрузчик ядра расположен в каталоге **`%ACM_PUBLIC%/boot`**, он отвечает за загрузку сервисов в каталоге **`%ACM_PUBLIC%/features`** с учетом настроек запрещающих и разрешающих запуск сервисов. И загрузку модулей в каталоге **`%ACM_PUBLIC%/modules`** в отдельных, изолированных загрузчиках.

В первую очередь запускается новая нить инициализации основных систем, а загрузчик ядра контроллирует выполнение нити. В случае когда инициализация ядра не может уложиться в отведенное время – система перезапускается. Затем производится загрузка и инициализация сервисов, после чего производится загрузка и инициализация модулей.

Нить инициализации

Для параметра **`ru.myx.ae3.properties.path.public`**: если он не указан, его значение устанавливается равным пути к каталогу откуда произведен запуск системы.

Для параметра **`ru.myx.ae3.properties.path.protected`**: если он не указан, его значение устанавливается равным **`<user.home>/acm.cm5/protected`**.

Для параметра **`ru.myx.ae3.properties.path.private`**: если он не указан, его значение устанавливается равным **`<user.home>/acm.cm5/private`**.

Для параметра **`ru.myx.ae3.properties.path.cache`**: если он не указан, его значение устанавливается равным **`%ACM_PRIVATE%/cache`**.

Для параметра **`ru.myx.ae3.properties.path.logs`**: если он не указан, его значение устанавливается равным **`%ACM_PRIVATE%/logs`**.

Для параметра **`ru.myx.ae3.properties.path.temp`**: если он не указан, его значение устанавливается равным **`%ACM_PRIVATE%/temp`**.

Создается коренной контекст VFS и в нем настраиваются пути public, protected, private, storage.

Производятся тесты и явная инициализация всех базовых систем.

Регистрируются преобразователи изображений.

Настраивается коренной ECMA контекст коренного сервера: в него кладутся объкты представляющие собой все статические API.

Инициализируется подсистема передачи данных по сети: NIO.

Инициализируется подсистема безопасной передачи данных: TLS.

Инициализируется подсистема протокола HTTP/HTTPS.

Инициализируется поддержка MIME форматов.

Регистрируется JDBC драйвер – трассировщик активности обмена с базой данных.

EventRecieverToStdout

Регистрируются основные сериализаторы и материализоторы маршаллинга.

Стартует интерфейс FILE_CONTROL.

Регистрируется модуль определения географической привязки по сетевому адресу.

Регистрируются скриптовые языки: Ecma, Tpl, Xslt, Map, Text, Null.

Регистрируется поддержка WebDAV.

Стартует монитор жизнеспособности системы, он работает до завершения работы системы и контроллирует наличие свободной памяти, количество ниток и должен перезапускать систему при обнаружении критических проблем.

Загрузка сервисов

Версия сервиса и его имя содержатся в имени файла, которое построено по следующему принципу:

7) Папка **<feature_name>-<version1>.<version2>[.<version3>[.<version4>]]/**

8) Файл **<feature_name>-<version1>.<version2>[.<version3>[.<version4>]].jar**

9) Файл **<feature_name>-<version1>.<version2>[.<version3>[.<version4>]].zip**

Каждый **versionX** должен содержать целое число от 0 до 127. Последовательность **versionX** определяет версию сервиса. Загрузчик подключает последнюю версию для каждого сервиса. Если в имени файла сервиса не содержится обязательной части шаблона – файл будет проигнорирован загрузчиком.

Производится запуск всех разрешенных сервисов. Для этого последовательно осуществляется вызов метода main от главного класса каждого сервиса. Этот медот должен осуществить регистрацию возможностей данного сервиса в системе.

Загрузка модулей

Версия модуля и его имя содержатся в имени файла, которое построено по следующему принципу:

10) Папка **<module_name>-<version1>.<version2>[.<version3>[.<version4>]]/**

11) Файл **<module_name>-<version1>.<version2>[.<version3>[.<version4>]].jar**

12) Файл **<module_name>-<version1>.<version2>[.<version3>[.<version4>]].zip**

Каждый **versionX** должен содержать целое число от 0 до 127. Последовательность **versionX** определяет версию модуля. Загрузчик подключает последнюю версию для каждого модуля. Если в имени файла модуля не содержится обязательной части шаблона – файл будет проигнорирован загрузчиком.

