IRAS API SERVICES INTERFACE SPECIFICATIONS

Property Tax Calculator

Last updated on: 31 Jan 2019

Version No.: 1.0.4

Table of Contents

1. Introduction

The Inland Revenue Authority of Singapore (IRAS) provides application programming interface (API) services to allow application developers to submit and retrieve tax related matters using HTTP requests. Most of the APIs will be in the form of a JSON web service which reduces client/server coupling and thus enabling easier integration between IRAS' service with external developers.

There will be a variety of services available in due time. While some services require a simple GET, others may be secured and require credentials that can be passed via HTTP header parameters which are as follows :

X-IBM-Client-Id	String containing the client ID of the application invoking IRAS API. This value will be provided to the application vendor by IRAS. E.g. a1234b5c-1234-abcd-efgh-a1234b5cdef
X-IBM-Client-Secret	String containing the client secret of the application invoking IRAS API. This value will be provided to the application vendor by IRAS. E.g. a12345bC67e8fG9a12345bC67e8fG9a12345bC67e8fG9

This document serves to help developers consume the API services provided by IRAS.

2. Registration at API Portal

Application developers are required to register for an account at https://apisandbox.iras.gov.sg/ to subscribe to IRAS API services for Sandbox Testing and at https://apiservices.iras.gov.sg/ for Production.

A computer-generated email will be sent to the subscriber's email account for account activation of the API Portal.

3. API Services

IRAS will provide various API services for the calculation of Property Tax. The following sections describe the request and response for each of the services.

The table below shows the list of Property Tax Calculation API services currently available in IRAS.

S/No	Name of API Services	Description	URL
1	Property Tax Calculation using Property Details	This API can be used for calculating the property tax payable by entering the building address, postal code, land description or property tax reference number, together with the property type, occupancy status (if necessary) and tax period.	For Sandbox Testing: https://apisandbox.iras.gov.sg/iras/sb/PTTaxCalc/CalPptyTaxByPptyDet For Production Usage: https://apiservices.iras.gov.sg/iras/prod/PTTaxCalc/CalPptyTaxByPptyDet
2	Property Tax Calculation using Annual Value	This API can be used for calculating the property tax payable by entering the annual value, together with the property type, occupancy status (if necessary) and tax period.	For Sandbox Testing: https://apisandbox.iras.gov.sg/iras/sb/PTTaxCalc/CalPptyTaxByAV For Production Usage: https://apiservices.iras.gov.sg/iras/prod/PTTaxCalc/CalPptyTaxByAV
3	Property Tax Apportionment Calculation using Property Details	This API can be used for calculating the property tax payable of the Seller and Purchaser by entering the building address, postal code, land description or property tax reference number, together with the purchaser's tax liability date.	For Sandbox Testing: https://apisandbox.iras.gov.sg/iras/sb/PTTaxCalc/CalPptyTaxApportionmentByPptyDet For Production Usage: https://apiservices.iras.gov.sg/iras/prod/PTTaxCalc/CalPptyTaxApportionmentByPptyDet

3.1 General Information

3.1.1 Production Usage

Approval is required to use this service.

The following parameters must be populated in the HTTP header:

X-IBM-Client-Id	String containing the client ID of the application invoking IRAS API. This value will be provided to the application vendor by IRAS.
X-IBM-Client-Secret	String containing the client secret of the application invoking IRAS API. This value will be provided to the application vendor by IRAS.

3.1.2 Common Interface Information

- JSON is case sensitive by specifications.
- All date strings are to be represented in compliance to the ISO-8601 standard.
- All string fields are subject to validation of the following acceptable characters that is allowed (in red)
    - [a-zA-Z0-9‘@#()\-,./&+_] (**Note**: whitespace is included)
- All properties follow the camel-case convention.
- Unless stated as optional, all JSON object properties must be specified.
- Unless otherwise specified, all JSON services are invoked using HTTP verb POST.
- All input data format are as specified like the following:

