Windows Azure Resource Provider API

Draft – APIs are subject to change

[1. Overview 3](#_Toc335863053)

[2. About the Azure Store 3](#_Toc335863054)

[A. Introduction 3](#_Toc335863055)

[B. User Experience 3](#_Toc335863056)

[C. Onboarding Process 7](#_Toc335863057)

[3. Terminology 7](#_Toc335863058)

[A. Illustrated Example 7](#_Toc335863059)

[B. Subscription 7](#_Toc335863060)

[C. Resource Provider 7](#_Toc335863061)

[D. Resource Type 8](#_Toc335863062)

[E. Resource 8](#_Toc335863063)

[F. Cloud Service 8](#_Toc335863064)

[4. API Contract 9](#_Toc335863065)

[G. Subscription Notifications 9](#_Toc335863066)

[H. Provisioning Resources 11](#_Toc335863067)

[Response Body Not Conforming to Contract 19](#_Toc335863068)

[I. Change Management Using ETags 19](#_Toc335863069)

[J. Updates to an Existing Resource 20](#_Toc335863070)

[K. Gets on Existing Resources 21](#_Toc335863071)

[L. Deletes on Existing Resources 23](#_Toc335863072)

[Response Body & Dealing with Failures 24](#_Toc335863073)

[5. Resource Quotas 25](#_Toc335863074)

[6. Billing Events 25](#_Toc335863075)

[7. Azure to Resource Provider Communication 25](#_Toc335863076)

[8. Single Sign-on (SSO) 26](#_Toc335863077)

[9. Change Log 27](#_Toc335863078)

# Overview

Azure provides an extensible platform that supports onboarding partner services which users can purchase through the Azure Store. Azure end users get access to great services built by partners, and partners gain access to a set of cloud developers and a revenue stream.

This document provides a brief overview of the Azure Store and the business process required for onboarding to it. Mostly, however, it focuses on technical implementation details.

Throughout this document, we will use example of a fictitious Contoso Corporation, which sells a cloud database service known as ContosoDB through the Azure Store.

The APIs and user experience shown in this document are **not** **final and are subject to change**.

# About the Azure Store

## Introduction

The Azure Store is a one stop shop for developer services and premium datasets that can be used with and for Windows Azure.

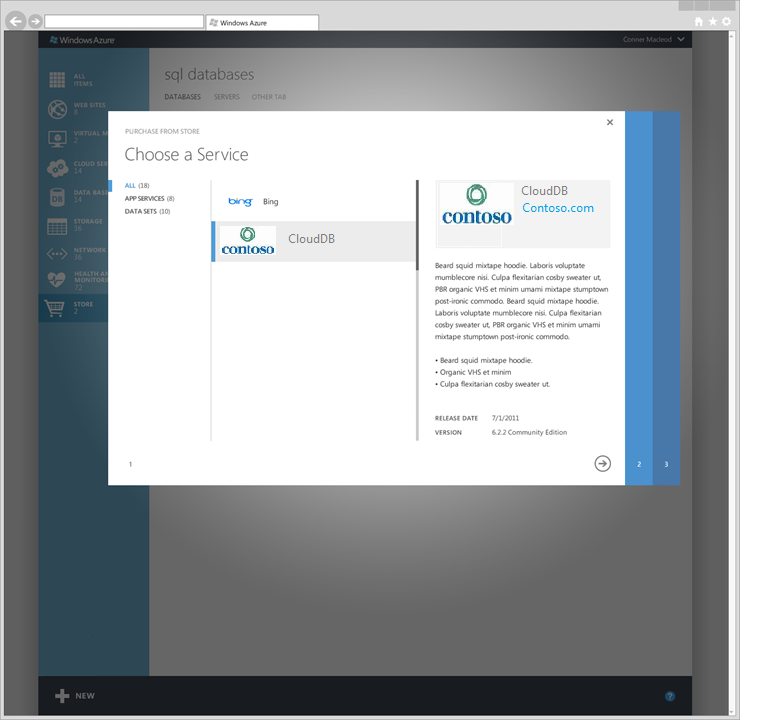
With the Azure Store, developers will have the ability to provision, manage, and scale 1st party and 3rd party developer services and premium datasets from right within their Windows Azure management experience. Hence, the Store represents a unique opportunity for sellers of cloud services to connect with a global pool of Windows Azure developers**.**

In the first release, the Azure Store only supports prepaid price tiers. For example, Contoso can define three plans: Free ($0.00), Silver ($9.99) and Gold ($29.99). Each plan is paid per month by the user, and there is no proration.

