Tamigo Web Services 1.7.7

Contents

Summary

Tamigo offers a set of REST based services for you to create new ways to interact with Tamigo. The services are divided into two categories.
The first set of services can be called using an application access, and is meant to be used for integrating third party applications with Tamigo
The other set of services uses a Tamigo user as authentication, and will return data based on the users context. These services can be used to create other user experiences for Tamigo, and we use it ourselves to power the mobile applications for Tamigo.
The services are all capable of returning xml or json-formatted data. Which response you get will be based on type of input and the specified content-type in the HTTP headers.
Services URL: https://api.tamigo.com can also be used without SSL.

Application Services

Login Service

Application Login

To login as an application first log in to Tamigo as an administrator. Go to Configuration -> Manage Application Keys to create a new application key.
The application key is used for authentication of your application and it is therefore paramount that you secure it properly.
If your key has been compromised, it is possible to regenerate the key to ensure the security of your Tamigo instance.
The response returned from the service contains the SessionToken, use this token in subsequent calls to the service to authenticate yourself.

Request (JSON)

```
POST /login/application
Content-Type: application/json

{"Name":"micros", "Key":"longstringasyourpasscodegeneratedfromtheapplication"}
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{"DefaultDepartmentId":"",
" DefaultDepartmentName ":"",
"DefaultCompanyId":"",
"DefaultCompanyName":"",
"Email":"",
"Password":"",
"Role":"Application",
"MenuId":"5",
"SessionToken":"e2c152f3-49ea-4b50-948b-bc0520261737"}
```

Partner Login

To login as a Partner, you need to get a login from Tamigo (support@tamigo.com). The partner key is used for authentication of you and it is therefore paramount that you secure it properly.
If your key has been compromised, you need to contacts as right away, so we can disable it and give you a new key.
The response returned from the service contains the SessionToken, use this token in subsequent calls to the service to authenticate yourself.

Request (JSON)

```
POST /login/application
Content-Type: application/json

{"Name":"micros", "Key":"longstringasyourpasscodegeneratedfromtheapplication"}
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{"DefaultDepartmentId":"",
" DefaultDepartmentName ":"",
"DefaultCompanyId":"",
"DefaultCompanyName":"",
"Email":"",
"Password":"",
"Role":"Application",
"MenuId":"5",
"SessionToken":"e2c152f3-49ea-4b50-948b-bc0520261737"}
```

Departments

Tamigo offers an way to get a list of all departments for a company.

Get All Departments

Retrieve a list of all departments of the company. Select whether you want to include virtual departments or not.

Request (JSON)

```
GET /departments/application/?securityToken=<token>
&includeParents={includeParentDepartments}
Content-Type: application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[{"DepartmentId":"00000000-0000-0000-0000-
000000000001","IsDefault":false,"Name":"Alle
afdelinger","ParentDepartmentId":null},
...
{"DepartmentId":"dc1a04be-fb08-443f-aa3f-
06caa7baab28","IsDefault":false,"Name":"Copenhagen","ParentDepartmentId":"00000
000-0000-0000-0000-000000000001"}
```

Attendance

Tamigo offers the attendance service, which allows you to register check-in and check-out for employees directly in Tamigo from a third-party system (e.g. Point-Of-Sale system).
The Attendance Service requires you to set up keys in Tamigo for the departments and employees whom should be able to use the service.
To set up these keys, login to Tamigo as a Planner or Administrator. Go to Configuration -> Departments, to set the Department ID (POS Key ID). Go to Configuration -> POS Keys, to set the corresponding keys for the Employees.

Check-in

Use check-in to register an employee has arrived at work. Tamigo will register the check-in at the time you send, and match it to the employee and department. As time rounding rules can be applied, the service returns the rounded time.

Parameter	Format	Example
departmentKey	Tekst	A123, 1234567, abcdefg

employeeKey	Tekst	A123, 1234567, abcdefg
checkInTime	yyyy-MM-ddThh:mm:ss	2011-09-02T12:37:00
sessionToken	as returned by login	766c9732-e2d1-46d1-ae3e-a74c560bb8e6

Request (JSON)

```
POST /attendance/checkin/?token=<sessionToken> HTTP/1.1
Content-Type: application/json

{"DepartmentKey":"<departmentKey>",
"EmployeeKey":"<employeeKey>","Time":"<checkInTime>"}
```

Response (JSON)

```
HTTP/1.1 200 OK
```

Check-out

Use check-out if you want to check out from work. Usage as check-in.

Parameter	Format	Eksempel
departmentKey	Tekst	A123, 1234567, abcdefg
employeeKey	Tekst	A123, 1234567, abcdefg
checkInTime	yyyy-MM-ddThh:mm:ss	2011-09-02T12:37:00
sessionToken	Som modtaget ved login	766c9732-e2d1-46d1-ae3e-a74c560bb8e6

Request (JSON)

```
POST /attendance/checkout/?securityToken=<sessionToken> HTTP/1.1
Content-Type: application/json

{"DepartmentKey":"<departmentKey>",
 "EmployeeKey":"<employeeKey>",
 "Time":"<checkInTime>"}
```

Response (JSON)

```
HTTP/1.1 200 OK
```

Export Absence

Use to export absence from 1 or more departments

It takes the following parameters:

StartDate: date formatted like MM-DD-YYYY

EndDate: date formatted like MM-DD-YYYY

DepartmentId: optional. If left out, result is returned for all departments

Request (JSON)

GET
/Leave/ByDate/?startDate=<startDate>&endDate=<endDate>&departmentId=<departmentId>securityToken=<sessionToken>
HTTP/1.1
Content-Type application/json

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[{"AbsenceType":"Vacation",
"Date":"\/Date(1324854000000+0100)\/",
"Name":"John Doe",
"WageSystemKey":"00044"},
...
{"AbsenceType":"Vacation",
"Date":"\/Date(1324940400000+0100)\/",
"Name":" John Doe ",
"WageSystemKey":"00044"}]
```

Import Revenue

Use to import the daily revenue.
It takes the following parameters:

ApplicationName: Same name as used when logging ind (can be left out)
Key: Same key as used when loggin in (can be left out)
Content: A csv format formatted like this: departmentId, date (YYYY-MM-DD)], revenue
Format: should use “Standard”
DateTimeRecived: the time you send the request
Type: for now send NULL

Request (JSON)

POST /Revenues/?securityToken=<sessionToken> HTTP/1.1
Content-Type application/json

```
{
 "ApplicationName": "<ApplicationName>",
 "Key": "<ApplicationKey>”,
 "Content": [
  "100,2011-01-01,1000",
  "100,2011-01-02,1200"
 ],
```

```
 "Format": "Standard",
 "DateTimeReceived": "\/Date(1327996809388+0100)\/",
 "Type": null
}
```


Response (JSON)
HTTP/1.1 200 OK

Import Revenue Over day/ Footfall

Use to import the daily revenue.
It takes the following parameters:

ApplicationName: Same name as used when logging ind (can be left out)
Key: Same key as used when loggin in (can be left out)
Content: A csv format formatted like this:
amount;amountType;startDateTime;endDateTime;posId;statusType;employeePosKey
Format: should use “Standard”
DateTimeRecived: the time you send the request
Type: for now send NULL

Amount: The number of either revenue or footfall.
Amountype: 1 = Revenue, 2 = Footfall (customers)
StartDateTime: Start of data to import (should be whole quarters start and end 01-01-2014 13:00 to 01-01-2014 13:15)
EndDateTime: End of data to import
posId: Department Store Id
statusType: 1 = Actual, 2 = forecast.
employeePosKey: Should only be filled if its revenue and employee specific

Request (JSON)
POST /Revenues/UploadRevenueOverDay?securityToken=<sessionToken> HTTP/1.1
Content-Type application/json


```
{
 "ApplicationName": "<ApplicationName>",
 "Key": "<ApplicationKey>”,
 "Content": [
  "12;2;2014-01-01 13:00;201-01-01 13:15;999;1;NULL",
  "6;2;2014-01-01 13:15;201-01-01 13:30;999;1;NULL",
 ],
 "Format": "Standard",
 "DateTimeReceived": "\/Date(1327996809388+0100)\/",
 "Type": null
```

```
}
```

Response (JSON)
HTTP/1.1 200 OK

Import Transactions

Use to import the daily revenue including related transactions.
It takes the following parameters:
DepartmentKey: Id of the department
EmployeePosKey: for now send NULL
Time: Time of the transaction
Amount: Total amount of included lines
Lines: Array with the following parameters:
Count: Number of items
Price: Item price
ProductKey: Product key of product
ProductName: Name of product

Request (JSON)
POST /Revenues/UploadTransactions?securityToken=<sessionToken> HTTP/1.1
Content-Type application/json

```
[
 {
  "DepartmentKey": "100",
  "EmployeePosKey": null,
  "Time": "\/Date(1412589803535+0200)\/",
  "Amount": 1000.5,
  "Lines": [
   {
    "ProductKey": "Some-product-key",
    "ProductName": "Some-product-name",
    "Count": 2,
    "Price": 500.25
   }
  ]
 }
]
```

Response (JSON)
HTTP/1.1 200 OK

Employee Service

The employee service allows you to retrieve and update employee information for use in a third party system (e.g. Salary System). Employee service also allows you to create new employees, planners or administrators from third party applications.

Retrieve Employees

If you want a list of employees this is where you go. All employees in the company will be retrieved.

Request (JSON)