3.1.3 Common Response Payload

All response payloads share the following common fields:

data	Object	The data property will be populated differently based on the API that is being invoked.
returnCode	Integer	**10** : Success - The request was successfully processed. **20** : Warning - The request was successfully processed. However, there are non-fatal issues. Please refer to the “info” object for diagnostic information. **30** : Failure – The request was not processed. Refer to “info” object for error information.
info	Object	This complex object holds any diagnostic information that will allow developers to debug their failed requests.
info.message	String	Diagnostic message in the event of warning or error.
Info.messageCode	Integer	Integer code signifying the type of error or warning. **850300 :** Request object is null – The incoming JSON request is null. **850301** : Arguments error – There is an error with one of the arguments provided. **850302** : Generic error – There is an exception within the

		service. **850303** : Service is inactive. **850304** : Service is not authorized for usage based on the provided credentials. **850305** : Invalid test user – The input fields provided are not valid for sandbox testing. **400348:** Property record not found. **10454 :** Property is not in Valuation List.
info.fieldInfoList	Array	An array for FieldInfo objects.
info.fieldInfoList.field	String	Name of the field that resulted in a warning / error.
Info.fieldInfoList.message	String	Diagnostic message provided to aid consumer's developers.

3.2 Property Tax Calculator with Property Details

3.2.1 Request Payload

criteria	String	Search Criteria: 1 = **Search by building address** (blkHouseNo, streetName are required fields. storeyNo, unitNo where applicable) 2 = **Search by land description** (surveyDistrict, mkTsNo, lotNo are required fields) 3 = **Search by property tax reference number** (pptyTaxRefNo is required field) 4 = **Search by postal code** (postalCode is required field. storeyNo, unitNo where applicable)
postalCode	String	Postal Code
storeyNo	String	Storey Number
unitNo	String	Unit Number
blkHouseNo	String	Blk / House Number
streetName	String	Street Name
surveyDistrict	String	Survey District 1 : MK 2 : TS
mkTsNo	String	MK or TS number according to survey district
lotNo	String	Lot Number
plotPartParcel	String	Plot / Part / Parcel
pptyTaxRefNo	String	Property Tax Reference Number
propertyClass	String	Property Classification 1 : Residential (occupancyStatus is required field for residential) 2 : Non-residential
occpyStatus	String	Occupancy Status 1 : Owner-occupied 2 : Let-out 3 : Vacant
periodFrm	String	Period from date of current year to take into consideration for tax computation, specified in ISO-8601 format
periodTo	String	Period to date of current year to take into consideration for tax computation, specified in ISO-8601 format

Sample JSON request payload


```
{
  "criteria": "1",
  "blkHouseNo": "999",
  "streetName": "PRINCE AVE 4",
  "storeyNo": "08",
  "unitNo": "111",
  "postalCode": ""
  "surveyDistrict": "",
  "mkTsNo": "",
  "lotNo": "",
  "plotPartParcel": "",
  "pptyTaxRefNo": "",
  "propertyClass": "1",
  "occpyStatus": "1",
  "periodFrm": "2018-01-01",
  "periodTo": "2018-12-25"
}
```

3.2.2 Response Payload

data	Object	The object payload containing the reference code of successful submission.
data.	String	Annual value of property.
data.netTaxPayable	String	Net tax payable.
data.periodFrm	String	Period from date taken into consideration for tax computation, specified in ISO-8601 format
data.periodTo	String	Period to date taken into consideration for tax computation, specified in ISO-8601 format
returnCode	As per Section 3.1.3	
info		
info.message		
info.messageCode		
info.fieldInfoList		
info.fieldInfoList.field		
info.fieldInfoList.message		
info.fieldInfoList.recordID		

Sample success JSON response payload

```
{
   "returnCode": 10,
   "info":  {
      "fieldInfoList": []
   },
   "data": {
     "netTaxPayablec: "280.00",
     "periodFrm": "2018-01-01",
     "periodTo": "2018-12-25"
   }
}
```

Sample error JSON response payload

```
{
   "info":{
      "fieldInfoList":{
         "fieldInfo":[]
       },
      "message":"Property is not in Valuation List",
      "messageCode":"10454"
   },
   "returnCode":"20"
}
```