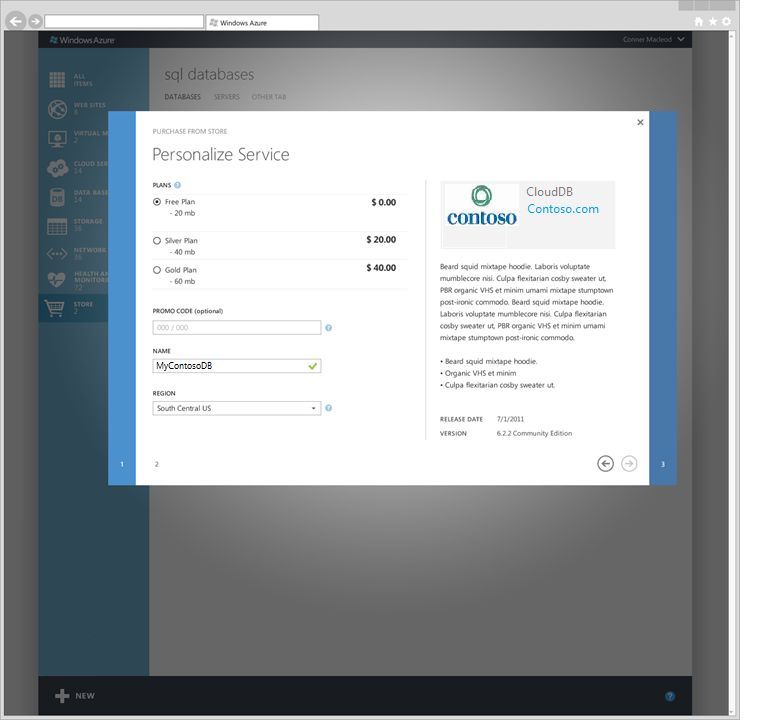
## User Experience

The Azure Store user experience (UX) will be into the user experience for the Azure Developer Portal, making purchasing items from the Azure Store as simple as creating a Website or Storage Account. The user will purchase services through a wizard.

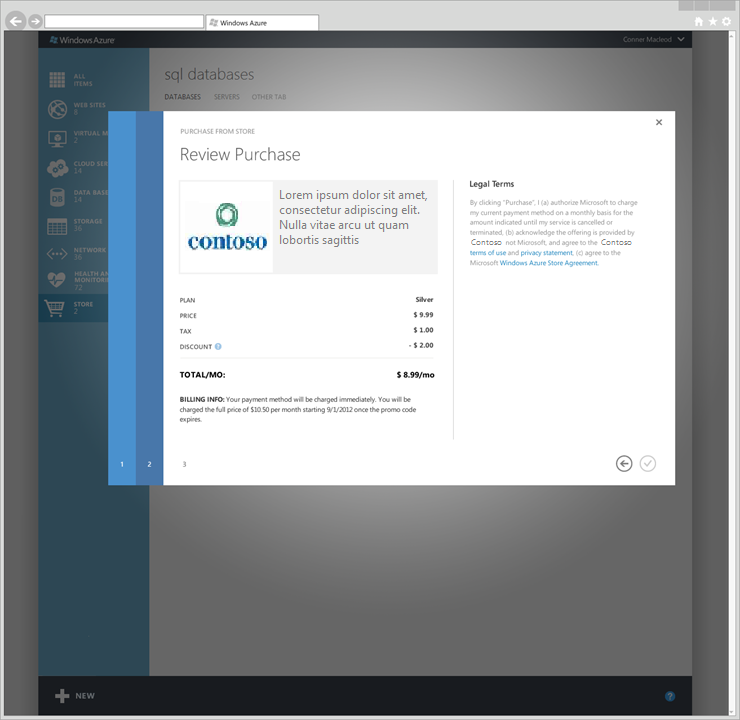
The first page of the wizard lists offers from partners:



The second page allows the user to choose a partner-defined plan, and name the resource. Note that partners can create free as well as paid plans:



The last page of wizard gives the user a chance to review the purchase, including the partner’s terms of service and privacy statement:



## Onboarding Process

# Terminology

## Illustrated Example

Abby has a single “Pay-as-you-go” **subscription** on Azure. She purchases a ContosoDB from the Azure Store UI. The Contoso **Resource Provider** receives a set of requests, which it responds to by creating a ContosoDB **resource**, which is an instance of the ContosoDB **Resource Type.** The Azure Developer Portal places this **resource** inside a **Cloud Service**. Contoso **Resource Provider** might have other resource types like ContosoQueue which the user can also provision.

## Subscription

Every Azure user is required to purchase a subscription before they can use any resources. Resources created within a single subscription are part of a single bill. A user may have several subscriptions e.g. “Pay-as-you-go” and “3-month free trial”, but typically users have at most one or two subscriptions. In the Azure Developer Portal, the subscription name is displayed in various locations.

Resource names within a Cloud Service must be unique.

## Resource Provider

All resources in Azure are backed by a Resource Provider (RP). An RP has to implement the RESTful contract described in this document. Conceptually, an RP is responsible for creating resources of a particular Resource Type.

Intrinsic settings are parameters that control the behavior of the RP, and they can be versioned independently and transparently by the RP. For example, an RP that has a database Resource Type may have three intrinsic settings: Database Name, Username and Password. Going forward they may have a new version which supports connection\_urls. Using versions they can manage multiple resources in Azure with different versions having different intrinsic settings.

