Integration Documentation v1.5

TiDaeX™ Account Initial Setup

The first step is registering a company on TiDaeX™, and providing company details, primary location/depot details and primary user contact details (Must have a valid email address).

Once your company has been created, you will receive, via email, login credentials for your primary contact. Using these credentials, you should create a specific account in TiDaeX™ for your software to use to interact with the API.

User Authentication and security tokens

With your account set up you can call [TiDaeXAPI/TiDaeXUserLogin]. By providing the 'username' and 'password' fields, a successful call to this method will return a token, which is to be used for authentication when calling other methods on the API. Token's are unique per user account account and will become invalidated upon every subsequent login call, or after 12 hours has passed.

There is a call [TiDaeXAPI/ValidateSessionToken] which can be used to check whether a given token is still valid. If a token is invalid, a response of 'Unauthorised' will be returned.

Most, if not all API calls require providing your security token, encoded within is your unique user ID.

Unique Company ID

Some API calls will require you to provide the unqiue ID of the company your account is registered against. If you require this ID, make a call to [TiDaeXAPI/GetCompanyId_ForUserId] suppling your security token to obtain it.

Getting Vehicles, Fleets and Fitters

To obtain a single vehicle, fleet or fitter ID's there are 'GET' methods to see if the vehicle/fleet/fitter exists; [TiDaeXAPI/GetVehicleDetails_ForVRM], [TiDaeXFleetAPI/GetFleetDetails_ByFleetCode], [TiDaeXAPI/GetFullUserDetails]. If you want to fetch bulk lists of these entities, use the following methods instead; [TabletAPI/GetAllVehicles_ForUserId], [TiDaeXFleetAPI/LoadAllOurFleets], [TiDaeXAPI/LoadUserData].

Creating Vehicles

To create a vehicle, make a call to [TiDaeXAPI/CreateVehicle]. If the vehicle already exists [TiDaeXAPI/CreateVehicle] will return the existing vehicle's ID, else if the vehicle is new, then the ID of the newly added vehicle will be returned.

When creating a vehicle, the 'fleetNumber' field is an optional alternative to the 'vrm' field. Some customers have vehicles that have no registration attached and as such populate the 'fleetNumber' field with their own 'unique' identifier for that vehicle.

**Note: The vehicle's 'fleetId' needs to exist on the system before creating the vehicle, and you can only create vehicles against fleets that your company owns.*

Once you have a valid TiDaeX™ vehicle ID, follow one of the two paths for setting up the axle configuration.

1. Group all the axle data together and make 1 call to [TiDaeXAPI/ UpdateListOfAxleDetails].
2. Make a call to [TiDaeXAPI/ ClearAxleConfigs], then call [TiDaeXAPI/SubmitAxleDetails] for each axle on the vehicle.

It doesn't matter which path you choose, but there is a difference between them.

[TiDaeXAPI/ UpdateListOfAxleDetails]is 'self-contained' and takes all the axle configs at once, invalidating any pre-existing configs for the vehicle, and adds all the new ones.

[TiDaeXAPI/SubmitAxleDetails] simply adds an additional axle config to the current config. On its own, it will not overwrite any existing configs held for the vehicle (hence the initial call to [TiDaeXAPI/ ClearAxleConfigs] in my 2nd path example above.

The 'Client_AxleConfig' structure used for submission on the payload for [TiDaeXAPI/ UpdateListOfAxleDetails], is outlined below;

```
"axleConfigId", int (NOT required - Primary Key)
"axleType", string (REQUIRED - **AXLE TYPE STRING** - SEE BELOW)
"axleOrder", int (REQUIRED - Zero indexed – Front to Back - axle order)
"numTires", int (REQUIRED - Number of tyres/wheels on the axle)
"vehicleId", int (REQUIRED - TiDaeXVehicle ID)
"createdDate", datetime (REQUIRED – Date of creation)
"lastModifiedDate", datetime (NOT Required)
"current", bool (NOT Required – Automatically set to 'TRUE')
"createdBy", int (NOT Required – Automatically set to user Id of account making request)
"lastModifiedBy", int (NOT Required)
```

Creating Fleets

To create a fleet, make a call to [TiDaeXFleetAPI/SubmitFleetDetails], if the fleet already exists '-1' will be returned. If the fleet is new, [TiDaeXFleetAPI/SubmitFleetDetails] will return the ID of the newly added fleet.