3.3 Property Tax Calculator with Annual Value

3.3.1 Request Payload

annualValue	String	Annual Value of property
occpyStatus	String	Occupancy Status 1 : Owner-occupied 2 : Let-out 3 : Vacant
periodFrm	String	Period from date of current year to take into consideration for tax computation, specified in ISO-8601 format
periodTo	String	Period to date of current year to take into consideration for tax computation, specified in ISO-8601 format
propertyClass	String	Property Classification 1 : Residential (occupancyStatus is required field for residential) 2 : Non-residential

Sample JSON request payload

```
{
  "annualValue": "20000.00",
  "propertyClass": "1",
  "occpyStatus": "1",
  "periodFrm": "2018-01-01",
  "periodTo": "2018-12-25"
}
```

3.2.2 Response Payload

data	Object	The object payload containing the reference code of successful submission.
data.	String	Annual value of property.
data.netTaxPayable	String	Net tax payable.
data.periodFrm	String	Period from date taken into consideration for tax computation, specified in ISO-8601 format
data.periodTo	String	Period to date taken into consideration for tax computation, specified in ISO-8601 format
returnCode	As per Section 3.1.4	
info		
info.message		
info.messageCode		
info.fieldInfoList		
info.fieldInfoList.field		
info.fieldInfoList.message		
info.fieldInfoList.recordID		

Sample success JSON response payload

```
{
  "returnCode":"10",
  "info":{
    "fieldInfoList":{
       "fieldInfo":[]
    }
  },
  "data":{
     "periodFrm":"2018-01-01T00:00:00",
     "periodTo":"2018-05-31T00:00:00",
     "netTaxPayable":"2083.33"
  }
}
```

Sample error JSON response payload

```
{
   "returnCode":"30",
   "info":{
     "messageCode":"850301",
     "message":"Arguments Error",
     "fieldInfoList":{
     "fieldInfo": [
       {
          "message":"Period start date later than end date",
          "field":"periodFrm / periodTo"
       }
     ]}
   }
}
```

3.4 Property Tax Apportionment Calculator with Property Details

3.4.1 Request Payload

criteria	String	Search Criteria: 1 = **Search by building address** (blkHouseNo, streetName are required fields. storeyNo, unitNo where applicable) 2 = **Search by land description** (surveyDistrict, mkTsNo, lotNo are required fields) 3 = **Search by property tax reference number** (pptyTaxRefNo is required field) 4 = **Search by postal code** (postalCode is required field. storeyNo, unitNo where applicable)
postalCode	String	Postal Code
storeyNo	String	Storey Number
unitNo	String	Unit Number
blkHouseNo	String	Blk / House Number
streetName	String	Street Name
surveyDistrict	String	Survey District 1 : MK 2 : TS
mkTsNo	String	MK or TS number according to survey district
lotNo	String	Lot Number
plotPartParcel	String	Plot / Part / Parcel
pptyTaxRefNo	String	Property Tax Reference Number
dateBuyerLiability	String	The date of the buyer's liability for tax computation, specified in ISO-8601 format

Sample JSON request payload

```
{
  "criteria": "1",
  "blkHouseNo": "999",
  "streetName": "PRINCE AVE 4",
  "storeyNo": "08",
  "unitNo": "111",
  "postalCode": ""
  "surveyDistrict": "",
  "mkTsNo": "",
  "lotNo": "",
  "plotPartParcel": "",
  "pptyTaxRefNo": "",
  "dateBuyerLiability": "2018-06-15"
}
```

3.2.2 Response Payload

data	Object	The object payload containing the reference code of successful submission.
data.sellerNetTaxPayable	String	Net tax payable by the Seller.
data.buyerNetTaxPayable	String	Net tax payable by the Buyer.
data.annualValue	String	Annual Value of the Property
data.taxRate	String	Owner-occupier or non-residential tax rate
returnCode	As per Section 3.1.4	
info		
info.message		
info.messageCode		
info.fieldInfoList		
info.fieldInfoList.field		
info.fieldInfoList.message		
info.fieldInfoList.recordID		

