Autotask Web Services API

API version 1.5.2

Created: 4/16/2010

Last updated on 10/8/2014

Table of Contents

API Revision History

Version	Date	Revision
1.1	06/21/2007	Initial release.
1.2	07/015/2007	For Ticket entity: Ticket.AllocationCodeID and Ticket.AssignedResourceRoleID have become reference fields. Entities added: AllocationCode, Appointment, BillingItem, Invoice, Phase, Project, Role, ServiceCall, Service CallTask, ServiceCallTaskResource, ServiceCallTicket, ServiceCallTicketResource, Task, and TimeEntry
	08/16/2007	Updated base URL information in Getting Started
	08/06/2008	For Installed Product entity – changed Entity Description table to indicate that Can Create and Can Update are now enabled; added the following fields: ContractID, CreateDate, DailyCost, HourlyCost, MonthlyCost, Notes, NumberOfUsers, PerUseCost, ServiceBundleID, ServiceID, SetupFee
	11/20/2008	Add Product and Product Vendor entities.
	11/20/2008	Add SubType field to Billing Item entity.
1.3	11/20/2008	Change data type for id fields for all entities from "integer" to "long".
	01/20/09	Clarified definition of Read Only or Required attributes and updated those attributes for following Entity fields (see About ReadOnly and Required Attributes on page 4): AccountNote: LastModifiedDate, Required - No; CompletedDateTime, ReadOnly - YesPhase: LastActivityDateTime, Required – No; PhaseNumber, Required – No; Sheduled, Required - NoTask: TaskNumber, Required - NoTicket: CreateDate, Required – No; LastActivityDate, Required – No; TicketNumber, Required – No; Ticket Note: CreatorResourceID, Required – No; LastActivityDate, Required - No Installed Product entity: Updated out of order column content Ticket entity - added the following SLA related fields:FirstResponseDueDateTime, ResolutionPlanDateTime, ResolutionPlanDueDateTime, ResolvedDateTime, ResolvedDueDateTime, ServiceLevelAgreementHasBeenMet, ServiceLevelAgreementID Contract entity – added the following SLA related field:ServiceLevelAgreementID
	02/23/2009	Update Web Services URL shown in Getting Started section as per new URLs provided by IT Services.
	02/27/2009	Corrected error reported in QueryXML example, notequals changed to notequal.
	03/17/2009	Contract entity – added the following fields: IsDefaultContract OverageBillingRate SetupFee Contract entity – Updated the following field: TimeReportingRequiresStartStopTime is now Picklist = YesI nstalled Product entity – added the following fields:VendorID Contact ID Type

Version	Date	Revision
	6/09/2009	TimeEntry entity: added following fields DateWorked Type
1.4	3/17/2009	Added the following 12 Contract related entities: ContractBlock, ContractFactor, ContractMilestone, ContractNote, ContractRate, ContractRetainer, ContractService, ContractServiceAdjustment, ContractServiceBundle, ContractServiceBundleAdjustment, ContractServiceBundleUnit, ContractServiceUnitAdded Opportunity Entity Added Quote Entity and following 5 Quote related entities: QuoteItem, QuoteLocation, Service, ServiceBundle, ShippingType Added AccountLocation entity In InstalledProduct Entity, updated the following fields: ServiceID, ServiceBundleID: No longer Picklist, now Reference Service and ServiceBundle. Add getZoneInfo() API Call
	4/07/2009	Added the following 7 Inventory related entities: InventoryItem, InventoryItemSerialNumber, InventoryLocation, InventoryTransfer, PurchaseOrder, PurchaseOrderItem, PurchaseOrderReceive
	4/14/2009	Updated InventoryTransfer entity: Removed InventoryItemID, added FromLocationID and ProductID. Updated PurchaseOrder entity: Terms changed to PaymentTerm

Version	Date	Revision
	6/09/2009	TimeEntry entity - Now allows Create and Update Following fields added: DateWorked, Type Following field removed: Date NOTE: Web Services API TimeEntry does not support the interface Bill Immediately feature. All time entries through Web Services API must be approved and posted. Service entity and ServiceBundle entity Following field added: UnitCost ContractServiceAdjustment and ContractServiceBundleAdjustment entities Following field added: AdjustedUnitCost Addition of a Service (or ServiceBundle) to a recurring service contract no longer requires the ContractService (or ContractServiceBundle) entity. If the API cannot find the correct ContractService (or ContractServiceBundle) entity, the API will create it. ContractRetainer entity Following field added: paymentID Field label changed:Contract Object ID is now ContractID DatePaid field was removed from the Autotask interface but remains in the ContractRetainer entity for version 1.4. It will be removed in a future re-versioned release. ContractServiceUnit Following fields added: Cost VendorAccountID ContractServiceBundleUnit Following field added: Cost NOTE: ContractServiceBundleUnit cannot have a Vendor association. QuoteItem When QuoteItem.Type = 11 (Service), UnitCost defaults to the unit cost specified for that service in the Admin module. When QuoteItem.Type = 12 (Service Bundle), UnitCost defaults to the sum of the unit costs for all services within that bundle. Installed Product entity Data corrected for one field: Type is no longer Required Quote entity Data corrected for one field: id field is Read Only QuoteItem entity Data corrected for one field: id field is Read Only QuoteLocation entity Multiple Resources on a Ticket is currently not supported in Web Services for Create or Update

Version	Date	Revision
		Although the Autotask interface now allows multiple resources on a ticket, the current versions of Web Services do not. Web Services is, however, aware of and will check existing multiple resources on a ticket and will not allow any resource to be assigned as primary resource if that resource is already a secondary resource. Added License Agreement to document Added Use Types table to AllocationCode entity information
	7/01/2009	Contract entity: OverageBillingRate field no longer required for block hour type contracts. Installed Product entity: Following field added -InstalledByID Type field no longer required ContractBlock entity: Hours field will not accept negative values HourlyRate field will not accept negative values
	7/21/2009	Invoice entity Following fields added: IsVoided, VoidedDate, VoidedByResourceID
	9/10/2009	TimeEntry entityFollowing fields added: CreatorResourceID, LastModifiedDateTime (date and time last modified), and LastModifiedResourceID TimeEntry now respects the Autotask Proxy Time Entry workflow policy, allowing TimeEntry Create or Update for Timesheet Approvers and Administrators as per the workflow policy setting. Project entity Following fields added: Department, LineofBusiness
	11/17/2009	ProductVendor entity Noted behavior that on Create, if no default vendor exists, newly created ProductVendor will be set as default. TimeEntry entity Added an additional condition: Time Entries cannot be created for Projects with Project Type of Archived (1), Template (3), Baseline (8).
	12/09/2009	Document change only. No change to API. Added table containing five XML special characters and entity references required to escape them.
	01/19/2010	BillingItem entity Added new field, TaxDollars, for retainer tax where applicable ContractRetainer AmountApproved field will include retainer tax that has been deducted, where applicable. AccountLocation entity Added note indicating that Web Services cannot respect UI condition that applies Site Configuration UDF requirement to Customer type accounts only

Version	Date	Revision
	02/09/2010	Ticket entityAdded new field, Resolution Added condition indicating that for Contact ID field, Contact. AccountID must = Account or Parent Account associated with ticket. AccountLocation entity Removed note re: Site Configuration UDFs that was added 1/19/2010 and added updated information for Site Configuration UDFs in Web Services.
	04/12/2010	Document only- No API change: Updated text in getting started NOTE: to reflect earlier addition of limited release URL to access WSDL and Web Services.
1.5	04/16/2010	TimeEntry entity Added the following fields: AllocationCodeID, ContractID, NonBillable, ShowOnInvoice Opportunity entity Added the following fields: Rating, TotalAmountMonths Quote entity Added the following fields: CalculateTaxSeparately, GroupByProductCategory, ShowEachTaxInGroup
	05/11/2010	Added note re: SOAP header
	08/25/2010	Entity added: ClientPortalUser
	09/07/2010	API call added: getThresholdAndUsageInfo () API request threshold information added
	09/021/2010	Doc update only: updated description of API call threshold.
	10/22/2010	Document change: added text to TimeEntry entity Behaviors and query() API call description to indicate that time entries and queries are in Eastern Standard Time (EST).
	11/08/2010	Documentation correction only. Correct myService URL for getZoneInfo() sample code
	11/10/2010	Updated query() api call to indicate that querying on protected UDFs is case sensitive. Added conditional requirement to Ticket, Task, and TimeEntry entities to indicate that attempt to create resource + role combinations where the Role is inactive will trigger an error message
	11/16/2010	Documentation update only: Added note to indicate that AllocationCodeID on ticket may be required if workflow policy to require Work Type on a ticket is enabled.
	12/01/2010	Documentation update only: Ticket entity – corrected CreateDate data type, changing datetime to date.

Version	Date	Revision
	01/06/2011	Project entity CompanyOwnerResourceID field is no longer required. Resource entity Added the following field: DefaultServiceDeskRoleID Documentation update only: Added character length for all fields with String datatype.
	02/08/2011	getZoneInfo() API call Updated to return a value for the field Database Type to indicate whether the database is type Pro or Go!. Document changes: section on User-defined Fields moved - now follows entities, before API: calls. Minor editing, no content changes. Added text to indicate entities that are not available in Autotask Go! TimeEntry entity: added text to indicate that when querying, in order to return only tickets or only tasks, the query must specify the correct Type value.
	02/23/2011	Product entity: PeriodType field is no longer required. For API Attachments, added one new entity and three new API calls: AttachmentInfo entityQuery only. See entity description. GetAttachment(), DeleteAttachment() and CreateAttachment() API calls. See API call descriptions.
	03/02/2011	Installed Product entity: SerialNumber (string) - character limit changed to 100
	05/04/2011	Added base URL information for London Data Center (formerly Global 1)
	07/07/2011	Added base URL information for North America zone change to America East and America West.
	08/09/2011	Added 2 new entities: Expense Report Expense Item TimeEntry entity LastModifiedDateTime data type changed from Integer to DateTime QuoteItem entity QuoteItem entities can now be deleted, that is CanDelete now = True Added Pre-release zone information to "Getting Started"
	09/21/2011	Note added to TicketNote entity re: excluding workflow notes from query API calls - getThresholdAndUsageInfo() call Added sample code delete() call added (currently for QuoteItem only)

Version	Date	Revision
	10/11/2011	BillingItem, Contract, Ticket, Task, and Project Entities: New field added - PurchaseOrderNumber
	10/23/2011	Contact entityAccountID field is now read only (to prevent possible data corruption)
	11/21/2011	Add AU Zone, ww6, to list of URLs Delete() API call Added sample code Installed Product entity Active field datatype changed from short to boolean, per change to AT Database
	02/28/2012	ExpenseReport entity: New field added - ApprovedDate Contract entity: New field added: OpportunityID TaskNote entity added ProjectNote entity added Added Webservices URLs for localization: Germany and China
	03/06/2012	Account and Contact entities: Country field character limit increased to 100
	05/15/2012	ProjectNote entity Correction - Announce field added Added Spanish Zone
	05/30/2012	Ticket entity Allow update on ticket with inactive attribute value if that value is not being changed; cannot assign new inactive value BillingItem entity New field added: ExpenseItemID

Version	Date	Revision
	06/05/2012	Invoice entity Following fields added: PaidDate, TotalTaxValue Contact entity State field increased to 40 characters Account entity Added the following fields: Active field (Account Active), ClientPortalActive, TaskFireActive State field increased to 40 characters Resource entity Enabled update() when logged in user has System Administrator level access. Added following fields: DateFormat, TimeFormat, Password, HomePhone, PayrollType, TravelAvailabilityPct InstalledProduct entity Added the following field: InstalledByContactID InstalledProduct update no longer requires that the resource associated with InstalledByID have Status = Active.
	07/25/2012	Invoice entity - Documentation **Correction** PaidDate field is NOT read only
	10/03/2012	Added two new entities: SalesOrder BillingItemApprovalLevel Added fields to the following entities: To the InventoryItem entity: Reserved Picked To the InventoryLocation entity: IsDefault ResourceID To the Opportunity entity: CloseDate SalesOrderID To the Product entity: DoesNotRequireProcurement To the QuoteItem entity: AverageCost HighestCost To the Role entity: SystemRole To the TimeEntry entity: BillingApprovalDateTime, BillingApprovalLevelMostRecent, BillingApprovalResourceID
	10/30/2012	To the PurchaseOrderItem entity: Added SalesOrder field

Version	Date	Revision
	01/09/2012	To the AttachmentInfo entity: new field, ContentType, added GetAttachment() Added note re: value returned is no longer a full path for Autotask attachments. Currently returns only filename and extension. Client Portal attachments are not affected. Code sample was updated. To the Resource entity Added requirements for Password field. To the ClientPortalResource entity Added additional options for Password field requirements.
	01/23/2013	Added the following entities: AccountToDo ActionType ContractCost ProjectCost TicketCost To Product entity Added Behavior: DoesNotRequireProcurement field defaults to False To the BillingItem entity: The following fields were added: ContractCostID, ProjectCostID, TicketCostID **Service** entity updated to allow create() and update()
	02/05/2013	Added the following entities: UserDefinedFieldDefinition UserDefinedFieldListItem Document only Correction: PurchaseOrderReceive entity does not accept update() Document only Correction: For Service entity: Name, PeriodType, and UnitPrice are required. Document only Correction: QuoteItem.LineDiscount is Required. When no LineDiscount value is provided, LineDiscount must = 0. Clarification added: AccountLocation allows update() only when UDFs that can be updated are added to the entity. Only those UDFs can be updated.

