Платформа VNetGIS.com

Руководство Программиста

«виртуального картографического сервера»

Версия прогр. обеспечения	0.3
Версия документа	0.0.1
Дата модификации	16.03.2003

2003

Содержание

1. Введение

1.1. Используемые термины:

Картографический сервер – программное обеспечение, получающее HTTP-запросы и формирующее ответ в виде «рисунка карты» или специальном текстовом формате (результат поискового запроса, «подсказки» и т. д.)

WEB-сервер – сервер клиента, на котором размещен «картографический интерфейс». Клиент (при помощи браузера) «обращается» к WEB-серверу для просмотра карты, поиска улицы и др.

Клиент – любой пользователь Интернет, указавший в браузере адрес страницы **WEB-сервера**, на которой расположен картографический интерфейс.

Картографический интерфейс – программный код (HTML, JavaScript, DHTML и др.) размещенный на WEB-сервере, позволяющий клиенту получить доступ к интерактивным картам.

1.2. Условные обозначения

\n – символ «новая строка» (шестнадцатеричное значение в коде ASCII = 0A)
Внимание! – необходимо обратить особое внимание, на указанные особенности использования данной функции сервера.
WEB – указанный параметр настраивается через ВЕБ- интерфейс картографического сервера (смотри «Руководство администратора»)
[…] – указывает условное название переменной, подробно описано ниже в тексте.
Пример: map_size=[MAP_SIZE]
Далее в тексте описан параметр MAP_SIZE:
MAP_SIZE – указывает реальное ....
В реальном результате запроса будет указано:
map_size=2

1.3. История изменений.

Версия 2.1 (08.06.2011)
1. Добавлен протокол JSONP lg=p

Версия 2.0 (17.12.2007)

1. Переменная scale теперь является пиксельным масштабом. Т.е. отношением длины в мировых координатах к длине в пикселях.
2. Переменная mc больше не отдаётся в результате запроса new, т.к. фактически эквивалентна переменной scale.
3. Переменная `map_pix_size` в результате запроса new, get_state `отдаётся в JavaScript (lg=j) отдаётся массивом, а не строкой.`

```
  Было: var map_pix_size='480,400';
  Сейчас: var map_pix_size=new Array(480,400);
```

4. В результате запроса lsearch, переменная prefix ВСЕГДА содержит НАИБОЛЬШУЮ общую часть результата.

5. Параметр usig в запросах для проверки подлинности пользователя.
6. **obj_rgx** пока не реализован

1. Общие принципы

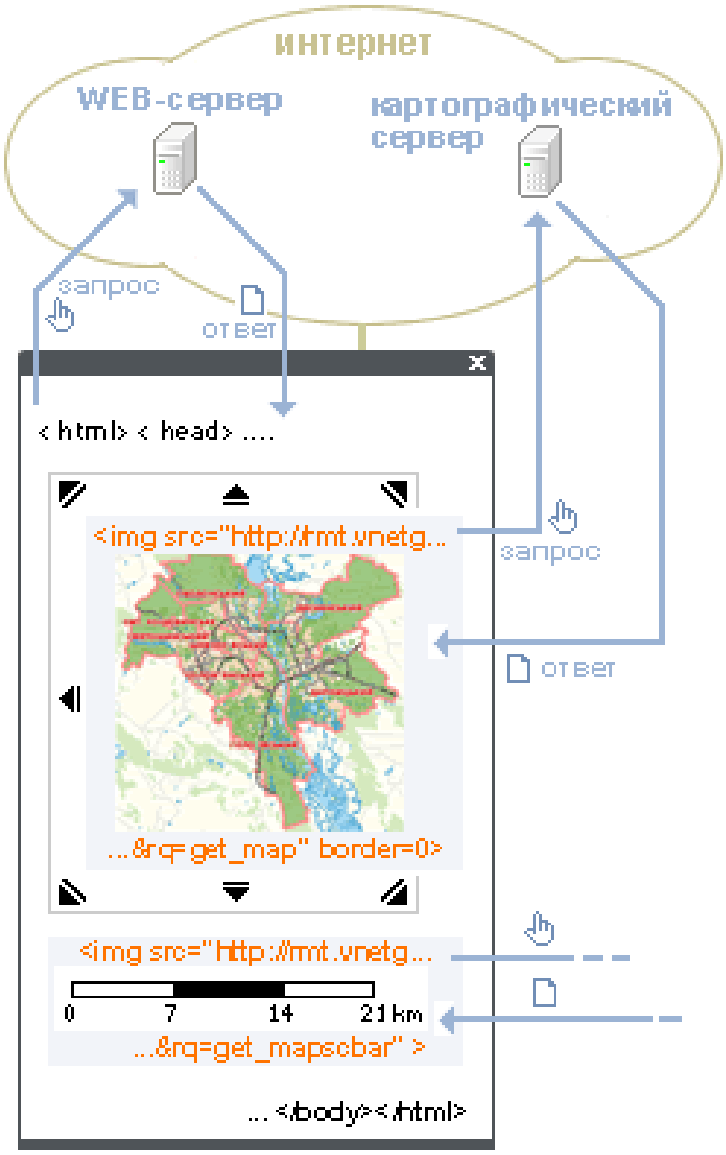

Рис. 1 Общая схема взаимодействия
Клиент - WEB-сервер – карт. Сервер

Упрощенное описание формирования «картографического интерфейса»:

1. Клиент указывает адрес страницы, содержащей «картографический интерфейс».
2. Браузер клиента выполняет HTTP-запрос к серверу и получает HTML-страницу. Страница содержит ссылки на различные объекты (картинки и др.)
3. Браузер загружает все необходимые графические объекты. Среди которых находится и ссылка на рисунок карты, картографический навигатор и масштабную линейку. В запросе содержатся все параметры, характеризующие создаваемый рисунок (масштаб, размер рисунка и т.д.)
4. Браузер пользователя отображает HTML-страницу содержащую рисунок участка карты, картографический навигатор и масштабную линейку.

Это очень упрощенная схема. Подобное «взаимодействие» не позволяет создать разграничение полномочий (права на доступ к карт. серверу и «виртуализацию» параметров конкретного пользователя).
В протокол взаимодействия с сервером построен на принципе «сеансов» («сессий»)
Для каждого нового клиента создается новая сессия, характеризующаяся уникальным идентификатором (в дальнейшем – UAMAP_CUID) и «шагом сессии» (ST). «Новый клиент» - пользователь, открывший страницу с картографическим интерфейсом на WEB-сервере с **вновь созданным** идентификатором UAMAP_CUID.
WEB-сервер управляет созданием UAMAP_CUID и контролирует правильность и уникальность идентификатора. Для «защищенных» серверов идентификатор создается по определенному закону (при этом на WEB-сервере необходимо установить дополнительное программное обеспечение), что **гарантирует** проверку прав доступа к данным ресурсам и защищенность каждого сеанса (если данные передаются по HTTPS)

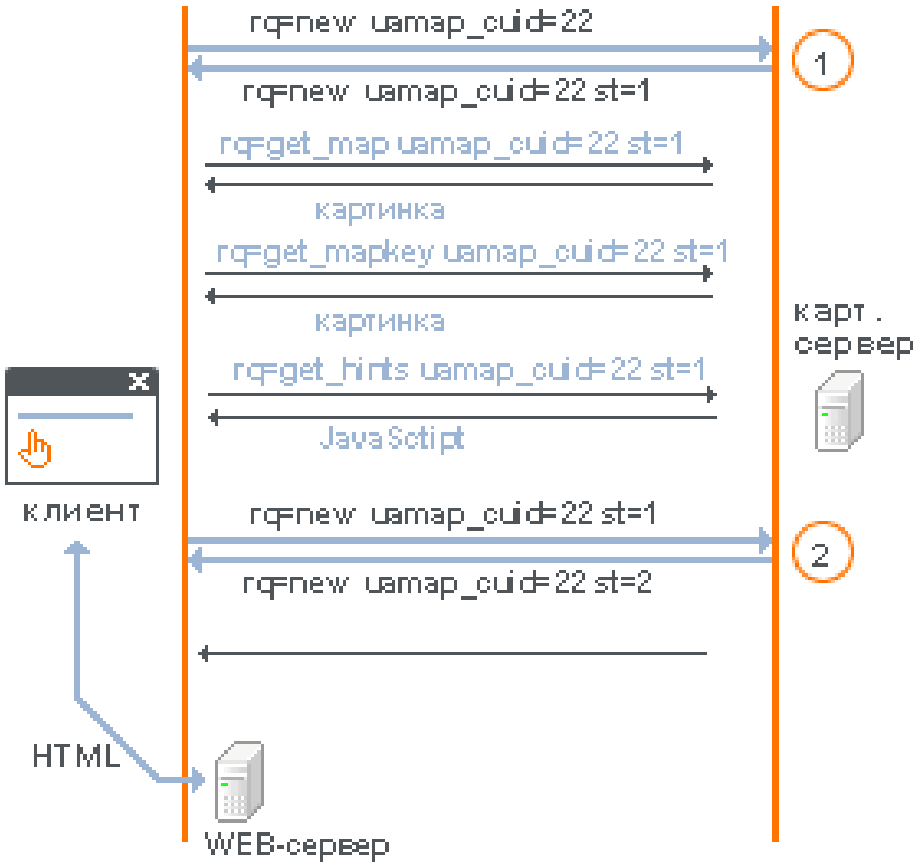

Рис 2. Взаимодействие между клиентом и сервером

Рассмотрим структуру взаимодействия между клиентом и сервером (рис.2):

1. Пользователь указывает в браузере URL страницы WEB-сервера с картографическим интерфейсом (впервые попадает на указанную страницу).
2. WEB-сервер формирует HTML-страницу (с использованием PHP/Pearl/ASP/…) с указанием уникального UAMAP_CUID (на рис. 2 он равен «22»).
3. После загрузки страницы (к примеру, по «событию» onLoad) выполняется **команда доступа** (**rq=new**) к серверу. Если **команда доступа** выполнился успешно - картографический сервер открывает сессию, возвращает общие параметры и значение ST=1(«шаг сессии»).

4. Браузер пользователя загружает HTML-страницу и начинает загрузку графических элементов страницы.
5. Среди графических элементов также находится «ссылка» на рисунок карты (результат выполнения **команды доступа**)

<img src=http://rmt.vnetgis.com?rq=get_map&uamap_cuid=22&st=1&l=ru&map=kiev>

и картографический навигатор

<img src=http://rmt.vnetgis.com?
rq=get_mapkey&uamap_cuid=22&st=1&l=ru&map=kiev>.

Если все параметры указаны правильно - картографический сервер вернет браузеру клиента рисунок карты и карт. навигатора.

6. Также можно в пределах одного «шага сессии» получить различную справочную информацию (выполнив **запрос данных**).

К примеру – «информационные подсказки»
(document.all.get_hints.src = 'http://rmt.vnetgis.com/?
rq=get_hints&uamap_cuid=22&st=1&map=kiev&lg=j&l=ua'). Сервер вернет массив в формате JavaScript.

7. Пользователь нажимает на «увеличить масштаб» («линк» на странице картографического интерфейса). При этом запрос (**команда доступа)** направляется на картографический сервер с указанием UAMAP_CUID и текущего значения ST. Далее повторяются действия, описанные в п.4 – 6.

Как только пользователь закрыл страницу или выполнил любое действие, приведшее к утрате правильного значения UAMAP_CUID и ST – создается новая сессия. Результат нескольких (**WEB**) последних «шагов сессии» хранится на сервере. Что позволяет получить результат (к примеру, рисунок карты) предыдущего запроса.

Если WEB-сервер не зарегистрирован на карт. сервере – запросы, поступившие от клиента, обработаны не будут.

2. Картографический сервис

Взаимодействовать с карт. сервером могут:
1. пользователи WEB-серверов (через браузер)
2. программные комплексы (формирующие запрос в общем формате)
3. «офисные» системы (написанные на VBA и работающие в офисных прогр. продуктах. К примеру, MS Word)
и др.

Все перечисленные категории пользователей можно условно разделить на **WEB-клиентов** и **программные комплексы**.

3.1. WEB-клиенты

Пользователи WEB, использующие картографический сервис на одном из WEB-серверов, взаимодействующем с карт. сервером. Как правило пользователь использует картографический интерфейс (с определенными сервисными функциями). Иногда, это просто изображение участка карты, сопровождающее аналитическую статью. При этом увеличить масштаб или изменить размер будет не возможно. На странице будет размещено изображение определенного участка карты.

Есть два принципиально различных способу взаимодействия:

- **«Переадресация»** - все **запросы** к WEB-серверу проходят через картографический сервер (для страниц, на которых размещены с карт. материалы).
- «**Запрос-ответ**» - **запрос** к картографическому серверу выполняет клиент (к примеру, программа на JavaScript)

У каждого способа есть как свои преимущества так и недостатки. Как правило, используют оба способа взаимодействия для одного картографического интерфейса.
В разделах 3.1.1. и 3.1.2. приведены упрощенные примеры доступа к карт. серверу. Полный перечень действующих примеров с исходными текстами доступен на http://demo.vnetgis.com/

3.1.1 «Переадресация» **(перезагружаемые интерфейсы)**

Рис. 3.

Пользователь выполнил «запрос» к WEB-серверу (submit). Все параметры запроса направляются на картографический сервер (рис. 3, шаг 1).

HTML:
<form action=http://rmt.vnetgis.com/ method=post>
<input type=hidden name=rq value=new>
<input type=hidden name=uamap_cuid value=22>
<input type=hidden name=st value=1>
<input type=hidden name=map value=kiev>
…
<input type=submit name=OK>
</form>

Картографический сервер обрабатывает запрос и переадресует запрос (указывая все необходимые параметры) на WEB-сервер. То есть, ответ направляется браузеру клиент, получив ответ браузер тут же отправляет запрос на WEB-сервер. При этом передаются все параметры исходного запроса (рис. 3, шаг 2). При переадресации используется метод GET (смотри Приложение 2. Описание HTTP-протокола), что накладывает ограничения на размер и количество передаваемых параметров (около 2 Кб).

WEB-сервер, получив запрос от клиента, формирует необходимую HTML-страницу и возвращает ее браузеру клиента (рис. 3, шаг 3). При этом в теле страницы присутствуют ссылки на рисунок карты, картографический навигатор и т.д. При формировании страницы WEB-сервер указывает полученные от карт. сервера (в переадресованном запросе) UAMAP_CUID и новое значения ST.