Sample success JSON response payload

```
{
   "returnCode": 10,
   "info":  {
      "fieldInfoList": []
   },
   "data": {
     "sellerNetTaxPayable”: "400.00",
     "buyerNetTaxPayable”: "800.00”,
     "annualValue”: "20400.00”,
     "taxRate”: "Owner-Occupier Tax Rate”
   }
}
```

Sample error JSON response payload

```
{
   "returnCode":"30",
   "info":{
     "messageCode":"850301",
     "message":"Arguments Error",
     "fieldInfoList":{
     "fieldInfo": [
       {
          "message":"Invalid Buyer’s Liability Date",
          "field":"dateBuyerLiability"
       }
     ]}
   }
}
```

3.5 Sandbox Testing

As explained in section 2, developers can first create an account in the Sandbox environment to make API calls to our Sandbox URL. This allows the developers to mimic the characteristics of our production environment and create a simulated response from our API.

3.5.1 Property Tax Calculator by Property Details Service

Input	Expected Output

<pre>{
"criteria": "1",
"blkHouseNo": "151B",
"streetName": "KINGS RD",
"storeyNo": "09",
"unitNo": "05",
"propertyClass": "1",
"occpyStatus": "2",
"periodFrm": "2018-02-01",
"periodTo": "2018-07-01"
}</pre>
	<pre>{
	"info":{
	"fieldInfoList":{
	"fieldInfo":[]
	}
	},
	"returnCode":"10",
	"data":{
	"netTaxPayable":"3618.19",
	"periodTo":"2018-07-01T00:00:00",
	"periodFrm":"2018-02-01T00:00:00"
	}
	}</pre>

<pre>{
"criteria": "4",
"storeyNo": "09",
"unitNo": "05",
"postalCode": "268159",
"propertyClass": "1",
"occpyStatus": "1",
"periodFrm": "2018-01-01",
"periodTo": "2018-12-31"
}</pre>
	<pre>{
	"info":{
	"fieldInfoList":{
	"fieldInfo":[]
	}
	},
	"returnCode":"10",
	"data":{
	"periodFrm":"2018-01-01T00:00:00",
	"periodTo":"2018-12-31T00:00:00",
	"netTaxPayable":"2844"
	}
	}</pre>

<pre>{
"criteria": "3",
"pptyTaxRefNo": "0200320A",
"propertyClass": "1",
"occpyStatus": "1",
"periodFrm": "2018-01-01",
"periodTo": "2018-09-30"
}</pre>
	<pre>{
	"info":{
	"fieldInfoList":{
	"fieldInfo":[]
	}
	},
	"returnCode":"10",
	"data":{
	"periodFrm":"2018-01-01T00:00:00",
	"periodTo":"2018-09-30T00:00:00",
	"netTaxPayable":"2133"
	}
	}</pre>

<table>
<tr>
<td>

```
{
  "criteria": "3",
  "pptyTaxRefNo": "0200320B",
  "propertyClass": "2",
  "periodFrm": "2018-06-01",
  "periodTo": "2018-12-31"
}
```

</td>
<td>

```
{
   "info":{
      "fieldInfoList":{
         "fieldInfo":[{
            "field":"pptyTaxRefNo",
            "message":"Value is not valid"
         }]
      },
      "message":"Arguments Error",
      "messageCode":"850301"
   },
   "returnCode":"30"
}
```

</td>
</tr>
<tr>
<td>

```
{
  "criteria":"3",
  "pptyTaxRefNo":"0200322G",
  "propertyClass":"2",
  "periodFrm": "2018-06-01",
  "periodTo": "2018-12-31"
}
```

</td>
<td>

```
{
   "info":{
      "fieldInfoList":{
         "fieldInfo":[]
       },
      "message":"Property is not in
Valuation List",
      "messageCode":"10454"
   },
   "returnCode":"20"
}
```