Version	Date	Revision
	05/07/2013	**Added the following entities:** Tax TaxCategory TaxRegion **The following entities were modified as noted:** Account entity Fields added - AdditionalAddressInformation, TaxExempt, TaxID, TaxRegion Changes to Country field as described under Account entity Conditions and Requirements. AllocationCode entity Field added - TaxCategoryID The Taxable field is no longer used. Values passed in to this field will be ignored and queries will return the empty value " ". Taxable status is now determined by TaxRegion and TaxCategory. BillingItem entity Fields added - LineItemID, MilestoneID, ServiceID, ServiceBundleID, VendorID Contact entity Field added - ExternalID Changes to Country field as described under Contactentity Conditions and Requirements. Invoice entity Fields added - BatchID, DueDate, TaxRegionName The TaxGroup field is no longer queryable. Project entity ExtProjectType is not queryable and no longer accepts values on create() and update(). Type picklist includes only Internal, Client, Proposal, Template. It no longer includes Archived, Business Objective, Inactivated. Existing projects of the retired types have been updated as described in the Project entity description. PurchaseOrder and Quote entities The TaxGroup field now passes TaxRegionID; when not provided, no taxes are applied. QuoteItem entity Fields added - TotalEffectiveTax, TaxCategoryID IsTaxable is now determined by QuoteItem.TaxCategoryID and TaxGroup (Tax Region ID) for the associated Quote entity, and TotalEffectiveTax value. Resource entity Field added: InternalCost SalesOrder entity Changes to ShipToCountry and BillToCounty fields as described under SalesOrder entity Conditions and Requirements. Notes added to SalesOrder entity description about change to ShipToCountry and BillToCounty fields. Task entity Fields added - IsVisibleInClientPortal, CanClientPortalUserCompleteTask Priority is no longer required. Will default to 0.

Version	Date	Revision
	05/16/2013	QuoteItem entity TaxCategoryID now accepts the value passed in. If no value is provided then ProductID, CostID, and ExpenseID will pull the value from the associated Alloc-ationCode.
	06/20/2013	getZoneInfo() now returns CI number (as integer) for database being accessed
	06/24/2013	AllocationCode entity: Product allocation codes (type 7) have been moved to type 4. Queries on type 7 allocation codes will be re-directed to type 4. Existing integrations can continue to use type 7 codes, now mapped to type 4. Product entity: Note added re: changes to Product allocation code (type 7).
	07/17/2013	Added the following entity: ContractTicketPurchase
	07/25/2013	NOTE ADDED: Project entity RE: change in behavior for Project.StartDate with new project schedule design DOCUMENTATION CORRECTIONS Corrected the content of the tables for the following entities as indicated: ContractService - Added InvoiceDescription ContractServiceBundle - Added InvoiceDescription ExpenseItem - Added PurchaseOrderNumber InstalledProduct - Added ParentInstalledProductID Quote - Added ShowEachTaxInGroup Resource - Added AccountingReferenceID ServiceBundle - Added InvoiceDescription
	07/26/2013	Updated Ticket entity to reflect changes that occurred with the release of Problem Man-agement in Autotask (Q1 Projects release). To Ticket entity added the following fields: ProblemTicketID TicketType
	07/31/2013	Resource entity Suffix field changed to picklist Corrected update for 07/25/2013 (transposition of project/product)
	08/07/2013	Task entity and Phase entity: Condition added: ProductID cannot be changed.
	08/27/2013	Account entity: AccountNumber field updated to accept 50 characters

Version	Date	Revision
	09/17/2013	Account entity – fields added: BillToAdditionalAddressInformation BillToAddress1 BillToAddress2 BillToAddressToUse BillToAttention BillToCity BillToCountryID BillToState BillToZipCode CountryID InvoiceMethod InvoiceNonContractItemsToParentAccount Contact entity – field added: CountryID Sales Order entity – fields added: BillToCountry ID ShipToCountryID ClientPortalUser entity - note added re: how to determine Taskfire Users by Security Level Added the following **entities**: AdditionalInvoiceFieldValue Country InvoiceTemplate PaymentTerm
	09/24/2013	Added Best Practices (document update only)
	10/8/2013	DOCUMENT ONLY: Added item under Best Practice
	10/22/2013	The TicketNote entity must respect the edit ticket note permissions implemented with Autotask release 2013.2 (October 22,). Permissions are assigned to the user's security level. Note added to TicketNote entity topic.
	10/22/2013	Entity added: InternalLocation
	11/05/2013	Resource entity: on query() or update(), if the InternalCost date range does not include the current date, the Internal Cost is not Active and Resource.InternalCost is set to 0. Documentation update only: update() call - added information to clarify that, on update, if no value is provided for fields that are not read only, the fields will be cleared. Best Practices - added "On update() Provide Data for All Non-Read Only Fields" Getting Started - added information to clarify need to update Sample Code with zone specific information
	12/18/2013	For query() XML, an error is generated when querying any entity if the following requirements are not met: Any field tag must contain an expression tag An expression tag must contain an op attribute

Version	Date	Revision
	01/07/2014	getEntityInfo(): The API entities now respect Autotask custom security settings for the logged in user. Use getEntityInfo() to retrieve information about the logged in user's access. getFieldInfo() now returns, for Picklist values IsActive IsSystem. User-defined Fields required in the UI are not required by the API. Contract entity, field added: SetupFeeContractCode ContractCost entity, fields added: ContractServiceBundleID ContractServiceID ProjectCost entity, fields added: ContractServiceBundleID ContractServiceID ContractServiceAdjustment entity, field added: AdjustedUnitCost Service entity, field added: ServiceLevelAgreementID ServiceBundle entity, field added: ServiceLevelAgreementID ShippingType entity, field added: AllocationCodeID Task entity, field added: RemainingHours Task entity, the following fields are required only when task has assigned resources AllocationCodeID DepartmentID Ticket entity, fields added: ChangeApprovalBoard ChangeApprovalType ChangeApprovalStatus ChangeInfoField1 ChangeInfoField2 ChangeInfoField3 ChangeInfoField4 ChangeInfoField5 ContractServiceBundleID ContractServiceID LastCustomerNotificationDateTime LastCustomerVisibleActivityDateTime TicketCost entity, fields added: ContractServiceBundleID ContractServiceID

Version	Date	Revision
		TimeEntry entity, fields added: ContractServiceBundleID ContractServiceID The following entities can now be created and updated: ServiceCallTask ServiceCallTaskResource ServiceCallTicket ServiceCallTicketResource The following entities were added: AccountTeam ChangeRequestLink Department TicketChangeRequestApproval The picklist flag has been removed from the following Ticket entity fields: AccountID, AllocationCodeID, AssignResourceRoleID, ContactID, ContractID, InstalledProductID
	01/09/2014	Document correction only: for getEntityInfo(), corrected labels for new fields returned. For example, "CanICreate" updated to "UserAccessForCreate"
	02/26/2014	BillingItem entity field added: LineItemFullDescription ContractRetainer entity: Condition added IsPaid value cannot be updated if ContractRetainer is associated to a ContractCost.
	03/12/2014	New entities added: TaskSecondaryResource TicketSecondaryResource TaskPredecessor ResourceRole Ticket entity: On update(), Priority and InstalledProduct fields are validated only when the field is changed. Validation will fail if the field is changed to an inactive value. Ticket entity: On update(), the ContactID field is validated only when the field is changed. Validation will fail if the field is changed to an inactive value.
	03/26/2014	Implemented a usage based latency tied to the threshold limit on API requests. Refer to "Usage Based Latency" on page 40.
	04/09/2014	No API changes. Added information to the Getting Started chapter of the documentation regarding differences between User Interface terminology and the API entity and field names. "Autotask UI and Web Services Terminology Differences" on page 41

Version	Date	Revision
	04/23/2014	New entity added: ServiceBundleService (allows developers to query for Services assigned to a Service Bundle and add and remove Service Bundle Services through the API) **AttachmentInfo** entity field added (to address issues with deletion of, and missing ParentID for, Opportunity attachments through the API): OpportunityID When AttachmentInfo.Type = Opportunity, then AttachmentInfo.OpportunityID is required and ParentID is not required. Documentation update only: Added information about the CountryID field under Conditions and Requirements for Account and Contact entities.
	05/07/2014	**AttachmentInfo** entity change in requirements. With the May 7 maintenance release, Opportunity type attachments must specify one of the following combinations: a Type ID and ParentID, or TypeID and OpportunityID, or TypeID and both ParentID and OpportunityID. OpportunityID value must be provided in order to query by OpportunityID. **BillingItem** entity new field added: LineItemGroupDescription holds description for BillingItems grouped on an invoice.
1.5.0	07/16/2014	Implemented WSDL version changes at the 0.0.0 level to track all WSDL updates. Initial version is 1.5.0. Version stored in WSDL. To query for the current WSDL version, use new API call "GetWsdlVersion()" on page 250 AllocationCode entity field changed: GeneralLedgerCode field is now a picklist. Field is read only. Use getFieldInfo() to return the picklist number value and label (name that appears in the UI menu).
	08/21/2014	Invoice entity: To maintain consistency between data submitted via the API and via the UI, Invoice.PaidDate uses only date information and now ignores time stamp information.
1.5.1	09/03/2014	**Contact** entity, the following fields were added BulkEmailOptOut: boolean, read/write, cannot query BulkEmailOptOutTime: datetime, read only NamePrefix: integer, picklist, read/write NameSuffix: integer,picklist, read/write SurveyOptOut: boolean, read only, cannot query FacebookURL: read/write TwitterURL: read/write LinkedInURL: read/write
1.5.2	10/07/2014	Two new entities added. These entities will allow user to create Installed Product (Configuration Item) Types and associate Udfs to the types: InstalledProductType InstalledProductTypeUdfAssociation

API License Agreement

WARRANTY DISCLAIMER - THE APIs are provided "AS IS" without warranty of any kind. Except to the extent required by applicable law, Autotask disclaims all warranties, whether express, implied or statutory, regarding the APIs, including without limitation any and all implied warranties of merchantability, accuracy, results of use, reliability, fitness for a particular purpose, title and non-infringement with third-party rights. Further, Autotask disclaims any warranty that your use of the APIs will be uninterrupted or error free.

COMMERCIAL USE – You may build applications using the API for commercial use. It is highly recommended that You contact our Autotask After-market Sales Team before building any commercial applications. Please send an outline of your planned application to aftermarketsales@autotask.com. We will respond within 48 hours to let you know whether the feature is part of the short-term roadmap or being developed by a partner. We will also discuss the possibility of including your proposed application in the after-market store and whether there are opportunities for joint sales efforts.

SUPPORT AND UPGRADES – Autotask does not guarantee upward compatibility but will make every effort to do so. You are not entitled to any support for the APIs beyond their standard support for Autotask. Autotask provides support for the current and previous two versions of the API.

PROPRIETARY RIGHTS - The APIs and all intellectual property rights in and to the APIs are and shall at all times remain the sole and exclusive property of Autotask and are protected by applicable intellectual property laws and treaties. Autotask is not restricted from developing and/or adding any features necessary to the Autotask product which may compete or overlap with applications built by You.

INDEMNITY - You agree that Autotask shall have no liability whatsoever for any use You make of the APIs. You shall indemnify and hold harmless Autotask from any and all claims, damages, liabilities, costs and fees (including reasonable attorneys' fees) arising from your use of the APIs.

TERM AND TERMINATION - Either party may terminate this Agreement at any time, for any reason, or for no reason including, but not limited to, if You violate any provision of this Agreement. You agree to be fully responsible for your own conduct and content while using the API, and for any consequences thereof. You agree to use the API only for purposes that are legal and not malicious.

Changes by Release: June 2009 and After

This section provides a more detailed summary of the changes listed in the API Revision History.

June 2009

The June 2009 release Web Services changes should not impact existing Web Services functionality.

TimeEntry entity

TimeEntry entity now allows Create and Update.

Two fields added: DateWorked (stores the date of the work for which time is being entered)

Type (picklist, stores the type of work done)

One field removed: Date (this field stores the same information as DateWorked)

NOTE: Web Services API TimeEntry does not support the interface Bill Immediately feature. All time entries created and updated through Web Services API must be approved and posted.

Service and ServiceBundle entities

One field added: UnitCost (stores the Unit Cost for the service or service bundle)

ContractServiceAdjustment and **ContractServiceBundleAdjustment** entities

One field added:AdjustedUnitPrice (stores the new, adjusted Unit Price)

Process to add Service or Service Bundle to a Contract has been streamlined as follows:

NOTE: The following information applies to ContractService and ContractServiceBundle and ContractServiceAdjustment and ContractServiceBundle Adjustment.