```
GET /employees/?token=<token>
Content-Type: application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[{"Email":"user1@tamigo.com","EmployeeId":"9c0f571e-cd0e-4e38-ac51-
ecf965cd3de6","IsPlanner":false,"Name":"Søren","PosKey":null,
"DepartmentRoles":null },
                                        ...
{"Email":"tpl125@tamigo.com","EmployeeId":"d1d0b7db-c774-4e81-b094-
d1d2acaa1d83","IsPlanner":false,"Name":"Janik","PosKey":null,
","DepartmentRoles ":null }]
```

Retrieve Detailed Employees

500 employees of a company will be retrieved including relational data like notes, salary weight and competencies. Employees comes in alphabetical order and can be paged. You will get the first 500 employees if excluding page from the query or setting it to 0.

Full request (JSON)

```
GET/employees/getemployeedetails/?token=<token>&page=<pagenumber>&includedelete
d<true,false>
Content-Type: application/json
```

Minimal request (JSON)

```
GET/employees/getemployeedetails/?token=<token>
Content-Type: application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{
    "About": "",
```

```
"AbsenceBalance": [
  {
    "Name": "Vacation",
    "Period": "CalendarYear",
    "Unit": "Day",
    "Value": 25,
    "Year": 2013
  }
],
"Address1": "",
"Address2": "",
"BankAccount": "",
"BankRegistration": "",
"Birthdate": "1984-01-01",
"CanAddOwnShiftsOnSmartphone": false,
"CanApproveEmployeeChanges": false,
"CanEditActualHours": false,
"CanEditPlan": false,
"CanSendSms": false,
"CarRegistration": "",
"City": "",
"Competencies": [{
    "Name": "",
    "ShortCode": "",
    "Status": false
  }],
"CurrentWageRateType": "Monthly",
"CustomColumns": [{
    "Description": "",
    "Name": "",
    "Type": "CheckBox",
    "Value": "False"
  }],
"CustomRoles": [{
    "CustomRoleModels": [
      {
        "IsAdminRole": false,
        "IsEmployeeRole": false,
        "IsPlannerRole": true,
        "Name": "Shop Manager"
      }
    ],
    "DepartmentId": "49664884-8382-4712-9953-fb36ca13fd1b"
  }],
"DeletedOn": null,
"EmployeeDocuments": [{
    "Description": "",
    "DocumentGroupId": "c0ba547f-7557-4b76-a1e7-0b18155fe99a",
    "FileName": "Contract.jpg"
  }],
"EmployeeId": "709c28a1-bc89-4f01-ab08-e61fcfed614f",
"EmployeeIsAdmin": false,
```

```
    "EmployeeType": "",
    "EmployerNumber": "",
    "EnforceMaxWeeklyWorkingHours": false,
    "From": "2000-01-01",
    "HaveRightApproveEmployeeChanges": false,
    "HeadCount": null,
    "HistoricalWages": [{
        "StartDate": "2000-01-01",
        "Wage": 120
     }],
    "HomeDepartment": "1616a1e1-c351-4ef3-8f23-27561409bb2b",
    "IdNumber": "",
    "IsUserEnabled": true,
    "MaximumHours": 40,
    "MaximumWageRate": 10,
    "MinimumHours": 30,
    "Name": "Janik",
    "NotAvailable": [{
        "From": "03:00",
        "To": "06:00",
        "WeekDayIndex": 3
      }],
    "Notes": [{
        "Author": "4a408a58-f9d6-4898-80a5-c3ef5667d2c6",
        "Comment": "",
        "Date": "2016-03-17"
      }],
    "Phone": "123456",
    "Phone2":"123456",
    "ReasonForTermination": "",
    "Roles": [{
        "Admin": false,
        "DepartmentId": "1616a1e1-c351-4ef3-8f23-27561409bb2b",
        "Employee": true,
        "Planner": false
      }],
    "SalaryWeightGroups": [{
        "StartDate": "2000-01-01",
        "Weights": [{
            "OrganizationalUnitId": "1616a1e1-c351-4ef3-8f23-27561409bb2b",
            "WeightPercent": 100
          }]
      }],
    "SendShiftReminder": false,
    "StandardHours": null,
    "To": "2010-01-01",
    "WageNumber": "1234",
    "WorkTravelTime": 30,
    "Zip": ""
  }
]
```

Create Employees

You can create a new employee, planner or administrator. If you leave department blank the default department will be used, and if password is blank tamigo will be used as password.

Request (JSON)

```
POST
/Employees/?securityToken={tokenId}&departmentId={departmentId}&password={passw
ord}
Content-Type: application/json

{"Email": "karina@tamigo.com",
 "Name": "Karina Jensen",
 "Phone": "88888888",
"EmployeeNumber": "1234",
"WageSystemKey": "4321",
"Role": "name of the role ex. Employee, Planner, Administrator",
"IsEnabled": "true"
}
```

Response (JSON)

```
HTTP/1.1 200 OK
{ Success":true}
```

Create/Update Employee with POS-key

You can create and update employees.

Requirements:

- Employee Information along with the DepartmentKey and employee POS-Key must be provided.
- If employee's POS-Key is already available in the database, then employee record will be updated.
- Employee's end date is not always expected.
- If email provided already exists in the system, it will not be added for employee – but employee will still be created/updated.
- DepartmentKey represents department id, and must be valid for employee to be imported/updated.

Notes:

- Role accepts both standard Tamigo role names as well as custom roles. If no matching name is found, employee will be created with Employee role.

Request (JSON)

```
POST /employees/CreateUpdateEmployees/?securityToken=<token> HTTP/1.1
Content-Type: application/json
```

```
{
    "ApplicationName": "<ApplicationName>",
    "Key": "<ApplicationKey>”,
    "Employees": [
         {
            "FirstName":"Jani",
            "LastName":"Doe",
            "Email":"support@tamigo.dk",
            "ActivateEmployee":true ,
            "SocialSecurityNumber":"45621297894",
            "WageSystemKey":"",
            "WageModel":"Fuldtid",
            "WageModelDescription":"",
            "Role":"Employee",
            "AddressLine1":"Kristania Gade 7",
            "AddressLine2":"abc",
            "PostCode":"2000",
            "City":"Copenhagen",
            "MobilePhoneNumber":"57845667",
            "HomePhoneNumber":"542698712",
            "DepartmentKey":"1002",
            "SetAsHomeDepartment":true,
            "BankRegistrationNumber":"3223",
            "BankAccountNumber":"0139-38392029",
            "MonthlySalary":null,
            "ContractHours":null,
            "PosKey":"123456",
            "DateOfBirth":null,
            "StartDate":null,
            "EndDate":null,
            "HourlyRate":null,
            "EmployeeId":null,
            "Contents":""

         }]
 }
```

Response (JSON)

```
HTTP/1.1 200 OK
```

Create/Update Employee with Custom fields

You can create and update employees, and assign values to custom fields for each employee.

Requirements:

- Combination of employee name + payroll number is used as employee’s unique identification for data upload. During import, if combination is found in database, employee data will be updated, otherwise a new employee will be created.

- If email provided already exists in the system, it will not be added for employee – but employee will still be created/updated.
- DepartmentKey represents store id, and must be valid for employee to be imported/updated.
- Custom fields in Tamigo have a UniqeId that they are identified with. Currently this property does not have a user interface, and therefore in case you will create new custom fields, contact Tamigo to set up their UniqueId property.

Notes:

- Role accepts both standard Tamigo role names as well as custom roles. If no matching name is found, employee will be created with Employee role.
- You can choose to specify either WageModel (name) or WageModelDescription. If neither of them match your current setup in Tamigo, a random wage model will be assigned.
- To make sure salary is set up correctly, make sure to send MonthlySalary or HourlyRate depending on whether employee's wage model is monthly or hourly paid.

Request (JSON)

```
POST /employees/CreateUpdateEmployees/?securityToken=<token> HTTP/1.1
Content-Type: application/json

{
    "Employees": [
        {
           "FirstName":"Jani",
           "LastName":"Doei",
           "Email":"support@tamigo.dk",
           "ActivateEmployee":false,
           "SocialSecurityNumber":"0123-456789",
           "WageSystemKey":"WageSystem EmployeeKey",
           "WageModel":"Fuldtid",
           "WageModelDescription":"",
           "Role":"Employee",
           "AddressLine1":"Kristania Gade 7",
           "AddressLine2":"abc",
           "PostCode":"2000",
           "City":"Copenhagen",
           "MobilePhoneNumber":"12345678",
           "HomePhoneNumber":"12345678",
           "DepartmentKey":"34",
           "SetAsHomeDepartment":true,
           "BankRegistrationNumber":"0123",
           "BankAccountNumber":"00456789",
           "MonthlySalary":null,
           "HourlyRate":null,
           "ContractHours":null,
          "PosKey":"123456",
           "DateOfBirth":null,
           "StartDate":"\/Date(1455577200000+0200)\/",
           "EndDate":null,
```