HTML:

```
<input type=image src="http://rmt.vnetgis.com/?
uamap_cuid=22&st=1&map=kiev&rq=get_map&l=" border=0 name=map width=400
height=390>
<img SRC="http://rmt.vnetgis.com/?uamap_cuid=22&st=2&map=kiev&rq=get_mapscbar&l="
border=0  name=uamap alt="">
<input type=image src="http://rmt.vnetgis.com/?
uamap_cuid=22&st=2&map=kiev&rq=get_mapkey&l=" border=0 width=100 height=98
name=key_uamap>
```

При первой загрузке страницы довольно сложно выполнить **команду доступа** (rq=new), а лишь потом – запрашивать картинки. По этому допускается выполнение запроса на получение графических элементов карты без открытия сессии. При этом необходимо корректно указать UAMAP_CUID и ST (=1). Сессия откроется при первом запросе к серверу с корректными значениями. Параметры отображения карты (указанные в сессии) будут соответствовать начальным значениям (**WEB**).

3.1.2. «Запрос-ответ» **(не перезагружаемые интерфейсы)**

Клиент формирует запрос – сервер возвращает результат в нужном формате. Подобная схема взаимодействия, являющаяся единственно возможной для **программных комплексы**, довольно сложно реализуется в WEB-приложений. Вся логика взаимодействия должна быть реализована на скриптовом языке (JavaScript, VBScript) в рамках картографического интерфейса.

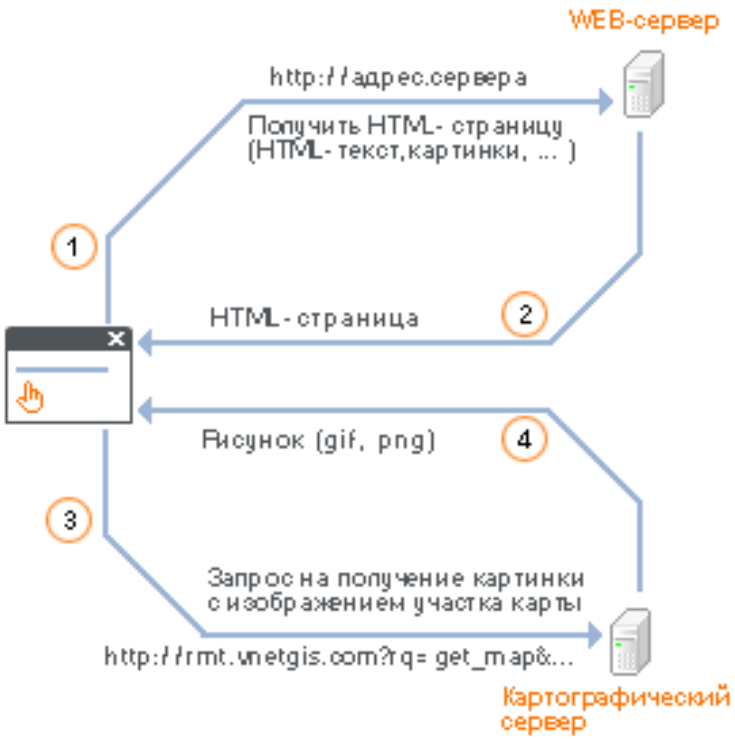

Рис. 4.

Пользователь выполнил «запрос» к WEB-серверу (рис.4, шаг 1). Загрузился HTML-код страницы (рис.4, шаг 2). Далее браузер обрабатывает полученный код и выполняет JavaScript-код указанный в теге <script language="JavaScript">…..</script>.
В начале страницы необходимо выполнить **команду доступа** (rq=new)

HTML:

```
<script language="JavaScript">
var uamap_cuid=22;
var cmd=”recenter”;
var cmd_param=””;
var st=1;
var Lang=”ru”;

if(isIE) document.all.get_jnew.src='http://rmt.vnetgis.com/?uamap_cuid='+uamap_cuid+
'&map=kiev&rq=new&log=0&cmd='+cmd+'&cmd_param='+cmd_param+
'&lg=j&st='+st+'&l='+Lang;

else document.layers['get_jnew1'].src='http://rmt.vnetgis.com/?uamap_cuid='+uamap_cuid+
'&map=kiev&rq=new&log=0&cmd='+cmd+'&cmd_param='+cmd_param+
'&lg=h&st='+st+'&l='+Lang;
</script>
```

В данном случаи результат запроса не анализируется. Важен сам факт выполнения запроса.
Далее необходимо получить результат выполнения запроса (рис.4, шаг 1 - 2).

HTML:

```
<img src="../img/1.gif" border=0 name="map" width=400 height=390>
<img src="../img/1.gif" border=0 name="map_scbar">
<img src="../img/1.gif" border=0 width=100 height=98 name="nav">

<script language="JavaScript">
document.images['map'].src=
'http://rmt.vnetgis.com/?uamap_cuid='+uamap_cuid+'&map=kiev&rq=get_map&st='+st;

document.images['nav'].src=
'http://rmt.vnetgis.com/?uamap_cuid='+uamap_cuid+'&map=kiev&rq=get_mapkey&st='+st;

document.images['map_scbar'].src=
'http://rmt.vnetgis.com/?uamap_cuid='+uamap_cuid+'&map=kiev&rq=get_mapscbar&st='+st;
</script>
```

Взаимодействие между клиентом и WEB-сервером отделено от взаимодействия между картографическим сервером и клиентом. Что позволяет создавать «не перезагружаемые» интерфейсы. При этом HTML-страница загружается один раз, а при выполнении изменения масштаба (к примеру) выполняется лишь запрос карт. серверу.

3.2. Программные комплексы

Очень часто при создании различных программных комплектов возникает потребность в доступе к пространственным данным. К примеру, для создания диспетчерской системы

автомобильного транспорта необходимы карты определенного региона. Программное обеспечение может без труда получить доступ к картографическому серверу (также по протоколу HTTP). В основном взаимодействие проходит по стандартной схеме «**Запрос-ответ**», описанной в п. 3.1.1.

4. Параметры запроса.

Запрос передается по протоколу HTTP с использованием метода POST или GET (смотри Приложение 2). Значения и названия параметров не зависит от метода.

Общий перечень параметров запроса к картографическому серверу

№	Описание	Название	Тип	Обяза-тельный	Примечание
1.	Идентификатор сессии	uamap_cuid	строка	да	
2.	Шаг сессии	st	число	да	
3.	Системное название карты	map	строка	да	
4.	Тип запроса	rq	строка	да	
5.	Название параметра управления	cmd		нет	Используется если rq=new
6.	Значение параметра управления	cmd_param		нет	Используется если rq=new
7.	Записывать лог-информацию о выполнении запроса	log	число	нет	0 или 1-ца
8.	Формат результата запроса	lg	символ	нет	
9.	Язык, на котором будет сформирован результат	l	строка	нет	Украинский/ русский/ английский
10.	Адрес WEB-сервера, на котором расположен интерфейс	host		нет	лучше не использовать
11.	Установить фиксированный размер рисунка карты	set_mapsize		нет	Запрос к препроцессору
12.	Сигнатура	`usig`	строка	В зависимости от настроек	Сигнатура для проверки подлинности пользователя
13.	Желаемая кодировка	`uchset`	строка	Не обязательный	По умолчанию кодировка win1251. Для того чтобы трактовать url и отдавать результат в utf8 добавьте в запрос `uchset=utf8`
14.	Callback-функция для протокола JSONP (lg=p)	`jsonp`	строка	Не обязательный	

Существует 2-ве «пары» специальных параметров:

key_uamap_x, key_uamap_y (или key_uamap.x, key_uamap.y) – координаты нажатия на рисунке карты
uamap_x, uamap_y (или uamap.x, uamap.y) – координаты нажатия на рисунке карты
Одновременно может указываться лишь одна «пара». Параметры передаются месте с некоторыми командами доступа.

Также, для «управляемых слоев» (**WEB**) необходимо указать их состояние (включен/выключен). По умолчанию, если имя слоя (системное) отсутствует в списке параметров – слой выключен. Если слой включен необходимо указать параметр (имя слоя) =1.
Пример: **&house=1&base=1&** или **&house=on&base=on&**

4.1. Идентификатор сессии (uamap_cuid).

Идентификатор сессии – уникальная строка, сформированная по определенному закону. Для «общедоступных» картографических баз данных WEB-сервер формирует произвольную уникальную последовательность символов (не больше 16-ти символов). Для «защищенных» - на WEB-сервере должен быть установлен программный модуль, создающий идентификатор по определенному закону. Что позволяет на сервере гарантировано идентифицировать пользователя и проверить права доступа.

Пример: uamap_cuid=3e638fc73a230

4.2. Шаг сессии (st).

1	Сессия			
2	Шаг 1	Шаг 2	Шаг 3	…
3	**rq=new** cmd=…	**rq=new** cmd=…	**rq=new** cmd=…	
4	rq=get_map rq=get_mapkey …	rq=get_map rq=get_mapkey …	rq=get_map rq=get_mapkey …	

1. Все запросы выполняются в пределах одной сессии
2. Одна сессия содержит произвольное количество шагов
3. Для одного «шага сессии» допустима только одна команда доступа (rq=new). Точнее, выполнение команды доступа приводит к формированию результата запроса и возврату пользователю номера шага, содержащего результат запроса.
4. Для одного «шага сессии» можно получить результирующие даны не ограниченное количество раз (изображение карты, картографического навигатора, масштабной линейки и т.д.).

Возможно получить результат нескольких предыдущих шагов. Количество шагов, для которых сервер хранит данные настраивается (**WEB**).

Пример: st=12

4.3. Системное название карты (map).

Указывает к какой именно картографической базе данных обращается пользователь. Если пользователь не имеет доступа к указанной карте – запрос будет отвергнут.

Пример: map=kiev

4.4. Тип запроса (rq).

Определяет тип запроса: команда доступа (=new) или получение результата (get_map, get_mapkey, get_mapscbar, get_hints, get_state, get_profile, get_pkg, get_maplist)

Запрос, без указания **rq** будет отвергнут сервером.

Пример: rq=new

4.5. Название и значение параметра управления (cmd, cmd_param).

Указывает какое именно действие, необходимо выполнить по команде доступа. Параметр указывается только в для команды доступа. Без указания **cmd** запрос будет отвергнут. Все допустимые названия параметра управления перечислены в п.5.

Пример: cmd=set_zoom_size&cmd_param=2

4.6. Записывать лог-информацию о выполнении запроса (log).

При разработке и отладке нового картографического интерфейса иногда необходимо узнать, как именно картографический сервер обработал запрос и уточнить возникшие ошибки. Укажите данный параметр для формирования сервером LOG-информации о выполнении запроса. В дальнейшем, полученную информацию можно просмотреть, воспользовавшись интерфейсом «Администратора виртуального сервера» (и отправить в службу поддержки).

Внимание! После окончания разработки уберите данный параметр (или укажите **log=0**) чтоб не создавать излишней нагрузки на сервер.

Пример: log=1

4.7. Формат результата запроса (lg).

Указывает, в каком именно формате необходимо вернуть результат запроса (для команды доступа и запроса «получение результата»).
Возможные значения:
t – текстовый формат
j – в формате JavaScript
v - в формате ActivScript (Flash)
d – HTTP (по умолчанию)
i – возвратить рисунок участка карты
p – в формате JSON (JSONP)

Структура и формат для каждого типа запросов детально описаны в п.5-6

Пример: lg=t

4.8. Язык, на котором будет сформирован результат (l).

Для каждого картографического банка данных текстовая информация может хранится на украинском, русском и английском языках. При выполнении команды доступа можно указать (cmd=lsearch), на каком именно языке указаны данные для поиска. А при получении результата (к примеру, рисунка карты) на каком языке должны быть текстовые надписи.
Допустимые значения:
ua – украинский язык (по умолчанию)
ru - русский язык
en – английский язык

Пример: l=ru

4.9. Адрес WEB-сервера, на котором расположен интерфейс (host).

Необходим для правильной идентификации пользователя. Если карт. сервер не получил от клиента адрес WEB-страницы, на которой расположен интерфейс, запрос будет отвергнут (referer). Для некоторых запросов (rq = get_map / get_mapkey / get_mapscbar) допустимо указание URL-страницы. Если referer, с указанием адреса картографического интерфейса, отсутствует – будет использован адрес, указанный в **host.**

4.10. Препроцессор для команд доступа

Предположим, необходимо изменить масштаб и изменить размер рисунка карты. Для этого необходимо выполнить две команды доступа:

- изменить масштаб (**cmd=zoom_in**)

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&**cmd=zoom_in&cmd_param=232,100**&st=3&l=ru

- изменить размер рисунка (**cmd=set_mapsize**)

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&**cmd=set_mapsize&cmd_param=2**&st=3&l=ru

Но для этого пользователю придется дважды «перегрузить страницу» (для перезагружаемых интерфейсов) ! А это - дополнительные не удобства для пользователя и неоправданная нагрузка на карт. сервер.

Для решения подобных проблем существует «**препроцессор**». Запросы к препроцессору обрабатываются перед выполнением команды доступа. Что позволяет перед выполнением основной команды (**cmd**), изменить значения ключевых параметров, влияющих на результат.

4.10.1. Установить фиксированный размер рисунка карты (set_mapsize).

Позволяет перед выполнением команды доступа, указать фиксированный индекс рисунка карты. Не допустим для следующих параметров запроса:

- **cmd=zoom_in**
- **cmd=lsearch**

Пример: http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&cmd=zoom_out&cmd_param=232,100&**set_mapsize=2**&st=3&l=ru

4.10.2. Управление отображением объектов слоя (obj_rgx).

Если необходимо выполнить предварительную фильтрацию элементов слоя, перед его визуализацией – укажите параметр:
obj_rgx=[название слоя]!![текстовый фильтр]
(описание правила формирования текстовых фильтров – «Руководство администратора»)

Пример: http://rmt.vnetgis.com/?
uamap_cuid=3e638fc73a230&map=kiev&rq=new&cmd=zoom_out&
cmd_param=232,100&**obj_rgx=_auto!!/^32|^78/**&st=3&l=ru

(Во второй версии пока не реализовано)

4.11. Сигнатура (usig)

Сигнатура нужна для проверки подлинности пользовательского сайта. Обязательно или нет использовать сигнатуру, регулируется в настройках пользователя на сервере.
Чтобы сформировать сигнатуру, добавьте параметр usig в запрос. Значение параметра представляет собой строку длинной 16 символов:
<8 символов время сигнатуры><8 символов подпись>
Т.е. фактически строка сигнатуры – это 2 числа в HEX формате.

Время сигнатуры – время, с которым формировалась подпись. Это число, которое возвращает функция time() – количество секунд с 1 Января 1970 г, по UTC. Сигнатура имеет ограниченный срок действия (по умолчанию 2 часа), после этого времени сигнатура считается некорректной.
<время сигнатуры>=hex(time(0))

Подпись – используется для проверки пользователя.
<подпись>=hex(crc32(to_lower(md5(<секретный пароль>+uamap_cuid+map+<время сигнатуры>))));

Секретный пароль – пароль, который хранится на картографическом сервере в настройках пользователя и на пользовательском сайте. !!!Не размещайте пароль в коде HTML страницы с картой.