Previously, Web Services required two entities to add a Service to a contract: the ContractService entity that relates the Service to the Contract, and the ContractServiceAdjustment entity that adds or adjusts the number of units of the service that are available to the specified contract.

Now you can add a Service to a contract using only the ContractServiceAdjustment entity. When the Web Services API creates a ContractServiceAdjustment entity, it checks for the ContractService entity and, if no ContractService entity exists, Web Services will create the entity.

The ContractService and ContractServiceBundle entities are still in place, allowing you to continue to use the two step procedure if desired.

ContractRetainer entity

One field added: paymentID (picklist, stores the payment type)

Label changes: Contract Object ID label has been changed to ContractID label

NOTE Re: DatePaid field - This field was removed from the Autotask interface but remains in the ContractRetainer entity for version 1.4. It will be removed in a future release where the Web Services API moves to a new version.

ContractServiceUnitentity

Two fields added: **Cost** (stores the Unit Cost); **VendorAccountID** (if a Vendor is associated with the service, stores the Vendor ID)

ContractServiceBundleUnit

One field added: Cost (stores the Unit Cost)

Note: Unlike the ContractServiceUnit, a ContractServiceBundleUnit cannot have a Vendor association.

QuoteItem

When QuoteItem.Type = 11 (Service), UnitCost defaults to the unit cost specified for that service in the Admin module.

When QuoteItem.Type = 12 (Service Bundle), UnitCost defaults to the sum of the unit costs for all services within that bundle.

Installed Product entity

Data corrected for one field: Type is no longer Required

Quote entity

Data corrected for one field: id field is Read Only

QuoteItem entity

Data corrected for one field: id field is Read Only

QuoteLocation entity

Data corrected for one field: id field is Read Only

Multiple Resources on a Ticket is currently not available for Create and Update in Web Services

Although the Autotask interface now allows multiple resources on a ticket, the current versions of Web Services do not. Web Services is, however, aware of and will check existing multiple resources on a ticket and will not allow any resource to be assigned as primary resource if that resource is already a secondary resource.

Added Use Types table to AllocationCode entity information.

July 1, 2009

Contract entity

OverageBillingRate field no longer required for block hour contracts.

Installed Product entity

Added the following field: **InstalledByID** (Read only, not required)

July 21, 2009

Invoice entity

Three fields added: **IsVoided** (Read Only, Not Required); **VoidedDate** (Read Only, Not Required); **VoidedByResourceID** (Read Only, Not Required)

September 10, 2009

TimeEntry entity

Three fields added: **CreatorResourceID**, **LastModifiedTime** (date and time last modified), and **LastModifiedResourceID**

TimeEntry now respects the Autotask Proxy Time Entry workflow policy setting, that is, when the Proxy Time Entry workflow policy is enabled for Timesheet Approvers, the Timesheet Approvers for the TimeEntry owner can create and update a TimeEntry. When Proxy Time entry is enabled for Administrators, an Administrator can create or update a TimeEntry for all active resources. When the workflow policy is disabled, only the Time Entry owner can create and update time entries.

Project entity

Two fields added: **Department**, **LineofBusiness**

November 11, 2009

ProductVendor entity

Added note describing new behavior; that is, on Create, if no default ProductVendor exists, the newly created ProductVendor will be set as default.

TimeEntry entity

Added an additional condition: Time Entries cannot be created for Projects with Project Type of Archived (1), Template (3), Baseline (8).

December 9, 2009

QueryXML

Document update only. No change to API. Added to document: a list of five special XML characters and the entity references required to escape them.

January 19, 2010

BillingItem entity

Added new field, TaxDollars, for retainer tax where applicable.

ContractRetainer

AmountApproved field will include retainer tax that has been deducted, where applicable.

AccountLocation entity

Added note indicating that Web Services cannot respect UI condition that applies Site Configuration UDF requirement to Customer type accounts only.

February 9, 2010

Ticket entity

Added new field, Resolution.

Added condition indicating that for Contact ID field, Contact.AccountID must = Account or Parent Account associated with ticket.

AccountLocation entity

Removed note re: Site Configuration UDFs that was added 1/19/2010 and added updated information for Site Configuration UDFs in Web Services.

April 12, 2010 - Text update only

No change to Web Services API

Updated text in getting started NOTE to match previous update information for three URL variations that appear for WSDL and Web Services.

April 16, 2010

Reversion Web Services API to version 1.5

TimeEntry entity

Added the following new fields: AllocationCodeID, ContractID, NonBillable, ShowOnInvoice.

Opportunity entity

Added the following new fields: Rating, TotalAmountMonths.

Quote entity

Added the following new fields: CalculateTaxSeparately,GroupByProductCategory, ShowEachTaxInGroup.

May 11, 2010

Added note re: SOAP header, pg. "Getting Started" on page 37

August 25, 2010

New Entity added: ClientPortalUser

September 7, 2010

New API call added: getThresholdAndUsageInfo()

Added information on threshold for API calls

September 21, 2010

Document update only: updated threshold description to provide information on how requests are calculated.

October 22, 2010

Document update only: added text to TimeEntry entity Behaviors and query() API call description to indicate that time entries and queries are in Eastern Standard Time (EST)

November 8, 2010

Document update only: Corrected myService URL in getZoneInfo() sample code

November 10, 2010

Updated query() api call to indicate that querying on protected UDF's is case sensitive.

Added conditional requirement to Ticket, Task, and TimeEntry entities to indicate that an attempt to create using a resource + role combinations where the role is inactive will trigger an error message.

November 16, 2010

Documentation update only: Added note to indicate that AllocationCodeID on ticket may be required if Autotask workflow policy to require Work Type on a ticket is enabled.

December 1, 2010

Documentation update only: Corrected Ticket entity CreateDate field datatype, changing datetime to date.

January 6, 2011

Project entity

CompanyOwnerResourceID field is no longer required.

Resource entity

Added the following field: DefaultServiceDeskRoleID

Documentation update only: added character length for all String datatype fields

February 8, 2011

getZoneInfo() API call

Now returns Database Type field to indicate database type value of Pro or Go!

Documentation updated:

Section titled "User-defined Fields" has moved. It is now after Entities and before API Calls. Also some minor editing, no content changes.

Added text to indicate entities that are not available in Autotask Go!

February 23, 2011

Product entity

PeriodType field is no longer required.

For API support of Attachments, added descriptions for the following entity and three API calls:

New entity - AttachmentInfo

Three new API calls – GetAttachment, DeleteAttachment, CreateAttachment

For details see the entity and API call descriptions.

March 2, 2011

InstalledProduct entity

SerialNumber (string) - character limit changed to 100

May 4, 2011

Added base URL information for London Data Center (formerly Global 1)

July 7, 2011

Added base URL information for North America zone change to America East and America West.

August 9, 2011

Added Pre-release zone information to "Getting Started" section

Added two new entities:

Expense Report

Expense Item

TimeEntry entity

LastModifiedDateTime data type changed from Integer to DateTime

QuoteItem entity

Quote Items can now be deleted, that is, for the QuoteItem entity, "Can Delete" now = True.

September 21, 2011

Note added to TicketNote entity re: excluding workflow notes from query

API calls -

getThresholdAndUsageInfo() call

Added sample code

delete() call added (currently for QuoteItem only)

October 11, 2011

BillingItem, **Contract**, **Ticket**, **Task**, and **Project** Entities:

New field added - PurchaseOrderNumber

October 23, 2011

Contact Entity:

AccountID field is now read only (to prevent possible data corruption)

November 21, 2011

Add Australia zone (ww6) to list of URLs

delete API call

Added sample code

InstalledProduct entity

Active field datatype changed from short to boolean, per change to AT Database

February 28, 2012

ExpenseReport entity:

New field added: ApprovedDate

Contract entity:

New field added: OpportunityID

TaskNote entity added

ProjectNote entity added

Added Webservices URLs for localization: Germany and China

March 6, 2012

Account and **Contact** entities:

Country field character limit increased to 100

End of Updates Available in Non-English Zones

May 15, 2012

ProjectNote entity:

Correction -added Announce field

Added Spanish zone

May 30, 2012

Ticket entity

Allow update on ticket with inactive attribute value if that value is not being changed; cannot assign new inactive value

BillingItem entity

New field added: ExpenseItemID

June 4, 2012

Invoice entity

Fields added: PaidDate, TotalTaxValue

Contact entity

State field increased to 40 characters

Account entity

Fields added: Active, ClientPortalActive, TaskFireActive

State field increased to 40 characters

Resource entity

Enabled Update when logged in user has System Administrator level access.

Added the following fields: DateFormat, TimeFormat, Password, HomePhone, PayrollType, TravelAvailabilityPct

BillingItem entity

Field added: ExtendedPrice

InstalledProduct entity

Update no longer requires that the resource associated with InstalledByID have Status = Active.

Field added: InstalledByContactID

July 25, 2012

Invoice entity - Document correction

PaidDate field is NOT marked as ReadOnly

October 2, 2012

Added two new entities:

SalesOrder

BillingItemApprovalLevel

Added fields to the following entities:

To the **InventoryItem** entity

Reserved

Picked

To the **InventoryLocation** entity

IsDefault

ResourceID

To the **Opportunity** entity

CloseDate

SalesOrderID

To the **Product** entity

DoesNotRequireProcurement

To the **QuoteItem** entity

AverageCost

HighestCost

To the **Role** entity

SystemRole

To the **TimeEntry** entity

BillingApprovalDateTime

BillingApprovalLevelMostRecent

BillingApprovalResourceID

October 30, 2012

To the **PurchaseOrderItem** entity

Added the SalesOrder field

January 9, 2013

To the **AttachmentInfo** entity table

Added the ContentType field. The field was previously added to this entity but not noted in the document

GetAttachment()

Added note re: value returned is no longer a full path for Autotask attachments. Currently returns only filename and extension. Client Portal attachments are not affected.
Code sample was updated.

To the **Resource** entity

Added requirements for Password field.

To the **ClientPortalResource** entity

Added additional options for Password field requirements.

January 23, 2013

Added the following entities:

AccountToDo

ActionType

ContractCost

ProjectCost

TicketCost

To the **Product** entity

Added Behavior note: DoesNotRequireProcurement field defaults to False

To the **BillingItem** entity

The following fields were added: ContractCostID, ProjectCostID, TicketCostID

Service entity updated to allow create() and update()

February 5, 2013 (Maintenance Release)

Added the following entities:

UserDefinedFieldDefinition

UserDefinedFieldListItem

Document only corrections:

PurchaseOrderReceive entity does not accept update()

For Service entity: Name, PeriodType, and UnitPrice are required.

QuoteItem.LineDiscount is Required. When no LineDiscount value is provided, LineDiscount must = 0.

Clarification:

AccountLocation allows update() only when UDFs that can be updated are added to the entity. Only those UDFs can be updated.

May 7, 2013 (Spring Release LR)

Added the following entities:

Tax

TaxCategory

TaxRegion

The following entities were modified as noted:

Account entity

Fields added - AdditionalAddressInformation, TaxExempt, TaxID, TaxRegion

Notes added about changes to Country field.

AllocationCode entity

Field added - TaxCategoryID

The Taxable field is no longer used. Values passed in to this field will be ignored and queries will return the empty value " ". Taxable status is now determined by TaxRegion and TaxCategory.

BillingItem entity

Fields added - LineItemID, MilestoneID, ServiceID, ServiceBundleID, VendorID

Contact entity

Field added - ExternalID

Notes added about change to Country field.

Invoice entity

Fields added - BatchID, DueDate, TaxRegionName

The TaxGroup field is no longer queryable.

Project entity

ExtProjectType is not queryable and no longer accepts values on create() and update().

Type picklist includes only Internal, Client, Proposal, Template. It no longer includes Archived, Business Objective, Inactivated. Existing projects of the retired types have been updated as described in the Project entity description.

PurchaseOrder and **Quote** entities

The TaxGroup field now passes TaxRegionID; when not provided, no taxes are applied.

QuoteItem entity

Fields added - TotalEffectiveTax, TaxCategoryID

IsTaxable is now determined by QuoteItem.TaxCategoryID and TaxGroup (Tax Region ID) for the associated Quote entity, and TotalEffectiveTax value.

Resource entity

Field added: InternalCost

SalesOrder entity

Notes added about change to ShipToCountry and BillToCounty fields.

Task entity

Fields added - IsVisibleInClientPortal, CanClientPortalUserCompleteTask

Priority is no longer required. Will default to 0.

May 21, 2013

The following entity was modified:

QuoteItem entity

TaxCategoryID now accepts the value passed in. If no value is provided then ProductID, CostID, and ExpenseID will pull the value from the associated AllocationCode.

June 20, 2013

getZoneInfo() now returns the CI number, a unique identifier, for the database being accessed.

