EIDR

Registry Version Compatibility

2012-02-April

Document Version 0.5

TABLE OF CONTENTS

1 Overview

Every effort is made to maintain backwards compatibility with prior versions each time an update to the EIDR Registry is released, but this is not always possible. New features (procedures, attributes, controlled vocabularies, etc.) may be introduced that did not exist in prior versions while existing features may have to be changed in a significant way. To minimize the impact on existing SDK or API integration efforts, each new Registry release includes backwards compatibility support for the two most recent prior versions as outlined below. Users of these and older versions of the Registry integration tools will be identified through activity logs and notified in advance of changes that might have a material impact on their operation.

Terminology

Client – Software that connects to the EIDR Registry, making requests and receiving back responses, generally by integrating an EIDR SDK into a third party application or by calling the EIDR REST APIs directly.

Deprecated – A feature that is no longer part of the current version of the Registry or SDK, but that is still supported for a period of time in the interests of backwards compatibility. In general, the prior two versions are supported.

Discontinued – A feature that was previously deprecated, but is now so old that it is no longer supported by the Registry or SDK. In general, features that are more than two versions out of date.

2 Backwards Compatibility

Approach

Generally speaking, the main compatibility concern with each Registry release is to avoid returning an XML response to a client request made using an outdated, but still supported format that cannot be correctly interpreted by the client. The client identifies itself in each request to the registry using a header with a value indicating its version number.

> NOTE: If the client request is made using a discontinued version or a version for which a down-converted format cannot be produced, then a Compatibility Error is returned rather than run the risk of returning an XML response that might well be misinterpreted.

If a client makes a request with a release number for which the current response is incompatible in schema or semantics, then if it is possible to do so, the response is down-converted to a format that the client can accept. However, if the current response can be accepted by the client, then that is returned.

Client/Registry Version Support

This table covers the various relationships that may exist between client request and Registry versions and describes how the Registry will respond in each case.

> NOTE: The client identifies its version in each HTTP GET or POST request to the registry using a header called "EIDR-Version" with a value indicating its release as a set of integers separated by dots, having the form <integer>[.<integer>]*, where only the first two integers are significant for registry compatibility.

Client/Registry Relationship	**Request Compatible with Registry Version** NOTE: Response is as expected by the client or contains an expanded XML payload that can be ignored by the client.	**Request Incompatible with Registry Version** NOTE: The syntax or semantics of the new response is unintelligible to the old client.	**Request Does Not Exist in Current Registry Version** NOTE: A feature or function that is no longer supported by the Registry. For the reverse, where the Registry includes features unknown to the client, see Client/Request Mismatch.
Client Version < Registry Version Client versions will be supported through two subsequent Registry releases before being discontinued.	Registry responds using its current format with the current schema version in the header. If the request is unchanged between client and Registry versions, then the client will read the full response correctly. If the current Registry response includes new semantics not defined in the client version, the	If it is possible to do so, the Registry responds with a down-converted format using the newest schema that is compatible with the client version. Down-conversions are not always perfect. For example, if a 1.0.4 client registers an Edit through the 1.1 Registry with Edit Class = Ship and then reads that same record,	If the client request has been deprecated by the Registry, but not yet discontinued, the Registry responds with a down-converted format using the newest schema that is compatible with the client version, if one is available. Otherwise, the Registry responds with a Compatibility Error.

Client/Registry Relationship	Request Compatible with Registry Version	Request Incompatible with Registry Version	Request Does Not Exist in Current Registry Version
	client will ignore the additional material and correctly interpret the remainder of the response.	the Edit Class will be Airline. If an acceptable down-conversion is not available, then the Registry will respond with a Compatibility Error.	
Client Version = Registry Version The ideal state.	Standard operating conditions.	An error state that should not occur, so the Registry simply returns its current response.	The Registry responds with a Compatibility Error.
Client Version > Registry Version This is an exception that should never exist in production.	Returns a Compatibility Error. NOTE: An older Registry cannot possibly know if its intended response is compatible with a newer client request, or be expected to respond intelligibly to a request it does not recognize. This is an unusual condition that should only occur when someone tries to test an in-development SDK version against an earlier Registry, so it fails immediately.		

Client/Request Mismatch

A client could make a request that is not supported by its stated version, but that is supported by the Registry. For example, a client advertising itself as 1.0.4 could make a request that was not introduced until 1.1.* In this case, the Registry presumes that the client is prepared to receive the current response. It is up to the client to interpret it correctly.

* This could happen for a number of valid, if unusual, reasons: The client may have linked two different SDK versions to add new features to an existing system; a client built with an older SDK may include hand-coded calls for new features; or a client making direct REST API calls may have accidentally submitted the wrong version information.

Error States

Further details regarding Compatibility Errors, including the official error number, label, and any additional information are described in the *EIDR User's Guide* and *API Overview* (available at http://eidr.org/resources/).

3 Deprecated/Discontinued Clients

Deprecated Clients Versions

All EIDR Members are notified each time a new Registry version is released. With each release, the previous client version is deprecated. Any direct users of the now deprecated client will be identified in the Registry logs and advised to upgrade to the new client at their earliest convenience.

Deprecated Features in an SDK

To simplify the upgrade process, a new SDK version may contain deprecated APIs from prior versions. These deprecated calls can include both Registry features and SDK-specific changes such as updated method signatures. If included in a client integration, these will return deprecation warnings during compilation. Subsequent versions of the SDK will not include deprecated features that are more than two versions out of date.

Discontinued Client Versions

Prior to discontinuing a client version, current users will be identified based on their activity in the Registry logs and given at least two months' advance notice that they must upgrade their client to a newer version.

The EIDR Web Site

The Web UI provided by EIDR for access to the Registry (https://ui.eidr.org/) always matches the version of the Registry and does not provide backwards compatibility with prior client versions.