4.12. Callback функция для протокола JSON (jsonp)

В формате JSON (lg=p) результат отдаётся ввиде javascript объекта. Если указать параметр `jsonp=<имя пользовательской функции>` ,то результат вернётся ввиде `<имя пользовательской функции>(<javascript объект>)`

5. Команды доступа

Все операции по созданию рисунка карты и формированию информационных данных выполняются исключительно по команде доступа. В ответ на команду, сервер взращает результат (см. пункт 5.1.) в формате, определяемом параметром **lg**. (см. пункт 4.8.) В составе данных всегда (кроме **lg=1**) находится новое значение «шага сессии». Следующую команду необходимо выполнить именно с указанным значением **st**.
После успешного завершения команды доступа, можно выполнить любой **запрос данных** (см. пункт 6.), если указанная команда предполагает их наличие. К примеру, после выполнения команды с параметром (cmd) lsearch – данные не формируются.

При выполнении запроса необходимо указать шаг сессии, полученный после выполнения команды доступа (текучий «шаг сессии»). Количество запросов не ограничено, но нужно помнить, что на картографическом сервере хранятся данные лишь для ограниченного количество «шагов» (по умолчанию – 2-вух). К примеру, текущий st=9. Но всегда можно, указав st=8, получить данные предыдущей команды доступа (**WEB**)

5.1. Результат команды доступа

Пользователь выполнил команду доступа. Кто именно подразумевается под пользователем ? Как ранее указывалось пользователей можно условно разделить на **WEB-клиентов** и **программные комплексы** (см п. 3.). Причем категория **WEB-клиентов** также подразделяется на 2-ва класса, в зависимости от принципа «взаимодействия» с сервером (см п. 3.1). Если при взаимодействии «**Запрос-ответ»** формат результата, как правило, влияет лишь на удобство его анализа/обработки, то в случаи **«Переадресации»** - именно возвращаемый результат и выполняет переадресацию.
К примеру, WEB-клиент направил запрос (команду доступа) на картографический сервер. Сервер его обработал и вернул результат клиенту в формате (п. 4.7, **lg=d**):

Location: http://www.имя.сервера/map?uamap_cuid=3e6&map=kiev&rq=new&
cmd=zoom_out&cmd_param=232,100&st=3

Что, собственно, и инициирует выполнение запроса к WEB-серверу с указанием необходимых параметров в запросе (см. также 5.2.) И другой формат просто не позволит нормально функционировать «картографическому интерфейсу».
Но для анализа в **программном комплексе** подобная структура не очень удобна. А для «информационных подсказок» и вовсе не приемлема. По этому существует несколько форматов представления результата. Они перечислены в п. 4.7. Рассмотрим более детально структуру каждого из форматов.

lg=t - текстовый формат**:**
[название параметра]=[значение]**\n**
…

Пример:
uamap_cuid=3e638fc73a230
l=ru
map=kiev
index_scale=3
map_pix_size=400,390
index_img_size=2

scale=170000&st=2
mc=59.9721898372&
{имя слоя=1}

lg=j - в формате JavaScript**:**
Представляет собой блок программного кода на языке JavaScript.
var [название параметра]=’[значение]’;

Пример:

var uamap_cuid=’3e638fc73a230’;
var l=’ru’;
var map=’kiev’;
var index_scale=3;
var map_pix_size=new Array(400,390);
// var map_pix_size=’400,390’; в версии 1.x
var index_img_size=2;
var scale=170000;
var st=2;
//var mc=’59.9721898372’; в версии 1.x

var {имя слоя=1};

lg=j - в формате JSONP**:**
Представляет собой объект на языке JavaScript или вызов пользовательской функции с передачей этого объекта в качестве параметра.

Пример:

http://rmt.v2.vnetgis.com//?
rq=new&cmd=zoom_out&cmd_param=232,100&st=0&map=kiev&lg=p&l=ua&uamap_cuid=0976c&jsonp=MyFunction

MyFunction({"index_img_size":1,"index_scale":1,"l":"ua","map":"kiev","map_pix_size":[400,390],"scale":110.0,"st":1,"uamap_cuid":"0976c","vnetGroupState":{},"vnetLayerState":{"metro":0,"monum":0,"roadd":0,"roadd_vertexes":0,"roadt":0,"roadt_vertexes":0}}

v - в формате ActivScript (Flash)
Возвращает результат, в формате наиболее приемлемом для анализа в Flash-приложениях
[название параметра]=[значение]&…

Пример:

uamap_cuid=3e638fc73a230&l=ru&map=kiev&index_scale=3&map_pix_size=400,390
&index_img_size=2&scale=170000&st=2&mc=59.9721898372&
{имя слоя=1}

d – HTTP (по умолчанию)
По мимо результата, содержит команду «переадресации» и полный адрес картографического интерфейса (**WEB**).

Пример:

Location: http://www.имя.сервера/map?
uamap_cuid=3e638fc73a230&l=ru&map=kiev&index_scale=3&map_pix_size=400,390
&index_img_size=2&scale=170000&st=2&mc=59.9721898372&
{имя слоя=1}

i – возвратить рисунок участка карты
После выполнения команды, возвращается сформированный «рисунок карты». Что позволяет встраивать изображение карты прямо в различные информационные статьи и т.д. (без полноценного картографического интерфейса)

Пример:
Смотри пример на http://demo.vnetgis.com/.ua/other_rqnew_php/

2.1. Трансляция переменных

Все пользовательские переменные (содержащиеся в form), передаваемые при запросе картографическому серверу (тип взаимодействия - «Переадресация»), возвращаются клиенту. Предположим, картографическому интерфейсу (HTML-страница, загруженная в браузере у клиента) необходимо передать на WEB-сервер (с которого и была загружена страница), при нажатии на кнопке «увеличить масштаб», определенные параметры (по мимо параметров, необходимых картографическому серверу). А запрос-то адресован картографическому серверу:

<form action=http://rmt.vnetgis.com>
<input type=hidden name=st value=3>
…
<input type=hidden name=”моя_переменная_1” value=”значение”>
</form>

Для решения подобных проблем, картографический сервер выполняет трансляцию переменных, указанных в запросе. Все переменные, не «адресованные» карт. серверу, передаются WEB-серверу в исходном виде.
Переменные адресованные карт. серверу анализируются и частично изменяются.
Также дополнительно передаются переменные:
index_scale - индекс масштаба карты (фиксированное значение). Для команд устанавливающих произвольное значение масштаба указывается ближайшее значение в "линейке" масштабных значений.
scale - реальное значение масштаба
map_pix_size - размер рисунка в пикселях (формат - [ширина, высота])
index_img_size - индекс размера рисунка

Транслируются все переменные кроме зарезервированных: `rq, lr, bn, lg, key_uamap_x, key_uamap_y, uamap_x, uamap_y, rmt_id, set_mapsize, key_uamap.x, key_uamap.y, uamap.x, uamap.y, rmt_id, usig. Часть из этих переменных может заменяться новыми значениями в зависимости от запроса`

5.1. Перечень допустимых параметров управления

Если не указано дополнительно, результат команды доступа полностью соответствует формату, указанному в п. 5.1. Полный перечень команд управления:

- set_mapsize
- set_zoom_scl
- fixed_mov
- zoom_in
- zoom_out
- key_recenter
- key_recenter_scl
- key_recenter_fix_scl
- recenter
- recenter_scl
- recenter_fix_scl
- recenter_geo
- view_region
- view_region_geo
- redraw
- recenter_fix_scl_geo
- msearch
- lsearch

Полное описание параметров управления и пример использования, приведены в **Приложении 1. Параметры управления и их использование**

Внимание! Параметры msearch и lsearch описаны в разделе 5.2.

5.2. Параметр управления msearch и lsearch

Существует два специализированных параметра управления, предназначенных для поиска в пространственной базе данных. Если необходимо найти и отобразить на карте определенный объект – используйте **msearch**.
Для поиска и формирования результата в виде списка найденных объектов – воспользуйтесь **lsearch**.
У обоих запросов формат «значения параметра управления» идентичен.

[**системное название слоя**][**тип переменно**][**переменная**][**условие**]
[системное название слоя][тип переменно][переменная][условие]....

где:
системное название слоя - указывает название слоя, по которому будет осуществлен поиск.

тип переменной - определяет, к какому именно типу относится указанная «переменная».
Допустимые значения:
:: - ID объекта
!! – название объекта (или несколько символов из названия)

переменная - число или текст, который необходимо найти

Пример:

cmd=msearch
cmd_param=school::1234

условие - определяет условие, ??????????????????
Допустимые значения:
$ - логическое **И**
% - логическое **ИЛИ**

После выполнения команды доступа, будет сформирован рисунок участка карты с размещенным по центру объектом (в данном случаи – школой с ID=1234).

5.2.1. Параметр управления msearch

Позволяет найти и отобразить на карте необходимые объекты. При этом как именно будет отображена карта (масштаб, размер, выделение найденных объектов и др.) – настраивает «администратор картографического сервера» (**WEB**)
Для слоя, принадлежащего пользователю, под ID подразумевается идентификатор, указанный при занесении объекта в базу (**WEB** «ID в базе пользователя»)

К примеру, пользователю принадлежит слой с системным названием _aa («Автомобильные салоны»). Ему необходимо получить рисунок карты с «найденным» автосалоном с ID (в базе пользователя) 123456789. Для этого необходимо выполнить команду доступа:

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&
cmd=msearch&cmd_param=_aa::123456789&st=3&l=ru

Внимание! В данной реализации есть определенные ограничения:

- допустимо применение только поиска по ID объекта
- реализован поиск только по указанному слою (запрос вида gg::12+hh::34 не поддерживается)

5.2.2. **Параметр управления** lsearch

Позволяет выполнить поиск в базе данных картографического сервера и возвращает список найденных объектов и дополнительную информацию. Как правило, команда доступа с параметром **lsearch** применяется для поиска данных в слоях, не принадлежащих пользователю. К прмеру, в состав карты входит слой «станции метро». Указанный слой является «собственностью» картографического сервера, и не может быть передан администратору WEB-сервера для организации собственного поискового интерфейса, по объектам слоя. А пользователь, к тому же, не знает точного названия станции. Ему необходимо найти станцию метро, содержащую в названии «**площадь**». Подобных станций может быть несколько. Следовательно, необходимо выполнить запрос к картографическому серверу и получить все станции метро, содержащие в названии «**площадь**», и их ID.

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&
cmd=lsearch&cmd_param=metro!!площадь&st=3&l=ru

Внимание! Поиск будет выполнен по русскоязычным названиям (в запросе указан параметр **l=ru**)

Результат выполнения команды доступа, для cmd=**lsearch**, полностью отличается от формата, описанного в п. 5.1. После выполнения команды, сервер возвратит (для указанного выше запроса):

http://адрес.сервера/страницы.с.результатом.поиска/?l=ru**&txt=площадь&**
uamap_cuid=3e6cb43ca9d39&st=1&map=kiev**&cmd_param=metro!!площадь&**
cnt=4&bn=1&bs=10&lr=metro&find=8783:Площадь+Льва+Толстого,
8786:Почтовая+площадь,8776:Площадь+Независимости,
8770:Контрактовая+площадь

Результат (по умолчанию) содержит:
txt – искомый текст или число
cnt – количество найденных (в базе данных) записей
bn – номер «блока результата поиска»
bs – размер «блока результата поиска»
lr – название слоя, по которому выполнен поиск
find – результат поиска, в формате:
[ID]:[Название объекта],[ID]:[Название объекта], …

Может содержать:
prefix – общий текст, добавляемый ко всем названиям найденных объектов.

Полученный результат обрабатывается программой на WEB-сервере и формируется страница для браузера клиента, содержащая список найденных объектов и навигацию по «блокам результата поиска» (смотри п. **3.1.1.**).

Блок результата поиска – часть списка найденных, по запросу, объектов. Предположим, в результате поиска найдено 340 объектов. Весь перечень целиком не может быть передан из-за ограничения на размер **GET**-запроса – около 2 Кб (смотри Приложение 2. Краткое описание протокола **HTTP/1.1**) Кроме того, на WEB-сервере все равно необходимо выполнить «разбивку» результата на страницы (подобный механизм реализован во всех поисковых системах). По проще выполнить разбивку на картографическом сервере, а в запросе (команде доступа) указывать номер нужного «блока». В результате запроса, картографический сервер возвращает общее количество найденных объектов (**cnt**), размер «блока» (**bs - WEB**) и номер «бока».

Чтоб получить 2-й «блок», при поиске всех станций метро, содержащих в названии букву «**a**», необходимо выполнить запрос:

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&
cmd=lsearch&cmd_param=metro!!a&st=3&l=ru&**bn=2**

Результат запроса:

http://адрес.сервера/страницы.с.результатом.поиска/?st=1&map=kiev&
uamap_cuid=3e6cb43ca9d39&log=1&l=ru&txt=a&cmd_param=metro!!a&cnt=30&**bn=2**&
bs=10&lr=metro&find=8793:Крещатик,8791:Тараса+Шевченко,8790:Театральная,
8789:Республиканский+стадион,8788:Дворец+Спорта,8787:Дворец+Украина,
8786:Почтовая+площадь,8782:Петровка,8781:Печерская,8777:Минская

Во всех рассмотренных ранее примерах, приведены результаты запроса без указания формата (по умолчанию – **lg=d**). Для получения результата в другом формате – нужно указать соответствующее значение **lg**:

lg=t
Для «атомарных» переменных:
[переменная]=[значение]**\n**
для массива:
[переменная]{**\n**
[ID 1]:[значение 1]**\n**
[ID 1]:[значение 2]**\n**
...
}[переменная]

Всегда указаны переменные: **cnt, bs, lr**
Также может присутствовать: **prefix**

Пример:

cnt=30**\n**
bs=10**\n**
lr=metro**\n**
find{**\n**
8793:Крещатик**\n**
8791:Тараса Шевченко**\n**
8790:Театральная**\n**
8789:Республиканский стадион**\n**
8788:Дворец Спорта**\n**

8787:Дворец Украина\n
8786:Почтовая площадь\n
8782:Петровка\n
8781:Печерская\n
8777:Минская\n
}find

lg=j
Для «атомарных» переменных:
var [переменная]=”[значение]”;
для массива:
var [переменная]_id = new Array([ID 1],[ID 1],…);
var [переменная]_name = new Array(“[значение 1]”,”[значение 2]”,…);

Пример:

var cnt=”30”;
var bs=”10”;
var lr=”metro”;
var find_id = new Array(8793,8791,8790,8789,8788,8787,8786,8782,8781,8777);
var find_name = new Array(“Крещатик“,“Тараса Шевченко“,“Театральная“,“Республиканский стадион“,“Дворец Спорта“,“Дворец Украина“,“Почтовая площадь“,“Петровка“,“Печерская“,“Минская“);

lg=p
Тоже, что и для lg=j только упакованное в один объект.