June 26, 2013

AllocationCode entity:
Product allocation codes (type 7) have been moved to Material allocation codes (type 4). Queries on type 7 allocation codes will be re-directed to type 4. Existing integrations can continue to use type 7 codes, now mapped to type 4.

Product entity:
Note added re: changes to Product allocation code (type 7). Product entity requires ProductAllocationCodeID.

July 17, 2013

Added the following entity:

ContractTicketPurchase

July 25, 2013

NOTE ADDED to **Project** entity:
RE: change in behavior for Project.StartDate with new project schedule design

DOCUMENTATION CORRECTIONS

Corrected the content of the entity tables as indicated (no change to API):

ContractService - added InvoiceDescription

ContractServiceBundle - added InvoiceDescription

ExpenseItem - added PurchaseOrderNumber

InstalledProduct - added ParentInstalledProductID

Quote - added ShowEachTaxInGroup

Resource - added AccountingReferenceID

ServiceBundle - added InvoiceDescription

July 26, 2013

Updated Ticket entity to reflect changes that occurred with the release of Problem Management in Autotask (Q1 Projects release).

Added the following fields to **Ticket** entity:

ProblemTicketID

TicketType

July 31, 2013

Resource entity

Suffix field changed to Picklist

Corrected text error (transposition of project with product) in July 25 update

August 7, 2013

Task and **Phase** entities:

Condition added - ProductID cannot be changed.

August 27, 2013

Account entity:

AccountNumber field updated to accept 50 characters

September 2013

Account entity – fields added:

BillToAdditionalAddressInformation

BillToAddress1

BillToAddress2

BillToAddressToUse

BillToAttention

BillToCity BillToCountryID

BillToState

BillToZipCode

CountryID

InvoiceMethod

InvoiceNonContractItemsToParentAccount

Contact entity – field added:

CountryID

Sales Order entity – fields added:

BillToCountry ID

ShipToCountryID

ClientPortalUser entity -

note added re: how to determine Taskfire Users by Security Level

Added the following entities:

AdditionalInvoiceFieldValue

Country

InternalLocation

InvoiceTemplate

PaymentTerm

Added Best Practices (document only update)

October 8, 2013

Document update only: added content to Best Practices

October 21, 2013

The TicketNote entity must respect the edit ticket note permissions implemented with Autotask release 2013.2 (October 22,). Permissions are assigned to the user's security level. Note added to TicketNote entity topic.

October 22, 2013

Added the following entity:

InternalLocation

November 5, 2013

Resource entity:

on query() or update(), if the InternalCost date range does not include the current date, the Internal Cost is not Active and Resource.InternalCost is set to 0.

Documentation update only:

update() call - added information to clarify that, on update, if no value is provided for fields that are not read only, the fields will be cleared.

Best Practices - added "On update() Provide Data for All Non-Read Only Fields"

Getting Started - added information to clarify need to update Sample Code with zone specific information

December 18, 2013

query():A query now requires that a field tag must contain an expression tag, and an expression tag must contain an op attribute.

When querying on any entity, an error message is generated if the XML

- contains a field tag which does not contain an expression tag.

- contains an expression tag that does not contain an op attribute.

January 7, 2014

getEntityInfo(): The API entities now respect Autotask custom security settings for the logged in user. Use getEntityInfo() to retrieve information about the logged in user's access.

getFieldInfo() now returns, for Picklist values, IsActive and IsSystem.

User-defined Fields required in the UI are not required by the API.

Contract entity, field added:

SetupFeeContractCode

ContractCost entity, fields added:

ContractServiceBundleID

ContractServiceID

ProjectCost entity, fields added:

ContractServiceBundleID

ContractServiceID

ContractServiceAdjustment entity, field added:

AdjustedUnitCost

Service entity, field added:

ServiceLevelAgreementID

ServiceBundle entity, field added

ServiceLevelAgreementID

ShippingType entity, field added,

AllocationCodeID

Task entity, field added:

RemainingHours

Task entity, the following fields are required only when task has assigned resources

AllocationCodeID

DepartmentID

Ticket entity, fields added:

ChangeApprovalBoard

ChangeApprovalType (use default value unless otherwise specified by user)

ChangeApprovalStatus

ChangeInfoField1

ChangeInfoField2

ChangeInfoField3

ChangeInfoField4

ChangeInfoField5

ContractServiceBundleID

ContractServiceID

LastCustomerNotificationDateTime

LastCustomerVisibleActivityDateTime

TicketCost entity, fields added:

ContractServiceBundleID

ContractServiceID

TimeEntry entity, fields added:

ContractServiceBundleID

ContractServiceID

The following entities can now be created and updated:

ServiceCallTask

ServiceCallTaskResource

ServiceCallTicket

ServiceCallTicketResource

The following entities were added:

AccountTeam

ChangeRequestLink

Department

TicketChangeRequestApproval

The picklist flag has been removed from the following Ticket entity fields: **AccountID**, **AllocationCodeID**, **AssignResourceRoleID**, **ContactID**, **ContractID**, **InstalledProductID**

January 9, 2014

Document correction only:

for getEntityInfo(), corrected new field names, for example, "CanICreate" changed to "UserAccessForCreate"

February 26, 2014

BillingItem entity field added:

LineItemFullDescription

ContractRetainer entity condition added:

IsPaid value cannot be updated if ContractRetainer is associated to a ContractCost.

March 12, 2014

New entities added:

TaskSecondaryResource

TicketSecondaryResource

TaskPredecessor

ResourceRole

Ticket entity:

On update(), Priority and InstalledProduct fields are validated only when the field is changed. Validation will fail if the field is changed to an inactive value.

On update(), the ContactID field is validated only when the field is changed. Validation will fail if the field is changed to an inactive value.

March 26, 2014

Implemented a usage based latency tied to the threshold on API requests. Refer to "Usage Based Latency" on page 40.

April 9, 2014

No API changes. Added information to the Getting Started chapter of the documentation regarding differences between User Interface terminology and the API entity and field names. Refer to "Autotask UI and Web Services Terminology Differences" on page 41.

April 23, 2014

New entity added:

ServiceBundleService (allows developers to query for Services assigned to a Service Bundle and add and remove Service Bundle Services through the API)

AttachmentInfo entity field added (to address issues with deletion of, and missing ParentID for, Opportunity type attachments through the API):

OpportunityID

When AttachmentInfo.Type = Opportunity, then AttachmentInfo.OpportunityID is required and AttachmentInfo.ParentID is not required. AttachmentInfo.ParentID is required for all other attachment types.

Documentation Update Only: Added information about the CountryID field under Conditions and Requirements for Account and Contact entities.

May 7, 2014

AttachmentInfo entity change in requirements:

With the May 7 maintenance release, Opportunity type attachments must specify one of the following combinations, a Type ID and ParentID, or TypeID and OpportunityID, or TypeID and both ParentID and OpportunityID.

OpportunityID value must be provided in order to query by OpportunityID.

BillingItem entity new field added:

LineItemGroupDescription holds description for BillingItems grouped on an invoice.

July 16, 2014

version 1.5.0

Implemented **WSDL version** changes at the 0.0.0 level to track all WSDL updates.
Initial version is 1.5.0. Version is stored in WSDL. To query for the current WSDL version, use new API call "GetWsdlVersion()" on page 250

AllocationCode entity field changed

GeneralLedgerCode field is now a picklist. Field is read only. Use getFieldInfo() to return the picklist number value and label (name that appears in the UI menu).

August 21, 2014

Invoice entity:

To maintain consistency between data submitted via the API and via the UI, Invoice.PaidDate uses only date information and now ignores time stamp information.

September 3, 2014

reversion to **1.5.1**

Contact entity, the following fields were added:

BulkEmailOptOut - boolean, read/write, cannot query

BulkEmailOptOutTime - datetime, read only

NamePrefix - integer, picklist, read/write

NameSuffix - integer, picklist, read/write

SurveyOptOut - boolean, read only, cannot query

FacebookURL - read/write

TwitterURL - read/write

LinkedInURL - read/write

October 7, 2014

reversion to **1.5.2**

Two new entities added. these entities will allow user to create Installed Product (Configuration Item) Types and associate Udfs to the types.

InstalledProductType

InstalledProductTypeUdfAssociation

Introduction

The Autotask Web Services API allows developers to create software to access Autotask's data and functionality. You can extend existing Web based applications to allow your users to access Autotask from those applications and complete tasks like create Autotask tickets and accounts and retrieve, update, and delete selected Autotask data.

Authentication passes the end user's login name and password. Client access to data and actions is limited by the Autotask permission level of the logged in end user.

Autotask Web Services API is SOAP Based

The API is entity oriented and **SOAP**based. It uses XML format for data interchange. The SOAP API defines the available functionality via the WSDL. By using SOAP, Autotask provides a standards based API that is secure, reliable, and flexible. Developers can use their preferred language to develop complex integrations with Autotask.

Currently, this document provides codes samples in .NET only. We encourage developers who work with other languages to participate in the Autotask Community Developer Tools and API forum. See Autotask Community Support, below.

TIP: For developers working with **PHP**, OpenDNS Engineering has generously published their Autotask Web Services API client as open source. You can access it here: OpenDNS Open Sources Autotask Web Service API Client. For additional information, refer to OpenDNS Open Sources Autotask Web Service API Client.

Requirements

In order to utilize the Autotask Web Services API you must configure the authentication as discussed in "Getting Started" on page 37.

API Call Threshold

To ensure an acceptable response time for all Web Services API users, Autotask sets a limit on the number of calls per hour between the API and an individual database. The limit is high. In addition, a usage based latency is applied as you approach the threshold. If you are using the API as intended you should not see an impact. For additional details, see "API Call Threshold" on page 40.

Autotask Community Support

Visit the Autotask Community Developer Tools and API forum to gain valuable information and assistance from other developers working with the Autotask Web Services API.

Getting Started

NOTE: When accessing the WSDL and the web services, you must use the URL zone variation that is consistent with the base URL that you use to access the Autotask Web site.

Going forward, the URL for your Autotask access may change; we recommend that, when writing your applications, you store the URL information in an initialization or configuration file. For example, .NET developers can store the information in the Web.config file. This allows easy updating if the URL information changes.

WSDL Versions

The Web Services API uses a three decimal numbering system to indicate its most current version. The API is currently in version 1.X.X. The second place number increases by 1 when a significant update occurs that will likely impact existing integrations.

The third place number increases when any change is made to the Web Services WSDL. These changes will not impact existing integrations. It is possible that, at some point, the WSDL used for integration development will differ from the WSDL available in different Autotask zones, for example, when an Autotask release is rolled out over several weeks. In addition, when working with international customers, there is always a delay between an Autotask English language release and the same release in the localized zones.

You can determine what WSDL version is being accessed by the logged in user with the "GetWsdlVersion()" on page 250 call.

Before you begin:

Create a Proxy Class (Web Reference)

In your development environment, create a proxy class (Web Reference) for the Autotask Web Services API by referencing the WSDL file. The appropriate URL is determined by your Autotask zone.

NOTE: Because an Autotask client can be located in any of Autotask's zones, we recommend that you set up your application so that "getZoneInfo()" on page 251 is the first call made in order to determine the proper zone (and accompanying URL) to use for all API interactions. You can direct the getZoneInfo request to any zone.

The current zone variations for the URL (see Note above) of the WDSDL file for 1.5 version are:

NOTE: All customers from the North America zone (**https://webservices.autotask.net/atservices/1.5/atws.wsdl** and **https://webservices.autotask.net/atservices/1.5/atws.asmx)** have been moved to American East and America West.

America East	https://webservices3.autotask.net/atservices/1.5/atws.wsdl
America West	https://webservices5.autotask.net/atservices/1.5/atws.wsdl
London Data Center	https://webservices4.autotask.net/atservices/1.5/atws.wsdl (Formerly Global 1)
Australia	https://webservices6.autotask.net/atservices/1.5/atws.wsdl
Germany	https://webservices7.autotask.net/atservices/1.5/atws.wsdl
China	https://webservices8.autotask.net/atservices/1.5/atws.wsdl
Italy	https://webservices9.autotask.net/atservices/1.5/atws.wsdl

France	https://webservices10.autotask.net/atservices/1.5/atws.wsdl
Japan	https://webservices11.autotask.net/atservices/1.5/atws.wsdl
Spain, Latin America	https://webservices12.autotask.net/atservices/1.5/atws.wsdl
Limited Release	https://webservices1.autotask.net/atservices/1.5/atws.wsdl
Pre-release	https://webservices2.autotask.net/atservices/1.5/atws.wsdl

NOTE to Developers for International Customers: Autotask releases do not occur simultaneously across all zones. For this reason there will be some unavoidable differences between versions of the WSDL, especially between the localized and English versions. If you bind your code to the English WSDL for development and testing, before you make your code available to users in any localized zone, bind your code to the WSDL for that zone. You can then detect and correct for any differences. Monitor Autotask release notices to be aware when releases are occurring in different zones.

Currently, the non-English localized versions of Autotask do not include any API updates made after February of 2012.