Each resource provider is assigned a unique namespace within the Azure management URI. Resource Provider namespaces must be alphanumeric and special characters are not allowed. We suggest using concise English names since the namespace is exposed to users. Once assigned, the namespace cannot be changed.

## Resource Type

A single resource provider may expose several logical units of functionality, each of which may be instantiated separately. Each such unit is called a Resource Type. For example, a data storage RP might expose a structured storage Resource Type and another unstructured storage Resource Type. The customer can pick between the two kinds of storage by instantiating the appropriate Resource Type.

## Resource

A resource is the instantiation of a Resource Type provided by an RP. Resources are instantiated in the context of a Cloud Service.

## **Cloud Service**

Azure exposes the notion of a *Cloud Service* which is a container for multiple resources. Conceptually, a Cloud Service represents a “project” or an “application” since it is a logical container for a set of resources which are surfaced by one or more resource providers. A subscription may have several Cloud Services and every resource is created within a Cloud Service.

The concept of Cloud Service will not be visible to users in the first release of Azure Store. The term itself is subject to change.

Windows Azure will provide a Cloud Service ID to the RP hosting resources that belong to the Cloud Service. The provider in turn can use this ID to group resources and return an aggregate list of resources:



# API Contract

This section describes the interfaces between Azure and RPs.

## Subscription Notifications

RPs receive notifications regarding subscriptions that have been enabled to use that provider. These notifications are generated as a subscription goes through its lifecycle.

|  |  |  |
| --- | --- | --- |
| Notification | Resource provider action | Example |
| Registered | The subscription was entitled to use resources belonging to this RP. This happens when a user purchases an item from the Azure Store.  The RP must remember this event since Azure will use this subscription in future communications. The RP must perform any required processing before responding.  The registered event will also carry the Email address of the logged in user and the OptIn flag. The OptIn flag will let you know if the user has chosen to accept promotional emails from you or not.  **If the RP does not respond with an accept code(200 or 201) for the first time this event is sent, Azure will not send any resource provisioning call to the RP** | Contoso Corp.’s RP will make an entry in a table, recording that Subscription ID *x* is entitled to create a particular resource. It will also store the email address and the OptIn flag. |
| Disabled | The subscription has been disabled, generally due to fraud or non-payment.  The RP should ideally mark resources as read-only. This will mirror what Windows Azure typically does in case of non-payment. The state returned in the GET calls should be paused. | Contoso does not allow management actions on the user’s data at this point, whether through UI or command line.  Any data or resources already provisioned are maintained while the subscription is in this state.  Contoso’s RP marks Resource Status as “Paused” with additional details in the Sub Status field which indicates the reason. |
| Enabled | A subscription has been enabled (for instance, they have paid an overdue bill).  The RP should now enable full access to resources. | Any restrictions imposed when the subscription was Disabled should be removed. |
| Deleted | The customer has cancelled their Azure subscription.  The RP should clean up all the data within this subscription. Windows Azure data is retained for 90 days even after a delete event is received. The RP should record the retention policy in your terms of service document. | All of the subscription’s data should be cleaned up. |

**API**

|  |  |
| --- | --- |
| Method | Request URI |
| POST | https:// <registered-resource-provider-endpoint>/subscription/<subscription-id>/Events |

**Request Header**

|  |  |
| --- | --- |
| Header | Description |
| Content-Type | Required. Set to **application/xml**. |
| x-ms-version | Required. Specifies the version of the operation to use for this request. The value of this header will be set to 2012-03-01or later. |

**Request Body**

<?xml version="1.0" encoding="utf-8"?>

<EntityEvent>

<EventId>766ed3be-11be-4a88-a7c8-ba4286299066</EventId>

<EntityType>Subscription</EntityType>

<EntityState>Registered</EntityState>

<EntityId>

<Id>f6c18f8a-ab84-4e6d-b410-18710e8ef770</Id>

<Created>2012-10-12T06:42:36.8265209Z</Created>

</EntityId>

<OperationId>ae9a07ef-2306-40e0-bbe5-2821352a8c4d</OperationId>

<Properties>

<EntityProperty>

<PropertyName>ResourceType</PropertyName>

<PropertyValue>gamify</PropertyValue>

</EntityProperty>

<EntityProperty>

<PropertyName>EMail</PropertyName>

<PropertyValue>bharata@live-int.com</PropertyValue>

</EntityProperty>

<EntityProperty>

<PropertyName>OptIn</PropertyName>

<PropertyValue>True</PropertyValue>

</EntityProperty>

</Properties>

</EntityEvent>

|  |  |
| --- | --- |
| Element name | Description |
| EventID | The EventID denotes the Subscription ID for the Azure User that has been authorized to use the Provider. |
| EntityType | Will always be *Subscription*. |
| EntityState | One amongst Enabled | Disabled | Deleted | Registered as explained in the table above. |
| EntityId/Id | Subscription ID – this is the preferred one to use. |
| EntityId/Created | Creation date for the subscription |
| OperationId | A unique identifier for the notification event.  A resource provider **should be prepared to get duplicate notifications**. However, all duplicates will have the same OperationId and if the event has been processed the provider should respond with 200 OK. |
| Properties | This is a property bag and holds the properties listed below in a name value pair |
| Email | This is the email address of the logged in user. |
| OptIn | This is to let the resource provider know if the user has chosen to Opt into receiving promotional offers. |

**Note:**

The Portal will also have a user interface which will allow a user to change his/her email address as well as decide to optin or not.  Every time the user does so, you will get the registered event again.  If you are using the email to create an account, then use the subscriptionid<passed in the url and entity id> to decide whether you should create a new account or update an existing one.