Пример:

http://rmt.v2.vnetgis.com/?rq=new&cmd=lsearch&cmd_param=metro!!%E0&st=3&l=ru&map=kiev&lg=p&uamap_cuid=0976c&jsonp=MyFunction

MyFunction({"bn":1,"bs":20,"cnt":35,"find_id":[64905,8797,8760,84936,8761,84935,8795,8796,8768,8787,8788,8767,64906,8765,8769,8770,84937,8793,8772,8775],"find_name":["Академгородок","Арсенальна,"Берестейска,"Бориспольска,"Вокзальна,"Вырлица","Героев Днепра","Гидропарк","Дарница","Дворец \"Украина\"","Дворец спорта","Др Народов","Житомирска,"Золотые Ворота","Кловска,"Контрактовая площадь","Красный хутор","Крещатик","Левобережна,"Лесова],"index_img_size":1,"index_scale":1,"l":"ru","lr":"metro","map":"kiev","map_pix_size":[400,390],"scale":110.0,"st":4,"uamap_cuid":"0976c","vnetGroupState":{},"vnetLayerState":{"metro":0,"monum":0,"roadd":0,"roadd_vertexes":0,"roadt":0,"roadt_vertexes":0}}

5.3. Поиск по «адресному реестру» (параметр cmd=lsearch)

Некоторые базы данных крупных городов содержат «адресный реестр». Он состоит из 3-х слоев:

- районы города (площадной слой, системное название - **district**)
- улицы (линейный, системное название - **strt**)
- дома (точечный, системное название - **address**)

Указанные слои, при отображении на карте, ничем не отличаются от остальных слоев.

Единственное отличие – «логическая» связь между указанными слоями при поиске объекта (дома с указанным адресом).
Рассмотрим некоторые возможные запросы.

Нужно найти:

1. район, содержащий в названии “**днеп**”
cmd=lsearch&cmd_param=**district**!!днеп

2. район, с ID=2 (проверить существование указного ID, по ID получить его полное название и прочее)
cmd=lsearch&cmd_param=**district**::2

3. улицу, содержащую в названии “**зел**”
cmd=lsearch&cmd_param=strt!!зел

4. получить список адресов, расположенных на улице с ID=1221
cmd=lsearch&cmd_param=strt::1221

5. адрес, с ID=1441 (проверить существование указного ID, по ID получить номер дома и название улицы)
cmd=lsearch&cmd_param=**address**::1441

6. найти все адреса с номером **12**, содержащие в названии улиц, на которых они расположены, текст «**про**»
cmd=lsearch&cmd_param=**strt!!про$address!!12**

Как не сложно заметить, в 4-м примере, вместо полного названия улицы картографический сервер возвращает список адресов, расположенных на указанной улице (по мимо ее полного названия). При этом полное название содержится в параметре **prefix**, так при формировании результата поиска должно добавляется к каждому адресу.

В версии 2.х даже в таком запросе формируется наибольшая общая часть. Для того чтобы отделить улицу от домов используйте разделитель между ними (по умолчанию *)

Рассмотрим результат 6-го запроса. Сервер возвращает (по умолчанию, lg=d):

http://адрес.сервера/страницы.с.результатом.поиска/?st=1&map=kiev&
uamap_cuid=3e6cb43ca9d39&l=ru&txt=про,12&cmd_param=strt!!про$address!!12&
cnt=20&bn=1&bs=30&lr=address&find=28832:просп.+50-летия+Октября.*12,
23893:просп.+Владимира+Маяковского*12,38331:просп.+Воссоединения*12,
34338:просп.+Василия+Порика*12,32072:просп.+Космонавта+Комарова*12,
24358:просп.+Мира*12,20271:просп.+Победы*12,5348:просп.+Павла+Тычины*12,
30003:просп.+Радянской+Украины*12,42196:просп.+Героев+Сталинграда*12,
29052:просп.+Академика+Королева*12,27017:просп.+Академика+Глушкова*12,
9183:Червонозоряный+просп.*12,1478:Днепровый+пер.*12,7210:Минский+просп.*12,
32052:Отрадный+просп.*12,4928:Промысловая*12,29458:Прорезная*12,
30494:Профессора+Подвысоцкого*12,24792:Профсоюзная*12

Название улицы отделено от номера дома “*”-кой. При отображении ее можно заменить на любой нужный символ. К примеру – “№”.

6. Запрос данных

После выполнения команды доступа, необходимо получить «результат запроса». Для любого значения **cmd** (кроме **cmd=lsearch**), можно получить:

get_map – рисунок «карты» (rq=**get_map**)
get_mapkey – рисунок «картографического навигатора» (rq=**get_mapkey**)
get_mapscbar – рисунок «масштабной линейки» (rq=**get_mapscbar**)
get_hints – «информационные подсказки» (rq=**get_hints**)
get_state – дополнительная информация о текущем «шаге сессии» (rq=**get_state**)

6.1. Получить рисунок «карты» (rq=get_map)

Для получения рисунка карты, необходимо указать «идентификатор сессии», «шаг сессии», системное название карты и «язык» (на котором должны быть сформированы надписи на карте)

Если:
идентификатор сессии (uamap_cuid) = 3e6decdcb93da
шаг сессии (st) = 2
язык (l) = ua
то для отображения рисунка на HTML-странице необходимо указать:

<img src="**http://rmt.vnetgis.com/?uamap_cuid=3e6decdcb93da&st=2&map=kiev&rq=get_map&l=ua**">

В результате запроса сервер возвратить рисунок в PNG-формате (**WEB**)

6.2. Получить рисунок «картографического навигатора» (rq=get_mapkey)

Для получения рисунка карты, необходимо указать «идентификатор сессии», «шаг сессии», системное название карты и «язык» (на котором должны быть сформированы надписи на карте)

Если:
идентификатор сессии (uamap_cuid) = 3e6decdcb93da
шаг сессии (st) = 2
язык (l) = ua
то для отображения рисунка на HTML-странице необходимо указать:

<img src="**http://rmt.vnetgis.com/?uamap_cuid=3e6decdcb93da&st=2&map=kiev&rq=get_mapkey&l=ua**">

В результате запроса сервер возвратить рисунок в PNG-формате (**WEB**)

6.3. Получить рисунок «масштабной линейки» (rq=get_mapscbar)

Для получения рисунка карты, необходимо указать «идентификатор сессии», «шаг сессии», системное название карты и «язык» (на котором должны быть сформированы надписи на карте)

Если:
идентификатор сессии (uamap_cuid) = 3e6decdcb93da
шаг сессии (st) = 2
язык (l) = ua
то для отображения рисунка на HTML-странице необходимо указать:

<img src="**http://rmt.vnetgis.com/?uamap_cuid=3e6decdcb93da&st=2&map=kiev&rq=get_mapscbar&l=ua**">

В результате запроса сервер возвратить рисунок в PNG-формате (**WEB**)

6.4. Получить «информационные подсказки» (rq=get_hints)

Для получения информационных подсказок необходимо указать «идентификатор сессии», «шаг сессии», системное название карты, формат результата запроса и «язык» (на котором должны быть сформированы надписи на карте)

Если:
идентификатор сессии (uamap_cuid) = 3e6decdcb93da
шаг сессии (st) = 2
язык (l) = ua
формат результата запроса (lg)=j
то для загрузки подсказок на HTML-странице необходимо указать (для IE 5 и выше):

```
<script type="text/javascript" language="javascript" id="get_hints"></script>
<script language="JavaScript">
document.all.get_hints.src='http://rmt.vnetgis.com/?uamap_cuid=3e6decdcb93da&map=kiev&lg=j&rq=get_hints&st=2&l=ua';
function onload_get_hints()
{
        alert('Загрузка информационных подсказок завершена');
}
</script>
```

После выполнения запроса в пределах страницы будет доступен объект (массив) ObjInfo.

Использование «информационных подсказок» показано в следующих примерах:
- Интерфейс с поиском и информационными подсказками (http://demo.vnetgis.com/.ua/slim_search_hints_php/)
- Интерфейс (с управлением пластами и информационными подсказками) (http://demo.vnetgis.com/.ua/intern_layer_hints_php/)
- Интерфейс с поиском и информационными подсказками (http://demo.vnetgis.com/.ua/full_search_hints_php/)
- Пример отправки карты по почте (Удобный интерфейс) (http://demo.vnetgis.com/.ua/other_sendmail_full_php/)
- Получить рисунок карты за один запрос (с подсказками) (http://demo.vnetgis.com/.ua/other_rqnew_hint_php/)

Рассмотрим структура данных, возвращаемых сервером (если **lg=j**).

var ObjInfo= new Array(new Array (**[ID слоя],[X],[Y],'[Краткое название]',[ID объекта], [ID стиля отображения]**), ……);onload_get_hints();

где:

ID слоя – идентификатор слоя карты. Уникальное число в пределах одного узла картографической сети.

В версии 2.x ID слоя – это порядковый номер слоя в списке слоёв. Нумерация начинается с 0.

X , **Y** - относительные координаты объекта на «графическом представлении» карты (в пикселях)

Краткое название - краткое название объекта (для пользовательских объектов – указывается пользователем при занесении данных в базу сервера)

ID объекта - идентификатор объекта. Для слоя, принадлежащего пользователю – это идентификатор, указанный пользователем при создании объекта. Как правило, это идентификатор объекта в базе данных пользователя. Если слой принадлежит картографическому серверу = 0.

ID стиля отображения – идентификатор стиля отображения. Уникальное число (в пределах одной карты), присвоенное при создании стиля. Позволяет, при формировании «подсказки», отображать иконку, которой данный объект обозначен на карте. Если стиль не указан =0.

В версии 2.x ID стиля – это порядковый номер стиля в лое. Нумерация начинается с 0.

Пример результата запроса:

var ObjInfo= new Array(new Array (123,285,201,'ЖЕК N 1304',777,4), new Array (333,25,201,'Театр «Колесо»',555,4), new Array (123,28,201,'ЖЕК N 13',333,1));onload_get_hints();

Внимание! После окончания загрузки вызывается функция onload_get_hints(). Она должна быть определена в коде пользователя. (только для **lg=j**)

Результата запроса, для других форматов:

lg=v

l[num]=[**ID слоя**]&**x**[num]=[**X**]&**y**[num]=[**Y**]&**t**[num]=[**Краткое название**]&**r**[num]=[**ID объекта**]&**s**[num]=[**ID стиля отображения**]&
l[num]=[**ID слоя**]& … **count_tps=[колличество зписей]**

num – порядковый номер подсказки

количество записей – общее количество объектов, для которых присутствуют «подсказки».

Пример:

l0=123&x0=285&y0=201&t0=ЖЕК N 1304&r0=777&s0=4&

l1=333&x1=25&y1=201&t1=Театр «Колесо»&r1=555&s1=4&
l2=123&x2=28&y2=201&t2=ЖЕК N 13&r2=333&s2=1&count_tps=3

lg=t

[ID слоя]\n
[X]\n
[Y]\n
[Краткое название]\n
[ID объекта]\n
[ID стиля отображения]\n
\n
[ID слоя]\n
…

Пример:

123**\n**
285**\n**
201**\n**
ЖЕК N 1304**\n**
777**\n**
4**\n**
\n
333**\n**
25**\n**
201**\n**
Театр «Колесо»**\n**
555**\n**
4**\n**
\n
123**\n**
28**\n**
201**\n**
ЖЕК N 13**\n**
333**\n**
1**\n**

Внимание! Форматы lg=**d** и lg=**i** не допустимы. Запрос, без указания формата будет отвергнут.

6.5. Получить дополнительная информация о текущем «шаге сессии» (rq=get_state)

О каждом «шаге сессии» можно получить дополнительную информацию. Структура результата запроса, для разных форматов:

lg=h:

<script language="JavaScript">
var map_extent = Array(**[map_ext_minx]**,**[map_ext_miny]**,
[map_ext_maxx],**[map_ext_maxy]**);

```
var map_size = Array([map_pix_size_width],[map_pix_size_height]);
var cmd = '[cmd]';
var cmd_param = '[cmd_param]';
var map_scale = [map_scale];
var map_cuid = '[uamap_cuid]';
var map_st = [st];
var map_sc_idx=[index_scale];
var map_size_idx=[index_img_size];
</script>
```

где:

map_ext_minx, **map_ext_miny**, **map_ext_maxx**, **map_ext_maxy** – «реальные» координаты окна отображения карты, для которого будет сформирован рисунок.
map_pix_size_width, **map_pix_size_height** – ширина и высота рисунка карты.

Остальные параметры описаны ранее.

lg=j:

```
var map_extent = Array([map_ext_minx],[map_ext_miny],
[map_ext_maxx],[map_ext_maxy]);
var map_size = Array([map_pix_size_width],[map_pix_size_height]);
var cmd = '[cmd]';
var cmd_param = '[cmd_param]';
var map_scale = [map_scale];
var map_cuid = '[uamap_cuid]';
var map_st = [st];
var map_sc_idx=[index_scale];
var map_size_idx=[index_img_size];
onload_get_state();
```

lg=p:
Тоже, что и для lg=j, только упакованное в JavaScript объект

lg=v:

```
minx=[map_ext_minx]&
miny=[map_ext_miny]&
maxx=[map_ext_maxx]&
maxy=[map_ext_maxy]&
width=[map_pix_size_width]&
height=[map_pix_size_height]&
cmd = [cmd]&
cmd_param = [cmd_param]&
map_cuid=[uamap_cuid]&
map_st=[st]&
map_sc_idx=[index_scale]&
map_size_idx=[index_img_size]
```

lg=t (по умолчанию):

minx=[map_ext_minx]\n
miny=[map_ext_miny]\n
maxx=[map_ext_maxx]\n
maxy=[map_ext_maxy]\n
width=[map_pix_size_width]\n
height=[map_pix_size_height]\n
cmd = [cmd]\n
cmd_param = [cmd_param]\n
map_cuid=[uamap_cuid]\n
map_st=[st]\n
map_sc_idx=[index_scale]\n
map_size_idx=[index_img_size]\n

Пример запроса:

```
<script type="text/javascript" language="javascript" id="get_state"></script>
<script language="JavaScript">
document.all.get_state.src='http://rmt.vnetgis.com/?uamap_cuid=3e6decdcb93da&
map=kiev&lg=j&rq=get_state&st=2&l=ua';
function onload_get_state()
{
        alert("Загрузка информации закончена успешно");
}
</script>
```

В ответ на подобный запрос, картографический сервер вернет исходный текст на JavaScript:

```
var map_extent = Array(299756.775945,5561564.8026,
349711.352302,5610270.51454);
var map_size = Array(400,390);
var cmd = 'recenter';
var cmd_param = '30,100';
var map_scale = 24000;
var map_cuid = '3e6decdcb93da';
var map_st = 2;
var map_sc_idx=1;
var map_size_idx=1;
onload_get_state();
```

Внимание! После окончания загрузки вызывается функция onload_get_state(). Она должна быть определена в коде пользователя. (только для **lg=j**)