Use these Vehicle/Fleet ID's to pass in to the respective [TiDaeXAPI/LaunchJob] payload fields.

Work Items

Work items represent the work to be done on a job (fitting/removing/inspecting tyres) or the goods/services supplied (MVC, PSI adjust etc..) and the work item type reflects this.

Work items are linked to a job via the unique Job id and, critically, are hierarchically linked to each other, in order to group work together for a given wheel position.

In addition to their type, work item types have a classification ID to distinguish goods from services and to denote particular types as 'root' types for a single wheel position. These root types (classification type 1 "Wheel Position") are limited to 1 per wheel position and must be the parent item that all other items on that position are related to.

Note: The exception to this is 'Remove'. A 'remove' type item can be either a Parent or Child item, but occur no more than **once per position.*

**Note; the 'WheelPosition' field must be populated with an acceptable value. The format of this value is (for example): A2. The 'A' represents the axle (labelled ascendingly alphabetically from front-to-back) and the '2' represents the wheel on the axle (labelled ascendingly numerically from left-to-right, capped at maxium of 8 wheels per axle).*

The WorkItem structure as documented on Swagger [TiDaeXAPI/LaunchJob]:

```
"WorkItemId", long – Server generated unique id
"Barcode", string – Free-text field
"CasingBarcode", string – Free-text field
"CasingDestinationId", long – Id field captured on device.
"Comments", string – Free-text field
"EANNumber", string – Free-text field
"HasBeenRegrooved", bool
"HasRFID", bool
"IsNew", bool
"IsParent", bool – Denotes work item is a parent item.
"IsRemould", bool
"JobId", long – Links work to a job, MUST be supplied.
"ParentWorkItemId", long – If item is a parent, MUST be set to 0, otherwise use id of
parent item, that this item is linked to.
"Pressure", float
"Quantity", int – Unless supply only, this should be set to 1.
"SerialNumber", string – Free-text field
"TorquedTo", float
"TreadDepth_Inner", float
"TreadDepth_Middle", float
"TreadDepth_Outer", float
"TyreBrandId", long – Server unique ID for Brand of tyre (if relevant)
"TyrePatternId", long– Server unique ID for Pattern of tyre (if relevant)
"TyreSizeId", long – Server unique ID for Size of tyre (if relevant)
"WheelPosition", string – MUST be set to a valid position value for the vehicle set on the
job, and be of format 'A1' or 'B3' etc…
"WorkItemTypeId", long – Unique ID of the work item type. There are currently 26 different
types, a list of which can be provided via a call on the API.
```

Work Item Types

As already mentioned, a list of the valid types for work items can be requested by calling [TiDaeXAPI/GetWorkItemTypes]. This provides the basic information contained in a type as well as the classification ID of the work type.

Launching a job

Launching a job into TiDaeX™ requires setting up a multipart-formdata POST to [TiDaeXAPI/LaunchJob]. Vehicles and Fleets and Fitters should be created, either in TiDaeX™ or via the API, prior to job launch.