**Response**

Resource providers should persist and process the event and return a HTTP 200 OK. In the event that there are failures, the provider must retry processing the event later. The Resource Provider must return an error response **only** if it needs the event to be retransmitted. The one exception to this rule is the *Registered* event which must be processed first.

Handling of notifications must idempotent as notifications may be retried.

## Provisioning Resources

Once a subscription has been **registered** to use a Resource Provider, any user with Service Admin or Co-Administrator for that subscription may create and manage Cloud Services containing resources implemented by the RP.

The provider will receive a subscription registration event for each subscription before the first resource is provisioned.

A Cloud Service may contain multiple resources under the same RP. Each such resource is provisioned with a separate call to the RP.

For example, in the topology below, when communicating with RP, Azure would do two PUTs, one each on Resource2 and Resource3.

Cloud Service

Resource1

Resource3

Resource2

**API**

|  |  |
| --- | --- |
| Method | Request URI |
| PUT | https:// <registered-resource-provider-endpoint>/ subscriptions/{subscriptionId}/cloudservices/{cloud-service-name}/resources/{resource-type}/{resource-name} |

**Arguments**

|  |  |
| --- | --- |
| Argument | Description |
| subscriptionId | The subscriptionId for the Azure User.  This is the same ID sent with the register notification. |
| cloud-service-name | The cloud-service-name uniquely identifies the Cloud Service within the user’s subscriptionId. In the first release the cloud service name will be defined by Azure and is not available to the user  NOTE: Cloud-service-name is **case insensitive**. |
| resource-type | The type of the resource – the resource providers declare the resource types they support at the time of registering with Azure. |
| resource-name | Required [Case insensitive], String  The name of the resource, chosen by the service author. Resource names must be **unique** within the resources belonging to the same resource provider for a given CloudService. |

**Request Header**

|  |  |
| --- | --- |
| Header | Description |
| Content-Type | Set to application/xml. |
| x-ms-version | Specifies the version of the Azure operation to use for this request. This is a version of the Azure API (Azure Communication Protocol) that the resource provider has indicated it supports and the resource provider can adjust that when it on-boards and updates its manifest. |
| x-ms-request-id | When Azure receives the request from the client it generates a unique request ID. This is passed to the Resource Provider to aid in trouble shooting and should be logged.  This request id should be treated as an opaque identifier |
| x-ms-client-request-id | Optional.  When a client calls into Azure it can fill in an x-ms-client-request-id to identify the request. Azure passes that along to aid in troubleshooting client issues. As with the previous field this logged to aid in troubleshooting and should not be used for any other purpose. |

**Request Body**

<Resource xmlns="http://schemas.microsoft.com/windowsazure" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">

<CloudServiceSettings>

<GeoRegion>West US</GeoRegion>

</CloudServiceSettings>

<ETag>decac2dc-879a-455a-9f00-30559ab06d3c</ETag>

</IntrinsicSettings>

<Plan>free\_gamify</Plan>

<PromotionCode i:nil="true"/>

<SchemaVersion>1.0</SchemaVersion>

<Type>gamify</Type>

</Resource>

|  |  |
| --- | --- |
| Element name | Description |
| ETag | Required, guid.  A unique number identifying this change to the Resource.  Azure might send the same request more than once in some conditions. Resource providers can use this ID to ignore the repeated requests. Even in such cases, the response returned should follow the standard rules. For example if you have output items being returned, you should return them in case you are called with the same ETag. |
| CloudServiceSettings/Geo Region | Required, String.  The geo region of the Cloud Service.  This would be one amongst the supported Azure Geo Regions  West US| East US| North Central US| South Central US| West Europe| North Europe| East Asia| Southeast Asia  An RP should use this to create the resource in the appropriate geo-affinity region. In case you provided a region list when registering on the Publisher Portal, Azure will only allow resources to be created within that subset of regions.  If you have not provided any supported regions at the time of registration, send back the same value that you receive.  For resources that are not located in Azure, the Geo Region can be ignored. |
| Schemaversion | Optional, String.  The version of the Intrinsic Properties for your Resource. If not specified, you must conform to the contract resource version you registered as default. If you have not registered any Schemaversion then you should not have this tag in your response.  Azure uses this version to validate that the response contains all the required OutputKeys. Refer to the [section below](file:///C:\Users\Administrator\Downloads\Azure%20Store%20Resource%20Provider%20API%20Contract.docx#_Response_Body_Not) on implications on failure to do so. |
| Plan | Required, String.  The plan of the resource, which you entered in the Publisher Portal.  The plan specifies the terms under which the resource type is being provisioned and will vary by resource type. Examples might include “Gold”, “Silver” or “Platinum”. |
| PromotionCode | Optional, String.  The promotion code for buying the resource. The Publisher Portal enables partners to create promotion codes that can be distributed to users who can redeem them in the Azure Store. This is provided to partners for analytics purposes. |
| Intrinsicsettings | Optional, String which is actually an XmlNode[]  Settings used to provision or configure the resource. The structure of this section is defined by the RP during registration in the Publisher Portal.  The order of parameters in the request is unspecified. RPs should not rely on any particular ordering. |