7. Специализированные запросы

Для работы некоторых приложений (к примеру, диспетчерской системы, являющейся клиентом картографической сети) необходима дополнительная информация о пользователе, картах и прочее. Для ее получения существуют дополнительные запросы:
get_profile – получить «профайл пользователя»
get_pkg – получить сервис-блок
get_maplist – получить список карт, доступных пользователю

Внимание! Данные (для перечисленных запросов) можно получить только в «текстовом» формате **lg=t**

7.1. Получить «профайл пользователя»

Данный запрос позволяет получить содержимое профайла указанного «хоста», получающего доступ к пространственным данным с виртуального сервера пользователя. В запросе необходимо указать параметр **host**, определяющий для какого именно WEB-сервера нужно получить данные.
Структура возвращаемых данных:

name=[MAP_NAME]\n
search.result.strt=[SEARCH_RESULT_STRT]\n
search.result.addr=[SEARCH_RESULT_ADDR]\n
search.result.marker=[SEARCH_RESULT_MARKER]\n
max_scale=[MAX_SCALE]\n
max_onestep_scale=[MAX_ONESTEP_SCALE]\n
step_scale.[**max_scale_num**]=[STEP_SCALE_**map_scale_num**]\n
…
count_map_size=[COUNT_MAP_SIZE]\n
max_map_size.0=[MAX_MAP_SIZE_0]\n
max_map_size.1=[MAX_MAP_SIZE_1]\n
map_size_list.[**map_size_num**].0=[MAP_SIZE_W_**map_size_num**]\n
map_size_list.[**map_size_num**].1=[MAP_SIZE_H_**map_size_num**]\n
…
max_mapkey_size.0=[MAX_MAPKEY_SIZE_0]\n
max_mapkey_size.1=[MAX_MAPKEY_SIZE_1]\n
mapkey_size.0=[MAPKEY_SIZE_0]\n
mapkey_size.1=[MAPKEY_SIZE_1]\n

max_extent_minx=[MAX_EXTENT_MINX]\n
max_extent_maxx=[MAX_EXTENT_MAXX]\n
max_extent_miny=[MAX_EXTENT_MINY]\n
max_extent_maxy=[MAX_EXTENT_MAXY]\n
init_extent_minx=[INIT_EXTENT_MINX]\n
init_extent_maxx=[INIT_EXTENT_MAXX]\n
init_extent_miny=[INIT_EXTENT_MINY]\n
init_extent_maxy=[INIT_EXTENT_MAXY]\n
l.[**num**].n=[MAP_LAYER_SYS_NAME]\n
l.[**num**].v=[MAP_LAYER_RULES]\n
l.[**num**].t=[MAP_LAYER_NAME]\n
l.[**num**].g=[MAP_LAYER_TYPE]\n

l.[**num**].s=[MAP_LAYER_SEARCH]\n
…

где:

MAP_NAME – системное название карты
SEARCH_RESULT_STRT – количество объектов адресного реестра, входящих в один «блок» результата поиска (улицы) Смотри п.5.5
SEARCH_RESULT_ADDR – … (дома)
SEARCH_RESULT_MARKER – количество маркерных объектов, входящих в один «блок» результата поиска. Смотри п.5.4.2
MAX_ONESTEP_SCALE – допустимое изменение фиксированного масштаба за один запрос

MAX_SCALE – количество «фиксированных масштабов» карты
STEP_SCALE_**map_scale_num** **–** значение фиксированного масштаба. Параметр указывается **map_scale_num** – раз.

MAX_MAP_SIZE_0, MAX_MAP_SIZE_1 – максимально допустимые значения рисунка карты (ширина и высота)
COUNT_MAP_SIZE – количество размеров рисунков карты
MAP_SIZE_W_**map_size_num**, MAP_SIZE_H_**map_size_num** **–** значения рисунка карты (ширина и высота). Параметр указывается **map_size_num** – раз.

MAX_MAPKEY_SIZE_0, MAX_MAPKEY_SIZE_1 – максимально допустимые значения рисунка картографического навигатора (ширина и высота)
MAPKEY_SIZE_0, MAPKEY_SIZE_1 – установленный размер рисунка картографического навигатора (ширина и высота)

MAX_EXTENT_MINX, MAX_EXTENT_MAXX,
MAX_EXTENT_MINY, MAX_EXTENT_MAXY – максимально разрешенный размер окна отображения карты

INIT_EXTENT_MINX,INIT_EXTENT_MAXX,
INIT_EXTENT_MINY,INIT_EXTENT_MAXY – начальный размер окна отображения карты

Далее следует описание параметров отображения/управления каждого слоя, входящего в состав данного проекта.
num – указывает порядковый номер слоя

MAP_LAYER_SYS_NAME – системное название слоя
MAP_LAYER_RULES – состояние управления
Допустимые значения:

0 - off
1 - on
3 - off_http
4 - on_http

MAP_LAYER_NAME – название слоя, соответствующее параметру **l** (на русском, украинском или английском)
MAP_LAYER_TYPE – тип слоя (линия/..)
Допустимые значения:

1 - линия

3 - полигон

4 - точечный

MAP_LAYER_SEARCH – допустим ли поиск по слою
Допустимые значения:

0 - без поиска

1 - адресный реестр

4 - маркерные объекты

5 - маркерные объекты (с адресной привязкой)

Пример запроса к серверу:

http://rmt.vnetgis.com/?uamap_cuid=3e6decdcb93da&map=kiev&lg=t&rq=get_profile&st=1&host=http://uamap.com/ua/map/kiev&l=ru

Пример «ответа сервера»:

name=kiev_region**\n**
search.result.strt=20**\n**
search.result.addr=30**\n**
search.result.marker=20**\n**
max_scale=9**\n**
max_onestep_scale=7**\n**
step_scale.1=310773**\n**
step_scale.2=170000**\n**
step_scale.3=90000**\n**
step_scale.4=50000**\n**
step_scale.5=20000**\n**
step_scale.6=12000**\n**
step_scale.7=8000**\n**
step_scale.8=4000**\n**
step_scale.9=2000**\n**
count_map_size=4**\n**
max_map_size.0=800**\n**
max_map_size.1=800**\n**
map_size_list.1.0=400**\n**
map_size_list.1.1=390**\n**
map_size_list.2.0=480**\n**
map_size_list.2.1=468**\n**
map_size_list.3.0=600**\n**
map_size_list.3.1=585**\n**

map_size_list.4.0=720\n
map_size_list.4.1=702\n
max_mapkey_size.0=200\n
max_mapkey_size.1=200\n
mapkey_size.0=100\n
mapkey_size.1=98\n
max_extent_minx=299756.775945\n
max_extent_maxx=349711.352302\n
max_extent_miny=5561564.8026\n
max_extent_maxy=5610270.51454\n
init_extent_minx=302861.853712\n
init_extent_maxx=346606.274535\n
init_extent_miny=5564592.25342\n
init_extent_maxy=5607243.06372\n
l.1.n=district\n
l.1.v=2\n
l.1.t=Районы\n
l.1.g=3\n
l.1.s=1\n
l.2.n=wood\n
l.2.v=2\n
l.2.t=Лес\n
l.2.g=3\n
l.2.s=0\n
l.3.n=bridge\n
l.3.v=2\n
l.3.t=Мосты\n
l.3.g=3\n
l.3.s=0\n
l.4.n=stadion\n
l.4.v=2\n
l.4.t=Стадионы\n
l.4.g=3\n
l.4.s=0\n
l.5.n=railway\n
l.5.v=2\n
l.5.t=Железные дороги\n
l.5.g=1\n
l.5.s=0\n
l.6.n=strt\n
l.6.v=2\n
l.6.t=Улицы\n
l.6.g=1\n
l.6.s=1\n
l.7.n=square\n
l.7.v=3\n
l.7.t=Площади\n
l.7.g=3\n
l.7.s=0\n
l.8.n=metro\n
l.8.v=4\n
l.8.t=Метро\n

l.8.g=4\n
l.8.s=4\n
l.9.n=address\n
l.9.v=2\n
l.9.t=Адреса\n
l.9.g=4\n
l.9.s=1\n

7.2. Получить сервис-блок

Для правильного функционирования некоторых сложных автоматизированных комплексов, взаимодействующих с серверами картографической сети, необходимы дополнительные справочные данные о пространственных банках данных. К примеру, для работы «Клиента диспетчерского сервера» необходима следующая дополнительная информация:

При чем в дальнейшем, для данного программного обеспечения, может понадобится другая информация. А хранение подобных данных в «клиенте» не позволит добавлять (и следить за изменением уже существующих) новые карты в автоматизированном режиме. Для решения данного вопроса существуют «сервис-блоки». Это обыкновенный файл, сформированный произвольным образом. При запросе «клиент» указывает уникальное название «блока» (зарезервированное за **определенным типом** программного обеспечения) в параметре **cmd**.

Пример запроса к серверу:

http://rmt.vnetgis.com/?uamap_cuid=3e6decdcb93da&map=kiev&lg=t&rq=get_pkg&st=1&cmd=gps

Ответ сервера:

fill_color=C4DEC1\n
name_ua=Київ\n
name_ru=Киев\n
name_en=Kiev\n
scale=10000\n
type=3\n
proj=proj=tmerc ellps=WGS84 lon_0=33E k=0.9996 x_0=500000 y_0=0\n
x=0\n
y=0\n

Внимание! Значение «сервис-блока» не зависит от параметров данного пользователя. Пользователь не может указать конкретные значения «сервис-блока» для собственных нужд. Для получения значения «блока» не нужно определять, какому именно пользователю он принадлежит, но получить значение возможно лишь пройдя стандартную процедуру идентификации.

Для создания нового типа (cmd=[ваш тип блока]) «сервис-блока» - обратитесь в службу поддержки.

7.3. Получить список карт, доступных пользователю

Каждый пользователь, в пределах «виртуального картографического сервера», получает доступ к ограниченному списку пространственных банков данных («картам»). Перечень банков определяется условиями доступа данного клиента к «картографической сети». Для получения «карт», доступных в данный момент, можно выполнить запрос **rq=get_maplist**.

Пример запроса к серверу:

http://rmt.vnetgis.com/?uamap_cuid=3e6decdcb93da&lg=t&rq=get_maplist&st=1

Ответ сервера:

Внимание! Список карт не зависит от данных конкретного «профайла пользователя». По этому наличие параметра **host** не обязательно. Но картографический сервер должен точно идентифицировать пользователя. Если параметр **referer** отсутствует – пользователь не будет определен, следовательно запрос будет отвергнут.

8. Поиск кратчайшего маршрута

Данная функциональность реализована только начиная с версии 2.0 (12.03.2009)

Для того чтобы поиск кратчайшего маршрута работал, в проекте карты должен быть описан хотя бы один граф. Граф – это связка двух слоёв: слой рёбер и слой вершин. Для карты может присутствовать более одного графа. По умолчанию выбирается первый граф.

Для поиска кратчайшего маршрута существуют такие команды:
spath – удалить существующий путь и добавить набор точек.
apath – добавить к пути набор точек
sspath – удалить существующий путь и добавить набор объектов.
aspath – добавить набор объектов к пути.

После выполнения этих команд построенный путь отображается на карте.
Эти команды дополнительно к тому, что возвращает стандартный запрос rq=new, возвращает переменные:
vnetDistance – расстояние пути.
vnetWeight – вес пути.

Дополнительным параметром для этих команд может быть **&vgraph=<имя_графа>**. Для явного указания, какой граф следует использовать при расчёте.

Вес пути может отличаться от расстояния. Для графа по умолчанию вес соответствует времени в секундах.

Для получения кратчайшего маршрута в текстовом виде существует запрос **`get_path`**

8.1. Команда spath

Название

Удалить существующий путь и добавить набор точек

Краткое описание

Удалить существующий путь и добавить набор точек из cmd_param.
Для очистки существующего пути указать пустой cmd_param.

CMD

spath

CMD_PARAM

Набор точек в мировой системе координат, разделённых «;» (точкой с запятой).
Разделитель между координатами одной точки – запятая.

Пример

http://rmt.v2.vnetgis.com/?rq=new&cmd=spath&cmd_param=315398,5589263;332888,5588603&uamap_cuid=49b929bc30ea4&map=kiev&lg=j&st=1&l=ua

Ищет путь между точками (315398;5589263) и (332888;5588603)

8.2. Команда apath

Название

Добавить к пути набор точек

Краткое описание

Добавить набор точек из cmd_param.

CMD

apath

CMD_PARAM

Набор точек в мировой системе координат, разделённых «;» (точкой с запятой). Разделитель между координатами одной точки – запятая.

Пример

http://rmt.v2.vnetgis.com/?rq=new&cmd=apath&cmd_param=320718,5586533&uamap_cuid=49b929bc30ea4&map=kiev&lg=j&st=4&l=ua

Добавлят точку (320718,5586533)

8.3. Команда sspath

Название

Удалить существующий путь и добавить набор объектов

Краткое описание

Удалить существующий путь и добавить набор объектов.

CMD

sspath

CMD_PARAM

Формат
cmd_param= такой же как в комманде msearch. Для очистки существующего пути указать пустой cmd_param.

Пример

http://rmt.v2.vnetgis.com/?rq=new&cmd=sspath&cmd_param=metro!!Харківська&
uamap_cuid=49b929bc30ea4&map=kiev&lg=j&st=6&l=ua

Удаляет старый путь и добавляет стартовую точку - метро «Харківська»

8.4. Команда aspath

Название

Добавить набор объектов к пути

Краткое описание

Добавить набор объектов к пути.

CMD

sspath

CMD_PARAM

Формат
cmd_param= такой же как в комманде msearch.

Пример

http://rmt.v2.vnetgis.com/?rq=new&cmd=aspath&cmd_param=metro!!Харківська&
uamap_cuid=49b929bc30ea4&map=kiev&lg=j&st=6&l=ua

Добавляет следующую точку - метро «Харківська»

8.5. Запрос get_path

Запрос служит для получения вычисленного пути в текстовом виде.

Пример:
http://rmt.v2.vnetgis.com/?rq=get_path&cmd_param=path&lg=t&uamap_cuid=49b929bc30ea4&map=kiev&st=7&l=ua&vpath=roadt

Параметры:

cmd_param - степень детализации пути. Необязательный параметр.
Возможные значения:
none - Только расстояние и вес
search_points - Только точки поиска
path - Путь без геометрии
line - Путь с геометрией линиями с двумя точками
polyline - Путь с нормальной детализацией геометрии

vpath – системное название графа. Необязательный параметр.
Если не указан, используется граф по умолчанию.

lg - Формат результата.

Вид текстового результата(lg=t):
points_count=<количество точек, между которыми ищется путь>
vnetDistance=<длина пути>
vnetWeight=<вес пути>
sp.0=<точка поиска>
sp.1=<точка поиска>
…
sp.< points_count -1>=<точка поиска>
p.0.0=<ребро>
p.0.1=<ребро>
…
p.0. < к-во рёбер в первом пути -1>=<ребро>
…
p.< points_count -2>.<к-во рёбер в последнем пути -1>=<ребро>

<точка поиска>:
Содержит координаты стартовой точки и путь до следующей точки поиска.