The LaunchJob payload as documented on Swagger [TiDaeXAPI/LaunchJob]:

```
"token", string – Your security token
"jobId", long – Only provide to edit a previously launched job
"integratedCreatedDate", dateTime – Provide to define when job was created (overrides
default creation date, normally time when TiDaeX™ creates job)
"additionalDetails", string – Free-text field for additional job details
"customerCode", string – Free-text field
"driverName", string – Free-text field
"driverTelephone", string – Free-text field
"allocatedfitterId", long – If not provided job will not be sent to a tablet/device. DO NOT
provide for a referred job!
"allocatedLocationId", long – Can be provided for a referrd job to send to a specific
network depot. *Note: if no Fitter or Location ID is provided upon launch, the job will be
```

created on the system but not sent anywhere. The job will then have to be edited and re-launched.
"fleetId", long – The TiDaeX™ ID of the Fleet TiDaeX™ holds the Vehicle against.
"jobType", int – 1 = Breakdown, 2 = Service Work, 3 = Supply only
"location", string – Free-text field
"networkRecipient", long – The TiDaeX™ ID of the Company to refer the job to.
"scheduledDate", dateTime – The date for the job/work
"vehicleId", long – The TiDaeX™ ID of the Vehicle. *Note: if jobType is 'Supply Only', then no vehicleId is required.
"vehicleAvailableStart", dateTime (optional)
"vehicleAvailableEnd", dateTime (optional) – Optional date fields for defining a period during which the Vehicle is available. *Currently not in use, but recorded on job*
"orderNumber" string – Free-text field
"externalPayloadField_1" string – Free-text field for pass-through data. Not touched/altered by TiDaeX.
"integratedCreationSource" bool – Should be set to TRUE, alerts the system this is an externally created job
"integratedCreatedDate" dateTime (optional) – pass in a date to override the default 'job.CreatedDate'. If not set, 'job.CreatedDate' will be set to DateTime.Now, when the call is made.
"jobSource" int – 0 = Own Job, 1 = Network referral
"workItemList[]", Client_WorkItem[] – A workItem entry should be added to the FormData payload for each workItem to be present on the job.

If the [TiDaeXAPI/LaunchJob], is successful a 200 response will be returned with the newly created JobID, and a count of the workItems created;

200 response string Format: "JobId = ;0; WorkItemListCount = ;0;".

Accept Job Referral

To accept a job referral, once you have the relevant job id, make a call to [TiDaeXAPI/AcceptJobReferral]. Once accepted, a job will then need to be edited and re-launched to allow you to allocate it to one of your fitters. Use [TiDaeXAPI/GetJobDetails_ForJobId], to get the job's details and pass them back in to [TiDaeXAPI/LaunchJob] along with the jobId and the Fitter Id you have selected, in order to send it out to a device.

Reject Job Referral

To reject a job referral, once you have the relevant job id, make a call to [TiDaeXAPI/RejectJobReferral].

Get Referred/Available Job Details

Make a call to [TiDaeXAPI/GetAllJobsInProgress_ForClient], passing in just your token on the payload.

A successful response will return a collection of jobs that are currently 'in progress' for your company.

**Note:* *Jobs that are 'completed' but in 'awaiting validation' status will still be included in this call.*

**Note:* *If the token you have is for a user with the role 'Location Manger' then the returned list will be constrained down to just those jobs pertaining to the user's managed locations.*

The response payload is as follows:

"companyJobId_Display", long – Your company specific 'seeded' job id
"JobId", long – The TiDaeX™ 'TRefXID' unique job Id
"CreatedDate", dateTime – Created date of the job
"createdDate_Display", string – A formated version of the job created date

"vehicleVRM", string – The Vehicle Registration Mark (VRM) for the job
"fleetName", string – The name of the Fleet on the job
"referralChildCompanyName", string – The name of the company this job was referred to
"ReferralChildCompanyId", long – The ID of the company this job was referred to
"referralParentCompanyName", string – The name of the company that referred this job
"ReferralParentCompanyId", long – The ID of the company that referred this job
"fitterId", long – The ID of the allocated fitter for the job
"ReferralChildJobId", long – The ID of the 'cloned' child job, when referring a job out.
"jobStatusId", int – The ID denoting which status the job is in
"jobSource", int – 0 = Own Job, 1 = Network referral
"allocatedLocationId", long – The ID of the allocated location/depot for the job
"depotName", string – The name of the location/depot the job is allocated to
"companyId", long – The ID of the company who created the job

Get Status Updates

Make a call to [TiDaeXAPI/GetAllUnCollectedStatusUpdates], passing in just your token on the payload.