Производится запуск всех найденных модулей. Для этого осуществляется вызоа метода main от главного класса каждого модуля. Этот медот должен осуществить регистрацию возможностей данного сервиса в системе.

Загрузка, модуль AE1

Модуль AE1 отвечает за инициализацию среды и подсистем AE1/RT3, которые представляют собой основу ACM.CMS.

Процесс начинается с определения умолчательного коренного каталога для серверов AE1 – проверяется параметр запуска **ru.myx.ae1.path.web**, если он не определен, то данным каталогом будет **%ACM_PROTECTED%/web**.

Затем в коренном контексте выполнения регистрируется Control API, которое свойственно для ACM.CMS.

Затем осуществляется инициализация переменных окружения из файла **%ACM_PROTECTED%/conf/properties.xml** (файл создается, если его нет).

После этого осуществляется инициялизация классов, указанных в файле **`%ACM_PROTECTED%/conf/initialize.xml`** (файл создается, если его нет). Это необходимо для регистрации драйверов JDBC и т.п.

После этого осуществляется регистрация всех серверов указанных в файле **`%ACM_PROTECTED%/conf/servers.xml`** (файл создается, если его нет).

Затем производится запуск интерфейсов описанных в файле**`%ACM_PROTECTED%/conf/interfaces.xml`** (файл создается, если его нет).

С этого момента система ACM.CMS может принимать запросы прользователей.

Затем модуль AE1 производит последовательный запуск всех серверов, чтобы убедиться, что любой сервер, вне зависимости от наличия к нему запросов со стороны пользователей, будет запущен (например, для выполнения фоновых задач).

Безопасность

Разделим возможные опасности на следующте группы:

1) получение доступа к закрытым или личным данным пользователей
2) возможность несанкционированой модификации данных
3) нарушение функционирования сервиса

Хранение паролей

Система не хранит паролей пользователей, вместо паролей хранятся результаты криптографической функции примененной к имени пользователя и паролю, при очередной проверке пароля система применяет ту же функцию к введенным пользователем данным, чтобы сравнить результат с хранящимся ключём. Таким образом – даже получив доступ к базе данных или к образу памяти JVM злоумышленник (или недобросовестный администратор) не сможет получить пароли пользователей (которые могут быть использованы этим пользователям и на других сайтах). Более того, при такой схеме хранения, даже если два пользователя будут иметь одинаковые пароли – этого нельзя будет понять по хранимым в системе данным.

Защита от подбора пароля

При каждой неудачной проверке пароля система производит задержку перед ответом препятствуя перебору пароля. При первой неудачной проверке пароля пользователя задержка составляет 2 секунды, с каждой последующей попыткой задержка увеличивается и достигает 10 секунд. Сброс счетчика отвечающего за задержку происходит после каждой удачной авторизации.

Это относительно важно, так как производительность системы высока и, даже, обычный тестовый сервер для одного пользователя может производить более 8000 проверок пароля в секунду по сети (при использовании HTTP – без шифрования) и в сотни раз больше из скрипта работающего локально на сервере. Задержка при неуспешных проверках позволяют снизить возможную скорость перебора более чем в сотни тысяч раз!

Защита от перехвата пароля

С самого начала система построена с поддержкой безопасной передачи данных с использованием шифрования (HTTPS – протокол HTTP поверх SSL). При нормальной (правильной) настройке сервера и точек доступа – любая проверка пароля (приглашение к вводу пароля и его передача от пользователя к серверу), также как и операции по смене пароля и редактированию личных данных производятся по шифрованному каналу, что препятствует перехвату этой информации на пути от компьютера пользователя к серверу.