<мировая к-та X>,<мировая к-та Y>,<вес>,<расстояние>,<ID объекта>,”<слой объекта>”,”<название объекта>”,<к-во рёбер в пути>

Если точка добавлена через spath или spath <ID объекта>=-1, слой и название пустые.

<ребро>:

<ID ребра>,<стартовая вершина>,<вес>,<расстояние>,<внешний идентификатор, с которым связано ребро>,”<название ребра>”,”< название стартовой вершины>”,<полилиния ребра>
Т.к. между точками поиска и вершинами графа добавляются хвостики, ID ребра и стартовая вершина могут не назначены, т.е. равны -1

<полилиния ребра>:
<линия ребра 0>;<линия ребра 1>;…<линия ребра N>

<линия ребра>:
<X0><пробел><Y0><пробел><X1><пробел><Y1><пробел>…<XN><пробел><YN>

Пример:

```
points_count=2
vnetDistance=1531.544955044444
vnetWeight=114.5385721913741
sp.0=333068.769761,5585795.477533,114.538572,1531.544955,8794,"metro","Харківська",9
sp.1=332998.600025,5586733.355371,0.000000,0.000000,-1,"","",0
p.0.0=-1,-1,1.659476,17.614406,-1,"","",333068.769761 5585795.477533 333051.142917 5585794.596053
p.0.1=76297,-1,6.286158,66.724045,18779,"","",333051.142917 5585794.596053 333075.063544
5585853.442668
p.0.2=1327,4862,10.946729,150.184936,104,"Миколи Бажана просп.","",333075.063544 5585853.442668
333222.071597 5585884.169649
p.0.3=31976,4866,22.475465,240.282572,18776,"","",333222.071597 5585884.169649 333164.234814
5585769.757484
p.0.4=76299,32767,11.417056,155.758766,1295,"Ревуцького","",333164.234814 5585769.757484
333129.127848 5585921.427827
p.0.5=76278,32758,8.505759,126.172635,1295,"Ревуцького","",333129.127848 5585921.427827
333101.722284 5586044.588134
p.0.6=76227,32743,1.196152,13.772360,0,"","",333101.722284 5586044.588134 333098.789056
5586058.044511
p.0.7=97474,32752,46.216899,675.725023,1295,"Ревуцького","",333098.789056 5586058.044511
332917.805859 5586705.541508
p.0.8=-1,-1,5.834878,85.310210,-1,"","",332917.805859 5586705.541508 332998.600025 5586733.355371
```

Вид результата для JavaScript (lg=j):

var vnetDistance=<длина пути>
var vnetWeight=<вес пути>
var vnetPathPoints=new Array(); - массив точек поиска вместе с рёбрами

vnetPathPoints[0]= <точка поиска>
vnetPathPoints[1]= <точка поиска>
…
vnetPathPoints[к-во точек поиска -1]= <точка поиска>
vnetPathPoints[0][7][0]= <ребро>
vnetPathPoints[0][7][1]= <ребро>
…
vnetPathPoints[0][7][< к-во рёбер в первом пути -1>]= <ребро>
…
vnetPathPoints[к-во точек поиска -1][7][< к-во рёбер в последнем пути -1>]= <ребро>
onload_path_points();

<точка поиска>:
Содержит координаты стартовой точки и путь до следующей точки поиска.

new Array(<мировая к-та X>,<мировая к-та Y>,<вес>,<расстояние>,<ID объекта>,'<слой объекта>','<название объекта>',new Array());

Если точка добавлена через spath или spath <ID объекта>=-1, слой и название пустые. Последний элемент – пустой массив для списка рёбер.

<ребро>:
new Array(<ID ребра>,<стартовая вершина>,<вес>,<расстояние>,<внешний идентификатор, с которым связано ребро>,'<название ребра>','< название стартовой вершины>', <полилиния ребра>);

Т.к. между точками поиска и вершинами графа добавляются хвостики, ID ребра и стартовая вершина могут не назначены, т.е. равны -1

<полилиния ребра>:
new Array(<линия ребра>,<линия ребра>…<линия ребра>)

<линия ребра>:
new Array(<X0>,<Y0>,<X1>,<Y1>,…<XN>,<YN>)

Пример:

```
var vnetDistance=3562.092729544784;
var vnetWeight=379.4833538791859;
var vnetPathPoints=new Array();
vnetPathPoints[0]=new
Array(333068.769761,5585795.477533,379.483354,3562.092730,8794,'metro','Харківська',new Array());
vnetPathPoints[1]=new Array(332668.600025,5586623.355371,0.000000,0.000000,-1,'','',new Array());
vnetPathPoints[0][7][0]=new Array(-1,-1,1.659476,17.614406,-1,'','',new Array(new
Array(333068.769761,5585795.477533,333051.142917,5585794.596053)));
vnetPathPoints[0][7][1]=new Array(76297,-1,6.286158,66.724045,18779,'','',new Array(new
Array(333051.142917,5585794.596053,333075.063544,5585853.442668)));
vnetPathPoints[0][7][2]=new Array(1327,4862,10.946729,150.184936,104,'Миколи Бажана
просп.','',new Array(new Array(333075.063544,5585853.442668,333222.071597,5585884.169649)));
vnetPathPoints[0][7][3]=new Array(31976,4866,22.475465,240.282572,18776,'','',new Array(new
Array(333222.071597,5585884.169649,333164.234814,5585769.757484)));
vnetPathPoints[0][7][4]=new Array(76299,32767,11.417056,155.758766,1295,'Ревуцького','',new
Array(new Array(333164.234814,5585769.757484,333129.127848,5585921.427827)));
vnetPathPoints[0][7][5]=new Array(76278,32758,8.505759,126.172635,1295,'Ревуцького','',new
Array(new Array(333129.127848,5585921.427827,333101.722284,5586044.588134)));
vnetPathPoints[0][7][6]=new Array(76227,32743,1.196152,13.772360,0,'','',new Array(new
Array(333101.722284,5586044.588134,333098.789056,5586058.044511)));
vnetPathPoints[0][7][7]=new Array(11768,32752,55.210309,902.470076,1295,'Ревуцького','',new
Array(new Array(333098.789056,5586058.044511,332831.801093,5586917.840804)));
vnetPathPoints[0][7][8]=new Array(76350,32799,1.262621,16.215568,0,'','',new Array(new
Array(332831.801093,5586917.840804,332836.357717,5586933.402996)));
vnetPathPoints[0][7][9]=new Array(11764,32804,20.887104,259.527844,1295,'Ревуцького','',new
Array(new Array(332836.357717,5586933.402996,332764.123144,5587179.278816)));
vnetPathPoints[0][7][10]=new Array(11762,32834,8.022620,68.777126,1295,'Ревуцького','',new
Array(new Array(332764.123144,5587179.278816,332759.624434,5587243.920002)));
vnetPathPoints[0][7][11]=new Array(97286,32825,2.143693,14.291165,0,'','',new Array(new
Array(332759.624434,5587243.920002,332749.492076,5587253.998331)));
vnetPathPoints[0][7][12]=new Array(19039,32830,0.922926,8.335110,18837,'','',new Array(new
Array(332749.492076,5587253.998331,332742.517963,5587258.555381)));
vnetPathPoints[0][7][13]=new Array(76390,32821,12.056263,109.268685,18789,'','',new Array(new
Array(332742.517963,5587258.555381,332657.539948,5587219.191428)));
vnetPathPoints[0][7][14]=new Array(11734,32734,2.668167,19.175864,1295,'Ревуцького','',new
Array(new Array(332657.539948,5587219.191428,332656.937970,5587200.118836)));
vnetPathPoints[0][7][15]=new Array(39514,9477,2.379410,14.258300,0,'','',new Array(new
Array(332656.937970,5587200.118836,332660.700751,5587186.365996)));
vnetPathPoints[0][7][16]=new Array(11732,9479,8.207016,56.723343,1295,'Ревуцького','',new
Array(new Array(332660.700751,5587186.365996,332702.678805,5587151.647844)));
vnetPathPoints[0][7][17]=new Array(76201,32726,21.017751,247.582502,1295,'Ревуцького','',new
Array(new Array(332702.678805,5587151.647844,332787.423375,5586922.841624)));
vnetPathPoints[0][7][18]=new Array(76183,32717,1.287973,15.349715,0,'','',new Array(new
Array(332787.423375,5586922.841624,332792.025628,5586908.198092)));
```

```
vnetPathPoints[0][7][19]=new Array(11756,32722,15.101666,207.430702,1295,'Ревуцького','',new
Array(new Array(332792.025628,5586908.198092,332858.884196,5586711.903962)));
vnetPathPoints[0][7][20]=new Array(76370,32808,148.071801,760.924530,1435,'Степана
Олійника','',new Array(new Array(332858.884196,5586711.903962,332685.614490,5586712.941550)));
vnetPathPoints[0][7][21]=new Array(-1,-1,17.757238,91.252475,-1,'','',new Array(new
Array(332685.614490,5586712.941550,332668.600025,5586623.355371)));
onload_path_points();
```

Вид результата для JSONP (lg=p):

Вид результата такой же, как для lg=j, только реузльтат упакован в JavaScript объект.

8.6. Соглашения по графам.

Часть стандартных карт имеет граф дорог. В таких картах присутствуют два графа:
roadt – граф дорог. Вес время в секундах.
roadd – граф дорог. Вес расстояние в метрах.

roadt используется по умолчанию.

Для обоих расстояние считается в метрах.
Внешний идентификатор ребра соответствует идентификатору, а заголовок названию улицы, к которой это ребро относится.

Приложение 1.

Параметры управления и их использование

Название

Установить фиксированный размер "рисунка карты"

Краткое описание

Указывается индекс размера (значения, соответствующие указанному индексу и их количество хранятся в профайле "сервера-клиента") **WEB**

CMD

set_mapsize

CMD_PARAM

Значение индекса (**1,2,3,...**).

Пример

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&**cmd=set_mapsize**&**cmd_param=2**&st=3&l=ru

Название

Установить указанный масштаб карты

Краткое описание

Указывается необходимый масштаба карты. Для каждой карты определяется интервал допустимых масштабов, в пределах которого должно находиться указываемое значение. В случаи выхода значения за рамки "допустимых масштабов" происходит коррекция устанавливаемого значения. (**WEB**)

В проф. пользователя хранится "начальное значения масштаба". При первом запросе к серверу (с новым UAMAP_CUID) будет создана карта именно с "начальным значения масштаба".

CMD

set_zoom_scl

CMD_PARAM

Устанавливаемое значение масштаба. В данном случаи указывается системное значение масштаба. Текущее значение масштаба возвращается в параметре **scale**

Пример

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&**cmd=set_zoom_scl&cmd_param=200000**&st=3&l=ru

Название

Фиксированное перемещение центра карты

Краткое описание

Происходит сдвиг видимой части картинки карты (на константное значение) в указанном направлении (**WEB**)

Значение "сдвига" указывается в проф. пользователя. По умолчанию равно: (0.4) * (ширины или высота рисунка)

CMD

fixed_mov

CMD_PARAM

Направление сдвига (относительно центра рисунка):

left – в лево
right – в право
top – в верх
bottom – в низ
right_top – в право и вверх
right_bottom – в право и в низ
left_top – в лево и вверх
left_bottom – в лево и в низ

Пример

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&**cmd=fixed_mov&cmd_param=right_bottom**&st=3&l=ru

Название

Фиксированное изменение масштаба с рецентровкой "картинки" карты (увеличение)

Краткое описание

Происходит сдвиг видимой части картинки карты в точке нажатия на "картинке" карты (центровка карты) с изменением масштаба на один “шаг” (относительно текущего значения “фиксированного масштаба”)

CMD

zoom_in

CMD_PARAM

Координаты (в пикселях) точки нажатия на карте. Именно данная точка будет расположена в центре рисунка карты (после изменения масштаба)
Формат : x,y

Внимание ! Если указаны координаты в параметрах **uamap_x, uamap_y** (координаты точки "нажатия") – значение параметра cmd_param не используется

Пример

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&**cmd=zoom_in&cmd_param=232,100**&st=3&l=ru

Название

Фиксированное изменение масштаба с рецентровкой "картинки" карты (уменьшение)

Краткое описание

Происходит сдвиг видимой части картинки карты в точке нажатия на "картинке" карты (центровка карты) с изменением масштаба на один “шаг” (относительно текущего значения “фиксированного масштаба”)

CMD

zoom_out

CMD_PARAM

Координаты (в пикселях) точки нажатия на карте. Именно данная точка будет расположена в центре рисунка карты (после изменения масштаба)
Формат : x,y

Внимание ! Если указаны координаты в параметрах **uamap_x, uamap_y** (координаты точки "нажатия") – значение параметра cmd_param не используется

Пример

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&**cmd=**zoom_out**&cmd_param=200,120**&st=3&l=ru

Название

Рецентровка "картинки" карты (при нажатии на "картографическом навигаторе")

Краткое описание

Происходит сдвиг видимой части картинки карты в точку, соответствующую координатам "точки нажатия" на картографическом навигаторе

CMD

key_recenter

CMD_PARAM

Не используется

Внимание ! Должны присутствовать дополнительные параметры **key_uamap_x, key_uamap_y** - координаты (в пикселях) точки нажатия на карте. Именно данная точка будет расположена в центре рисунка карты (после изменения масштаба)

Пример

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&**cmd=key_recenter&cmd_param=&key_uamap_x=41&key_uamap_y=32**&st=3&l=ru

Название

Рецентровка "картинки" карты с изменением масштаба (при нажатии на "картографическом навигаторе")

Краткое описание

Происходит сдвиг видимой части картинки карты в точку, соответствующую координатам "точки нажатия" на картографическом навигаторе с одновременным изменением масштаба карты.

CMD

key_recenter_scl

CMD_PARAM

Устанавливаемое значение масштаба. В данном случаи указывается системное значение масштаба. Текущее значение масштаба возвращается в параметре **scale**

Внимание ! Должны присутствовать дополнительные параметры **key_uamap_x, key_uamap_y** - координаты (в пикселях) точки нажатия на карте. Именно данная точка будет расположена в центре рисунка карты (после изменения масштаба)

Пример

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&
cmd=key_recenter_scl&cmd_param=20000&key_uamap_x=30&key_uamap_y=20&
st=3&l=ru

Название

Рецентровка "картинки" карты с изменением “фиксированного масштаба” (при нажатии на "картографическом навигаторе")

Краткое описание

Происходит сдвиг видимой части картинки карты в точку, соответствующую координатам "точки нажатия" на картографическом навигаторе с одновременным изменением масштаба карты. При этом масштаб задается в «фиксированных значениях»

CMD

key_recenter_fix_scl

CMD_PARAM

Значение индекса (**1,2,3,...**).