The current variations for the URL of the web services are:

America East	https://webservices3.autotask.net/atservices/1.5/atws.asmx
America West	https://webservices5.autotask.net/atservices/1.5/atws.asmx
London Data Center	https://webservices4.autotask.net/atservices/1.5/atws.asmx (Formerly Global 1)
Australia	https://webservices6.autotask.net/atservices/1.5/atws.asmx
Germany	https://webservices7.autotask.net/atservices/1.5/atws.asmx
China	https://webservices8.autotask.net/atservices/1.5/atws.asmx
Italy	https://webservices9.autotask.net/atservices/1.5/atws.asmx
France	https://webservices10.autotask.net/atservices/1.5/atws.asmx
Japan	https://webservices11.autotask.net/atservices/1.5/atws.asmx
Spain, Latin America	https://webservices12.autotask.net/atservices/1.5/atws.asmx
Limited Release	https://webservices1.autotask.net/atservices/1.5/atws.asmx
Pre-release	https://webservices2.autotask.net/atservices/1.5/atws.asmx

Please anticipate regular updates and the release of new versions of the Web Service API. New versions will, for the most part, support backward compatibility through multiple releases, but there may be changes to functionality that have an impact on your application. Please check Autotask release notes and any additional notifications that provide information on Web Services changes and updates.

Autotask will not continue to support each release indefinitely. We will provide advance notice of when we anticipate ending support of any version.

TIP: We strongly suggest that you update your application with each Web Service API release, or monitor the impact that changes might have on your application and plan accordingly, especially if Autotask plans to remove support for the API version that you are using.

NOTE: The WSDL includes a SOAP Header that is unrelated to general API usage. Unless our Web Services support or dev team indicates otherwise, you can ignore this header.

Configure programmatic authentication.

You can use your preferred language to configure your programmatic authentication to the Web Services API. This documentation currently includes sample authentication code in the following language(s):

VB.NET

TIP: For developers working with PHP, Open DNS Engineering has published their Autotask Web Services API client as open source. You can access it here: OpenDNS Open Sources Autotask Web Service API Client. For additional information, refer to OpenDNS Open Sources Autotask Web Service API Client.

VB.NET Sample Code

The following is sample VB.NET code to configure authentication to the Web Services API. This code assumes that you have a Web Reference class for the API named "WebReference".

NOTE: This is **sample code**. Please modify the sample to refer to your own Autotask Zone as described below.

To Use the Sample Code Below

You must replace the 2 that follows "webservices" in the URL with your Autotask **zone number**;

for example, if you use the America East zone, change the 2 to 3, that is,

"https://webservices3.autotask.net/atservices/1.5/atws.asmx".

If you are unsure of the number associated with your zone, refer to the URLs listed above.

TIP: If you do not know your Autotask zone, see "getZoneInfo()" on page 251.

```
Dim strWebURI As String = "https://webservices2.autotask.net/atservices/1.5/atws.asmx"
Dim myService As New WebReference.ATWS
myService.Url = strWebURI
Dim cred As New System.Net.NetworkCredential("myUserName@domain.com", "myPassword")
Dim credCache As New System.Net.CredentialCache
credCache.Add(New Uri(myService.Url), "Basic", cred)
myService.Credentials = credCache
```

NOTE: The Web Services API respects the Autotask security level permissions of the logged in user.

Time Data Storage and Return

Note that the Web Services API stores and returns all time data in Eastern Standard Time (EST).

API Call Threshold

To ensure an acceptable response time for all Web Services API users and prevent inadvertent coding errors from crippling the system, Autotask sets a limit on the number of external requests allowed per hour between an individual database and the API. That is, on receipt of a request, the API will check the number of requests received for that database in the preceding 60 minutes.

The threshold is generous and **you should not see an impact with normal API usage**.

It is possible to reach the threshold, however, if you are running multiple high usage integrations or one integration that is calling the API on multiple threads. If you should actually reach the threshold, API service is **temporarily suspended** and an error message is generated.

Usage Based Latency

The Web services API will add a usage based latency as you approach your threshold. The latency is calculated to prevent the number of calls within an hour from reaching the threshold amount. This preferred model allows applications to continue processing, rather than being interrupted by a suspension of service.

The latency is based on the number of requests submitted, as a percentage of the number of requests allowed, as follows:

0-49.99%, latency = 0 sec

50.00-74.99%, latency = .5 sec

75.00+%, latency = 1 sec

For example, if the threshold = 10,000 requests per hour, at 5,000 requests the API will add .5 seconds to each request up to 7,499. At 7,500 requests, the latency increases to 1 second.

If your Service is suspended

In addition to the error message returned through the API, Autotask sends an email notification if usage exceeds the threshold. The following order of preference applies.

1. If you have set up an "Unsuccessful Ticket Creation" email address for your ATES mailbox under Incoming Email Processing, and that address is configured to receive human-readable notifications, the warning is sent to that email address. If multiple addresses are configured, we will send to all of them.
2. If the email address described above is not found, the email is sent to your Support Email Address.
3. If no Support Email Address is found, the email is sent to the primary email address of the logged-in user responsible for the violation.

Use the getThresholdAndUsageInfo() API Call

You can use the API call, **getThresholdAndUsageInfo()**, to determine your database threshold and monitor how close you are to reaching it. For details about this call, see "getThresholdAndUsageInfo()" on page 247.

NOTE: The limit can be adjusted under special circumstances. Please contact Autotask Customer Support to request an adjustment.

For additional information on working with the Web Services API, check out the Autotask Community Developer Tools and API forum.

Autotask UI and Web Services Terminology Differences

The Autotask user interface is dynamic and subject to changes as user needs evolve. The API is code based and therefore requires a level of consistency that does not apply to the UI. For this reason, the terminology used to describe the Web services entities and field names can differ from the terms used to describe the same entities and fields in the UI.

The following table lists the primary terminology differences at this time.

Term in UI	Additional Context Information	Term in API
Billing address	Address of the company to bill	BillToAddress
Block Hour Multiplier		ContractFactor, BlockHourFactor
Charge	When referring to any billable or non-billable monetary amount for an Inventory Item, Product, or Material Cost associated with a Project, Ticket, or Contract. In the interface, cost is used when referring to a Unit Cost or amount charged by Vendor	Cost
Classification	Replaces the Key Account Icon classifications	KeyAccountIcon
Company or, in the Client Portal, Client	Company replaces Account in most usage. For Client Portal, Client is used instead of Company (Client Portal client instead of Client Portal account). In the interface, Account remains Account in word pairings where account means sales account, financial account, or log in account, for example, Account Manager, Synchronization Account, User Account.	Account
Configuration Item		InstalledProduct
Destination Company		ToAccount (as in BillableToAccount)
General Ledger Account	Also previously referred to as GL Account, GL Code, and General Ledger Code.	GeneralLedgerCode
Lead Source	The source of a sales lead	LeadReferral
Name	The following entities now have Names instead of Titles: Reference, Attachment, Opportunity , Appointment, Project, Sales Order, Issue, Item, Activity, Milestone, Phase, Wizard. **Note:** In the UI, Contacts have both a name and a title	Title
Promised fulfillment date	Used with Sales Orders, the date you expect to deliver goods and services.	PromisedDueDate

Term in UI	Additional Context Information	Term in API
Promised fulfillment date	Used with Opportunities, the date you expect to deliver goods and services	ProjectedLiveDate
Quote	Used for all references to a quote, including eQuote	Equote

API Best Practices

These suggestions can help you to insure the best results from the Autotask Web Services API.

Know Your Zones

Autotask has multiple zones. They are generally allocated geographically, but a customer can be located in any zone. The zone number appears in the customer's Autotask URL.

The Web services and WSDL URLs are specific to the customer's zone. Autotask releases do not occur simultaneously across all zones. For this reason there will be some unavoidable differences between versions of the WSDL.This difference is most noticeable between the localized and English versions.

To avoid problems when you make your code available to customers in multiple zones, do the following:

- To determine the proper zone (and accompanying URL) to use for all API interactions, set up your application so that "getZoneInfo()" on page 251 is the first call made.
- If you bind your code to an English language WSDL for development and testing, before you make your code available to users in any localized zone, bind your code to the WSDL for that zone. You can then detect and correct for any differences.

For a list of WSDL and Web Service URLs for all zones, refer to "Getting Started" on page 37.

Do Not Use GetZoneInfo To...

Do not use GetZoneInfo to validate that you have the correct username and password for API access. GetZoneInfo does not validate against username or password. It validates against the base URL only.

Use Multiple Queries to Retrieve Over 500 Records

When constructing queries, be aware that Autotask will return only 500 records at once, sorted by id. If there are over 500 records that meet your criteria, you must create multiple queries where each query filters for id value > the maximum value received in the previous query.

Use LastModifiedDate to Increase Query Efficiency

Many API users regularly use API queries to update Autotask data cached in an external system. Cached data is then used by applications, for example, dashboards, that allow users to quickly access selected data. With a maximum return of 500 records at once, users with extensive databases require multiple calls. This option is time intensive and, when traffic volume is high, can tax the API resources.

An alternative to returning all records is to formulate queries to return only records that have changed since the last query. You can do this for selected entities using the LastModifiedDate or LastActivityDate fields.

- **LastModifiedDate** stores the most recent date and time that the entity was updated or changed.
- **LastActivityDate** stores both the most recent date and time that the entity was modified, as described for LastModifiedDate, and the most recent date and time that any activity related to the entity occurred. For example, this might include a note or time entry added to the entity.

You can determine if the available fields meets your needs.

Not all entities have these fields exposed. With the exception of the Contact entity, no entity has more than one of the two fields available.

The following table indicates which entities have LastModifiedDate or LastActivityDate fields exposed.

NOTE: The field names for some entities use only Date and others use DateTime. Both names return the same data and it is always a datetime data type.

Entity	Use this field
Account	LastActivityDate
AccountNote	LastModifiedDate
AccountToDo	LastModifiedDate
Contact	LastActivityDate and LastModifiedDate
ContractNote	LastActivityDate
Phase	LastActivityDateTime
ProjectNote	LastActivityDate
Service	LastModifiedDate
ServiceBundle	LastModifiedDate
ServiceCall	LastModifiedDateTime
Task	LastActivityDateTime
TaskNote	LastActivityDate
Ticket	LastActivityDate
TicketNote	LastAcivityDate
TimeEntry	LastModifiedDateTime

StatusLastModifiedDate

Three entities include a StatusLastModifiedDate field: Contract Cost, ProjectCost, and TicketCost. The StatusLastModifiedDate field indicates changes to **the cost status only** , for use with the procurement work-flow. These entities do not have an option to determine if/when other fields were modified.

Entity	Use this field
ContractCost	StatusLastModifiedDate
ProjectCost	StatusLastModifiedDate
TicketCost	StatusLastModifiedDate

Use the Entity Returned by update() and create() to Obtain Id Only

You can use the entity returned by create() and update() to obtain the Id field content, but do not rely on the rest of the content. Some entity fields cannot be calculated immediately during create and update. In the returned entity , those fields may not be accurate.

If you need to access the entity for information other than the Id, query for the entity after the create/update succeeds.

Be Sure You Have the Correct Autotask Security Level

The API respects user security levels within Autotask. If the user you are authenticating with has access to certain features in Autotask through the UI, then they will have the same access through the API. To determine the logged in user's access, use "getEntityInfo()" on page 242.

To be safe and to avoid the possibility that you will be denied access unexpectedly, authenticate as a user with Full Access security level.

On update() Provide Data for All Non-Read Only Fields

When you update an entity, all entity fields are updated unless they are Read Only. You must provide a value for any field that is **not** Read Only, or that field will be cleared.

As best practice, you can create a test account, then update using the test account to make sure you are getting the expected results.

Check your database usage threshold and monitor the volume of your activity

To maintain optimum response time for all users, Autotask sets a limit on the number of external requests allowed per hour between an individual database and the API. Most users will never exceed the threshold. If you do exceed the threshold, your service is temporarily suspended.

You can avoid a service suspension by monitoring your usage. A usage-based latency is added when you reach 50% of allowed calls, and the latency increases when you reach 75% off allowed calls. In addition, you will receive a notification if you approach the threshold. For details on the latency and warning notification, refer to "API Call Threshold" on page 40.

You can use the getThresholdAndUsageInfo() API Call to learn the threshold for your database and monitor your current usage. Refer to "getThresholdAndUsageInfo()" on page 247.

About Autotask API Entities

The Autotask Web Services API presents selected Autotask entities as programming objects that allow the client to perform actions on the specific entity type. Each Autotask Entity object inherits from the Autotask base class Entity.

Currently the API provides the following actions: create, update, and query. A delete option is currently available for selected entities. The allowed actions are specific to the entity object, for example, although the client can create certain entity types, not all entity types accept a create() call. All entities can be queried.

NOTE: Although you can query all entities, some entities contain fields that cannot be queried. You can find these fields listed in the entity description under "**Fields that Cannot Be Queried**".