Также, следует отметить, что большинство браузеров не кешируют на диске страницы и cookies полученные по HTTPS и не сохраняют результаты заполнения форм – так что можно более спокойно входить в систему с «чужих» компьютеров – однако, это не исключает возможность «записи» нажатий на клавиши при наборе пароля специальной программой «троянским конем» на зараженном компьютере или от простого подглядывания «через плечо» ☺.

Защита от перехвата авторизационного «печенья» (cookie)

Поскольку HTTP является протоколом без «состояния» - для отслеживания этого самого состояния используются «сессии». В двух словах, сессия в системе ACM.CMS состоит из набора данных хранимого на сервере и уникального идентификатора, к которому привязаны

эти данные, передаваемого сервером клиенту и, впоследствии, передаваемого клиентом к серверу при каждом запросе в рамках текущей сессии. Состояние авторизации хранится для каждой сессии, после успешной авторизации система формирует новый идентификатор сессии, и, до тех пор, пока текущая сессия считается авторизаванной, при правильной настройке сервера, этот новый идентификатор будет передаваться только по защищенному каналу, не позволяя злоумышленнику возпользоваться идентификатором сессии для получения доступа к закрытым данным или возможностям пользователя.

Более того, авторизационные данные сессии привязаны к определенному адресу в сети, с которого эта авторизация проводилась, поэтому, даже при такой настройке сервера, которая позволяет передачу этих данных по открытому каналу – злоумышленнику будет не достаточно просто перехватить идентификатор сессии для того, чтобы им воспользоваться.

Кеш

Кеш в оперативной памяти

Две основные задачи возложенные на кеш – это, собственно, тактическое кеширование конечных и промежутчных результатов выполнения «дорогих» операций и исключения ситуаций, когда одна кешируемая опирация с одними аргументами выполнялась бы одновременно в нескольких параллельных нитках.

Системный кеш – это не просто словарь ключей и значений: для его эффективной работы применяется масса сложных решений, позволяющих добиться следующих результатов:

1) запись и удаление элементов в кеше не производят блокирования чтения из кеша
2) запись и удаление элементов в кеше не проивзодят блокирования записи и удаление других элементов
3) опрерации по поддержанию кеша не блокируют других операций над кешем
4) при автоматическом освобождении элементов кеша учитывается не только максимальное время хранения элемента, но и время последнего доступа к нему
5) при отсутствии нужного элемента в кеше, операция для создания этого элемента будет проиведена единожды, даже при конкурентных запросах к этому элементу
6) блокировка при создании отсутствующего в кеше элемента не блокирует другие операции над кешем, включая чтения и создания других элементов
7) при попадании в кеш «одноразовых» значений, они автоматически преобразуются в «многоразовые»
8) при попадании в кеш «больших» по объему бинарных данных – эти данные сохраняются на диске для экономии оперативной памяти
9) при нехватке оперативной памяти осуществляется автоматическая чистка элементов кеша

Дисковый кеш

Основная задача дискового кеша – сохранение локально идентифицируемых данных, полученных из базы данных или других источников, которые требуют бережного и экономного обращения. Доплнительная, автоматически решаемая при этом задача – хранимые локально данные имеют произвольный доступ и значительно более гибки для любых дальнейших алгоритмов работы над ними.

Идентифицируемыми данными назовем те данные, для идентификаии которых можно получить определенный «ключ», значение которого будет постоянным и одинаковым для последующих обращений к тем же данным. Сами данные могут быть как постоянными, так и изменяемыми. Для изменяемых данных системы обменивается сообщениями об изменении данных для сброса кеша.

Поскольку система может работать в кластере, но даже и без работы в этом режиме в состоянии ворочить гигантскими объемами данных, любая возможность сократить нагрузку на базу данных и уменьшить количество запросов к ней стоит своих «свеч». Все операции по получению идентифицируемых дающие существенный вклад по снижению нагрузки на СУБД используют системный дисковый кеш.

Два основных требования к дисковому кешу – автоматическое поддержание актуального состояния и достаточная эффективность алгоритмох хранения и доступа к данным, чтобы быть заметно быстрее, чем полный цикл обращения к СУБД (формирование запроса, ожидание

ответа, разбор полученных данных) даже при работе одного экземпляра ACM.CMS с выделенной базой данных.