**Type of Intrinsic Settings**

Intrinsic Settings are represented as [Strings](http://msdn.microsoft.com/en-us/library/362314fe(v=vs.71).aspx) over the wire. It is an [Array](http://msdn.microsoft.com/en-us/library/system.array.aspx) of objects of type [XmlNode](http://msdn.microsoft.com/en-us/library/system.xml.xmlnode.aspx) which is wrapped in an element called ArrayOfXmlNode in the standard data contract namespace for the type.

If "x" is an array that contains attribute node "N" in namespace "ns" that contains "value" and an empty element node "M", the representation is as follows.

{"x":"<ArrayOfXmlNode xmlns=\"http://schemas.datacontract.org/2004/07/System.Xml\" a:N=\"value\" xmlns:a=\"ns\"><M/></ArrayOfXmlNode>"}

Attributes in the empty namespace at the beginning of XmlNode arrays (before other elements) are unsupported.

**Response**

The response includes an HTTP status code, a set of response headers and a response body.

Azure allows a Resource Provider **20 seconds** to perform the operation after which it terminates the connection. Azure will then resend request by issuing a PUT with the same ETag – In such cases the resource provider should respect the ETag and provide correct Idempotent behavior by providing a response similar to that of the initial PUT request, including the output items. In case you have one time output items, you will have to generate a new one to send back.

**Status Code**

A successful operation returns:

* 200 (OK) or 201 (Created): These are returned when the operation completes synchronously. RDFE does not differentiate between the two values.

If the **status code is in the 5xx range or a timeout occurs**, Azure will retry the operation by issuing a PUT with the same ETag.

Azure will treat 409 conflicts as errors and are not expected.

**If the status code has any other value [4xx], or if the retries also fail**, the resources will be treated as being in a failed state. In such cases the resource provider must use the OperationStatus/Error field in the response body to indicate the message that would be passed to the user, the timeouts will manifest as 500-Internal Server Error to the caller**.**

**Response Body – Please note that the xml tags have to be in alphabetical order.**

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

<Resource xmlns="http://schemas.microsoft.com/windowsazure">

<CloudServiceSettings/>

<ETag>1b3a65f1-bab5-414a-b5f4-41f4bd6bda3e</ETag>

<IntrinsicSettings>

<Key>key</Key>

<Value>Value</Value>

</IntrinsicSettings>

<Name>helloworld</Name>

<OperationStatus>

<Result>Succeeded</Result>

</OperationStatus>

<OutputItems>

<OutputItem>

<Key>connection\_url</Key>

<Value>http://testrails/randomizer/helloworld</Value>

</OutputItem>

<OutputItem>

<Key>key</Key>

<Value>123456</Value>

</OutputItem>

</OutputItems>

<Plan>free\_gamify</Plan>

<State>Started</State>

<SubState>Waiting for your order</SubState>

</Resource>

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
| Element name | Description |
| CloudServiceSettings/GeoRegion | Required, String.  The geo region of the Cloud Service within which this resource resides.  This would be one amongst the supported Azure GeoRegions  West US| East US| North Central US| South Central US| West Europe| North Europe| East Asia| Southeast Asia |
| ETag | This should be the same guid that Windows Azure passes to the RP. |
| IntrinsicSettings | Required, String which is actually an XmlNodes[]  Current Settings for the resource. The structure of this section is defined by the resource provider during registration. If the resource does not have any intrinsic properties, then this should be returned as an empty string. |
| State | Required, string.  The current resource status – must be one amongst Started| Stopped|Paused |
| SubState | Optional, string.  Resource Provider can use this to report more detailed State Attributes. |
| OutputItems/  OutputItem | Optional. Key:String, Value:String.  List of key-value pairs with the output from the resource, such as a resource name or connection string.  These keys must be registered with Azure during the onboarding process**.**  **During the first create call, the output items are mandatory if you have registered any. However if the RP gets another PUT call on the resource to edit anything, for ex the plan, the RP need not return any output items.** |
| OperationStatus/  Result | Required, string.  One of following values.   * Succeeded: The operation succeeded. * Failed: The operation failed. |
| UsageMeters/  UsageMeter/  Name  Used  Included  Unit | Optional.  This is used for reporting the current usage for the Resource. You can report on more than one meters.  Name: String, Name of the Meter  Used: String, The amount of Units used.  Included: String, Total units in the Plan.  Unit: String. Must be bytes|hours|generic |
| OperationStatus/  Error  /HttpCode  /Message | Optional. HttpCode:String, Message: String  Provides detailed information about failures in a structured way. Note that Azure will pass these to the caller.  HttpCode and Message are explained in detail in the section below. |

Strings which are returned from an RP will NOT be localized.