A successful response will return a collection of status updates that have not been flagged as 'collected' for every job for your company in the system.

The response payload is as follows:

"statusId", long – Unique id for a status update
"jobId", long – Your company specific 'seeded' job id (companyJobId)
"m_event", string – Status Type description e.g. "Delivered to Device"
"comment", string – Additional fitter comments
"fitter", string – Name of fitter that generated status or if N/A: "No Fitter Allocated."
"fitterId ", long – Id of fitter that generated status or if N/A: 0
"ExternalPayloadField_1", string – A copy of the 'pass-through' data included on the TiDaeXAPI/LaunchJob] payload when it was launched.

After a status update has been processed in your software, it should be flagged as 'collected' by calling [TiDaeXAPI/SetStatusUpdateAsCollected], passing in your token and the StatusId of the update. This will prevent this update from being returned on your next [TiDaeXAPI/GetAllUnCollectedStatusUpdates] call.

Get Completed Jobs

The process for getting completed Job data is the same as for Status Updates. Make a call to [TiDaeXAPI/GetAllUnCollectedCompletedJobs], passing in just your token on the payload.

A successful response will return a collection of jobs that are currently 'completed' for your company.

The response payload is as follows:

"jobId", long – Unique TrefXId of the job
"createdBy", long – User ID of who created the job
"createdDate", dateTime – Created date of the job
"createdDate_Display", string – String formatteed Created date of the job
"lastModifiedBy", long – User ID of last User to modify the job details
"lastModifiedDate", dateTime – Date of the last modification of the job details
"lastModifiedDate_Display", string – String formatted Date of the last modification of the job details
"fleetId", long – Id of the Fleet on the job
"fleetName", string – Name of the Fleet on the job
"companyId", long – Id of the company who created the job
"companyName", string – Name of the company who created the job
"location", string – Free text field of job location

"vehicleId", long – Id of the vehicle on the job
"vehicleVRM", string – Vehicle Registration Mark of the vehicle on the job
"fitterId", long – Id of the fitter who completed the job
"fitterName", string – Name of the fitter who completed the job
"depotId", long – Id of the home location of the Fitter who completed the job
"depotName", string – Name of the home location of the Fitter who completed the job
"jobSource", int – 0 = Own Job, 1 = Referred job
"jobStatusId", int – Id of the current status of the job
"jobStatus_Display", string – Description of the current status of the job
"companyJobId", long – 'Seeded' Id of the job
"companyJobId_Display", string – String formatted 'Seeded' Id of the job
"completedDate", dateTime – Completed date of the job
"completedDate_Display", string – String formatted Completed date of the job
"validatedDate", dateTime – Validated date of the job
"validatedDate_Display", string - String formatted date
"processedDate", dateTime – NOT IN USE
"processedDate_Display", string – NOT IN USE
"cancelledDate", dateTime – Cancelled date of the job
"cancelledDate_Display", string - String formatted date
"startedDate", dateTime –Date the job was started on the mobile device
"startedDate_Display", string - String formatted date
"signatureDate", dateTime – Date the job was signed off
"signatureDate_Display", string - String formatted date
"jobTypeId ", int - 1 = Breakdown, 2 = Service Work, 3 = Supply only
"jobType_Display", string – Description of the JobType
"isTrailer", bool
"customerCode", string – Free text field
"driverName", string
"driverTelephone", string
"recipient", int – Company ID of recipient of referred job
"recipientName", string – Name of the company/recipient of referred job
"scheduledDate", dateTime – Initial scheduled date of the job
"scheduledDate_Display", string – String formatted date
"additionalDetails", string – Free text field
"workItemList", List<Client_WorkItem> – **See Below [1]**
"photoList", string – **See Below [2]**
"completed", bool – Whether a job has been completed
"paused", bool – Whether a job is currently paused
"processed", bool – NOT IN USE
"awaitingValidation", bool – Whether a job is currently awaiting validation
"validated", bool – Whether a job has been validated
"cancelled", bool – Whether a job is cancelled
"referralAccepted", bool – Whether a referred job has been accepted
"referralRejected", bool – Whether a referred job has been rejected
"sentDate", dateTime – Date when PDF was emailed out
"sentDate_Display", string – String formatted date
"axleConfigChanged", bool – Whether the vehicle axle config was altered on this job
"probedId", string – Id of the bluetooth probe used (if any)
"torqueWrenchNumber", string
"torqueTagNumber", string
"torqueTarget", int – Torque setting nuts were tightened to
"recordedDistanceCovered", int – Recorded mileage
"mileageNotAvailable", bool – Whether the mileage could not be read/recorded
"signatureImage_Driver", string – Signature Image
"signatureImage_Fitter", string – Signature Image
"signature_DriverName", string – Name of the Driver who signed the job off
"signature_FitterName", string – Name of the Fitter who signed the job off
"signature_Timestamp", dateTime – Date when job was signed off
"signature_CustomerRole", string – If Driver was not available, the role of the authorised person who signed the job off
"signature_DriverNotAvailable", bool – Whether the driver was not available to sign the job off
"wheelRemoved", bool – Indicates if a wheel was removed during the job
"wheelRemovedReason", string – NOT IN USE
"certifiedGoods", bool – Wether the driver has certified that the stated goods were actually supplied