About Entity Fields

Each entity type object contains an array of properties that describe instances of the entity type. The properties represent data fields. The client can access and act on the entity's field parameters: supplying, updating, or deleting the field data. The allowed actions are specific to the field, for example, when updating an entity, not all fields within the entity can be updated. When querying, a small number of entities contain fields that cannot be queried. See the **Note** above.

For information on the currently available entities and fields consult the individual descriptions for each entity. See "Entities" on page 48. Some field data may vary between Autotask implementations and cannot be provided in this document, for example, picklist values and user-defined fields. For complete field descriptions specific to your Autotask implementation, use the "getFieldInfo()" on page 244or "getUDFInfo()" on page 248 API calls.

NOTE: The Autotask Web Services API stores and returns all entity time data in Eastern Standard Time (EST).

About the id Field

The id field belongs to each entity and acts as the unique identifier. It is created by the system and cannot be changed by the user application; for example, if you attempt to specify a value for the id field when using the create() API call, the create will fail.

About Required and Read Only Field Attributes

For the Autotask Web Services API, the Read Only and Required field attributes have the following meanings:

- Read Only - Read Only fields cannot be changed in an update call.
- Required – Required fields must be present when you attempt a create() call.

It is possible for a field to be both Read Only and Required. Some Read Only fields must be supplied for a create (), so they are Required but, once the entity has been created, they cannot be changed.

About Entity Descriptions

For a list of all currently available entities, see "Entities" on page 48, or check the Table of Contents. The

Entities list provides a page reference to additional information about the entity and its standard Autotask fields.

Each entity description includes the following information:

- Which actions can be performed on the entity.

  Note that actions are governed by the permissions of the logged in end user; for example, although an Account entity allows a create() call, the logged in end user may not have permission to create an Account entity.

- A listing of **Fields that Cannot Be Queried**, where applicable.
- A table that lists all standard Autotask fields by Field Name and provides the following information for each field: Field Name, Label, Data Type, Read Only, Is Required, Reference Name (provided only if the field is a reference), Picklist, and Picklist Parent Column Name (only if the field is a picklist child). Can Query is not included because non-queryable fields are listed under a separate heading. Refer to the previous bullet point.

In addition, to obtain current information on Entities and their field data specific to your implementation, use the following two API calls:

- For standard Autotask fields for each entity and all field data specific to your Autotask implementation, use the "getFieldInfo()" on page 244 API call.
- For user-defined fields (UDFs) and UDF field data for each entity that allows User Defined Fields, use the "getUDFInfo()" on page 248 API call.

Entities

The API currently exposes the following entities. Each of the entities inherits from the Autotask base class Entity.

Entity	Description
"Account" on page 54	Describes an Autotask Account. An account represents a company or organization that you do business with.
"AccountLocation" on page 58	Takes on the UDFs that hold the site setup information for the Account represented by AccountId.
"AccountNote" on page 59	Describes any sort of note created by an Autotask user and associated with an Account entity, as opposed to a Ticket Note.
"AccountTeam" on page 63	Describes the resources associated with an Account Team. In Autotask, the account team associates resources an account. The resources then have access to the account data when their security level allows account access at the Mine level.
"AccountToDo" on page 61	Describes an Autotask To-Do, a scheduled item associated with an Account that appears on the user's Autotask calendar. It can be assigned to any resource and associated with an Contract, Ticket, or Opportunity.
"ActionType" on page 64	Describes an Action Type assigned to a CRM Note or To-Do. The Action Type specifies the type of activity scheduled by the to-do or associated with the note and the ActionType View controls where the Note or To-do appears in the user interface.
"AllocationCode" on page 66	Describes an Autotask Allocation Code. An allocation code represents one of six billing item categories: Work Types and Internal Allocation Codes (Admin > Company Setup), Material Cost Codes and Product Codes (Admin > Products and Services > Products), Milestone Codes (Admin > Contracts), and Recurring Contract Service Codes (Admin > Products and Services > Services).
"Appointment" on page 69	Describes an Autotask Appointment, that is, scheduled calendar time that is not a service call.
"AttachmentInfo" on page 70	Describes an Attachment in Autotask. Attachments are external documents that are associated with an Autotask Account, Ticket, Project, or Opportunity entity.
"BillingItem" on page 72	Describes a billable item in Autotask that has been approved and posted. A billing item may or may not be included in an invoice and billed to the customer.
"BillingItemApprovalLevel" on page 75	Describes a multi-level approval record for an Autotask time entry. It allows developers to use the API to implement multi-tier approval for Autotask time entries. Only available through the Web Services API.
"ChangeRequestLink" on page 76	Describes the associations between Change Request tickets and both Incidents and Problems. Change request tickets are part of the Autotask change Management feature set.
"ClientPortalUser" on page 77	Describes an Autotask Account Contact that has been assigned access to the Client Access Portal.

Entity	Description
"Contact" on page 79	Describes an Autotask Contact. A Contact is an individual associated with an Account.
"Contract" on page 83	Describes an Autotask Contract. Contracts specify a billing arrangement with an Account. Autotask currently provides five contract types: Time and Materials, Fixed Price, Block Hours, Retainer, and Recurring Service.
"ContractBlock" on page 86	Describes an Autotask Contract Block which represents a block of hours purchased for a Block Hour type Contract.
"ContractCost" on page 88	Describes a cost associated with an Autotask contract.
"ContractFactor" on page 91	Describes an Autotask Block Hour Factor, an option used with Block Hour type Contracts. It allows you to compensate for the Block Hour fixed rate by applying a multiplier to specific role rates.
"ContractMilestone" on page 92	Describes a billing milestone for an Autotask Fixed Price type Contract. Billing milestones define tangible work or measured progress that must be completed before billing can take place.
"ContractNote" on page 94	Describes a note associated with an Autotask Contract. Notes are used to track information, update the status of the associated contract, and communicate with resources and customers.
"ContractRate" on page 95	Describes an Autotask Contract Rate. A Contract Rate is associated with a Role and is specific to a contract. You use it to override your company's standard role rate for labor tracked against the contract.
"ContractRetainer" on page 96	Describes a payment amount applied to a Retainer type contract and sets the time period covered by the purchase.
"ContractService" on page 101	Describes an Autotask Service added to a Recurring Service contract.
"ContractServiceBundle" on page 103	Describes an Autotask Service Bundle added to a Recurring Service contract. Autotask Service Bundles group multiple Services for purchase.
"ContractServiceAdjustment" on page 102	Describes an adjustment to the quantity of units of a Contract Service entity that are added to a Recurring Service Contract. It can only be created; it CANNOT be queried or updated. Changes made to the Contract using the ContractServiceAdjustment entity affect only the quantity of units.
"ContractServiceBundleAdjustment" on page 104	Describes an adjustment to the quantity of units of a Service Bundle that are added to a Recurring Service Contract. It can only be created; it cannot be queried or updated. Changes made to the Contract using the ContractServiceBundleAdjustment entity affect only the quantity of Contract Service units.
"ContractServiceUnit" on page 108	Describes the number of units of a specific service that are associated with a Recurring Service contract for a specific date range. Used for billing purposes.
"ContractServiceBundleUnit" on page 106	Describes the number of units of a specific service bundle that are associated with a Recurring Service contract for a specific date range. Used for billing purposes.
"ContractTicketPurchase" on page 110	Describes a payment amount applied to the purchase of (or pre-payment for) one or more Service Desk Tickets through a Per Ticket Contract.

Entity	Description
"Country" on page 112	Describes a Country as defined in the Autotask CRM module. Country entity is referenced by other entities and specifies the display name, address format, 2 letter county code, and ISO standard name associated with the country.
"Department" on page 98	Describes an Autotask Department. A Department is an association used to manage resources, especially when assigning project tasks.
"ExpenseItem" on page 113	Describes a line item associated with an Expense Report entity. It allows expense line items to be created, queried, and updated through the API.
"ExpenseReport" on page 116	Describes Expense Reports created in Autotask and is used to submit expense line items for approval and reimbursement.
"InstalledProduct" on page 118	Describes Autotask Configuration Items (previously known as Installed Products). Configuration items are Products that are associated with an Account entity.
"InstalledProductType" on page 120	Describes a Type, for example, printer, server, or workstation, assigned to a Configuration Item in Autotask. The InstalledProductType associates one or more User-defined Fields with configuration items of the same type.
"InstalledProductTypeUdfAssociation" on page 121	Describes one or more User-defined Fields associated with an InstalledProductType. An InstalledProductType is assigned to a Configuration Item in Autotask.
"InternalLocation" on page 123	Describes a Location defined in Company Setup in the Autotask Admin module. Every resource is associated with a location. The time zone and holiday set of the associated location are applied to the resource's time entries and schedules.
"InventoryItem" on page 125	Describes an Autotask Product that is associated with an Inventory Location in the Autotask Inventory module.
"InventoryItemSerialNumber" on page 127	Describes a serial number associated with an Inventory Item. It allows users to track and manage Inventory Items created from Autotask Products that require a unique serial number.
"InventoryLocation" on page 128	Describes an Autotask Inventory Location, that is, a physical or virtual place where your company stores or assigns inventory items.
"InventoryTransfer" on page 130	Describes a transaction where a specified quantity of one Inventory Item entity is transferred from the item’s currently assigned Inventory Location to another Inventory Location. It does not describe transactions where an Inventory Item is associated with an Account as a Configuration Item.
"Invoice" on page 132	Describes an Autotask Invoice. Invoices include Billing Items that have been approved and posted and are being billed to a customer or presented for information purposes only.
"InvoiceTemplate" on page 134	Describes a template that defines the content and appearance of an Autotask Invoice.
"Opportunity" on page 137	Describes an Autotask Opportunity. An opportunity is a forecasted piece of business, that is, an identifiable prospect that needs a product or service and offers a potential sale, project, or contract. This entity allows you to track the progress of the opportunity and generate sales forecasts.

Entity	Description
"PaymentTerm" on page 140	Describes an Autotask Payment Term. A payment term specifies conditions and requirements for payment due on an Autotask invoice; for example, Net 30 days.
"Phase" on page 141	Describes an Autotask project Phase. Phases allow users to break projects into sub-groups of project tasks.
"Product" on page 143	Describes an instance of hardware, software, or a material item in Autotask that a company sells or supports for customers.
"ProductVendor" on page 145	Describes a Vendor type Account that is associated with an Autotask Product.
"Project" on page 147	Describes an Autotask Project. A project defines and organizes a group of related tasks, events, and documents.
"ProjectCost" on page 150	Describes a cost associated with an Autotask Project.
"ProjectNote" on page 153	Describes notes created by an Autotask user and associated with a Project entity.
"PurchaseOrder" on page 155	Describes an Autotask Inventory module Purchase Order.
"PurchaseOrderItem" on page 158	Associates a Product entity with a PurchaseOrder entity.
"PurchaseOrderReceive" on page 160	Describes a transaction where a specified quantity of a Purchase Order Item is "received", that is, debited from the Quantity value of the associated Purchase Order Item and added to the Quantity On Hand value of the Inventory Item.
"Quote" on page 162	Describes a Quote in Autotask. Must be associated with an Autotask Opportunity entity and allows users to specify and track multiple products, services, labor items, etc., that further define the Opportunity.
"QuoteItem" on page 164	Describes an Autotask Quote Item. Quote Items define a line item added to an Autotask Quote.
"QuoteLocation" on page 167	Describes a location associated with an Autotask Quote that defines address information for a ShipToLocationID and/or BillToLocationID.
"Resource" on page 168	Describes an Autotask Resource. Autotask Resources are employees, contractors, or consultants with access to a company's Autotask system.
"ResourceRole" on page 171	Describes a Resource - Role relationship.
"Role" on page 172	Describes an Autotask Role. Roles are associated with a department, have a standard billing rate, and are assigned to resources for billing purposes.
"SalesOrder" on page 174	In Autotask, a sales order is associated with an Opportunity. It provides a method to track cost items generated from an Autotask Quote.
"Service" on page 177	Describes a deliverable item that represents a pre-defined unit of work performed for a set price and billed at regular intervals, for example, a "Disk Backup" performed for one computer.
"ServiceBundle" on page 179	Describes a group of Service entity items that are priced and billed as one component of Recurring Service type contract.
"ServiceBundleService" on page 181	Describes a Service entity assigned to a ServiceBundle entity.