</td>
</tr>
<tr>
<td>

```
{
  "criteria":"2",
  "surveyDistrict":"1",
  "mkTsNo":"1",
  "lotNo":"102",
  "propertyClass":"2",
  "periodFrm":"2018-05-01",
  "periodTo": "2018-09-10"
}
```

</td>
<td>

```
{
   "info":{
      "fieldInfoList":{
         "fieldInfo":[]
      }
   },
   "returnCode":"10",
   "data":{
      "periodFrm":"2018-05-01T00:00:00",
      "periodTo":"2018-09-10T00:00:00",
      "netTaxPayable":"3105.56"
   }
}
```

</td>
</tr>
<tr>
<td>

```
{
  "criteria":"3",
  "pptyTaxRefNo":"0200321P",
  "propertyClass":"2",
  "periodFrm": "2018-06-01",
  "periodTo": "2018-12-31"
}
```

</td>
<td>

```
{
   "info":{
      "fieldInfoList":{
         "fieldInfo":[]
       },
      "message":"Property record not
found",
      "messageCode":"400348"
   },
   "returnCode":"20"
}
```

</td>
</tr>
</table>

3.5.2 Property Tax Calculator by Annual Value Service

Input	Expected Output
{ "annualValue": "50000", "propertyClass": "2", "periodFrm": "2018-01-01", "periodTo": "2018-05-31" }	{ "returnCode":"10", "info":{ "fieldInfoList":{ "fieldInfo":[] } }, "data":{ "periodFrm":"2018-01-01T00:00:00", "periodTo":"2018-05-31T00:00:00", "netTaxPayable":"2083.33" } }
{ "annualValue": "59000", "propertyClass": "1", "occpyStatus": "2", "periodFrm": "2018-02-01", "periodTo": "2018-07-01" }	{ "returnCode":"10", "info":{ "fieldInfoList":{ "fieldInfo":[] } }, "data":{ "periodFrm":"2018-02-01T00:00:00", "periodTo":"2018-07-01T00:00:00", "netTaxPayable":"2834.84" } }
{ "annualValue": "90000", "propertyClass": "1", "occpyStatus": "3", "periodFrm": "2018-01-01", "periodTo": "2018-12-31" }	{ "returnCode":"10", "info":{ "fieldInfoList":{ "fieldInfo":[] } }, "data":{ "periodFrm":"2018-01-01T00:00:00", "periodTo":"2018-12-31T00:00:00", "netTaxPayable":"12000" } }

{
"annualValue": "8000",
"propertyClass": "1",
"occpyStatus": "1",
"periodFrm": "2018-01-01",
"periodTo": "2018-05-31"
}
	{
	"returnCode":"10",
	"info":{
	"fieldInfoList":{
	"fieldInfo":[]
	}
	},
	"data":{
	"periodFrm":"2018-01-01T00:00:00",
	"periodTo":"2018-05-31T00:00:00",
	"netTaxPayable":"0"
	}
	}

{
"annualValue": "100000",
"propertyClass": "1",
"occpyStatus": "1",
"periodFrm": "2018-02-01",
"periodTo": "2018-12-31"
}
	{
	"returnCode":"10",
	"info":{
	"fieldInfoList":{
	"fieldInfo":[]
	}
	},
	"data":{
	"periodFrm":"2018-02-01T00:00:00",
	"periodTo":"2018-12-31T00:00:00",
	"netTaxPayable":"5023.33"
	}
	}

3.5.3 Property Tax Apportionment Calculator by Property Details Service

Input	Expected Output

{
"criteria":"1",
"dateBuyerLiability":"2019-01-31",
"blkHouseNo":"16",
"storeyNo":"01",
"streetName":"Tagore Dr",
"unitNo":"10"
}
	{
	"returnCode":"10",
	"info":{
	"fieldInfoList":{
	"fieldInfo":[]
	}
	},
	"data":{
	"annualValue":"20400.00",
	"buyerNetTaxPayable":"275.56",
	"sellerNetTaxPayable":"179.11",
	"taxRate":"Owner-Occupier Tax Rate"
	}
	}