The maximum size of a response that Azure will accept from the Resource Providers is 1 MB. Any response greater than 1 MB in size will be dropped by Azure and will return **500 Internal Server Error** to the client. In general, APIs exposed by the Resource Provider should be designed to transmit relatively little data.

**Resource State**

Azure will define following basic resource States for a Resource. In addition it also provides a pass- through channel for the RPs to flow Resource specific granular state information back to the user.

|  |  |  |
| --- | --- | --- |
| State | Description | Example |
| Started | Resource is available without problems. | ContosoDB is functional correctly |
| Stopped | Resource was stopped due a user action | The user manually stops their ContosoDB e.g. through the partner’s management UI |
| Paused | The resource was temporarily put on hold | User’s bill has not been paid. Contoso halts access to ContosoDB in response. |

Azure will drop status fields outside of the above set and in such cases report the Resource State as “Unknown” to the end user.

**Dealing with Operation Errors**

Azure provides an Operation Status section where in case of errors the result must be set to “Failed”. This prompts Azure to look in the error block. The error block should be used to further classify the cause of the failure using the Http code in the 4xx range and supplementing that with a useful message about the error.

<OperationStatus>

<Result>Failed</Result>

<Error>

<HttpCode>http-code-4xx-range</HttpCode>

<Message>detailed-error-message</Message>

</Error>

</OperationStatus>

Azure expects resource providers to fill in the HTTP error code in the 4xx range for such issues and supplement that with an error message.

Responses received from the RP are passed through to the client and it is the responsibility of the Resource Provider to ensure that information in the body is **appropriate for public consumption** (e.g., no stack traces, no non-public terminology, etc.).

Errors in the 5xx range effect the entire operation hence should be reported at the HTTP message response level.

### Response Body Not Conforming to Contract

In cases when the response doesn’t conform to the above contract, Azure will deem the operation as unsuccessful and will **NOT** generate any corresponding billing events. The resource will be marked as “provisioning failed” and only the “Delete” action that will be enabled.

## Gets on Existing Resources

Azure based on user activity will periodically request the current State of the resource using a GET on the following URIs.

**API**

|  |  |  |
| --- | --- | --- |
| Method | Request URI | Description |
| GET | https:// <registered-resource-provider-endpoint>/ subscriptions/{subscriptionId}/cloudservices/{cloud-service-name}/resources/{resource-type}/{resource-name} | Returns the Resource [{resource-type}/{resource-name}] within the CloudService. |
| GET | https:// <registered-resource-provider-endpoint>/ subscriptions/{subscriptionId}/cloudservices/{cloud-service-name}/ | Returns a CloudService [{cloud-service-name}] as a Collection of Resources |

**Arguments**

|  |  |
| --- | --- |
| Argument | Description |
| subscriptionId | The subscriptionID for the Azure User.  This is the same ID sent with the register notification. |
| cloud-service-name | The cloud-service-name uniquely identifies the Cloud Service within the user subscriptionID.  NOTE: Cloud-service-name is **case insensitive**. |
| resource-type | The type of the resource – the resource providers declare the resource types they support at the time of registering with Azure. |
| resource-name | Required [Case insensitive], String  The name of the resource, chosen by the service author. Resource names must be **unique** among the resources belonging to a CloudService and supported by a single resource provider |
|  |  |

**Request Body**

Empty

**Response**

The response includes an HTTP status code, a set of response headers and a response body.

**Status Code**

A successful operation returns status code 200 (OK). If the service doesn’t exist, the return code must be 404 (Not Found).

**Response Headers**

Only standard HTTP headers are required in the response.

**Response Body**

The response body should be the same as that returned in the response for [Provisioning](file:///C:\Users\Administrator\Downloads\Azure%20Store%20Resource%20Provider%20API%20Contract.docx#_Step_2_–) or Update flow with the following changes:

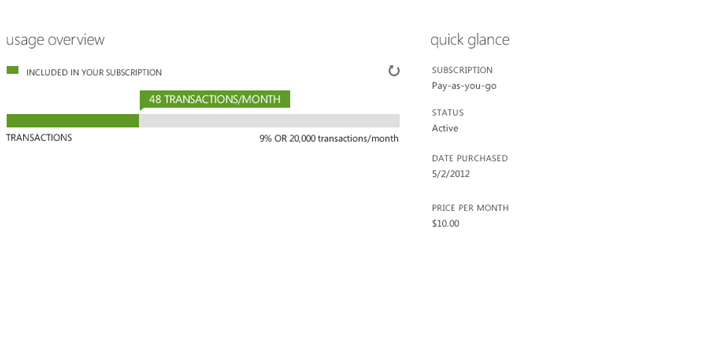
For response to https:// <registered-resource-provider-endpoint>/ subscriptions/{subscriptionId}/cloudservices/{cloud-service-name}/

The response body should include Name, Plan and Type values. Please see the example for details

The GET should not have the output items.