```
            "CustomColumns":
                      [
                                {"Name": "initals","Value": "123"},
                                {"Name": "SomeList", "Value": "kørekort"}
                      ]
         }
   ]
 }
```

Response (JSON)

```
HTTP/1.1 200 OK
```

If data import went well, nothing will be returned from the API. But if any errors occurred, the error information will be provided in XML string:

```
<string xmlns="http://schemas.microsoft.com/2003/10/Serialization/">Invalid
token</string>
```

Delete Employee

Requirements:

- Employee POS-key is required to delete a particular employee's record.

Request (JSON)

```
POST /employees/DeleteEmployee/?posKey=<POS-key>&securityToken=<token> HTTP/1.1
Content-Type: application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
```

Absence Service

The employee service allows you to create, retrieve and delete employee absences from third party applications.

Create absence

Create an employee absence (one or multiple) by passing employee POS-key and absence type.

Requirements:

- Employee POS-key, Start date and End date of absence, Absence type and Absence is approved or not are required.

Request (JSON)

```
POST /LeaveRequests/Create/?securityToken=<token>
Content-Type: application/json

   [{
     "LeaveRequestType":6,
     "PosKey":"0002",
     "FromDate":"\/Date(1467370454000)\/",
     "ToDate":"\/Date(1468148054000)\/",
     "Comment":"",
     "RequestedDate":"\/Date(1464778454000)\/",
     "Approved":null | true | false,
     "Rejected": null | true | false,
     "ApprovedDate":null,
     "NumberOfHours":null,
     "DepartmentId":<Store id> | null,
     "EmployeeLeaveRequestId":null
     }]
```

Response (JSON)

```
HTTP/1.1 200 OK
```

Create / Update sick absence

Create an employee sick absence by passing employee POS-key, absence date, Leave request type Id and a flag that whether employee is sick or not sick.

Requirements:

- When creating employee sick absence, IsSick flag is required to be true and absence date passed will be considered as the date when the employee sickn absence starts. Later when IsSick flag is passed as false, absence date provided will be the date when employee sick absence ends.

Request (JSON)

```
POST /LeaveRequests/CreateUpdateSickLeave/?securityToken=<token>
HTTP/1.1
Content-Type: application/json
{
    "EmployeePosKey":"1000",
    "LeaveDate":"\/Date(1467370454000)\/",
    "IsSick":true,
    "LeaveRequestTypeId":6
}
```

Response (JSON)

```
HTTP/1.1 200 OK
```

Delete an absence

Requirements:

- Employee POS-key, absence start date and End date of absence, Absence name are required.

Request (JSON)

```
POST /LeaveRequests/Delete/?posKey=0002&dateFrom=2016-07-01&dateTo=2016-07-10&leaveType=6&securityToken=<token>
Content-Type: application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
```

Delete all absences

Deletes all employee absences that are related to employee POS-key and are within a given start and end date range.

Requirements:

- start date and end date of absences are required.
- Employee POS-key is an optional parameter.

Request (JSON)

```
POST /LeaveRequests/Delete/All/?posKey=0002&dateFrom=2016-07-01&dateTo=2016-07-10&securityToken=<token>
Content-Type: application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
```

Get all absences by date range

Get all absences that are within a given start and end date range, including overlapping absences

Requirements:

- Start date and end date of absences are required.

Request (JSON)

```
GET /LeaveRequests/All/?startdate=2016-06-09&enddate=2016-06-
10&securityToken=<token>
HTTP/1.1
Content-Type: application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
[{
    "DepartmentName": "Department Name",
    "EmployeeId": " employee Id",
    "EmployeeName": "Jan Doe",
    "LeaveApproved": true,
    "LeaveEndDate": "/Date(1465596000000+0200)/",
    "LeaveRejected": false,
    "LeaveStartDate": "/Date(1465509600000+0200)/",
    "LeaveType": "Ferie"
  }]
```

Kpi Services

Upload Kpi data (Budget revenue, Actual Revenue, Footfall)

This service makes it possible to upload arbitrary data to Tamigo for showing in numerous places. For example, can hourly budget be shown on the daily overview for planning on an hourly basis**. Requires Application Login.**

For other types of data contact Tamigo.

POST

/KPI/Upload/{status}/{type}/{period}/?securityToken={tokenId}
Content-Type: application/json

Status: Actual = 1 Forecast = 2 Budget = 3
Type: Revenue = 1 Customers = 2
Period (Time interval): Quarter = 1 Half hour = 2 Hour = 3 Day = 4 Week = 5 Month = 6

Body

```
{
"DepartmentId": "10",
 "Amount": "560,44",
 "Start": ""\/Date(1412589803535+0200)\/"
},{
"DepartmentId": "12",
```

```
 "Amount": "560,44",
 "Start": ""\/Date(1412589803535+0200)\/"
}
```

User Services

Login Service

User Login

To use user services you need to login as a user of Tamigo, providing their email and password for authentication. The response returned from the service contains the SessionToken, use this token in subsequent calls to the service to authenticate yourself.

Request (JSON)

```
POST /Login/
HTTP/1.1
Content-Type: application/json

{"Email": "karina@tamigo.com",
  "Password": "password",
  "DefaultCompanyId": "3e7131b4-d2e7-44ab-a4bf-5ecbcceff011"}
```

The *DefaultCompanyId* its an optional parameter, if we left blank *DefaultCompanyId*, the service do login to Default Department.

Request (XML)

```
<Login xmlns="http://schemas.datacontract.org/2004/07/Tamigo.Services.Entities"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
<Email>[Email]</Email>
 <Password>[Password]</Password>
</Login>
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{"DefaultDepartmentId":"1ca845e9-28f4-5f79-9517-6d0eac942564",
"DefaultDepartmentName":"Ny Bistro",
"DefaultCompanyId":"1fc5e5e9-28f4-42e6-9917-6faf8c98da64",
"DefaultCompanyName":"Ny Bistro",
"Email":"test@example.com",
"Password":"",
"Role":"Planner",
"SessionToken":"1fcbace9-28f4-4e76-9917-6f0ebc98da64"}
```

Response (XML)

```
<Login xmlns="http://schemas.datacontract.org/2004/07/Tamigo.Services.Entities"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
<DefaultDepartmentName>[DepartmentName]</DefaultDepartmentName>
<Email>[Email]</Email>
<Password/><Role>[Role]</Role><SessionToken>[SessionToken]</SessionToken>
</Login>
```

Get Token

Returns the token information. Available for All.

Request (JSON)

```
GET /Login/?securitytoken={sessionToken}
Content-Type: application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{"DefaultDepartmentId":"1ca845e9-28f4-5f79-9517-6d0eac942564",
"DefaultDepartmentName":"Ny Bistro",
"DefaultCompanyId":"1fc5e5e9-28f4-42e6-9917-6faf8c98da64",
"DefaultCompanyName":"Ny Bistro",
"Email":"test@example.com",
"Password":"",
"Role":"Planner",
"SessionToken":"1fcbace9-28f4-4e76-9917-6f0ebc98da64"}
```

Menu service

The menu service contains everything about menu items.

Mobile Menu

This service returns a list of menu items. The service determines by the companyid and role from the token which menu is to be returned.
The menu contains headers (parentId = null) and an item (parentId is equal to header id).
The list is sorted the right way by the service.
Version determines what the menu will include. Currently highest version of menu is **2**.

Request (JSON)

```
GET /Menu/mobile/?securitytoken={sessionToken}&Version={version}
Content-Type: application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
```


```
"[{\"DefaultChildId\":null,\"Description\":\"\",\"Enabled\":true,\"Id\":1024,\"
OrderIndex\":1,\"ParentId\":null,\"ResourceId\":\"InfoHeader\",\"Role\":\"4ac90
ed7-edd4-4e4c-98d9-
3aa4b1b85d7d\",\"Selected\":false,\"Url\":null},{\"DefaultChildId\":null,\"Desc
ription\":\"\",\"Enabled\":true,\"Id\":1025,\"OrderIndex\":1,\"ParentId\":1024,
\"ResourceId\":\"FrontPage\",\"Role\":\"4ac90ed7-edd4-4e4c-98d9-
3aa4b1b85d7d\",\"Selected\":false,\"Url\":null }]"
```

Shift service

The shift service can be used to retrieve information on rosters and upcoming shifts.

My overview

Returns a list of the logged in employees upcoming shifts. Dates are expressed as milliseconds since EPOCH (1970-1-1 00:00:00). Available for All.

Request (JSON)

```
GET /shifts/future/?securitytoken={sessionToken}
Content-Type: application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[{"BreakCode":"Q",
      "Comment":"",
      "DepartmentName":"Ny Bistro",
      "EmployeeName":null,
      "EndTime":"\/Date(1318860000000+0200)\/",
      "StartTime":"\/Date(1318831200000+0200)\/"},
                                          ...
{"BreakCode":"Q",
      "Comment":"",
      "DepartmentName":"Ny Bistro",
      "EmployeeName":null,
      "EndTime":"\/Date(1318946400000+0200)\/",
      "StartTime":"\/Date(1318917600000+0200)\/"}]
```

Response (XML)