<table>
<tr>
<td>

```
{
      "criteria":"2",
      "dateBuyerLiability":"2019-
      12-03",
      "lotNo":"12",
      "mkTsNo":"1",
      "plotPartParcel":"",
      "surveyDistrict":"1"
}
```

</td>
<td>

```
{
  "returnCode":"10",
  "info":{
    "fieldInfoList":{
      "fieldInfo":[]
        }
   },
  "data":{
      "annualValue":"180000.00",
      "buyerNetTaxPayable":"1403.23",
      "sellerNetTaxPayable":"28596.77",
      "taxRate":"Non-Residential Tax
      Rate"
  }
}
```

</td>
</tr>
<tr>
<td>

```
{
      "criteria":"3",
      "dateBuyerLiability":"2019-
      05-05",
      "pptyTaxRefNo":"1287341G"
}
```

</td>
<td>

```
{
  "returnCode":"10",
  "info":{
    "fieldInfoList":{
      "fieldInfo":[]
        }
   },
  "data":{
      "annualValue":"8100.00",
      "buyerNetTaxPayable":"3.33",
      "sellerNetTaxPayable":"179.11",
      "taxRate":"Owner-Occupier Tax
      Rate"
  }
}
```

</td>
</tr>
<tr>
<td>

```
{
      "criteria":"4",
      "dateBuyerLiability":"2019-
      11-10",
      "postalCode":"503957",
      "storeyNo":"29",
      "unitNo":"03"
}
```

</td>
<td>

```
{
  "returnCode":"10",
  "info":{
    "fieldInfoList":{
      "fieldInfo":[]
        }
   },
  "data":{
      "annualValue":"43800.00",
      "buyerNetTaxPayable":"620.50",
      "sellerNetTaxPayable":"4854.50",
      "taxRate":" Non-Residential Tax
      Rate "
  }
}
```

</td>
</tr>
</table>

4. Sample Code (C#)

```
using System;
using System.Net;
using System.IO;
using System.Text;

// jsonData – contains data from Section 3.1.1 of this document
public static void callWebAPI(string jsonData, string url)
{
        try
        {
            var httpWebRequest = (HttpWebRequest)WebRequest.Create(url);
            httpWebRequest.ContentType = "application/json;";
            httpWebRequest.Method = "POST";

            //Step 1: Enter the Client-Id given by IRAS
            httpWebRequest.Headers["Client-Id"] = "{YOUR_CLIENT_ID}";
            //Step 2: Enter the Client-Secret given by IRAS
            httpWebRequest.Headers["Client-Secret"] ="{YOUR_CLIENT_SECRET}";

            // Step 3: Call API using POST
            using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
            {
                streamWriter.Write(jsonData);
                streamWriter.Flush();
                streamWriter.Close();
            }

            // Step 3a: Output response
            var httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
            using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
            {
                var result = streamReader.ReadToEnd();
                //print the received reponse
                Console.WriteLine(result);
            }
        }
        catch (WebException e)
        {
            if (!string.IsNullOrEmpty(e.Message))
            {
                // Step 3b: Print general errors
                Console.WriteLine("Exception - ");
                Console.WriteLine(e.Message);
            }

            if (e.Response != null)
            {
                // Step 3c: Print Output response exception
                Stream receiveStream = e.Response.GetResponseStream();
                StreamReader readStream = new StreamReader(receiveStream, Encoding.UTF8);
                // print the error received from Server
                Console.WriteLine("Response error received - ");
                Console.WriteLine(readStream.ReadToEnd());
            }
        }
}
```

The information provided is intended for better general understanding and is not intended to comprehensively address all possible issues that may arise. The contents are correct as at 31 Jan 2019 and are provided on an "as is" basis without warranties of any kind. IRAS shall not be liable for any damages, expenses, costs or loss of any kind however caused as a result of, or in connection with your use of this document.

While every effort has been made to ensure that the above information is consistent with existing policies and practice, should there be any changes, IRAS reserves the right to vary our position accordingly.