The fields for operation result and error information are required as well. These are also used to report the Status of the resources so you should strive to provide the latest.

The get call should also return the usage information that will be displayed in the portal. The portal will display this data as per the screen shot below:



To display this bar the following information is needed

1. Name of the meter – In this case it is transactions
2. Used – This is the amount that has been used – in this case it is 48 transactions
3. Included- this is the amount that the user is entitled to – In this case it is 20,000 transactions.

GET https://<registered-resource-provider-endpoint>/ subscriptions/{subscriptionId}/cloudservices/{cloud-service-name}/Resources/{resource-type}/{resource-name}

**Response body: Please note that the tags have to be in alphabetical order**

<Resource xmlns="http://schemas.microsoft.com/windowsazure">

<CloudServiceSettings>

<GeoRegion>usnorth</GeoRegion>

</CloudServiceSettings>

<ETag>100-100-10203-302012</ETag>

<Name>testgamify</Name>

<OperationStatus>

<Result>Succeeded</Result>

</OperationStatus>

<Plan>free\_gamify</Plan>

<State>Started</State>

<SubState>Waiting for your order</SubState>

<Type>gamify</Type>

</Resource>

GET https:// <registered-resource-provider-endpoint>/ subscriptions/{subscriptionId}/cloudservices/{cloud-service-name}/

**Response body: Please note that the tags have to be in alphabetical order**

<CloudService xmlns="http://schemas.microsoft.com/windowsazure">

<GeoRegion>usnorth</GeoRegion>

<Resources>

<Resource>

<ETag>100-100-10203-302012</ETag>

<Name>testgamify</Name>

<OperationStatus>

<Result>Succeeded</Result>

</OperationStatus>

<Plan>free\_gamify</Plan>

<State>Started</State>

<SubState>Waiting for your order</SubState>

<Type>gamify</Type>

</Resource>

<Resource>

<ETag>100-100-10203-302012</ETag>

<Name>newgamify</Name>

<OperationStatus>

<Result>Succeeded</Result>

</OperationStatus>

<Plan>free\_gamify</Plan>

<State>Started</State>

<SubState>Waiting for your order</SubState>

<Type>gamify</Type>

</Resource>

<Resource>

<ETag>100-100-10203-302012</ETag>

<Name>testetagamify</Name>

<OperationStatus>

<Result>Succeeded</Result>

</OperationStatus>

<Plan>free\_gamify</Plan>

<State>Started</State>

<SubState>Waiting for your order</SubState>

<Type>gamify</Type>

</Resource>

</Resources>

</CloudService>

Resource providers should respond to this request in less than 2 seconds. The maximum response size is 1 MB. Any response greater than 1 MB in size will be dropped by Azure and will return **500 Internal Server Error** to the client. Similarly responses not received within 2 seconds will result in the resource State to be reported as “Unknown” with empty intrinsic settings.

## Deletes on Existing Resources

**API**

|  |  |
| --- | --- |
| Method | Request URI |
| DELETE | https:// <registered-resource-provider-endpoint>/ subscriptions/{subscriptionId}/cloudservices/{cloud-service-name}/ |
| DELETE | https://<registered-resource-provider-endpoint>/ subscriptions/{subscriptionId}/cloudservices/{cloud-service-name}/Resources/{resource-type}/{resource-name} |

**Arguments**

|  |  |
| --- | --- |
| Argument | Description |
| subscriptionId | The subscriptionID for the Azure User.  This is subscription ID is passed in the Registered Event |
| cloud-service-name | The cloud-service-name uniquely identifies the Cloud Service within the user subscriptionID.  NOTE: Cloud-service-name is **case insensitive**. |
| resource-type | The type of the resource – the resource providers declare the resource types they support at the time of registering with Azure. |
| resource-name | Required [Case insensitive], String  The name of the resource, chosen by the service author. Resource names must be **unique** among the resources belonging to a CloudService and supported by a single resource provider |

**Request Header**

**Same as before.**

**Request Body**

Empty

**Response**

The response includes an HTTP status code, a set of response headers and a response body.

**Status Code**

A successful operation returns status code 200 (OK). If the service doesn’t exist, the return code must be 404 (Not Found).

If the status code is 404 (NotFound), Azure assumes that the resources have already been deleted. If the status code is in the 500-599 range or a timeout occurs, Azure will retry the operation. If the status code has any other value or the retries also fail, the resources will be treated as being in a failed state.

**Response Headers**

Only standard HTTP headers are required in the response.

### Response Body & Dealing with Failures

The response body is optional.