```
<ArrayOfShift
xmlns="http://schemas.datacontract.org/2004/07/Tamigo.Services.Entities"
xmlns:i=\"http://www.w3.org/2001/XMLSchema-instance">
<Shift>
<BreakCode>Q</BreakCode>
<Comment/>
<DepartmentName>Herning</DepartmentName>
<EmployeeName i:nil=\"true\"/>
<EndTime>2011-12-19T06:00:00</EndTime>
<IsAvailable>false</IsAvailable>
<IsExchange>false</IsExchange>
<ShiftId>2f5e8bcc-a419-4c2b-891a-54fa4592bd52</ShiftId>
<StartTime>2011-12-18T23:00:00</StartTime>
</Shift>
</ArrayOfShift>
```

Authorized roster - today

This resource returns a list of the authorized roster for the day including employees that are on leave, signified by an associated activity. Available for all Roles. If department is equal to “all”, shift from all rosters in all departments are returned.

Request (JSON)

```
GET /shifts/today/?securitytoken={sessionToken}(&departmentId={departmentId})
Content-Type: application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[{"BreakCode":"Q","Comment":"",
"DepartmentName":"Ny Bistro","EmployeeName":"",
"EndTime":"\/Date(1316700000000+0200)\/",
"StartTime":"\/Date(1316671200000+0200)\/"},
                                        ...
{"BreakCode":"Q","Comment":"",
"DepartmentName":"Ny Bistro","EmployeeName":"",
"EndTime":"\/Date(1316700000000+0200)\/",
"StartTime":"\/Date(1316671200000+0200)\/"}]
```

Response (XML)

```
"<ArrayOfShift
xmlns="http://schemas.datacontract.org/2004/07/Tamigo.Services.Entities"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
<Shift>
<BreakCode>Q</BreakCode>
<Comment/>
<DepartmentName>København</DepartmentName>
<EmployeeName/>
<EndTime>2011-12-13T16:00:00</EndTime>
<StartTime>2011-12-13T08:00:00</StartTime>
</Shift>
</ArrayOfShift>"
```

Authorized roster – arbitrary day

This resource returns a list of the authorized roster for the day including employees that are on leave, signified by an associated activity. Available for all Roles. If department is equal to “all”, shift from all rosters in all departments are returned.

Request (JSON)

```
GET /shifts/day/{date}/?securitytoken={sessionToken}(&departmentId={departmentId})
Content-Type: application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[{"BreakCode":"Q","Comment":"",
"DepartmentName":"Ny Bistro","EmployeeName":"",
"EndTime":"\/Date(1316700000000+0200)\/",
"StartTime":"\/Date(1316671200000+0200)\/"},
                                        ...
{"BreakCode":"Q","Comment":"",
"DepartmentName":"Ny Bistro","EmployeeName":"",
"EndTime":"\/Date(1316700000000+0200)\/",
"StartTime":"\/Date(1316671200000+0200)\/"}]
```

Authorized roster - today

This resource returns a list of the authorized roster for the day including employees that are on leave, signified by an associated activity. Available for all Roles. If department is equal to “all”, shift from all rosters in all departments are returned.

Request (JSON)

```
GET /shifts/today/?securitytoken={sessionToken}(&departmentId={departmentId})
Content-Type: application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[{"BreakCode":"Q","Comment":"",
"DepartmentName":"Ny Bistro","EmployeeName":"",
"EndTime":"\/Date(1316700000000+0200)\/",
"StartTime":"\/Date(1316671200000+0200)\/"},
                                        ...
{"BreakCode":"Q","Comment":"",
"DepartmentName":"Ny Bistro","EmployeeName":"",
"EndTime":"\/Date(1316700000000+0200)\/",
"StartTime":"\/Date(1316671200000+0200)\/"}]
```

Response (XML)

```
"<ArrayOfShift
xmlns="http://schemas.datacontract.org/2004/07/Tamigo.Services.Entities"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
<Shift>
<BreakCode>Q</BreakCode>
<Comment/>
<DepartmentName>København</DepartmentName>
<EmployeeName/>
<EndTime>2011-12-13T16:00:00</EndTime>
<StartTime>2011-12-13T08:00:00</StartTime>
</Shift>
</ArrayOfShift>"
```

Authorized roster – period of dates by employee

This resource returns a list of the authorized roster for the period of dates for the employee selected. Available for all Roles. If department is equal to “all”, shift from all rosters in all departments are returned.

Request (JSON)

```
GET
/shifts/period/{startDate}/{endDate}/?securitytoken={sessionToken}(&departmentI
d={departmentId})&employeeId=employeeId
Content-Type: application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[{"BreakCode":"Q","Comment":"",
"DepartmentName":"Ny Bistro","EmployeeName":"",
"EndTime":"\/Date(1316700000000+0200)\/",
"StartTime":"\/Date(1316671200000+0200)\/"},
                                        ...
{"BreakCode":"Q","Comment":"",
"DepartmentName":"Ny Bistro","EmployeeName":"",
"EndTime":"\/Date(1316700000000+0200)\/",
"StartTime":"\/Date(1316671200000+0200)\/"}]
```

Response (XML)

```
"<ArrayOfShift
xmlns="http://schemas.datacontract.org/2004/07/Tamigo.Services.Entities"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
<Shift>
<BreakCode>Q</BreakCode>
<Comment/>
<DepartmentName>København</DepartmentName>
<EmployeeName/>
<EndTime>2011-12-13T16:00:00</EndTime>
<StartTime>2011-12-13T08:00:00</StartTime>
</Shift>
</ArrayOfShift>"
```

Get all types of shifts – Original, planned and actual shifts

This resource returns a list of the all types of shifts for the period of dates. Available only for all Application. If department is equal to “all”, shift from all rosters in all departments are returned.

Request (JSON)

```
GET
/shifts/all/{startDate}/{endDate}/?securityToken={tokenId}&departmentId={depart
mentId}") Content-Type: application/json
```

Response (JSON)

```
Host: api.tamigo.com
Content-Type: application/json
Accept: application/json


[{

    "Date": "/Date(1453071600000+0100)/",
    "DepartmentId": "dc1a04be-fb08-443f-aa3f-06b0a7bffb28",
    "DepartmentKey": "835",
    "DepartmentName": "Copenhagent",
    "EmployeeId": "00000000-0000-0000-0000-000000000000",
    "FirstName": null,
    "EmployeePosKey": "E100",
    "Email": "user@tamigo.com",
    "OriginalShift": {
      "ActivityName": null,
      "Comments": "",
      "Shift": 7.25,
      "ShiftActivityId": null,
      "Sum": null
    },
    "PlannedShift": {
      "ActivityName": null,
      "Comments": "",
      "Shift": 7.25,
      "ShiftActivityId": null,
      "Sum": null
    },
    "ActualShift": {
      "ActivityName": null,
      "Comments": null,
      "Shift": null,
      "ShiftActivityId": null,
      "Sum": null
    },

    "WageNumber": null
  }
]
```

Response (XML)


```
"<ArrayOfEmployeeShift
xmlns="http://schemas.datacontract.org/2004/07/Tamigo.Services.Entities"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
    <EmployeeShift>
        <Date>2016-01-05T00:00:00</Date>
        <DepartmentId>dc1a04be-fb08-443f-aa3f-06b0a7bffb28</DepartmentId>
        <DepartmentKey>835</DepartmentKey>
        <DepartmentName>Copenhagent</DepartmentName>
        <EmployeeId>00000000-0000-0000-0000-000000000000</EmployeeId>
        <FirstName i:nil="true"/>
        <OriginalShift>
            <ActivityName i:nil="true"/>
            <Comments/>
            <Shift>9.00</Shift>
            <ShiftActivityId i:nil="true"/>
            <Sum i:nil="true"/>
        </OriginalShift>
        <PlannedShift>
            <ActivityName i:nil="true"/>
            <Comments/>
            <Shift>9.00</Shift>
            <ShiftActivityId i:nil="true"/>
            <Sum i:nil="true"/>
        </PlannedShift>
         <ActualShift>
            <ActivityName i:nil="true"/>
            <Comments i:nil="true"/>
            <Shift i:nil="true"/>
            <ShiftActivityId i:nil="true"/>
            <Sum i:nil="true"/>
        </ActualShift>

        <WageNumber i:nil="true"/>
    </EmployeeShift>
```

Get relevant planned shift

This returns the most relevant planned shift for the employee. By relevant means the shift that has not ended yet.

Request (JSON)

```
GET /shifts/planned/relevant/?securitytoken={sessionTokenContent-Type:
application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{"BreakCode":"Q","Comment":"",
"DepartmentName":"Ny Bistro","EmployeeName":"",
"EndTime":"\/Date(1316700000000+0200)\/",
"StartTime":"\/Date(1316671200000+0200)\/"},
                                        ...
{"BreakCode":"Q","Comment":"",
"DepartmentName":"Ny Bistro","EmployeeName":"",
"EndTime":"\/Date(1316700000000+0200)\/",
"StartTime":"\/Date(1316671200000+0200)\/"}
```

Response (XML)