В дисковом кеше хранится три основных вида информации, любой файл в дисковом кеше представляет собой информацию одного из перечисленных видов:

1) информация об узлах дерева хранилища – информация о самом узле, списки дочерних элементов;
2) информация об объектах хранилища (каждый объект может быть привязан к нескольким узлам дерева);
3) блобы – цельные куски данных представленные в виде файлов;

Хранилище S3

Очередь обновлений

Очередь отложенных обновлений

При изменении данных хранилища зачастую требуется внесение массы изменений в базу данных. Для более быстрой реакции эти изменения разделены на две группы: изменения, которые нужно производить сразу и изменения, которые можно отложить и произвести на фоне. Задачи по отложенным изменениям сохраняются в таблицу s3ChangeQueue, после этого они последовательно обрабатываются в том же порядке сервером ответственным за обновления. В каждый момент времени определено не более одного сервера ответственного за обновление. Такой подход позволяет избавиться от значительной части конкурентных обновлених базы данных и возможных deadlocks.

Среди отложенных обновлений следующие операции:

1) create(lnkGuid) – индексация созданного объекта

2) create-global(lnkGuid) – индексация и создание синхронизированных копий созданного объекта

3) resync(lnkGuid) – создание синхронизированных копий объекта

4) update(lnkLuid) – переиндексация или удаление индексов объекта

5) update-all(objId) – переиндексация или удаление индексов всех линков на данный объект (если число линков более 8 – происходит разложение задачи на соответствующее число «update»)

6) update-object(objId) – проверка числа оставшихся линков на данный объект и чистка (history, extra, data) если ссылок больше нет

7) clean(lnkLuid, objId) – очистка индексов одного из линков объекта и «update-object»

8) clean-start(lnkGuid) – получает список детей в иерархии дерева для указанной ссылки на объект и производит разложение задачи на соответствующее число «delete-item»

9) clean-all(objId) – удаляет все ссылки на объект, удаляет их индексы и вызывает «update-object»

10) upgrade-index(idxVersion) – выбирает LIMIT_BULK_TASKS*8 объектов хранилища с индексами устаревшей версии, и если результат не равен пустому множеству – сохраняет для каждого из них задачу «update» и одну задачу «upgrade-index» для последующего продолжения обновления индексов

11) delete-item(lnkLuid, objId) – удаление алиасов, индексов, ссылки на объект и произведение «update-object»

12) recycle-start(lnkLuid, delId) – получение списка всех детей в иерархии дерева для указанной ссылки на объект, создание соответствующего числа задач «recycle-item» и создание задачи «recycle-finish» (delId == lnkGuid перемещаемого в корзину объекта)

13) recycle-item(lnkLuid, delId) – очистка индексов ссылки на объект и перенесение данной ссылки в корзину

14) recycle-finish(lnkLuid, delId) – очистка индексов и удаление коренной ссылки на объект перемещенной в корзину ветки дерева

Ротация серверов отвечающих за обновления осуществляется следующим образом: в таблице s3Locks каждые 5 минут каждый подключенный сервер пытается зарегистрироваться как сервер отвечающий за обновления. Сервер, получивший это право, пользуется им один час, после этого 10 минут отдыхает. Если к базе подключены другие сервера – в это время один из них получает право обрабатывать обновления. Если среди подключенных серверов имеются сервера разных версий (версия хранилища, а не СМ в целом), то в этом процессе участвуют только сервера с самой свежей версией.

Обработка событий осуществляется следующим образом:

1) обработчик выбирает LIMIT_BULK_TASKS (=32) задач

2) если список задач пуст – производит до 5ти операций по обновлению extra устаревших версий на новые версии (список экстр устаревших версий собирается по мере работы сервера)

3) производит последовательное выполнение полеченного списка задач

4) если выбранный список задач содержал менее LIMIT_BULK_TASKS элеменов и ни одна выполненная задача не породила новых задач, обработчик переходит в состояние отдыха на 15 секунд

5) перход к первому пункту.