Usually a resource provider would choose to have a response when a resource deletion fails, so that error information can be returned to the caller. Note that even in this case the status code should still indicate success, as a failure to delete one resource does not imply failure of the entire operation**.** If a response is specified, it should contain the same XML returned by the Provisioning and Update APIs in the [previous section](file:///C:\Users\Administrator\Downloads\Azure%20Store%20Resource%20Provider%20API%20Contract.docx#_Step_2_–).

# Resource Quotas

Azure will enforce following Quotas on the Cloud Service,

|  |  |
| --- | --- |
| Entity | Quota |
| ResourceProviderType/ResourceType | 5 – You can create only 5 resources of a Particular type within the CloudService. |

Provisioning APIs will fail with **412 Precondition Failed** when the Quota limits for any of the resources within the request are reached.

# Billing Events

Azure takes care of generating a bill for the user, and paying out partners. Be mindful that the following events can have a monetary COGS impact on your service:

|  |  |
| --- | --- |
| Event | Description |
| Provision | The user is immediately charged for the new plan (unless the plan is free). |
| Updates to Plan  Upgrades  Downgrades – Are NOT Allowed | Changes to the Plan which expands the service offering hence incurring a higher monetary charge.  The user is immediately charged for the new plan. Previously charged amounts, if any, are not refunded or prorated to the user. |
| Delete (de-provision) | This immediately removes the monetary relationship with the user. Previously charged amounts, if any, are not refunded or prorated to the user. |

# Azure to Resource Provider Communication

Azure uses certificate based authentication when calling into the resource provider. Resource providers **must support** multiple client certificates for the purpose of certificate rollovers. The thumbprints for the various environments are

Test: a9 be 06 e8 88 13 39 d7 b6 9f 84 21 f5 13 04 43 1b ed 0e e7

Production: will be provided in a subsequent update

# Single Sign-on (SSO)

In the first release of the Azure Store, Microsoft will provide a simplified management experience for all resources purchased through the Azure Store. This will be limited to obtaining connection information, upgrading the plan, deleting the resource and viewing useful links to tutorials and sample code.

Richer management will only be available through the partner’s own management portal. The user will be able to see the resource in the Azure Developer Portal, and click a button which will open a new browser window where the user will be silently signed-on without having to provide a username or password.

This section details on how SSO is enabled.

1. The user selects a purchased item in the Azure Developer Portal and clicks the *Manage* button.
2. Windows Azure looks up the RP’s SSO URL, which was provided by the partner on the Publisher Portal.

The SSO URL will look like this:

https:// <registered-resource-provider-endpoint>/ subscriptions/{subscriptionId}/cloudservices/{cloud-service-name}/resources/{resourcetype}/{resourcename}/**SsoToken**

1. Windows Azure invokes the URL, as part of the call the following information is provided to the RP
   1. Subscription ID
   2. Datetime stamp
   3. Cloud Service name
   4. Resource name
2. The RP generates a secret token based on the above parameters as well as a secret key that it keeps. This token is then passed back to Azure Developer Portal.

<SsoToken xmlns="<http://schemas.microsoft.com/windowsazure>">

  <TimeStamp>2012-10-05T05:09:03+00:00</TimeStamp>

  <Token>63e9a232b0bc8e5083571d1e72f58e5d670f6482</Token>

</SsoToken>

1. The Azure Developer Portal in the browser redirects to this new URL with the secret token in a new browser window. The URL also passes the same information as Step 3. Note that the timestamp passed is the same that was passed by the RP when the SsoToken API was called.

The URL will be based on the following pattern:

https://<resource provider sso url>?token=<token> &subid=<subid>&cloudservicename=<csname>&resourcetype=<resourcetype>&resourcename=<resname>&timestamp=2012-10-05T05:09:03+00:00

1. The RP receives the token as part of the request and validates it by regenerating it and comparing to the original one.  If it matches, the following happens
   1. RP generates a normal session by setting the session cookie with a timeout interval.
   2. RP returns the HTML payload that is displayed in the user’s browser.

# Change Log

|  |  |
| --- | --- |
| Date | Change |
| 8/27/12 | Only deletes of resource types are supported. Bulk deletion of all the resource types for a cloud service are no longer supported. |
| 8/28/12 | Environments and the process to onboard configurations |
| 8/29/12 | Update to the structure of subscription notifications. The URI includes subscription id which is also passed in the body of the notification. |
| 8/29/12 | Update to the requirements around GeoRegions. For RPs that are not located in Azure the geo-region can be ignored |
| 9/8/12 | Updated the onboarding XML |
| 9/8/12 | Update to the authentication mechanism to account for the use of a client certificate instead of PKI |
| 9/9/12 | Clarification to indicate that cloud service names cannot be selected by the user |
| 9/10/12 | Update to indicate that ETag is returned on GETs for resources as well as update to discussion on use of ETAGs and Incarnation IDs |
| 9/20/12 | Dropped the discussion around environments and onboarding XML  Update to indicate that the subscription notifications will arrive on /subscriptions/{subid}/events |
| 9/21/12 | Added the usage xml section to the Provisioning and GET resource call. This can be used by RPs to send usage to the portal |
| 10/12/12 | Added details around the register event call and fixed numerous issues. |