```
"<Shift
xmlns="http://schemas.datacontract.org/2004/07/Tamigo.Services.Entities"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
<BreakCode>Q</BreakCode>
<Comment/>
<DepartmentName>København</DepartmentName>
<EmployeeName/>
<EndTime>2011-12-13T16:00:00</EndTime>
<StartTime>2011-12-13T08:00:00</StartTime>
</Shift>
```

Vacant shifts

Deprecated, use Bid service instead. This resource is only available for backwards compatibility. Returns list of shifts that are open to take. Available for all Roles.

Request (XML)

```
GET
/Shifts/available/?securitytoken={sessionToken}(&departmentId={departmentId})
Content-Type: application/json
```

Response(XML)

```
<ArrayOfAvailableShift
xmlns="http://schemas.datacontract.org/2004/07/Tamigo.Services.Entities"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
<AvailableShift>
<BidPlaced>false</BidPlaced>
<BreakCode>Q</BreakCode>
<Comment/><DepartmentId>dc1a04be-fb08-443f-aa3f-06b0a7bffb28</DepartmentId>
<DepartmentName>København</DepartmentName>
<EmployeeHaveShiftSameDay>false</EmployeeHaveShiftSameDay>
<EndTime>2011-12-13T16:00:00</EndTime>
<ExtendedShiftEndTime>0001-01-01T00:00:00</ExtendedShiftEndTime>
<ExtendedShiftStartTime>0001-01-01T00:00:00</ExtendedShiftStartTime>
<IsExtended>false</IsExtended>
<OtherEmployeesHaveBidOnShift>false</OtherEmployeesHaveBidOnShift>
<ShiftId>d54f1963-bdd4-4929-a053-4efb947f96b2</ShiftId>
<ShiftProviderId>00000000-0000-0000-0000-000000000000</ShiftProviderId>
<ShiftProviderName>Ledig</ShiftProviderName>
<StartTime>2011-12-13T08:00:00</StartTime>
<Week>50</Week>
</AvailableShift>
…
</ArrayOfAvailableShift>
```

Update shift

Use this to update your shift resources. It is important to send all information about the shift back, even if it is not changed. All changes between the original shift and the new shifts will be seen as updates. So if an original shift comment is omitted the comment will be deleted.
This service is also used for employees to set their shifts as “Vacant” by setting “IsAvailable” to “true”. If an employee uses this service, only the “IsAvailable” property will be used. If a planner changes employee on a shift, exchanges and bids will be removed if possible. If the “EmployeeId” is changes to “null” “IsAvailable” will be set as “true” on the server.
This service returns a “Response Message”. If “Success” is true everything is nice. If “Success” is “false” a “Message” will be supplied with a description of the error. This message should be shown to the user.

To update a shift with empty endtime, set endtime to a date less than 01-01-1900.

Request (JSON)

```
PUT /Shifts/{ShiftId}/?securitytoken={sessionToken}

{
  "ShiftId": "fb60311f-e88c-4a6a-9bc4-6ad09fd6a7b5",
  "StartTime": "\/Date(1355986847000+0100)\/",
  "EndTime": "\/Date(1356008447000+0100)\/",
  "Comment": "Early",
  "BreakCode": "P",
  "IsAvailable": false,
  "IsExchange": false,
  "EmployeeId": "4a6b1e63-666d-4955-b42b-20a289da3414",
  "ShiftActivityId": "00000000-0000-0000-0000-000000000000"
}
```

Response (JSON)

```
HTTP/1.1 200 OK

{"Message":null,"Success":true}
```

Add shift

A planner can use everything on a shift besides the IsAvailable and the “IsExchange” property. If the “EmployeeId” is send as “null”, “IsAvailable” will be set as “true” on the server.
This service returns a “Response Message”. If “Success” is true everything is nice. If “Success” is “false” a “Message” will be supplied with a description of the error. This message should be shown to the user.
“type” should be either “planned” or “actual” to define if you are adding a planned or an actual shift.

Request (JSON)

```
POST /Shifts/{type}/?securitytoken={sessionToken}

{
  "StartTime": "\/Date(1355986847000+0100)\/",
  "EndTime": "\/Date(1356008447000+0100)\/",
  "Comment": "Early",
  "BreakCode": "P",
  "IsAvailable": false,
  "IsExchange": false,
  "EmployeeId": "4a6b1e63-666d-4955-b42b-20a289da3414",
  "ShiftActivityId": "00000000-0000-0000-0000-000000000000"
}
```

Response (JSON)

```
HTTP/1.1 200 OK

{"Message":null,"Success":true}
```

Delete shift

A planner or administrator can delete a shift, if they have the role in the specific department.

Request (JSON)

DELETE /Shifts/{shiftId}/?securityToken={tokenId}

Response (JSON)

```
HTTP/1.1 200 OK

{"Message":null,"Success":true}
```

Bid service

The Bid service gives the user a view on shifts that have been offered, and it enables bidding for these shifts.

Get available shifts

This resource returns a list of shifts that are open to bid on. This resource is available for all roles.

Request (JSON)

```
GET /bids/Available/?securitytoken={sessionToken}
```

Response (JSON)

```
HTTP/1.1 200 OK

[{"BreakCode":"Q","Comment":"",
"DepartmentName":"Ny Bistro","EmployeeName":"",
"EndTime":"\/Date(1316700000000+0200)\/",
"StartTime":"\/Date(1316671200000+0200)\/"},
...
{"BreakCode":"Q","Comment":"",
"DepartmentName":"Ny Bistro","EmployeeName":"",
"EndTime":"\/Date(1316700000000+0200)\/",
"StartTime":"\/Date(1316671200000+0200)\/"}]
```

Response (XML)

```
"<ArrayOfAvailableShift
xmlns="http://schemas.datacontract.org/2004/07/Tamigo.Services.Entities"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
<AvailableShift>
<BidPlaced>false</BidPlaced>
<BreakCode>Q</BreakCode>
<Comment/>
<DepartmentId>dc1a04be-fb08-443f-aa3f-06b0a7bffb28</DepartmentId>
<DepartmentName>København</DepartmentName>
<EmployeeHaveShiftSameDay>false</EmployeeHaveShiftSameDay>
<EndTime>2011-12-21T16:00:00</EndTime>
<ExtendedShiftEndTime>0001-01-01T00:00:00</ExtendedShiftEndTime>
<ExtendedShiftStartTime>0001-01-01T00:00:00</ExtendedShiftStartTime>
<IsExtended>false</IsExtended>
<OtherEmployeesHaveBidOnShift>false</OtherEmployeesHaveBidOnShift>
<ShiftId>dbdb767f-1415-4cdb-8763-1a7c2281f1b8</ShiftId>
<ShiftProviderId>00000000-0000-0000-0000-000000000000</ShiftProviderId>
<ShiftProviderName>Ledig</ShiftProviderName>
<StartTime>2011-12-21T08:00:00</StartTime>
<Week>51</Week>
</AvailableShift>
</ArrayOfAvailableShift>"
```

Lay bid for available shift

The logged in employee bids on an available shift
This is only available to Employees. Planners cannot bid on shifts

Request (JSON)

```
PUT /Shifts/Available/{shiftId}/?securitytoken={sessionToken}
```

Response (JSON)

```
HTTP/1.1 200 OK
```

Accept bid on available shift

Use this resource to decide a bid for an available shift. Available only to Planners.
Use the "IsAccepted" property to accept or reject the request

Request (JSON)

```
PUT /Requests/{requestId}/?securityToken={sessionToken}

{
  "RequestId": "dbdb767f-1415-4cdb-8763-1a7c2281f1b8",
  "IsAccepted": true
}
```

Request (XML)

```
<ShiftRequest
xmlns="http://schemas.datacontract.org/2004/07/Tamigo.Services.Entities"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  <ProviderId>00000000-0000-0000-0000-000000000000</ProviderId>
  <RequestId>dbdb767f-1415-4cdb-8763-1a7c2281f1b8</RequestId>
  <StartDateTime>01-01-0001 00:00:00</StartDateTime>
  <EndDateTime>01-01-0001 00:00:00</EndDateTime>
  <ShiftHours>0</ShiftHours>
  <RequesterId>00000000-0000-0000-0000-000000000000</RequesterId>
  <IsAccepted>True</IsAccepted>
</ShiftRequest>
```

Response (JSON)

```
HTTP/1.1 200 OK
```

Withdraw vacant shift

Withdraw shift if it has been made vacant. The bid cannot be withdrawn if other people have bid on the shift.

Request (JSON)

```
PUT /bids/withdraw/{ShiftId}/?securitytoken={sessionToken}
```

Response (JSON)

```
HTTP/1.1 200 OK

{"Message":null,"Success":true}
```

Shift Exchange Service

The Shift exchange service can be used to make exchanges between two shifts, as opposed to the Bid service, where there is only one shift involved.

Create new Shift Exchange

This creates a new Shift Exchange. Available for Employee.

Request (JSON)