Все изменения одного цикла производятся в одной транзакции, что также позволяет сократить число транзакций в базе данных и ускорить обработку очереди.

Подключения к СУБД

Увеличение конкурентности запросов к СУБД не приводит к росту её производительности, более того, организовав очередь запросов к базе данных, мы достигаем следующие два цели:

1) приоритезация запросов и обработка запросов в очереди в соответствии с их приоритетом

2) отслеживание одинаковых запросов в очереди и их однократное выполнение (работает только при непустой очереди, но именно в такие моменты является актуальным)

Каждый экземпляр хранилища S3 создает следующие соединения к СУБД:

1) одно соединение для загрузки линков хранилища; при каждой итерации обработчика происходит загрузка данных всех ожидающих в очереди линков за один запрос

2) одно соединение для загрузки иерархии объектов хранилища; все задачи в обработчике очереди на загрузку иерархии пользуется одним и тем же, предварительно подготовленным запросом

3) одно соединение для выполнения всех поисковых запросов: getAliases, hasExternal, hasExternals, loadExternal, searchIdentity, searchLinks, searchLocal, searchLocalAlphabet

4) одно соединение для обмена сообщениями об обновлениях объектов хранилища; это же соединение используется для обновления и выполнения отложенных задач, если данный экземпляр хранилища является ответственным за это в данный момент

5) по одному дополнительному соединению на коммит каждой транзакици

Работа с дисковым кешем

Продуктивность дисковой системы является ключевым моментом в общей продуктивности типичного веб-сервера. Это обусловлено тем, что диски имеют значительно меньшую скорость,

чем оперативная память. Дисковая система не любит сотни конкурентно читающих в пишуших ниток – при таком сценарии её эффективность сильно падает. До недавнего времени, максимальную производительность дисковых операций можно было достичь при последовательной работе с диском одним процессом на диск (многодисковые RAID массивы могут выполнять одновременно операции на отдельных физических дисках). Однако, благодаря технологии NCQ и возможностям некоторых операционных систем планировать выполнение ожидающих запросов к диску в соответствии с расположением головок, оптимальное число одновременно работающих с диском процессов должно быть несколько раз больше числа дисков и клеблется в пределах 8-32 штуки.

Для работы с дисковым кешем система использует фиксированное число обработчиков запросов, что позволяет боле продуктивно пользоваться дисковой системой и использовать довольно большие и постоянно переиспользуемые внутренние буферы для формирования данных на запись и разбора данных после чтения.

Общее

Основные преимущества:

1) Реализована на Java – кроссплатформенность
    a. Все основные системы доведены по скорости до норм общепризнаных решений
    b. Предоставляет простой и эффективный доступ для использования любый java-библиотек
2) EcmaScript среда – общеизвестный и стандартный язык
    a.
    b. Система управления пакетами позволяет создавать свои и использовать чужие библиотеки для решения типичных задач
3) Прозрачная работа с персистентными объектами
    a.
4) Прозрачные транзакции / Медленные транзакции
    a. При отсутствии явного управления транзакциями в нити выполнения – транзакция автоматически создается при первом изменении в персистентных данных и коммитится при успешном завершении выполнения данной нити
    b. Для других процессов изменения произведенные в данной транзакции не видны пока она не завершена и, также, не видны если эти другие процессы имеют собственную активную транзакцию начавшуюся раньше.
5) Явный интерфейс к дереву объектов с возможностями поиска данных и статистической обработки
    a. Системные интерфейсы имеют точку входа через свойство глобального объекта «ae3», через выражение «ae3.vfs.root» доступен коренной узел иерархии объектов в файлоподобном виде.
    b. VFS поддерживает поиск по значениям полей, вхождение в массив, условия больше и меньше для числовых значений, нахождение ссылок на объект…

Обновление версий

В данном разделе предоставляется информация по действиям, которые требуется производить при обновлении версий системы, это этинственный раздел, в котором, так или иначе, содержится информация не только по текущей версии системы, но и о предыдущих версиях.