Entity	Description
"ServiceCall" on page 183	Describes an Autotask service call and its relationship to four other service call related entities. Service calls are instances of time, with specified start and stop times, that are scheduled to perform work for an Account.
"ServiceCallTask" on page 185	Describes an Autotask project task assigned to a service call.
"ServiceCallTaskResource" on page 186	Describes an Autotask resource assigned to a task that is assigned to a service call.
"ServiceCallTicket" on page 188	Describes an Autotask ticket assigned to a service call.
"ServiceCallTicketResource" on page 189	Describes an Autotask resource assigned to a ticket that is assigned to a service call.
"ShippingType" on page 191	Describes an Autotask Shipping Type. A shipping type defines a carrier for a product shipment. It can be associated with a Quote entity.
"Task" on page 192	Describes an Autotask Task. Tasks are associated with a Project and define work that must be done.
"TaskNote" on page 195	Describes notes created by an Autotask user and associated with a Task entity.
"TaskPredecessor" on page 196	Describes a predecessor/successor arrangement between two project schedule items.
"TaskSecondaryResource" on page 197	Describes a resource assigned to a task but not as the primary resource.
"Tax" on page 198	Describes the tax rate charged to a customer for specific goods or services purchased in a specified tax region.
"TaxCategory" on page 200	Describes the tax rate for a specific billing item. Used to determine a customer's total taxes on billing items.
"TaxRegion" on page 201	Describes a geographic area where billing items have the same tax rate. Used to determine a customer's total taxes on billing items.
"Ticket" on page 202	Describes an Autotask Ticket. Tickets define service requests within the Autotask system.
"TicketChangeRequestApproval" on page 207	Describes a record of approval for a ticket change request. The change request approval process is part of the Autotask Change Management feature set.
"TicketCost" on page 209	Describes a cost associated with an Autotask Ticket.
"TicketNote" on page 212	Describes a note created by an Autotask user and associated with a Ticket entity.
"TicketSecondaryResource" on page 214	Describes a resource assigned to a ticket but not as the primary resource.
"TimeEntry" on page 215	Describes an Autotask Time Entry. A time entry allows an Autotask resource to enter ticket and task time (Labor) and general or regular time (non-customer facing time).

Entity	Description
"UserDefinedFieldDefinition" on page 221	This entity defines a User-defined Field (UDF) in Autotask. User-defined Fields are custom fields that each Autotask customer can add to their Account, Contact, Opportunity, Sales Order, Projects, Products, Configuration Items, Ticket, and Task tables.
"UserDefinedFieldListItem" on page 224	Describes a list item associated with a UserDefinedFieldDefinition entity that has DataType = List

Account

The Account entity describes an Autotask Account. In Autotask, an account represents a company or organization that you do business with. Autotask users manage Accounts through the CRM module (CRM > Accounts), or the Directory module (Directory > Accounts).

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	Account
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	•

Fields that Cannot Be Queried

The following fields from this entity will return an error when queried.

- AssetValue
- SICCode
- StockMarket
- StockSymbol

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

Conditions and Requirements

In Autotask , the Country fields now provide a countries pick list. The following conditions apply to the Country fields in the API.

NOTE: The Country field is not the same as the CountryID field. The CountryID field references the ID field of the Country entity. If a value is provided for the Country field, the API will use that value and, based on the rules below, map that value to a valid Country entity. If no value is provided for the CountryID field, the ID value for the matching Country entity will automatically populate the CountryID field.

- On create() and update():

  If the value provided for Country matches either a standard country abbreviation, an ISO standard country name, or an Autotask country display name, the value is mapped to that country.

  If no match is found, the value is mapped to "Other". The passed in text is stored.

In the UI, "Other" appears in the Country field as "Other [stored text value]". These "Other" values are not available for selection in the UI.

- On query():

  If the entity is mapped to a country, then the country display value will be returned.

  If the country value is mapped to "Other", the stored text value is returned.

- On query() by Country:

  The system searches only the country display name.

  If the entity's Country value is not mapped to an Autotask country, then you must query for Country = "Other"; that is, the string "Other", not the stored text value. This will return all entities where the country value is non-standard.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountName	{LT:Account} Name	string (100)		●			
AccountNumber	{LT:Account} Number	string (50)					
AccountType	{LT:Account} Type	short		●		●	
Active	Account Active	boolean					
AdditionalAddressInformation	Additional Address Information	string (100)					
Address1	Address 1	string (128)					
Address2	Address 2	string (128)					
AlternatePhone1	Alternate Phone 1	string (25)					
AlternatePhone2	Alternate Phone 2	string (25)					
AssetValue	Asset Value	double					

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
BillToAdditionalAddressInformation	Bill To Additional Address Information	string (100)					
BillToAddress1	Bill To Address Line 1	string (150)					
BillToAddress2	Bill To Address Line 2	string (150)					
BillToAddressToUse	Bill To Address to Use	integer	●			●	
BillToAttention	Bill To Attention	string (50)					
BillToCity	Bill To City	string (50)					
BillToCountryID	Bill To Country ID	integer			Country		
BillToState	Bill To {LT:State}	string (128)					
BillToZipCode	Bill To {LT:ZipCode}	string (50)					
City	City	string (30)					
ClientPortalActive	Client Portal Active	boolean	●				
CompetitorID	Competitor	integer				●	
Country	Country	string (100)					
CountryID	Country ID	integer			Country		
CreateDate	Create Date	datetime	●				
Fax	Fax	string (25)					
id	id	long	●	●			
InvoiceMethod	Transmission Method	integer				●	
InvoiceNonContractItemsToParentAccount	Invoice non contract items to Parent {LT:Account}	boolean					
KeyAccountIcon	Key Account Icon	integer				●	

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
LastActivityDate	Last Activity Date	datetime	●				
MarketSegmentID	Market Segment	integer				●	
OwnerResourceID	{LT:Account} Owner	integer		●	Resource		
ParentAccountID	{LT:ParentAccount}	integer			Account		
Phone	Phone	string (25)		●			
PostalCode	{LT:ZipCode}	string (10)					
SICCode	SIC Code	string (32)					
State	{LT:State}	string (40)					
StockMarket	Stock Market	string (10)					
StockSymbol	Stock Symbol	string (10)					
TaskFireActive	TaskFire Active	boolean	●				
TaxExempt	Tax Exempt	boolean					
TaxID	Tax ID	string (50)					
TaxRegionID	Tax Region ID	integer			TaxRegion		
TerritoryID	Territory Name	integer				●	
WebAddress	Web	string (255)					

AccountLocation

The AccountLocation entity takes on the UDFs that hold the site setup information for the account represented by AccountId.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	AccountLocation
Can Create:	
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	•

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

Conditions and Requirements

NOTE: Site Configuration UDFs are associated with the AccountLocation entity. In Autotask, required Site Configuration UDFs are required for *Customer type* accounts only. The Web Services API, however, will show all Site Configuration UDFs for all Account Locations as not required until an Update is performed. When you perform an Update, the Web Service API will perform a check and, for Account Locations where Account.Type = Customer, any required Site Configuration UDF will be required. If no data is provided for a required UDF, Web Services cannot complete the update.

- AccountID, id, and LocationName are Read Only; therefore, update() is allowed only when the entity has User-defined Fields that allow update.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountID	{LT:Account} ID	integer	•	•	Account		
id	id	long	•	•			
LocationName	{LT:Account} Name	string (100)	•				

AccountNote

The AccountNote entity describes any sort of note created by an Autotask user and associated with an Account entity. Autotask users manage Account Notes through the CRM module (CRM > Accounts).

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: AccountNote
Can Create: •
Can Update: •
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountID	{LT:Account}	integer		•	Account		
ActionType	TypeValue	integer		•		•	
AssignedResourceID	Assigned Resource	integer		•	Resource		
CompletedDateTime	DateCompleted	datetime	•				
ContactID	Contact	integer			Contact		
EndDateTime	EndDate	datetime		•			
id	id	long	•	•			
LastModifiedDate	DateStamp	datetime	•				
Name	Name	string (128)					

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Note	Detail	string (32000)					
OpportunityID	ProposalID	integer			Opportunity		
StartDateTime	StartDate	datetime		•			

AccountToDo

This entity describes a To-Do created by an Autotask user and associated with an Autotask Account. In Autotask, a To-Do is a scheduled item that appears on the user's Autotask calendar and can be assigned to another resource. It requires an Action Type and can be associated with an Opportunity, Ticket, or Contract that is associated with the same Account as the To-Do. Autotask users manage Account To Dos through the CRM module (CRM > To-Dos).

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: AccountToDo
Can Create: •
Can Update: •
Can Query: •
Can Delete: •
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- ContactID, ContractID, OpportunityID, and TicketID must reference a Contact, Contract, Opportunity, or Ticket entity with the same AccountID as the AccountToDo.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountID	{LT:Account}	long		•	Account		
ActionType	Action Type	integer		•		•	
ActivityDescription	Description	string (3200)					
AssignedToResourceID	Assigned To Resource	long		•	Resource		

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
CompletedDate	Completed Date	datetime					
ContactID	Contact	long			Contact		
ContractID	Contract	long			Contract		
CreateDateTime	Create Date Time	datetime	●				
CreatorResourceID	Creator Resource	long	●		Resource		
EndDateTime	End Date Time	datetime		●			
id	{LT:Account} To Do ID	long	●	●			
LastModifiedDate	Last Modified Date	datetime	●				
OpportunityID	Opportunity	long			Opportunity		
StartDateTime	Start Date Time	datetime		●			
TicketID	Ticket	long			Ticket		

AccountTeam

This entity describes the resources associated with an Account Team. In Autotask, the account team associates resources with an account. The resources then have access to the account data when their security level allows account access at the Mine level.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: AccountTeam

Can Create: •

Can Update: •

Can Query: •

Can Delete: •

Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountID	{LT:Account}	long		•	Account		
id	{LT:Account} Team ID	long	•	•			
ResourceID	Resource	long		•			

ActionType

This entity describes an Action Type assigned to a CRM Note or To-Do. The Action Type specifies the type of activity scheduled by the to-do or associated with the note. The View associated with the Action Type controls where in Autotask the user can view the note or to-do.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: ActionType
Can Create: •
Can Update: •
Can Query: •
Can Delete: •
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Active	Active	boolean		•			
id	ActionType ID	long	•	•			
Name	Name	string (32)		•			
SystemActionType	System Action Type	boolean	•				
View	View	integer		•		•	

AdditionalInvoiceFieldValue

This entity describes the values for custom Additional Invoice Fields that users can add to Autotask for use with Autotask Invoice Templates. Autotask invoice templates define the appearance and content of an invoice generated in Autotask. Users can add Additional Invoice Fields as variables to the template. The user can then specify a value for the field(s) when one or more invoices are processed and that value will apply to all invoices for that session.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: AdditionalInvoiceFieldValue
Can Create:
Can Update:
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AdditionalInvoiceFieldID	Additional Invoice Field ID	long	•	•	Account	•	
FieldValue	Field Value	string (100)	•	•			
id	Additional Invoice Field Value ID	long	•	•			
InvoiceBatchID	Batch ID	long	•	•			

AllocationCode

The AllocationCode entity describes an Autotask Allocation Code. An allocation code represents a billing item category. There are six types of allocation codes: Work Types, Material, Internal Time, Expense Categories, Recurring Contract Service, and Milestone. Administrators add and manage allocation codes through the Admin module (Admin >Site Setup > Billing Codes).

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	AllocationCode
Can Create:	
Can Update:	
Can Query:	•
Can Delete:	
Can Have UDFs:	

Fields that Cannot Be Queried

The following fields from this entity will return an error when queried.

- Taxable

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

NOTE: As of May 2013, AllocationCode no longer determines whether or not an item is taxable. A QuoteItem or BillingItem is taxable when the item is associated with both a TaxRegion and TaxCategory and they equal the TaxRegion and TaxCategory fields of a Tax entity with a value > 0. Values passed to AllocationCode.Taxable will be ignored and when the field is queried the empty value "" will be returned.

NOTE: To accommodate changes in the Autotask database and UI while maintaining compatibility within the API, Product allocation code, type 7, now maps to Material (cost) allocation code, type 4. Pre-existing type 7 codes have been added to type 4 codes because Product allocationcodes have been eliminated from the database. Existing queries for type 7 will be redirected to type 4.

Conditions and Requirements

- GeneralLedgerCode is now a pick list. Previously the GeneralLedgerCode field returned only the GeneralLedgerCode id. The API did not provide an option to connect the id with the name that appears in the UI drop down menu. The getFieldInfo() call will now return the GeneralLedgerCode picklist value and label value (GeneralLedgerCode name).

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Active	Active	boolean		●			
AllocationCodeType	Allocation Code Type	integer				●	
Department	Department ID	integer					
Description	Description	string (500)					
ExternalNumber	Number	string (100)					
GeneralLedgerCode	General Ledger Code	integer				●	
id	id	long	●	●			
Name	Name	string (200)					
Taxable	Taxable	boolean					
TaxCategoryID	Tax Category ID	integer			TaxCategory	●	
Type	Type	integer				●	
UnitCost	Unit Cost	double		●			
UnitPrice	Unit Price	double		●			
UseType	Use Type	integer				●	

AllocationCode Use Types

The following table describes the values in the UseType field picklist. These values determine how an AllocationCode entity can be used in Autotask.