"device_UUID", string – The UUID of the device used to complete the job
"device_Platform", string – The operating system of the device used to complete the job
"device_Model", string – The model number of the device used to complete the job
"numMessages", int – NOT IN USE
"vehicleAvailableStart", dateTime – Start of vehicle available period
"vehicleAvailableEnd", dateTime – End of vehicle available period
"vehicleAvailableStart_Display", string – String formatted date
"vehicleAvailableEnd_Display", string – String formatted date
"referralParentCompanyId", long – Company ID of the company who referred the job
"referralParentCompanyName", string – Name of the company who referred the job
"referralChildCompanyId", long – Company ID of the company who the job was referred to
"referralChildCompanyName", string – Name of the company the job was referred to
"referralParentJobId", long – INTERNAL USE ONLY – Id of the parent the referred job was cloned from
"referralChildJobId", long – INTERNAL USE ONLY – Id of the child clone of this job
"rates_AdminFee", bool – Does an 'Admin Fee' apply to this job
"rates_DayRate ", bool – Does an 'Day Rate' apply to this job
"rates_NightRate", bool – Does an 'Night Rate' apply to this job
"rates_SiteCharge", bool – Does an 'Site Charge' apply to this job
"fittersNotes", string – Any fitter notes on the job
"thirdPartyReference", string – Free text field
"orderNumber", string – Free text field
"ExternalPayloadField_1", string – Pass-through field for external data
"FollowUpWorkRequired", bool – Is any further work required in the future as a result of work carried out on this job
"CollectedByIntegratedParty", bool – Has this job been flagged as 'collect' by an external integrated software system
"IntegratedCreationSource", bool – Denotes a job originated from outside of TiDaeX.
"CamAxleConfig", string – NOT IN USE
"JobHasPhotos", bool – Denotes a job has photos associated with it

[1] workItemList – List<Client_WorkItem>

The 'Client_WorkItem' has the following structure:

"WorkItemId", long – Unique Id of the work item
"Barcode", string – Free text field
"CasingBarcode", string – Free text field
"CasingDestinationId", long – 1=Bank,2=COPStock,3=Dealer,4=Driver,5=FleetLocation, 6=MajorRepair,7=Manufacturer,8=Scrap
"Comments", string – Free text field
"CompletedDate", dateTime – FOR INTERNAL DEVICE USE ONLY
"EANNumber", string – Free text field
"HasBeenRegrooved", bool – Whether a tyre has been re-grooved
"HasRFID", bool – Whether a tyre has RFID/NFC
"IsNew", bool – Whether a tyre is new
"IsRemould", bool – Whether a tyre is remoulded
"JobId", long – Unique TRefXID of the job
"ParentWorkItemId", long – Unique Id of the work item this item is linked to
"Pressure", decimal
"Quantity", int – Number of tyres (only relevant for supply only)
"SerialNumber", string – Free text field
"TorquedTo", decimal – Nm value of torque
"WheelRemovalReason", string – NOT IN USE
"TyreRemovalReason", string – Free text field
"AdditionalTyreRemovalReasons", string – Free text field
"OtherTyreRemovalReasons", string – Free text field
"WheelRemoved", bool – Whether the wheel was removed
"NoTyreRemoved", bool – Whether no tyre was removed
"Tablet_WorkItemId", long – FOR INTERNAL USE ONLY
"Tablet_ParentWorkItemId", long – FOR INTERNAL USE ONLY
"Tablet_CreatedOnTablet", bool – FOR INTERNAL USE ONLY
"TreadDepth_Inner", decimal – Tread depth in mm
"TreadDepth_Middle", decimal – Tread depth in mm
"TreadDepth_Outer", decimal – Tread depth in mm

"TyreBrandId", long – TiDaeX™ Unique Id
"TyrePatternId", long – TiDaeX™ Unique Id
"TyreSizeId", long – TiDaeX™ Unique Id
"TyreBrand_Display", string – Brand Name
"TyrePattern_Display", string – Pattern Name
"TyreSize_Display", string – Size Name
"WheelPostion", string – The Wheel position on the vehicle e.g. "A1"
"SwapFromPosition", string – The position this tyre was swapped from
"WorkItemType_Display", string – The name of the work item type
"WorkItemTypeId", long – **See Below [3]**
"FollowUpWorkRequired", bool – Whether any further work is required

[2] photoList – The photoList will be null if there are no photos on the job. If there are photos, it will be an empty list. This is deliberate, as there is a max JSON payload length and including the photos on all jobs would add a lot of data to the payload. To obtain any job photos, make a call to [TiDaeXAPI/GetAllPhotos_ForJobId], passing in your token and a job id.

[3] Work Item Type – A work item can have any of the following types, that can be obtained by calling [TiDaeXAPI/GetWorkItemTypes], passing in your token.

WorkTypeId	Description	ClassificationId
1	Fit	1
2	Remove	1
3	Inspect	1
4	Fit COP	1
5	Re-groove	3
6	Punc. Repair	3
7	Tread Repair	3
8	Major Repair	3
9	Balance	3
10	Re-Torque	3
11	PSI Adjusted	3
12	30min Re-Torque	3
13	Turn on rim	3
14	Twinning	3
15	Valve	2
16	Extension	2
17	Ext. Bracket	2
18	HPVC/MVC	2
19	Tube	2
20	Flap	2
23	Wheel	2
24	Supply	1
25	Wheel Swap	1
26	Work	1
27	Quick Inspect	1

[4] Axle Types

Making a call to [TiDaeXAPI/GetAxleTypes] provides the following results;

```
    "AxleTypeId": 1,
    "AxleTypeName": "axle_steer"

    "AxleTypeId": 2,
    "AxleTypeName": "axle_drive"

    "AxleTypeId": 3,
    "AxleTypeName": "axle_free_rolling"

    "AxleTypeId": 4,
    "AxleTypeName": "axle_steer_drive"

    "AxleTypeId": 5,
    "AxleTypeName": "axle_trailer"

    "AxleTypeId": 6,
    "AxleTypeName": "axle_trailer_lift"

    "AxleTypeId": 7,
    "AxleTypeName": "axle_trailer_steer"

    "AxleTypeId": 8,
    "AxleTypeName": "axle_steer_lift"

    "AxleTypeId": 9,
    "AxleTypeName": "axle_spare"
```


Provide the 'AxleTypeName' in the 'axleType' on the [TiDaeXAPI/ UpdateListOfAxleDetails] or [TiDaeXAPI/SubmitAxleDetails] payload.