DIGITAL PAYMETER API

(WORKING DRAFT)

OVERVIEW

The Digital Paymeter system offers two programming interfaces into the content metering rules engine. The first of these is tailored to the current Digital Paymeter HTML-based "lightbox" solution where all user facing elements are controlled by Syncronex/Dti. This is the default solution typically put in place to meter browser-based content (aka – web sites).

The second interface is specific to external systems (native smart device applications, replica edition systems, and other non-Syncronex/Dti systems wishing to leverage the consistent set of content metering business rules established by the Newspaper customer.

The following sections describe these two interfaces. The *Default Interface* is the HTML-based solution used exclusively by the current Syncronex/Dti lightbox features and the newer *Standard Interface* is to be used by external systems.

DEFAULT SERVICE

The out-of-box functionality (light box overlay, etc.) relies on an internal web-based programming interface to handle the meter processing rules. This is a web service endpoint that is callable from any client capable of making http GET requests to a web server. However, this endpoint is currently tied to the Digital Paymeter default light box mechanism. Because of this, the inputs and outputs are *not* generic in nature. The server's response is assuming that the caller is a JavaScript function running on the end-user's browser from within a specific content page.

ENDPOINT

```
https://server:port/{co}/{property}/API/svcs/meter
```

{co} and {property} are specific to a customer and determined at implementation time.

INPUT

The default API endpoint expects the following input arguments (as query string arguments)

sessionId	A base64 encoded GUID value that identifies the current anonymous user making the request for content. If blank (or if given id is not found) the server will assume this is a user

	never seen before and will issue a new id (see OUTPUT section below)
contentId	A text value indicating the category of content being viewed.
externalId	*Reserved*
referrer	The referring page. System uses this value in order to process any exempted referrers logic. For instance, links to content via Yahoo.com might be exempt from any metering.
page	The specific page being viewed. This is used to populate dynamic overlay content with appropriate return or cancel url links (so the user can be returned to the content she was attempting to view after going through a sign-in or registration process.
formFactor	(optional). Server will use this to decide which dynamic overlay content to send back to caller. Only significant value is "mobile". Any other value (or no value at all) results in the default overlay content.
callback	Used by the calling JavaScript to indicate the JSONP callback function to invoke. The payload returned from the method call is a JSONP package (see OUTPUT)

EXAMPLE

Used in an html script tag:

```
<script type='text/javascript'
src='https://stage.syncaccess.net/sync/demo/API/svcs/meter?sessionid=ZTY3ODY1M2EtMjdm
ZC00ZTQ0LThiNzEtMGZjMDhjZjdhNjYx&contentId=premium&externalId=&referrer=http://www.go
ogle.com&page=http://www.acmenews.com/topstories/foo.htm&callback=serverCallback >
```

OUTPUT

Server method responds with a JSONP payload. For example:

```
serverCallback( {
    authorized:false,
    sessionIdentifier: ZTY3ODY1M2EtMjdmZC00ZTQ0LThiNzEtMGZjMDhjZjdhNjYx,
    overlayContent: "<div><div>…</div></div>"
} )
```

Authorized	Text value: 'true' or 'false' denoting whether or not the request to view content was allowed.
sessionIdentifier	Base64 encoded GUID used to identify the anonymous user.
overlayContent	Blank when **authorized** is 'true'. Otherwise, this contains an arbitrary text value representing the raw html snippet that is used to render an overlay. It's assumed that the caller has an appropriate element in which to inject this snippet. Since the target element is also assumed to be an html element within an existing DOM (namely the current page being viewed) the overlay content snippet must not contain html tags like **<html>, <head>,<body>**

NOTES

As noted previously, this default API endpoint is tied to a specific calling mechanism. The caller in this case is the Digital Paymeter JavaScript code that is loaded into a newspaper content page via an html script tag. This is part of the standard *Integration between CMS and Digital Paymeter* setup. This specific JavaScript code gathers up specific local information (session id, page, referrer, etc.) makes the call to this default endpoint and then parses the response. In the event that the response contains an overlay, the calling script code creates and injects a couple of special html `<div>` elements that control the behavior of the light box. The script takes any overlay content returned by the server and injects it directly into the innermost of these `<div>` tags and ultimately appends the new html content to the current page. The result is that the browser immediately renders the light box and its content.