Внимание ! Должны присутствовать дополнительные параметры **key_uamap_x, key_uamap_y** - координаты (в пикселях) точки нажатия на карте. Именно данная точка будет расположена в центре рисунка карты (после изменения масштаба)

Пример

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&
cmd=key_recenter_fix_scl&cmd_param=2&key_uamap_x=30&key_uamap_y=20&
st=3&l=ru

Название

Рецентровка "картинки" карты (при нажатии на карте)

Краткое описание

Указать новую центральную точку рисунка карты

CMD

recenter

CMD_PARAM

Координаты (в пикселях) точки нажатия на карте. Именно данная точка будет расположена в центре рисунка карты (после изменения масштаба)
Формат : x,y

Внимание ! Если указаны координаты в параметрах **uamap_x, uamap_y** (координаты точки "нажатия") – значение параметра cmd_param не используется

Пример

http://rmt.vnetgis.com/?
uamap_cuid=3e638fc73a230&map=kiev&rq=new&**cmd=recenter&**
cmd_param=200,240&st=3&l=ru

Название

Рецентровка "картинки" карты (при нажатии на карте) с изменением масштаба

Краткое описание

Указать новую центральную точку рисунка карты и изменить масштаб.

CMD

recenter_scl

CMD_PARAM

Устанавливаемое значение масштаба. В данном случаи указывается системное значение масштаба. Текущее значение масштаба возвращается в параметре **scale**

Внимание ! Должны присутствовать дополнительные параметры **uamap_x, uamap_y** - координаты (в пикселях) точки нажатия на карте. Именно данная точка будет расположена в центре рисунка карты (после изменения масштаба)

Пример

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&
cmd=recenter_scl&cmd_param=3000&uamap_x=120&uamap_y=50&st=3&l=ru

Название

Рецентровка "картинки" карты (при нажатии на карте) с изменением масштаба. При этом масштаб задается в «фиксированных значениях»

Краткое описание

(Смотри ком. SET_ZOOM_SIZE)

CMD

recenter_fix_scl

CMD_PARAM

Значение индекса (**1,2,3,...**).

Внимание ! Должны присутствовать дополнительные параметры **uamap_x, uamap_y** - координаты (в пикселях) точки нажатия на карте. Именно данная точка будет расположена в центре рисунка карты (после изменения масштаба)

Пример

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&
cmd=recenter_fix_scl&cmd_param=2&uamap_x=120&uamap_y=50&st=3&l=ru

Название

Рецентровка "картинки" карты. Координаты "центральной точки" указываются в географических координатах

Краткое описание

CMD

recenter_geo

CMD_PARAM

Координаты (реальные значения координат) точки нажатия на карте. Именно данная точка будет расположена в центре рисунка карты (после изменения масштаба)
Формат : x,y

Внимание ! Если указаны координаты в параметрах **uamap_x, uamap_y** (координаты точки "нажатия") – значение параметра cmd_param не используется

Пример

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&
cmd=recenter_geo**&cmd_param=&uamap_x=54.55667676&uamap_y=26.877567675**
&st=3&l=ru

Название

Показать выбранный регион на рисунке карты (координаты в пикселях).

Краткое описание

Внимание! При показе "выбранный регион" будет отображен на рисунке определенного ранее размера (при этом произойдет изменение масштаба)

CMD

view_region

CMD_PARAM

Координаты необходимого региона на текущем рисунке карты (в пикселях)
Формат: **x1,y1;x2,y2**

Пример

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&**cmd=view_region&cmd_param=20,20;40,70**&st=3&l=ru

Название

Показать выбранный регион на рисунке карты. («реальные» координаты)

Краткое описание

CMD

view_region_geo

CMD_PARAM

Координаты необходимого региона на карте
Формат: **x1,y1;x2,y2**

Пример

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&
cmd=view_region_geo**&cmd_param=543434,2645435;5545456,2745435**&st=3&l=ru

Название

Перерисовать карту

Краткое описание

Перерисовать карту

CMD

redraw

CMD_PARAM

Не используется

Пример

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&
cmd=redraw**&cmd_param=**&st=3&l=ru

Название

Рецентровка "картинки" карты (при нажатии на карте указываются «реальные» координаты) с изменением масштаба. При этом масштаб задается в «фиксированных значениях»

Краткое описание

(Смотри ком. RECENTER_FIX_SCL)

CMD

recenter_fix_scl_geo

CMD_PARAM

Значение индекса (**1,2,3,...**).

Внимание ! Должны присутствовать дополнительные параметры **uamap_x, uamap_y** - координаты («реальные» значения) точки нажатия на карте. Именно данная точка будет расположена в центре рисунка карты (после изменения масштаба)

Пример

http://rmt.vnetgis.com/?uamap_cuid=3e638fc73a230&map=kiev&rq=new&
cmd=recenter_fix_scl_geo&cmd_param=2&uamap_x=2655677&uamap_y=5455677&
st=3&l=ru

Приложение 2.

Краткое описание протокола HTTP/1.1 [RFC2068]

Данное описание протокола довольно условное и сокращенное. Более подробно протокол **HTTP/1.1** описан в стандарте **RFC2068**. (Русскоязычный перевод находится здесь http://vnetgis.com/ru/page/rfc2068)

1. HyperText Transfer Protocol

HyperText Transfer Protocol - протокол обмена WWW - серверов

1.1. Назначение

HyperText Transfer Protocol (HTTP) - это протокол прикладного уровня, применяемый в распределенных информационных системах гипермедиа. HTTP используется проектом World Wide Web с 1990 года.

HTTP/1.0 предоставляет открытое множество методов, которые могут быть использованы для указания целей запроса. Они построены на дисциплине ссылок, где для указания ресурса, к которому должен быть применен данный метод, используется Универсальный Идентификатор Ресурсов (Universal Resource Identifier - URI), в виде местонахождения (URL) или имени (URN). Формат сообщений сходен с форматом Internet Mail или Multipurpose Internet Mail Extensions (MIME-Многоцелевое Расширение Почты Internet).

HTTP/1.0 используется также для коммуникаций между различными пользовательскими просмотрщиками и шлюзами, дающими гипермедиа доступ к существующим Internet протоколам, таким как SMTP, NNTP, FTP, Gopher и WAIS. HTTP/ 1.0 разработан, чтобы позволять таким шлюзам через proxy серверы, без какой-либо потери передавать данные с помощью упомянутых протоколов более ранних версий.

1.2. Общая структура HTTP

Общая структура HTTP основывается на парадигме запросов/ответов. Запрашивающая программа (обычно она называется клиент) устанавливает связь с обслуживающей программой-получателем (обычно называется сервер) и посылает запрос серверу в следующей форме: метод запроса, URI, версия протокола, за которой следует MIME-подобное сообщение, содержащее управляющую информацию запроса, информацию о клиенте и, может быть, тело сообщения. Сервер отвечает сообщением, содержащим строку статуса (включая версию протокола и код статуса - успех или ошибка), за которой следует MIME-подобное сообщение, включающее в себя информацию о сервере, метаинформацию о содержании ответа, и, вероятно, само тело ответа. Следует отметить, что одна программа может быть одновременно и клиентом и сервером. Использование этих терминов в данном тексте относится только к роли, выполняемой программой в течение данного конкретного сеанса связи, а не к общим функциям программы.
В Internet коммуникации обычно основываются на TCP/IP протоколах. Для WWW номер порта по умолчанию - TCP 80, но также могут быть использованы и другие номера портов

- это не исключает возможности использовать HTTP в качестве протокола верхнего уровня.

Для большинства приложений сеанс связи открывается клиентом для каждого запроса и закрывается сервером после окончания ответа на запрос. Тем не менее, это не является особенностью протокола. И клиент, и сервер должны иметь возможность закрывать сеанс связи, например, в результате какого-нибудь действия пользователя. В любом случае, разрыв связи, инициированный любой стороной, прерывает текущий запрос, независимо от его статуса.

2. HTTP запрос

2.1. Общие понятия

Запрос - это сообщение, посылаемое клиентом серверу. Первая строка этого сообщения включает в себя метод, который должен быть применен к запрашиваемому ресурсу, идентификатор ресурса и используемую версию протокола. Для совместимости с протоколом HTTP/0.9, существует два формата HTTP запроса:

```
Запрос = Простой-Запрос | Полный-Запрос
Простой-Запрос = "GET" SP Запрашиваемый-URI CRLF
Полный-Запрос = Строка-Статус
      *(Общий-Заголовок | Заголовок-Запроса | Заголовок-Содержания ) CRLF
      [ Содержание-Запроса ]
```

Если HTTP/1.0 сервер получает Простой-Запрос, он должен отвечать Простым-Ответом HTTP/0.9. HTTP/1.0 клиент, способный обрабатывать Полный-Ответ, никогда не должен посылать Простой-Запрос.

2.2. Строка Статуса

Строка **Статус** начинается со строки с названием метода, за которым следует **URI-Запрос** и использующаяся версия протокола. Строка **Статус** заканчивается символами CRLF. Элементы строки разделяются пробелами (SP). В Строке Статус не допускаются символы LF и CR, за исключением заключающей последовательности CRLF.

```
Строка-Статус = Метод SP URI-Запроса SP Версия-HTTP CRLF
```

Следует отметить, что отличие Строки **Статус Полного-Запроса** от **Строки Статус Простого-Запроса** заключается в присутствии поля Версия-HTTP.

2.3. Метод запроса

В поле **Метод** указывается метод, который должен быть применен к ресурсу, идентифицируемому **URI-Запрос**. Названия методов чувствительны к регистру. Существующий список методов может быть расширен.

```
Метод = "GET" | "HEAD" | "PUT" | "POST" | "DELETE" | "LINK" | "UNLINK" |
дополнительный-метод
```

Список методов, допустимых конкрутным ресурсом, может быть указан в поле **Заголовок-Содержание**. Тем не менее, клиент всегда оповещается сервером через код статуса ответа, допускается ли применение данного метода для указанного ресурса, так как допустимость применения различных методов может динамически изменяться. Если данный метод известен серверу, но не допускается для указанного ресурса, сервер должен вернуть код статуса "405 Method Not Allowed", и код статуса "501 Not Implemented", если метод не известен или не поддерживается данным сервером.
Общие методы HTTP/1.0 описываются ниже.

2.4. GET

Метод GET служит для получения любой информации, идентифицированной URI-Запросом. Если URI-Запрос ссылается на процесс, выдающий данные, в качестве ответа будут выступать данные, сгенерированные данным процессом, а не код самого процесса (если только это не является выходными данными процесса).

Метод GET изменяется на "условный GET", если сообщение запроса включает в себя поле заголовка "If-Modified-Since". В ответ на условный GET, тело запрашиваемого ресурса передается только, если он изменялся после даты, указанной в заголовке "If-Modified-Since".

Алгоритм определения этого включает в себя следующие случаи:

- Если код статуса ответа на запрос будет отличаться от "200 OK", или дата, указанная в поле заголовка "If-Modified-Since" некорректна, ответ будет идентичен ответу на обычный запрос GET.
- Если после указанной даты ресурс изменялся, ответ будет также идентичен ответу на обычный запрос GET.
- Если ресурс не изменялся после указанной даты, сервер вернет код статуса "304 Not Modified".

Использование метода условный GET направлено на разгрузку сети, так как он позволяет не передавать по сети избыточную информацию.

2.5. HEAD

Метод HEAD аналогичен методу GET, за исключением того, что в ответе сервер не возвращает Тело- Ответа. Метаинформация, содержащаяся в HTTP заголовках ответа на запрос HEAD, должна быть идентична информации HTTP заголовков ответа на запрос GET. Данный метод может использоваться для получения метаинформации о ресурсе без передачи по сети самого ресурса. Метод "Условный HEAD", аналогичный условному GET, не определен.

2.6. POST

Метод POST используется для запроса сервера, чтобы тот принял информацию, включенную в запрос, как субординантную для ресурса, указанного в Строке Статус в поле URI-Запроса. Метод POST был разработан, чтобы была возможность использовать один общий метод для следующих функций:

- Аннотация существующих ресурсов
- Добавление сообщений в группы новостей, почтовые списки или подобные группы статей
- Доставка блоков данных процессам, обрабатывающим данные
- Расширение баз данных через операцию добавления
- Реальная функция, выполняемая методом POST, определяется сервером и обычно зависит от URI- Запроса.
- Добавляемая информация рассматривается как субординатная указанному URI в том же смысле, как файл субординатен каталогу, в котором он находится, новая статья субординатна группе новостей, в которую она добавляется, запись субординатна базе данных.

Клиент может предложить URI для идентификации нового ресурса, включив в запрос заголовок "URI". Тем не менее, сервер должен рассматривать этот URI только как совет и может сохранить тело запроса под другим URI или вообще без него.

Если в результате обработки запроса POST был создан новый ресурс, ответ должен иметь код статуса, равный "201 Created", и содержать URI нового ресурса.

2.7. PUT

Метод PUT запрашивает сервер о сохранении Тело-Запроса под URI, равным URI-Запроса. Если URI-Запроса ссылается на уже существующий ресурс, Тело-Запроса должно рассматриваться как модифицированная версия данного ресурса. Если ресурс, на который ссылается URI-Запроса не существует, и данный URI может рассматриваться как описание для нового ресурса, сервер может создать ресурс с данным URI. Если был создан новый ресурс, сервер должен информировать направившего запрос клиента через ответ с кодом статуса "201 Created". Если существующий ресурс был модифицирован, должен быть послан ответ "200 OK", для информирования клиента об успешном завершении операции. Если ресурс с указанным URI не может быть создан или модифицирован, должно быть послано соответствующее сообщение об ошибке. Фундаментальное различие между методами POST и PUT заключается в различном значении поля URI-Запроса. Для метода POST данный URI указывает ресурс, который будет управлять информацией, содержащейся в теле запроса, как неким придатком. Ресурс может быть обрабатывающим данные процессом, шлюзом в какой-нибудь другой протокол, или отдельным ресурсом, допускающим аннотации. В противоположность этому, URI для запроса PUT идентифицирует информацию, содержащуюся в Содержание-Запроса. Использующий запрос PUT точно знает какой URI он собирается использовать, и получатель запроса не должен пытаться применить этот запрос к какому-нибудь другому ресурсу.

2.8. DELETE

Метод DELETE используется для удаления ресурсов, идентифицированных с помощью URI-Запроса. Результаты работы данного метода на сервере могут быть изменены с помощью человеческого вмешательства (или каким-нибудь другим способом). В принципе, клиент никогда не может быть уверен, что операция удаления была выполнена, даже если код статуса, переданный сервером, информирует об успешном выполнении действия. Тем не менее, сервер не должен информировать об успехе до тех пор, пока на момент ответа он не будет собираться стереть данный ресурс или переместить его в некоторую недостижимую область.