```
POST /exchanges/?securitytoken={sessionToken}

{"ProposerShiftId":"<shiftId1>","AccepterShiftId":"<shiftId2>"}
```

Response (JSON)

```
HTTP/1.1 200 OK
```

List of shifts to exchange with

This resource returns a list of shifts that the current shift can be exchanged with. Available for Employee.

Request (JSON)

```
GET /Exchanges/Available/?shiftId={shiftIdToExchange}&securityToken={token}
```

Response (JSON)

```
[{"EmployeeId":"AA1A148A-3138-4BE5-9892-4A6F6EE9F95B",
"Name":"Peter",
"HaveShifts":"false",
"BreakCode":"Q",
"EndDateTime":"\/Date(1316700000000+0200)\/",
"StartDateTime":"\/Date(1316671200000+0200)\/"},
"Comment":"Lukke",
"CanExchangeShift":"false"},
...
{"EmployeeId":"AA1A148A-3138-4BE5-9892-4A6F6EE9F95B",
"Name":"Peter",
"HaveShifts":"false",
"BreakCode":"Q",
"EndDateTime":"\/Date(1316700000000+0200)\/",
"StartDateTime":"\/Date(1316671200000+0200)\/"},
"Comment":"Lukke",
"CanExchangeShift":"true"}]
```

Get List of pending approvals (Employee and Planner)

Get a list of pending Exchange approvals.
An employee will get a list of exchanges that needs to be approved – this means that the employee accepts that he/she would like to exchange the selected shifts.
A Planner will get a list of exchanges that have been approved by the employee and needs to Accept the exchange before it is final.

The Property Violate11HourRule is only available when the planner is accepting the request.
If the property is True. There should be an alert.

Request (JSON)
GET /Exchanges/Requests/?securityToken={sessionToken}

Response (XML)

```
<ArrayOfShiftExchangeRequest
xmlns="http://schemas.datacontract.org/2004/07/Tamigo.Services.Entities"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
<ShiftExchangeRequest>
<Comment i:nil="true"/>
<EmployeeId1>38f02810-6ac4-40b4-b238-39236c826118</EmployeeId1>
<EmployeeId2>3203ba4b-5264-4f36-8daa-76e7e69d0f31</EmployeeId2>
<EmployeeName1>Peter</EmployeeName1>
<EmployeeName2>Karina</EmployeeName2>
<EndDateTime1>2011-12-24T16:00:00</EndDateTime1>
<EndDateTime2>2011-12-23T16:00:00</EndDateTime2>
<IsAccepted i:nil="true"/>
<RequestId>9aeb6e1a-d50a-40f0-bd79-0369fcc25ddc</RequestId>
<ShiftHours1>7.25</ShiftHours1>
<ShiftHours2>7.25</ShiftHours2>
<StartDateTime1>2011-12-24T08:00:00</StartDateTime1>
<StartDateTime2>2011-12-23T08:00:00</StartDateTime2>
<Violate11HourRule>False</Violate11HourRule>
<ShiftComment1>Åbner</ ShiftComment1>
< ShiftComment2>Lukker</ ShiftComment2>
< ShiftBreakCode1>P</ ShiftBreakCode1>
< ShiftBreakCode2>Q</ ShiftBreakCode2>
</ShiftExchangeRequest>
</ArrayOfShiftExchangeRequest>
```

Decide shift exchange

Approve or reject a shift exchange. The approval is a two-step process:

1. Approval by the employee who are exchanging his shift (the one that did not create the shift exchange)
2. Approval by the planner.

Available to All Roles.

Request (JSON)

```
PUT /Exchanges/Requests/{requestId}/ {isAccepted}/?securitytoken={sessionToken}
```

Response (JSON)

```
HTTP/1.1 200 OK
```

Withdraw shift exchange

Withdraw your shift exchange, if the other employee hasn't accepted it yet.

Request (JSON)

```
PUT /Exchanges/withdraw/{shiftId}/?securitytoken={sessionToken}
```

Response (JSON)

```
HTTP/1.1 200 OK

{"Message":null,"Success":true}
```

News Service

The News Service offer access to the information that is posted on the front page of Tamigo.

Get company and department news

This resource returns the content from the front page in Tamigo. The content is returned as HTML. This resource is available to all Roles.

Request (JSON)

```
GET /News/?securityToken={sessionToken})
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: text/html

<html><body>
<h1>{departmentNews}</h1>
<div id="departmentNews">
      {content}
</div>
<h1>{companyNews}</h1>
```

```
<div id="companyNews">
      {content}
</div>
```

Absence Service

The absence service enables the Employee to ask for absence, and the Planner can register absence (e.g. an employee no-show due to illness), the Planner can also approve or reject the Employee's absence requests.

Create new absence request / Register absence

When an employee wishes to plan absence he submits a absence request. When a Planner wishes to register absence he uses the same service. If a planner registers absence there is the option to make any shifts in the absence period "vacant".

Parameter	Format	Example
employeeId	Guid	766c9732-e2d1-46d1-ae3e-a74c560bb8e6
DateFrom	DateTime	2011-09-02T12:37:00
DateTo	DateTime	2011-09-02T12:37:00
leaveTypeId	as specified by leavetype service	1

Request (JSON)

```
POST
/LeaveRequests/?securityToken={sessionToken}&makeShiftsVacant={makeShiftsVacant
}

{"EmployeeId":"<employeeId>", "DateFrom":"<DateFrom>","DateTo":"<dateTo>",
"LeaveTypeId":"<leaveTypeId>, "Comment":"<comment>"}
```

Response (JSON)

```
HTTP/1.1 200 OK
```

List of future absence

This resource returns a list of the registered absence for the logged in employee.

Request (JSON)

```
GET /Leave/Future/?securityToken={sessionToken}
```

Response (JSON)

```
HTTP/1.1 200 OK

[{"LeaveRequestId":"766c9732-e2d1-46d1-ae3e-a74c560bb8e6",
"EmployeeId":"ffff9732-e2d1-46d1-ae3e-a74c560bb702",
“EmployeeName”:”Karina”,
"DateFrom":"\/Date(1316700000000+0200)\/",
"DateTo":"\/Date(1316700000000+0200)\/",
"LeaveTypeId":"1",
”LeaveTypeName”:”Vacation”}
,
...
```

{"LeaveRequestId":"766c9732-e2d1-46d1-ae3e-a74c560bb8e6",
"EmployeeId":"ffff9732-e2d1-46d1-ae3e-a74c560bb702",
“EmployeeName”:”Karina”,
"DateFrom":"\/Date(1316700000000+0200)\/",
"DateTo":"\/Date(1316700000000+0200)\/",
"LeaveTypeId":"1"
”LeaveTypeName”:”Vacation”}]

Decide leave requests

When a planner wishes to approve or deny a leave request he can update the corresponding leave request.
Available to Planner only.

Request (JSON)

```
PUT /LeaveRequests/{requestId}/?securityToken={token}
```

Request (JSON)

{ "LeaveRequestId": "<requestId>",
"IsApproved": “<bool>”,
"MoveShiftsToVacant":” <bool>”,
"SmsComment": "<comment>"}

Response (JSON)

```
HTTP/1.1 200 OK
```

List of leave types groups (New)

This resource returns a list of the currently available leave types groups for use by the leaverequest service.
Available to all roles.
If employeeId and datefrom is provided the groups a limited for that employee.

Request (JSON)

```
GET
/Leave/Groups/?securityToken={tokenId}&employeeId={employeeId}&dateFrom={dateFrom}
```

Response (JSON)

```
HTTP/1.1 200 OK

[{"Id":"766c9732-e2d1-46d1-ae3e-a74c560bb8e6", "Name":"Ferie"},
...
{"Id":"aaaaa732-e2d1-46d1-ae3e-a74c560bb8e6", "Name":"Syg"}]
```

List of leave types (Updated)

This resource returns a list of the currently available leave types for use by the leaverequest service.
Available to all roles, employees can only choose between a subset of leavetypes, as specified in Tamigo.

Update: If absenseGroupId is supplied the list will be filtered by the groups. If all possible leavetypes are returned.

Request (JSON)

```
GET
/Leave/LeaveRequestTypes/?securityToken={tokenId}&employeeId={employeeId}&absen
ceGroupId={absenceGroupdId}")
```

Response (JSON)

```
HTTP/1.1 200 OK

[{"LeaveTypeId":"766c9732-e2d1-46d1-ae3e-a74c560bb8e6", "Name":"Ferie"},
...
{"LeaveTypeId":"aaaaa732-e2d1-46d1-ae3e-a74c560bb8e6", "Name":"Syg"}]
```

List of employees from department

Returns the list of employees you can register absence on.

Request (JSON)

GET /Leave/Employees/?securityToken={token}

Response (JSON)

HTTP/1.1 200 OK

Requests Service

The request service offers the Planner an option to decide on bids from his Employees.

Pending bids for vacant shifts

This resource returns a list of pending bids. Available for Planners

Request (JSON)

```
GET /Requests/?securitytoken={sessionToken}
```

Response (JSON)