Информация по обновлениям не разделена на «администрирование» и «сборку» преднамеренно: обновление версии это дело, а у каждого дела одна голова. И если начальник, сборщик и администратор не являются одним лицом – то эта одна голова должна решать когда, кто и что будет делать, а если все эти функции сосредоточены в одном лице – то ему было бы удобнее видеть эту информацию собранную в одном месте.

Для каждой версии первым блоком указан краткий список основных изменений, затем идет информацию по действиям, связаным с переходом системы на очередную версию.

Списки действий разделены на две группы: «обязательно» и «желательно» - второй список обычно содержит действия, которые станут «обязаельными» при обновлении последущих версий.

Списки действий представлены в определенном порядке – в большинстве случаев действия связанные с обновлением версии необходимо производить именно в этом порядке (за иключением тех случаев, когда несколько действий идущих в списке одно за другим можно выполнить в произвольном порядке или одновременно).

Все переделки и операции по обновлению версий спроектированы с учетом следующих требований:

1) обеспечение возможности откатить на одну версию назад
2) при совместной работе нескольких экземпляров системы – допустимо различие на одну версию

При вводе новых форматов, отказа от части таблиц или переделке логики подсистем хранения и индексации (что сказывается на возможности работы разных версии системы с одной СУБД) выпускается промежуточная версия, которая умеет понимать/читать эти новые форматы, не пользоваться устаревшими полями и таблицами, но учитывать их в запросах к базе данных и т.п. Только после релиза этой версии выпускается следующая – которая будет реально сохранять даные в новых форматах и будет рассчитывать, что устаревших полей и таблиц уже нет.

Для успешного обновления через несколько версий – требуется обновлять эти версии последовательно. В общем случае это дает возможность отката изменения – так как обратная совместимость гарантируется только на одну версию. В случае обновления серверов кластера или серверов **работающих с одной базой данных** (например devel/test/live) версии систем установленных на этих серверах не должен различаться больше чем на единицу.

Также, за выполнение отложенных задач в кластере всегда отвечают только экземпляры ACM.CMS с высшей версией соответствующей подсистемы. В двух соседних версиях ACM.CMS каждой отдельной подсистеме соответствует либо одинаковая версия подсистемы, либо более новая версия подсистемы в более новой версии ACM.CMS если этого требует внутренний алгоритм перехода и обновления версий.

Переход на версию 678 с версии 677

Основные изменения в системе:

1) Больше совместимости с ECMA / JavaScript – работавший ранее код может не работать
2)

Обязательно:

1) для каждого DLS client удалить таблицу **d1ChangeLog**
2) если используется параметр запуска **serve.root**, заменить его на **ru.myx.ae1.path.web**.
3) заменить старые создания массивов вида **a = (1, 2, 3)** на **a = [1, 2, 3]**.
4) заменить во всех файлах сайтов и скриптах подстроку **Preffix** на **Prefix**.
5) В всех response / filter от типов описанных в файлах ***.scheme** убедиться они начинаются с **if(!HashValid(content)) return content;**.
6)

Желательно:

1) Заменить подстроку **for(** на **for each(** или **for keys(** в зависимости от смысла (**for keys** специвльно сделан для того, чтобы облегчить эту операцию). Также можно заменить на **for v677(** - тогда данный цикл будет работать по-старому.
2) **Create.list()** заменить на **[]**
3) **Create.map()** заменить на **{}**
4) В searchLocal: для лучшей производительности заменить условие **($state=2 | $state=4)** на **$listable=true**
5) В searchLocal: для лучшей производительности заменить условие **($state=2 | $state=5)** на **$searchable=true**
6) для каждой базы данных удалить таблицу **cmLocks**
7) В подключении к mssql поменять формат url на аналогичный указанному в файле примере **servers.xml.sample** расположенном в **%ACM_PROTECTED%/conf**.
8) При использовании mssql поменять имя драйвера в файле **interfaces.xml** на указанное в соответствующем примере **interfaces.xml.sample** в **%ACM_PROTECTED%/conf**.
9) Там, где используется собственная реализация глубокого копирования объектов типа **map** заменить вызовы на системную **Create.mapClone(map)** или **Default.mapPutRecursive(target, source)** и удалить реализацию.
10) Конструкции **for(var i in map.keySet())** заменить на **for(var i in map)**.
11) В циклах где используются методы **Entry.getGuid()** и **Entry.getKey()** рассмотреть целесообразность их замены на **Storage.getEntryGuid(entry)** и **Storage.getEntryKey(entry)**