2.9. LINK

Метод LINK устанавливает взаимосвязи между существующим ресурсом, указанным в URI-Запроса, и другими существующими ресурсами. Отличие метода LINK от остальных методов, допускающих установление ссылок между документами, заключается в том, что метод LINK не позволяет передавать в запросе Тело-Запроса, и в том, что в результате работы данного метода не создаются новые ресурсы.

2.10. UNLINK

Метод UNLINK удаляет одну или более ссылочных взаимосвязей для ресурса, указанного в URI- Запроса. Эти взаимосвязи могут быть установлены с помощью метода LINK или какого-нибудь другого метода, поддерживающего заголовок "Link". Удаление ссылки на ресурс не означает, что ресурс прекращает существование или становится недоступным для будущих ссылок.

2.11. Поля Заголовок-Запроса

Поля Заголовок-Запроса позволяют клиенту передавать серверу дополнительную информацию о запросе и о самом клиенте.

```
Заголовок-Запроса = Accept | Accept-Charset | Accept-Encoding |
                    Accept-Language | Authorization | From |
                    If-Modified-Since |
                    Pragma | Referer | User-Agent | extension-header
```

Кроме того через механизм расширения могут быть определены дополнительные заголовки; приложения, которые их не распознают, должны трактовать эти заголовки, как Заголовок-Содержание. Ниже будут рассмотрены некоторые поля заголовка запроса.

2.11.1. FROM

В случае присутствия поля From, оно должно содержать полный E-mail адрес пользователя, который управляет программой-агентом, осуществляющей запросы. Этот адрес должен быть задан в формате, определенном в RFC 822. Формат данного поля следующий: From = "From" ":" спецификация адреса.

Например:
From: webmaster@WWW.org

Данное поле может быть использовано для функций захода в систему, а также для идентификации источника некорректных или нежелательных запросов. Оно не должно использоваться, как несекретная форма разграничения прав доступа. Интерпретация этого поля состоит в том, что обрабатываемый запрос производится от имени данного пользователя, который принимает ответственность за применяемый метод. В частности, агенты-роботы должны использовать этот заголовок для того, чтобы можно было связаться с тем человеком, который отвечает за работу робота, в случае возникновения проблем. Почтовый Internet адрес, указывающийся в этом поле, не обязан соответствовать адресу того хоста, с которого был послан данный запрос. По возможности, адрес должен быть доступным Internet адресом вне зависимости от того, является ли он в действительности Internet E-mail адресом или Internet E-mail представлением адреса других почтовых систем.
Замечание: Клиент не должен использовать поле заголовка From без позволения пользователя, так как это может войти в конфликт с его частными интересами или с местной, используемой им, системой безопасности. Настоятельно рекомендуется предоставление пользователю возможности запретить, разрешить или модифицировать это поле в любой момент перед запросом.

2.12. If-Modified-Since

Поле заголовка If-Modified-Since используется с методом GET для того, чтобы сделать его условным: если запрашиваемый ресурс не изменялся во времени, указанного в этом поле, копия этого ресурса не будет возвращена сервером; вместо этого, будет возвращен ответ "304 Not Modified" без Тела-Ответа.

If-Modified-Since = "If-Modified-Since" ":" HTTP-дата

Пример использования заголовка:
If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT

Целью этой особенности является предоставление возможности эффективного обновления информации локальных кэшей с минимумом передаваемой информации. Тот же результат может быть достигнут применением метода HEAD с последующим использованием GET, если сервер указал, что содержимое документа изменилось.

2.13. User-Agent

Поле заголовка User-Agent содержит информацию о пользовательском агенте, пославшем запрос. Данное поле используется для статистики, прослеживания ошибок протокола, и автоматического распознавания пользовательских агентов. Хотя это не обязательно, пользовательские агенты должны всегда включать это поле в свои запросы. Поле может содержать несколько строк, представляющих собой название программного продукта, необязательную косую черту с указанием версии продукта, а также другие программные продукты, составляющие важную часть пользовательского агента. По соглашению, продукты указываются в списке в порядке убывания их значимости для идентификации приложения.

```
User-Agent = "User-Agent" ":" 1*( продукт )
продукт = строка ["/" версия-продукта]
версия-продукта = строка
```

Пример:

```
User-Agent: CERN-LineMode/2.15 libwww/2.17b3
```

Строка, описывающая название продукта, должна быть короткой и давать информацию по существу - использование данного заголовка для рекламирования какой-либо другой, не относящейся к делу, информации не допускается и рассматривается, как не соответствующее протоколу. Хотя в поле версии продукта может присутствовать любая строка, данная строка должна использоваться только для указания версии продукта. Поле User-Agent может включать в себя дополнительную информацию в комментариях, которые не являются частью его значения.

3. HTTP ответ

3.1. Структура ответа

После получения и интерпретации запроса, сервер посылает ответ в соответствии со следующей формой:

```
Ответ = Простой-Ответ | Полный-Ответ
Простой-Ответ = [ Содержание-Ответа ]
Полный-Ответ = Строка-Статус
        *( Общий-Заголовок | Заголовок-Ответа | Заголовок-Содержания) CRLF
        [ Содержание-Ответа ]
```

Простой-Ответ должен посылаться только в ответ на HTTP/0.9 Простой-Запрос, или в том случае, если сервер поддерживает только ограниченный HTTP/0.9 протокол. Если клиент посылает HTTP/1.0 Полный-Запрос и получает ответ, который не начинается со Строки-Статус, он должен предполагать, что ответ сервера представляет собой Простой-Ответ, и обрабатывать его в соответствии с этим. Следует заметить, что Простой-Ответ состоит только из запрашиваемой информации (без заголовков) и поток данных прекращается в тот момент, когда сервер закрывает сеанс связи.

3.2. Строка статуса

Первая строка Полного-Запроса является Строкой-Статус, состоящей из версии протокола, за которой следует цифровой код статуса и ассоциированное с ним текстовое предложение. Все элементы Строки-Статус разделены пробелами. Не разрешены символы CR и LF, за исключением завершающей последовательности CRLF.
Строка-Статус=Версия-HTTP SP Статус-Код SP Фраза-Об'яснение.
Так как Строка-Статус всегда начинается с версии протокола "HTTP/" 1*ЦИФРА "." 1*ЦИФРА (например HTTP/1.0), существование этого выражения рассматривается как основное для определения того, является ли ответ Простым-Ответом, или Полным-Ответом. Хотя формат Простого-Ответа не исключает появления подобной строки (что привело бы к неправильной интерпретации сообщения ответа и принятию его за Полный-Ответ), вероятность такого появления близка к нулю.

3.3. Статус-Код и пояснение к нему

Элемент Статус-Код представляет собой 3-х цифровой целый код, идентифицирующий результат попытки интерпретации и удовлетворения запроса. Фраза-Об'яснение, следующая за ним, предназначена для краткого текстового описания Статус-Кода. Статус-Код нацелен на то, чтобы его использовала машина, а Фраза-Об'яснение предназначена для человека. Клиент не обязан исследовать и выводить на экран Фразу-Об'яснение.
Первая цифра Статус-Кода предназначена для определения класса ответа. Последние две цифры не выполняют никакой категоризирующей роли. Существует 5 значений для первой цифры:

1xx: Информационный - Не используется, но зарезервирован для использования в будущем

2xx: Успех - Запрос был полностью получен, понят, и принят к обработке.

3xx: Перенаправление - Клиенту следует предпринять дальнейшие действия для успешного выполнения запроса. Необходимое дополнительное действие иногда может быть выполнено клиентом без взаимодействия с пользователем, но настоятельно рекомендуется, чтобы это имело место только в тех случаях, когда метод, использующийся в запросе безразличен (GET или HEAD).

4xx: Ошибка клиента - Запрос, содержащий неправильные синтаксические конструкции, не может быть успешно выполнен. Класс 4xx предназначен для описания тех случаев, когда ошибка была допущена со стороны клиента. Если клиент еще не завершил запрос, когда он получил ответ с Статус-Кодом- 4xx, он должен немедленно прекратить передачу данных серверу. Данный тип Статус-Кодов применим для любых методов, употребляющихся в запросе.

5xx: Ошибка Сервера - Сервер не смог дать ответ на корректно поставленный запрос. В этих случаях сервер либо знает, что он допустил ошибку, либо не способен обработать запрос. За исключением ответов на запросы HEAD, сервер посылает описание ошибочной ситуации и то, является ли это состояние временным или постоянным, в Содержание-Ответа. Данный тип Статус-Кодов применим для любых методов, употребляющихся в запросе.

Отдельные значения Статус-Кодов и соответствующие им Фразы-Об'яснения приведены ниже. Данные Фразы-Об'яснения только рекомендуются - они могут быть замещены любыми другими фразами, сохраняющими смысл и допускающимися протоколом.

```
Статус-Код = "200" ; OK |
             "201" ; Created |
             "202" ; Accepted |
             "203" ; Provisional Information |
             "204" ; No Content |
             "300" ; Multiple Choices |
             "301" ; Moved Permanently |
             "302" ; Moved Temporarily |
             "303" ; Method |
             "304" ; Not Modified |
             "400" ; Bad Request |
             "401" ; Unauthorized |
             "402" ; Payment Required |
             "403" ; Forbidden |
             "404" ; Not Found |
             "405" ; Method Not Allowed |
             "406" ; None Acceptable |
             "407" ; Proxy Authentication Required |
             "408" ; Request Timeout |
             "409" ; Conflict |
             "410" ; Gone |
             "500" ; Internal Server Error |
             "501" ; Not Implemented |
             "502" ; Bad Gateway |
             "503" ; Service Unavailable |
             "504" ; Gateway Timeout |
```

Код-Рассширения

Код-Расширения = ЦИФРА
Фраза-Об'яснение = строка *(SP строка)

От HTTP приложений не требуется понимание всех Статус-Кодов, хотя такое понимание, очевидно, желательно. Тем не менее, от приложений требуется способность распознавания классов Статус-Кодов (идентифицирующихся первой цифрой) и отношение ко всем Статус-Кодам статуса ответа, как если бы они были эквивалентны Статус-Коду x00.

3.4. Поля Заголовок-Ответа

Поля заголовка ответа позволяют серверу передать дополнительную информацию об ответе, которая не может быть внесена в Строку-Статус. Эти поля заголовков не предназначены для передачи информации о содержании ответа, передаваемого в ответ на запрос, но там может быть информация собственно о сервере.
Заголовок-Ответа= Public | Retry-After | Server | WWW-Authenticate | extension-header
Хотя дополнительные поля заголовка ответа могут быть реализованы через механизм расширения, приложения, которые не распознают эти поля, должны обрабатывать их как поля Заголовок-Содержание. Полное описание этих полей можно получить в спецификации протокола HTTP в CERN.

4. Содержание запроса и ответа

Как видно на рисунке все транзакции между клиентом и сервером происходят на верхнем уровне сетевой иерархии. Все HTTP-транзакции имеют один общий формат. Каждый запрос клиента и ответ сервера состоит из трех частей: строки запроса (ответа), раздела заголовка и тела.

Клиент инициирует транзакцию следующим образом:
Клиент устанавливает связь с сервером по назначенному номеру порта (по умолчанию - 80). Затем посылается запрос документа с указанием HTTP комманды (называемой методом), адреса документа и номера версии HTTP.

Например:
GET /index.html HTTP/1.0

Клиент посылает необязательную информацию заголовка, чтобы сообщить серверу информацию о своей конфигурации и данные о форматах документов, которые он может принимать. Вся информация заголовка указывается построчно и каждая строка содержит пару имя-значение. Заголовок завершается пустой строкой.

Например:
User-Agent: Mozilla/2.02Gold (WinNT; I)
Accept: image/gif, image/jpeg, */*

Послав запрос и заголовки, клиент может отправить дополнительные данные (тело запроса). Эти данные используются, например, CGI-программами, применяющими метод POST.

Сервер отвечает на запрос клиента следующим образом:

Первая часть ответа - строка состояния, содержащая версию протокола HTTP, код состояния и описание, которое представляет собой просто понятный для человека текст, поясняющий код состояния.

Например: HTTP/1.0 200 OK

После строки состояния сервер передает клиенту информацию заголовка ответа, содержащую данные о самом сервере и затребованном документе. Завершает заголовок пустая строка.

Например:
Date: Fri, 20 Sep 1996 08:17:58 GMT
Server: NCSA/1.5.2
Last-modified: Mon, 17 Jun 1996 21:53:08 GMT
Content-type: text/html
Content-length: 2482

Если запрос клиента успешен, то сервер посылает затребованные данные. Это может быть копия файла или документ сформированный "на лету". Если запрос клиента удовлетворить нельзя, то сервер передает дополнительные данные в виде удобного для человека разъяснения причин, по которым сервер не смог выполнить запрос.

В HTTP 1.0 после передачи сервером затребованных данных следует разъединение с клиентом и транзакция завершается, если не был передан заголовок Connection: Keep Alive. В HTTP 1.1 сервер по умолчанию не разрывает соединение и клиент может посылать другие запросы. Это позволяет сэкономить время и затраты клиента, которому не приходится заново соединяться с тем же сервером. Таким образом, в HTTP 1.1 транзакция может циклически повторяться, пока клиент или сервер не закроет соединение явно.

5. Методы

Метод - это HTTP-комманда, с которой начинается первая строка запроса клиента. Метод сообщает серверу о цели запроса. Чаще всего используются методы GET, HEAD и POST (регистр имеет значение).

5.1. GET

Этим методом запрашивается информация, расположенная на сервере в указанном в запросе месте. Методом GET web браузеры обычно получают документы для визуализации. Результатом такого запроса может быть, например, файл, лежащий на сервере или же сформированный специально для этого запроса.

Например:
GET / HTTP/1.0
User-Agent: Simple Web client by Eugen Kuleshov
Accept: */*

5.2. HEAD

Метод HEAD аналогичен методу GET, за исключением того, что сервер не посылает тело ответа, а только информацию заголовка о запрошенном файле или ресурсе. Обычно этот метод используется для получения информации о документе не получая документ.

Можно, например, затребовать следующую информацию:

- время изменения документа;
- размер документа;
- тип документа;
- тип сервера.

Следует отметить, что большая часть посылаемой сервером информации заголовка, не является обязательной и может представляться не всеми web серверами.

Например:
HEAD / HTTP/1.0
User-Agent: Simple Web client by Eugen Kuleshov
Accept: */*

5.3. POST

Этот метод позволяет посылать на сервер данные в запросе клиента. Эти данные могут быть использованы сервером для динамической генерации запрашиваемого документа, например для передачи входных данных для:

- сетевых служб (например телеконференций)
- программ с интерфейсом типа "коммандная строка"
- выполнения операций в базах данных

Например:

```
POST / HTTP/1.0
User-Agent: Simple Web client by Eugen Kuleshov
Accept: */*
Content-type: application/x-www-form-urlencoded
Content-length: 16

test=20&test2=22
```

Обратите внимание на атрибуты Content-type и Content-length они используются для того, чтобы указать серверу на тип кодирования тела запроса и дать информацию о длине тела.