```
[{
"Comment":"",
"EndDateTime":"\/Date(1324220400000+0100)\/",
"IsAccepted":null,
"RequestId":"6a957dc0-bdc8-47bf-9a0e-5f4d936f85da",
"RequesterId":"38f02810-6ac4-40b4-b238-39236c826118",
"RequesterName":"Peter",
"ShiftHours":7.25,
"StartDateTime":"\/Date(1324191600000+0100)\/",
"ProviderId":"00000000-0000-0000-0000-000000000000",
"ProviderName":"Ledig",
“Violate11HourRule”:”false”
},
...
{
"Comment":"",
"EndDateTime":"\/Date(1324220400000+0100)\/",
"IsAccepted":null,
"RequestId":"f2a77510-c92b-4543-984b-c7323fc3311b",
"RequesterId":"38f02810-6ac4-40b4-b238-39236c826118",
"RequesterName":"Peter",
"ShiftHours":7.25,
"StartDateTime":"\/Date(1324191600000+0100)\/",
"ProviderId":"00000000-0000-0000-0000-000000000000",
"ProviderName":"Ledig",
“Violate11HourRule”:”false”
}]
```

Decide bid for shift

Approve or reject a bid or shift exchange. Available to Planner, Administrator

Request (JSON)

```
PUT /Requests/{requestId}/?securitytoken={sessionToken}

{“RequestId”:”<requestID>”,”Approved”:<true/false>}
```

Response (JSON)

```
HTTP/1.1 200 OK
```

Contact Service

The contact service returns a list of contact information for employees and planners in the current department.

Contact List

This resource returns a list of available contacts, including email addresses and phone numbers.

Request (JSON)

```
GET /Contacts/?securitytoken={sessionToken}
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[{"Email":"karina@tamigo.com","FirstName":"Andreas
T","ImageUrl":"\/images\/small\/Andreas
T_Untitled.png","LastName":null,"MobilePhone":"+4511111111"},
...
{"Email":"peter@tamigo.com","FirstName":"Ulrik","ImageUrl":"","LastName":null,"
MobilePhone":"+4511111111"}]
```

Departments Service

The department service is used for operations to retrieve a list of departments for the company, and updating of the default department for a User.

All Departments

Get a list of the currently logged in user's available departments.

Request (JSON)

```
GET /Departments/?securitytoken={sessionToken}
Content-Type:application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[{"DepartmentId":"223a4516-41d5-4327-97ca-65c67d6b20a0",
"IsDefault":false,"Name":"Afdeling 1"},
                                        ...
{"DepartmentId":"c4dcc6f2-ff52-4a22-b29d-aae817b5c8e7",
"IsDefault":false,"Name":"Slik"}]
```

Set Default department

In v1 of our iPhone application you change the department that your currently looking at by setting the default department.

Request (JSON)

```
PUT /departments/c4dcc6f2-ff52-4a22-b29d-
aae817b5c8e7/?SecurityToken={sessionToken}
Content-Type: application/json

{"DepartmentId":"c4dcc6f2-ff52-4a22-b29d-aae817b5c8e7","IsDefault":true}
```

Response (JSON)

```
HTTP/1.1 200 OK
```

Revenue Over Day Service

The revenue over day service is used to upload data for forecasting purposes.
AmountTypeId 1 is Revenue and 2 is Customers. In the future this can be expanded.
StatusTypeId 1 is Actual data and 2 is Forecast data. 2 can be used to upload your own forecast data.
Notice that the types has to exist for the upload to succeed.
Notice that the time span for a single unit has to be less than 24 hours, more than 15 minutes and the upload will fail, if the span cannot be divided into 15 minute spans.

Upload single data unit

Upload actual or forecast data for a single department in a single period.

Request (JSON)

```
POST /Revenues /period/single/?securityToken={tokenId}
Content-Type:application/json

{"Amount":"<float>","AmountTypeId":"[1|2]", "End":"<DateTime>",
"PosStoreId":"<string>", "Start":"<DateTime>", "StatusTypeId":"[1|2]"}
```

Response (JSON)

```
HTTP/1.1 200 OK "true"
```

Upload list of data units

Upload actual or forecast data for multiple departments or/and multiple periods.

Request (JSON)

```
POST /Revenues /period/list/?securityToken={tokenId}
Content-Type:application/json

[{"Amount":"<float>","AmountTypeId":"[1|2]", "End":"<DateTime>",
"PosStoreId":"<string>", "Start":"<DateTime>", "StatusTypeId":"[1|2]"},
...
{"Amount":"<float>","AmountTypeId":"[1|2]", "End":"<DateTime>",
"PosStoreId":"<string>", "Start":"<DateTime>", "StatusTypeId":"[1|2]"]
```

Response (JSON)

HTTP/1.1 200 OK "true"

ICal service

The iCal service returns the users “ical link” this is his personal calendar in iCal format

Get ical link

Request (JSON)

```
GET /Calendar/link/?securityToken={tokenId}
Content-Type:application/json
```

Response (JSON)

```
HTTP/1.1 200 OK "true"
```

Upload list of data units

Upload actual or forecast data for multiple departments or/and multiple periods.

Request (JSON)

```
POST /Revenues /period/list/?securityToken={tokenId}
Content-Type:application/json

Result: "https:\/\/services.tamigo.com\/Calendar\/208ef757-4d91-4219-a9d6-
5761d5c03bf3\/Calendar.ics"
```

Response (JSON)

HTTP/1.1 200 OK "true"

Calendar comments service

This service returns the “comments” that a day in the roster have attached. It consists of two short texts like “We are closed” or “Holiday” and a long text that can be several hundred chars long.

It is advised to show the short comments at the top of the roster at all time. The long text should be visible with the press of a button.

Get the calendar comment

Request (JSON)

```
GET /comments/day/{date}/?securityToken={tokenId}
Content-Type:application/json
```

Response (JSON)

```
HTTP/1.1 200 OK "true"
```

Upload list of data units

Upload actual or forecast data for multiple departments or/and multiple periods.

Request (JSON)

```
POST /Revenues /period/list/?securityToken={tokenId}
Content-Type:application/json

Result: "https:\/\/services.tamigo.com\/Calendar\/208ef757-4d91-4219-a9d6-
5761d5c03bf3\/Calendar.ics"
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{
"CompanyCommentShort":"Kort besked",
"DepartmentCommentLong":"Rigtig lang besked!",
"DepartmentCommentShort":"Kort besked"
}
```

Roles service

The Roles service returns the roles that the user have in the specific department.

Get Roles

Request (JSON)

```
GET /Roles/?securityToken={tokenId}
Content-Type:application/json
```

Response (JSON)

```
HTTP/1.1 200 OK "true"

[{"Role 1"}, {"Role 2"}, {"Role 3"}]
```

The Set Role service, setting in the token the new role selected by the user.

Set Role

Request (JSON)

```
POST /Roles/?securityToken={tokenId}
Content-Type:application/json
```

Body

```
 “planner”
```

Response (JSON)

```
HTTP/1.1 200 OK "true"
```

Company service

The Get Employee Companies returns all the user's companies.

Get Employee Companies

Request (JSON)

```
GET /Companies/EmployeesCompanies/?securityToken={tokenId}
Content-Type:application/json
```

Response (JSON)

```
HTTP/1.1 200 OK "true"

[{"CompanyId":"766c9732-e2d1-46d1-ae3e-a74c560bb8e6", "Name":"Company Name"},
...
{" CompanyId ":"aaaaa732-e2d1-46d1-ae3e-a74c560bb8e6", "Name":"Company Name "}]
```

Breakcode service

The Get Breakcodes returns all the company's breakcodes.

Get Breakcodes

Request (JSON)

```
GET /Breakcodes/?securityToken={tokenId}
Content-Type:application/json
```

Response (JSON)

```
HTTP/1.1 200 OK "true"

[{"BreakCodeId":"766c9732-e2d1-46d1-ae3e-a74c560bb8e6", "Code":"Q", "Name":"Q",
"Description":"BreakCode1", "Hours":"5", "MinimumHours":"5"}]},
...
{" BreakCodeId ":"aaaaa732-e2d1-46d1-ae3e-a74c560bb8e6"," Code ":"P",
"Name":"P", "Description":"BreakCode2", "Hours":"8", "MinimumHours":"8"}]
```

Shift Activity service

Get Shift Activities

This Get method returns a list of all the company's shift activities

Request (JSON)

```
GET /ShiftActivities/?securityToken={tokenId}
Content-Type:application/json
```

Response (JSON)

```
HTTP/1.1 200 OK "true"

[{"ShiftActivityId":"9b8b2254-875f-41a1-99c7-1ca963dcf762", "Code":" 1",
"Name":"Syg", "Description":"", "Enabled":"True", "OrderIndex":"2"},
...
{" ShiftActivityId ":"3e7131b4-d2e7-44ab-a4bf-5ecbcceff011", " Code ":"0",
"Name":"Hej", "Description":"", "Enabled":"True", "OrderIndex":"1"}]
```

Actual shift service

Actual shifts – arbitrary day

This resource returns a list of the actuals shifts for the day including employees that are on leave, signified by an associated activity. Available for all. If department is equal to “all”, shift from all rosters in all departments are returned. If the endTime is equal to 1/1/1900 it means that the endtime is empty at the moment.

Request (JSON)