STANDARD SERVICE

The existing API is cumbersome for arbitrary third party applications due to its specificity. It's acknowledged that a more generalized API needs to be considered to provide more flexibility to customers while still leveraging the centralized rules processing engine. In this way, multiple external sources can all present a consistent set of behaviors to end users. For instance, a customer might wish to allow a *metered* approach to content from native device (iPhone, Android) that is the same as the behavior from a web browser.

The following sections describe the new *standard* metering API to accommodate external systems.

ENDPOINT

`https://server:port/{co}/{property}/API/svcs/meter/standard`

INPUT

The standard API endpoint expects the following input arguments (as the body of the http POST). These values can be supplied as a Json payload provide the caller is able to designate the appropriate *http content-type header*.

sessionId	A base64 encoded value that can be persisted client-side (i.e. as a cookie) that contains an anonymous GUID identifier (as in default API) .
userID	(Optional*). If the calling system has the internal UserId for the current user (i.e., following a successful login request via the *SubscriberInfo* API Method), it can be provided here to have the system authorize the *known* user instead of the *anonymous* user. * This property is only optional until the user triggers a paywall event. After that, the user will not be authorized via her anonymous sessionId.
contentId	A text value indicating the category of content being viewed. This would be identical to the existing default API. The server needs to know what content is being viewed to correctly process authorization rules.
externalId	*Reserved*
referrer	(optional). Administrators can configure the metering rules to exempt a user's view when that view is sourced from a search engine, social media link, etc. External systems can leverage this mechanism by providing this optional *referrer*. This *must* be a valid (in form) URI but does not necessarily have to point to any physical resource. I.e., *http://this.doesnt.exist/here.htm* The key aspect of the referrer is the host/domain portion (*this.doesnt.exist*) The domain value here is checked against a list of exempted referring domains on the server.
clientInfo	The default API (since it's assuming the caller is ultimately a browser) leverages the *User Agent* data sent as part of the HTTP Get request. A generic API (one that can't assume the nature of the client) would expect to get additional properties that can be used in the metering process. For instance, a device type, some kind of device identifier, etc. *The specifics of this information still need to be determined.*

OUTPUT

The standard API responds with discrete values that describe the results of the authorization request (this is in contrast to the default API that encodes many of these discrete values into the arbitrary html code snippet returned)

sessionIdentifier	Server will still respond with an encoded session identifier. The client can then persist this as necessary and use it in subsequent calls. The newer endpoint would likely include a subscriber id along with (or in place of) the anonymous session id.
statusCode	A more detailed authorization result beyond just a simple 'true' or 'false'. • 0 – Success. User is authorized for requested content • 100 – Warning. Anonymous user exceeded a warning threshold • 101 – Warning. View count not incremented due to exempt referrer • 102 – Warning. View count not incremented due to exempt IP • 103 – Warning. User is authorized but we need to issue them a warning • 200 – Failure. Anonymous user exceeded meter • 201 – Failure. Valid user has insufficient subscription level • 202 – Failure. Error • 203 – Failure. User is not authorized for requested content
statusMsg	Text value accompanying the status code that can be used by client to display info to end user.
viewCount	User's current number of views of the given content
remainingViews	Number of views remaining before hard pay wall event is triggered (a *StatusCode* of 200).
Authorized	Simple 'true'/'false' value that can be used in lieu of the *StatusCode* if the external system is simply looking for yes or no answer. *Note that a warning event (status code 100) results in a value of 'false' here.*
registerUrl	A complete Url that points to the appropriate registration page for the given system. Calling systems can use this value to push the user out to a browser so she can register for a new account.

EXAMPLE

A request similar to this:

https://stage.syncaccess.net/sync/demo2/api/svcs/meter/standard?format=json

with the following content (here using Json):

```
{
  "sessionId":"YTliMzYyYTktMWMyZC00NTc0LWE4NWMtN2JkMTA2YjAyMGQ3",
  "contentId":"demoContent",
  "referrer":"http://www.syncronex.com/page1.html",
  "clientInfo":"iPad Device"
```

```
}
```

Would result in data similar to this:

```
{
    "sessionIdentifier": "YTliMzYyYTktMWMyZC00NTc0LWE4NWMtN2JkMTA2YjAyMGQ3",
    "statusCode": "0",
    "statusMsg": "Success. User is authorized for requested content",
    "viewCount": "2",
    "remainingViewCount": "3",
    "authorized": "true",
    "registerUrl":
          "https://stage.syncaccess.net:443/sync/demo2/api/account/register"
}
```