Переход на версию 677 с версии 676

Основные изменения в системе:

1) Изменения в S3 – searchLocal осуществляет фильтрацию в яве для любых запросов, кроме запрсов, содержащих условия по полю $title или $owner
2) В условиях фильтрации S3 появились условия $listable и $searchable
3) В сортировках S3 появились значения «created», «$modified», «$modified-»
4) В сортировках S3 появились значения «listing», «$key», «$key-»
5) В S3 более агрессивная процедура обновления s3Extra – с целью сократить число записей старых версий сделанных без учета возможности сжатия данных.
6) Серьёзная оптимизация систем ввода/вывода в S3 и S4, позволяющая в одних местах сократить утилизацию памяти, в других местах увеличить производительность, а при записи в дисковый кеш гарантировать целостность данных в случае сбоя.
7) В CVS заведена папка sys-release для того, чтобы последняя стабильная версия всегда была доступна.
8) Изменения в MailAPI, позволяющие отправлять письма со встроенными картинками.

Обязательно:

1) для каждого хранилища s3 разрешить NULL в поле **`s3ChangeQueue.evtCmdDate`**
2) для каждого хранилища s3 удалить таблицу **`s3ChangeLog`**
3) для каждого DLS client разрешить NULL в поле **`d1ChangeQueue.evtCmdDate`**

Желательно:

1) В searchLocal: для лучшей производительности заменить условие ($state=2 | $state=4) на $listable=true
2) В searchLocal: для лучшей производительности заменить условие ($state=2 | $state=5) на $searchable=true
3) для каждого DLS client удалить таблицу **`d1ChangeLog`**
4) В подключении к mssql поменять формат url на аналогичный указанному в файле примере **`servers.xml.sample`** расположенном в **`%ACM_PROTECTED%/conf`**.
5) При использовании mssql поменять имя драйвера в файле **`interfaces.xml`** на указанное в соответствующем примере **`interfaces.xml.sample`** в **`%ACM_PROTECTED%/conf`**.

Переход на версию 676 с версии 675

Основные изменения в системе:

1) Изменения в системе управления пользователями, включая:
    a. При смене пароля через «забыл пароль» - меняются оба пароля пользователя
    b. При проверке пароля используется объект User, а не специальный запрос
2) В дисковом кеше хранилища изменен формат ключа searchLocal – при обновлении ожидается сильная нагрузка ввиду большого числа промаха кешей – хорошо бы обновлять в относительно спокойное время для сервера и прогнать линк-чекер для заполнения кеша.

Обязательно:

1) Конструкцию `SafeGet( a, b )` заменить на конструкцию `( a || b )`.
2) Конструкцию `HashCreate()` заменить на `Create.map()`
3) Конструкцию `HashCreate( x )` заменить на `Create.map( x )`
4) Конструкцию `ArrayCreate()` заменить на `Create.list()`
5) Конструкцию `ArrayCreate( x )` заменить на `Create.list( x )`
6) Конструкцию `HashGet( a, b )` заменить на `a[ b ]`
7) Конструкцию `HashPut( a, b, c )` заменить на `a[ b ] = c`
8) Конструкцию `GenerateUniqueID()` заменить на `Create.guid()`
9) Конструкцию `contentTypeForFileName( x )` заменить на `File.getContentTypeForName( x )`

Желательно:

1) проверить работу регистрации пользователей, логина, смены пароля
2) для каждого хранилища s3 разрешить NULL в поле `s3ChangeQueue.evtCmdDate`
3) для каждого хранилища s3 удалить таблицу `s3ChangeLog`
4) для каждого DLS client разрешить NULL в поле `dlChangeQueue.evtCmdDate`
5) Конструкцию `Split( a, b )` заменить на `splitRegex( a, b )` – идентичная или на `split( a, b )`, если не требуется поддержка регулярных выражений
6) Конструкцию `Split( a )` заменить на `split( a, ',' )`
7) Конструкцию `split( a )` заменить на `split( a, ',' )`
8) Конструкцию `Replace( a, b, c )` заменить на `replaceRegex( a, b, c )` – идентичная или на `replace( a, b, c )`, если не требуется поддержка регулярных выражений
9) Конструкцию `a.get( b )` заменить на `a[ b ]`.
10) Конструкцию `a.put( b, c )` заменить на `a[ b ] = c`.

Переход на версию 675 с версии 674

Основные изменения в системе:

1) полностью изменено устройство поисковой части поисковика используемого в модулях S3 и DLS – теперь он умеет разделять запросы «ИЛИ» на отдельные составные части, позволяя обработчику выполнять другие задачи между этими составными частями запроса

Обязательно:

1) во всех вызовах метода **`entry.search`** где используется параметр **`timeout`** со значением более 0, но менее 5000 заченить на осмысленное значение, ранее такие значения приравнивались к 5000
2) во всех вызовах метода **`entry.search`** где используется параметр **`timeout`** со значением 0 при параметре **`limit`** также равном 0 обдумать, действительно ли надо искать в обход кеша (результаты поиска кешируются на 5 минут), и, если нет уверенности – заменить на осмысленное значение

Желательно:

1) Конструкцию **`HashCreate()`** заменить на **`Create.map()`**
2) Конструкцию **`HashCreate( x )`** заменить на **`Create.map( x )`**
3) Конструкцию **`ArrayCreate()`** заменить на **`Create.list()`**
4) Конструкцию **`ArrayCreate( x )`** заменить на **`Create.list( x )`**
5) Конструкцию **`HashGet( a, b )`** заменить на **`a[ b ]`**
6) Конструкцию **`HashPut( a, b, c )`** заменить на **`a[ b ] = c`**
7) Конструкцию **`GenerateUniqueID()`** заменить на **`Create.guid()`**
8) Конструкцию **`contentTypeForFileName( x )`** заменить на **`File.getContentTypeForName( x )`**

Переход на версию 674 с версии 673

Основные изменения в системе:

1) серьезные изменения в скинах
2) систематизированы настройки системных каталогов

Обязательно:

1) Для каждого скина каждого сайта:
    a. Если нет файла skin.settings.xml – его требуется создать, или эти скины не будут видны системе.
    b. Добавить в skin.settings.xml параметр type со значением HIERARCHY – этот параметр теперь определяет типа скина, тип HIERARCHY соответствует тому единственному типу скина, который был до 674 версии, пример: `<type>HIERARCHY</type>`.
    c. Добавить в skin.settings.xml параметр chartset отображающий кодировку, в которой написаны скрипты скина, например: `<charset>windows-1251</charset>`.
    d. Добавить в skin.settings.xml параметр renderer с именем языка, на котором написаны скрипты скина, если такого параметра там еще нет (он поддерживался и раньше, но не был обязательным), например: `<renderer>ACM.JSCRIPT</renderer>`.
    e. Если в параметрах плагина SKINNER в config.xml было указан параметр personalized – требуется указать его в skin.settings.xml, например: `<personalized>true</personalized>`.
2) Из config.xml каждого сайта требуется убрать плагины SKINNER, WEBDAV и ADMIN – теперь они регистрируются автоматически и входят в состав RT3 сайта независимо от настроек.
3) Если у вас использовались ранее значения для каталогов %ACM_PROTECTED% и/или %ACM_PUBLIC% по умолчанию - укажите параметры явно в скрипте запуска системы - для них изменилось значение по умолчанию: было “текущий каталог в момент запуска”, стало “домашний каталог пользователя под которым произведен запуск”/acm.cm5/protected или private соответственно.

Желательно:

1) Конструкцию `SafeGet( a, b )` заменить на конструкцию `( a || b )`.