```
GET /actualshifts/
/day/{date}/?securitytoken={sessionToken}(&departmentId={departmentId})

Content-Type: application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[{"BreakCode":"Q","Comment":"",
"DepartmentName":"Ny Bistro","EmployeeName":"",
"EndTime":"\/Date(1316700000000+0200)\/",
"StartTime":"\/Date(1316671200000+0200)\/"},
                                        ...
{"BreakCode":"Q","Comment":"",
"DepartmentName":"Ny Bistro","EmployeeName":"",
"EndTime":"\/Date(1316700000000+0200)\/",
"StartTime":"\/Date(1316671200000+0200)\/",
"ShiftActivityId":"376a579a-acc9-4020-b313-d2e826b60b14",
"ShiftId":"376a579a-acc9-4020-b313-d2e826b60b14"}]
```

Actual plan day status

This service returns a true or false if the actual plan day is closed or open.

If closed = true. If Open = false.

Request (XML)

```
GET
/actualshifts/ status/day/
{date}/?securitytoken={sessionToken}(&departmentId={departmentId})


Content-Type: application/json
```

Response (JSON)

```
HTTP/1.1 200 OK

{"Message":’true’,"Success":true}
```

Actual shifts – actual shifts by employee

This resource returns a list of the actual plan for the past 60 days for an employee selected.
If department is equal to “all”, actual shifts from all employees departments is returned. If Departmentid is left empty defaultdepartment will be chosen. Or you could specify a departmentid.
Takese an employeeid. If the endTime is equal to 1/1/1900 it means that the endtime is empty at the moment.

Request (JSON)

```
GET /actualshifts
/past/{date}/?securitytoken={sessionToken}(&departmentId={departmentId}&employe
eId={employeeId})

Content-Type: application/json
```

Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

[{"BreakCode":"Q","Comment":"",
"DepartmentName":"Ny Bistro","EmployeeName":"",
"EndTime":"\/Date(1316700000000+0200)\/",
"StartTime":"\/Date(1316671200000+0200)\/"},
                                        ...
{"BreakCode":"Q","Comment":"",
"DepartmentName":"Ny Bistro","EmployeeName":"",
"EndTime":"\/Date(1316700000000+0200)\/",
"StartTime":"\/Date(1316671200000+0200)\/"}]
```

Close Actual Day

Use this to close a day of your actual shift plan. Supply the day and the department. Is available for planner and administrator.

Request (JSON)

```
PUT /actualshifts/
close/day/{date}/?securitytoken={sessionToken}(&departmentId={departmentId})
}

{
  "Departmentid": "fb60311f-e88c-4a6a-9bc4-6ad09fd6a7b5",
  "Date": "\/Date(1355986847000+0100)\/"
}
```

Response (JSON)

```
HTTP/1.1 200 OK

{"Message":'true',"Success":true}
```

Open Actual Day

Use this to close a day of your actual shift plan. Supply the day and the department. Is available for administrator, and if the settings is enabled in the Tamigo Web application. It will also be possible for planners to open a day in the current wage period, to see if it is possible use the service "Can planner reopen" .

Request (JSON)

```
PUT /actualshifts/
open/day/{date}/?securitytoken={sessionToken}(&departmentId={departmentId})


{
  "Departmentid": "fb60311f-e88c-4a6a-9bc4-6ad09fd6a7b5",
  "Date": "\/Date(1355986847000+0100)\/"
}
```

Response (JSON)

```
HTTP/1.1 200 OK

{"Message":'true',"Success":true}
```

Can Planner Reopen Actual Day

This service returns if it is possible for planners to reopen closed actual plans, in the current wage period.

Request (JSON)

GET /actualshifts/reopen/day/{date}/?securitytoken={sessionToken}

```
{
  "Date": "\/Date(1355986847000+0100)\/"
}
```

Response (JSON)

HTTP/1.1 200 OK

```
{Success":true}
```

Copy authorized plan to actual plan – For a Day

This copies an authorized plan to an actual plan. This is only allowed for administrators or planners.

Request (JSON)

POST /actualshifts/
copy/day/{date}/?securitytoken={sessionToken}(&departmentId={departmentId})

```
{
  "Departmentid": "fb60311f-e88c-4a6a-9bc4-6ad09fd6a7b5",
  "Date": "\/Date(1355986847000+0100)\/"
}
```

Response (JSON)

HTTP/1.1 200 OK

```
{"Message":'',"Success":true}
```

Services for partner login

Tamigo offers special services for partners. They will be listed below. Only a partner login has access to these services.

Create new company

This service creates a new company, with one administrator. Fill out the different parameters to set culture, package and name, email etc. for the administrator.

Request (JSON)

POST / Companies/?securitytoken={sessionToken}

Body:

Company

```
{
  "Name": "This is the company name",
"ContactPerson": "This is the administrators name",
"Address": "This is the Address",
"City": "This is the city",
"Email": "This is the email for the admin login",
"Password": "for the login",
"Phone": "This is the phone of the administrator",
"zipCode": "This is the zipcode",
"Culture": "This is the culture of the company",
"PackageType": "package type is ranged from 1-5",
}
```

Response (JSON)

HTTP/1.1 200 OK

{Message:”Company Created!” Success":true}

Touch Services for smartphones

Touch services for the smartphones are used in order to Check in/out, similar as in the main Touch App.

TouchCheckIn

The *TouchCheckIn* method is allowing the user to check in from a smartphone device.

Request (JSON)

POST/Attendance/TouchCheckIn/?securityToken={tokenId}

Body:

Null

Response (JSON)

TouchCheckIn is returning an *AttendanceInformation* type of object, which contains a *CheckInTime* updated with the *dateTimeCheckIn*.


```
{
"CheckInTime": "12:37",
"CheckOutTime": "null",
"CurrentTime": "12:48:32",
"CurrentStatus": "AllowCheckOut",
"EnableShiftBreakCodePanel": "true",
"IsCheckedInElsewhere": "false",
}
```


TouchCheckOut

Similar to the *TouchCheckIn* method, the *TouchCheckOut* calls the *CheckOut* service and returns an *Attendance Information* object, updated with the CheckOutTime. If breaktime is empty insert null.

Request (JSON)

POST /Attendance/TouchCheckOut/{breakcodeid}/?securityToken={tokenId}

Body:

Null

Response (JSON)


```
{
"CheckInTime": "12:37",
"CheckOutTime": "15:45",
"CurrentTime": "15:45:00",
"CurrentStatus": "AllowCheckIn",
"EnableShiftBreakCodePanel": "false",
"IsCheckedInElsewhere": "false",
}
```

GetTouchStatus

It returns an *AttendanceInformation* type of object, updated with the latest information.

Request (JSON)

GET /Attendance/GetTouchStatus/?securityToken={tokenId}

Body:

Null

Response (JSON)

TouchCheckIn is returning an *AttendanceInformation* type of object, which contains a *CheckInTime* updated with the *dateTimeCheckIn*.


```
{
"CheckInTime": "12:37",
"CheckOutTime": "null",
"CurrentTime": "12:48:32",
"CurrentStatus": "AllowCheckOut",
"EnableShiftBreakCodePanel": "true",
"IsCheckedInElsewhere": "false",
}
```


GetBreakCodes

It returns a list of *AttendanceBreakCode* type of objects, with the breakcodes corresponding to a certain employee.

Breakcodeid Is optional, if sent the service will return a list with the breakcodeid as the selected one.

Request (JSON)

GET /Attendance/GetBreakCodes/?securityToken={tokenId}&breakcodeid={breakcodeid}

Body:

Null

Response (JSON)


```
[{"BreakCodeID":"1",
"Code":"12",
"Enabled":"false",
"Hours":"0.12",
"Name":"Q",
"Selected":"true"},
                                        ...
[{"BreakCodeID":"4",
"Code":"30",
"Enabled":"false",
"Hours":"1.5",
"Name":"R",
"Selected":"true"}]
```

UpdateBreakCode

This method is used in the applications when an user is selecting a certain breakcode. Its purpose is to change the .Selected property of the selected breakcode. The method returns a *ResponseMessage,* which has the *Success=true* in case the breakcode was successfully updated, and false otherwise.

Request (JSON)

POST /AttendanceService/UpdateBreakCode/{breakcodeid}/?securityToken={tokenId}

Body:

Null

Response (JSON)

{Message:"", Success:"true"}

GetTouchAccess

The *GetTouchAccess* method is used in order to decide whether a device may or may not have access to the Touch application.

In case the company does not have the option *AlwaysCheckIPOnTouchLogin* enabled, the device using the Touch application will always be able to use it. Otherwise, a set of checks on the IP address is being made: used when components are first initialized, in the smartphone versions.

Request (JSON)

GET/AttendanceService/GetTouchAccess/ securityToken={tokenId}

Body:

Null

Response (JSON)

{Message:"" Success:"false"}