NOTE: If your company has enabled the Autotask Workflow Policy that requires a Work Type on a Ticket, the Type 1 AllocationCode is required to create or update a ticket

UseType	Used for:	UI Location
1	Tickets (Labor)	Admin > Site Setup > Billing Codes > Work Types tab
2	Expenses	Admin >Site Setup > Billing Codes > Expense Categories tab

UseType	Used for:	UI Location
3	Regular (Internal) Time	Admin > Site Setup > Billing Codes > Internal Time tab
4	Material (Ticket, Project, and Contracts Costs)	Admin > Site Setup > Billing Codes > Material tab
5	Services	Admin > Site Setup > Billing Codes > Recurring Contract Service tab
6	Milestones	Admin > Site Setup > Billing Codes > Milestone Codes
7	Products	To accommodate changes in the Autotask UI while maintaining data consistency within the API, Product allocation code, type 7, now maps to Material allocation code, type 4. Existing type 7 codes have been moved to type 4. When querying for product cost codes (type 7), query for type 4.

Appointment

The Appointment entity describes an Autotask Appointment, that is, scheduled time that is not a service call. Autotask users manage appointments through their calendar. In addition, Dispatchers can view and schedule appointments through the Dispatcher's Workshop.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: Appointment
Can Create: •
Can Update: •
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
CreateDateTime	Create Date	datetime	•				
CreatorResourceID	Created By	integer	•		Resource		
Description	Description	string (8000)					
EndDateTime	End Date	datetime		•			
id	id	long	•	•			
ResourceID	Resource	integer		•	Resource		
StartDateTime	Start Date	datetime		•			
Title	Appointment Title	string (256)		•			
UpdateDateTime	Update Date	datetime	•				

AttachmentInfo

The AttachmentInfo entity describes an Attachment in Autotask. Attachments are external documents that are associated with an Autotask Account, Ticket, Project, or Opportunity entity. Attachments in Autotask can be documents uploaded to the server, file links, folder links, and URLs.

NOTE: This entity is unique. It does not use the standard create(), update()and delete() API calls. Instead, you must use one of the following three calls: **GetAttachment()**, **CreateAttachment()**, and **DeleteAttachment()**. Refer to "Attachment API Calls" on page 230.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	AttachmentInfo
Can Create:	
Can Update:	
Can Query:	•
Can Delete:	
Can Have UDF:	

Fields that Cannot Be Queried

The following fields from this entity will return an error when queried.

- FileSize

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

Conditions and Requirements:

- The API size limit for individual attachment files is 6-7 MB.

  NOTE: This is less than the 10 MB limit for individual attachment files when working through the Autotask UI.

- File types are limited to the file types allowed in Autotask. For information, see "Adding Attachments" in the online Help. The maximum file size indicated in the Help does not apply to the API.
- To add, download, and delete attachments with the API, you must use the three attachment specific API

calls: CreateAttachment(), GetAttachment(), and DeleteAttachment().

- All AttachmentInfo entity types except for Opportunity type must specify a TypeID and ParentID

  Opportunity type attachments must specify a either a Type ID and ParentID, or TypeID and OpportunityID, or TypeID and both ParentID and OpportunityID.

  OpportunityID value must be provided in order to query by OpportunityID.
- To specify the associated entity, use Parent Type. Use the "getFieldInfo()" on page 244 API call to determine ParentType picklist values.
- To specify the type of attachment, use the Type field. Use the "getFieldInfo()" on page 244 API call to determine Type picklist values.
- To specify whether the attachment should be available to all Autotask users, including Client Portal users, or internal users only, use the Publish field. Use the "getFieldInfo()" on page 244 API call to determine Publish picklist values.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AttachDate	Attach Date	datetime	●				
AttachedByContactID	Attached By Contact	long	●		Contact		
AttachedByResourceID	Attached By Resource	long	●		Resource		
ContentType	Content Type	string (100)	●				
FileSize	File Size	long	●				
FullPath	File Name	string (255)	●	●			
id	Attachment ID	long	●	●			
OpportunityID	Opportunity ID	long			Opportunity		
ParentID	Parent ID	long	●	●			
ParentType	Parent Type	integer	●	●		●	
Publish	Publish	integer	●	●			
Title	Title	string (255)	●	●			
Type	Type	string (30)	●	●		●	

BillingItem

The BillingItem entity describes a billable item in Autotask that has been approved and posted. A billing item may or may not be included in an invoice and billed to the customer. In Autotask, BillingItems are managed in the Contracts module.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	BillingItem
Can Create:	
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Fields that Cannot Be Queried

The following fields from this entity will return an error when queried.

LineItemFullDescription

LineItemGroupDescription

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

NOTE: The BillingItem entity includes the WebServiceDate field that allows the external application to flag the date and time that the object is processed. This field is provided as a convenience for applications that work specifically with the BillingItem entity via the Web Services API. It does not appear in the Autotask interface.

Conditions and Requirements

- BillingItem.LineItemFullDescription contains the description for a BillingItem.
- When BillingItems are *grouped* on an invoice, the LineItemGroupDescription contains the description for the grouped items. LineItemFullDescription is also available for BillingItems that are grouped.
- When mulltiple BillingItems that are associated with the same InvoiceID share the same LineItemID, those items are grouped.

  It is possible for a BillingItem to have a unique LineItemID and still be "grouped" in a group of one item. To confirm the group status of a BillingItem, refer to Invoice.InvoiceEditorTemplateID and the InvoiceTemplate.GroupBy value.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountID	{LT:Account}	integer	●		Account		
AllocationCodeID	AllocationCodeID	integer	●		AllocationCode		
ApprovedTime	ApprovedDateTime	datetime	●				
ContractCostID	Contract Cost	long	●		ContractCost		
ContractID	ContractID	integer	●		Contract		
Description	Description	string (2000)	●				
ExpenseItemID	Expense Item	integer	●				
ExtendedPrice	Extended Price	double	●				
id	id	long	●	●			
InvoiceID	InvoiceID	integer	●		Invoice		
ItemApproverID	ItemApproverID	integer	●		Resource		
ItemDate	ItemDate	datetime	●				
ItemName	ItemName	string (255)	●				
LineItemFullDescription	Line Item Full Description	string (8000)	●				
LineItemGroupDescription	Line Item Group Description	string (8000)	●				
LineItemID	Invoice Line Item ID	long	●				
MilestoneID	Milestone ID	long	●		ContractMilestone		
NonBillable	NonBillable	integer	●	●			
OurCost	OurCost	double	●				
ProjectCostID	Project Cost	long	●		ProjectCost		
ProjectID	ProjectID	integer	●		Project		

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
PurchaseOrderNumber	Purchase Order Number	string (32)	●				
Quantity	Quantity	double	●				
Rate	Rate	double	●				
RoleID	RoleID	integer	●		Role		
ServiceBundleID	Service Bundle ID	long	●		Service Bundle		
ServiceID	Service ID	long	●		Service		
SubType	Sub Type	Integer	●	●		●	
TaskID	TaskID	integer	●		Task		
TaxDollars	TaxDollars	double	●				
TicketCostID	Ticket Cost	long	●		TicketCost		
TicketID	TicketID	integer	●		Ticket		
TimeEntryID	TimeEntryID	integer	●		TimeEntry		
TotalAmount	TotalAmount	double	●				
Type	Type	integer	●	●		●	
VendorID	Vendor ID	long	●		Account		
WebServiceDate	WebServiceDate	datetime					

BillingItemApprovalLevel

The BillingItemApprovalLevel entity describes a multi-level approval record for an Autotask time entry. It allows developers to use the API to implement multi-tier approval for Autotask time entries. This option is only available through the Web Services API.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: BillingItemApprovalLevel

Can Create: ●

Can Update:

Can Query: ●

Can Delete:

Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

Conditions and Requirements

- TimeEntryID and ApprovalResourceID must reference existing objects.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ApprovalDateTime	Approval Date Time	datetime		●			
ApprovalLevel	Approval Level	integer		●			
ApprovalResourceID	Approval Resource ID	integer		●	Resource		
id	id	integer	●	●			
TimeEntryID	Time Entry ID	integer		●	TimeEntry		

ChangeRequestLink

This entity describes the associations between Change Request tickets and both Incidents and Problems. Change request tickets are part of the Autotask change Management feature set. Change management features are only available in the Autotask UI when the Change Management module is enabled.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ChangeRequestLink
Can Create:	•
Can Update:	
Can Query:	•
Can Delete:	•
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

Conditions and Requirements

- ChangeRequestTicketID must be a valid ID for a ticket with TicketType = Change Request.
- ProblemOrIncidentTicketID must be a valid ID for a ticket with TicketType = Problem ticket or Incident ticket
- A Change Request ticket cannot be associated with a ticket with Ticket Type = Service Request.
- A Change Request ticket cannot be associated with another ticket with TicketType = Change Request.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ChangeRequestTicketID	Change Request Ticket ID	integer		•	Ticket		
id	Change Request Problem Incident ID	integer	•	•			
ProblemOrIncidentTicketID	Problem Or Incident Ticket ID	integer		•			

ClientPortalUser

This entity describes an Autotask Account Contact that has been assigned access to the Client Access Portal. In the Autotask Interface, Account Contacts are enabled as Client Portal Users in Admin > Client Access Portal > Manage Accounts > select Account > "Manage users for this account" or from the Client Portal tab of the Edit Contact window.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ClientPortalUser
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Fields that Cannot Be Queried

The following fields from this entity will return an error when queried.

- Password

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- To determine Taskfire users through the APi, query for ClientPortalUser.SecurityLevel = 4 and 5

  There are two Client Portal user security levels that allow access to Taskfire features: Resource (value = 4) and Administrator (value = 5). Query the ClientPortalUser entity for users with these security levels to return users that have access to Taskfire.

- For Password field

  Required on create()

  Not required on update() performed on existing ClientPortalUser. If password is set, then we will store in DB.

  Query will never return the Password field as part of the entity returned (it will always be null).

  To update for an existing entity, set the Password field to the new password to be stored on update().

Must enforce the following restrictions: 1) alpha-numeric characters only or contain one of these characters ~ @ # $ * 2) must be between 6-50 characters, 3) cannot contain spaces.

- SecurityLevel refers to Client Access Portal security levels.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ClientPortalActive	Client Portal Active	boolean		●			
ContactID	Contact ID	integer	●	●	Contact		
DateFormat	Date Format	integer		●		●	
id	Client Portal User ID	long	●	●			
NumberFormat	Number Format	integer		●		●	
Password	Password	string (50)					
SecurityLevel	Security Level	integer		●		●	
TimeFormat	Time Format	integer		●		●	
UserName	User Name	string (200)		●			

Contact

The Contact entity describes an Autotask Contact. A contact is an individual associated with an Account. Autotask users manage contacts through the CRM module (CRM > Contacts) or the Directory Module (Directory > Contacts).

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	Contact
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	•

Fields that Can Not Be Queried

Although this entity can be queried, it contains one or more fields that cannot be queried. If you attempt to query these fields, an error is returned.

- BulkEmailOptOut
- SurveyOptOut

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

NOTE: To prevent possible data corruption, the AccountID field is now read only.

Conditions and Requirements

- In Autotask, the Country fields now provide a Country pick list. The following conditions apply to the Country fields in the API.

  NOTE: The Country field is not the same as the CountryID field. The CountryID field references the ID field of the Country entity. If a value is provided for the Country field, the API will use that value and, based on the rules below, map that value to a valid Country entity. If no value is provided for the CountryID field, the ID value for the matching Country entity will automatically populate the CountryID field.

  On create() and update():

  If the value provided for Country matches either a standard country abbreviation, an ISO standard country name, or an Autotask country display name, the value is mapped to that country.

  If no match is found, the value is mapped to "Other". The text that was passed in is stored.

In the UI, "Other" appears in the Country field as "Other [stored text value]". These "Other" values are not available for selection in the UI.

On query():

If the entity is mapped to a country, then the country display value will be returned.

If the country value is mapped to "Other", the stored text value is returned.

On query() by Country:

The system searches only the country display name.

If the entity's Country value is not mapped to an Autotask country, then you must query for Country = "Other"; that is, the string "Other", not the stored text value. This will return all entities where the country value is non-standard.

- The BulkEmailOptOut field indicates whether or not this contact has opted out of receiving bulk emails generated by Autotask. BulkEmailOptOutTime indicates the date that the contact opted out. The CAN-SPAM act specifies 30 days as the minimum time to refrain from sending commercial emails to recipients that have chosen to opt out of bulk email. Since there may be a need to communicate technical or product information to users who have opted out of bulk email, Autotask does not force compliance with the 30 day opt out period.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountID	{LT:Account}	integer	•	•	Account		
Active	Active	integer		•			
AdditionalAddressInformation	Additional Address Information	string (100)					
AddressLine	Address 1	string (128)					
AddressLine1	Address 2	string (128)					
AlternatePhone	Alternate Phone	string (32)					
BulkEmailOptOut	Bulk Email Opt Out	boolean					
BulkEmailOptOutTime	Bulk Email Opt Out Time	datetime	•				
City	City	string (32)					

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Country	Country	string (100)					
CountryID	Contact Country ID	integer			Country		
CreateDate	Create Date	datetime	•				
EMailAddress	Email	string (50)					
Extension	Phone Ext.	string (10)					
ExternalID	External ID	string (50)					
FacebookURL	Facebook URL	string (200)					
FaxNumber	Fax	string (25)					
FirstName	First Name	string (20)		•			
id	id	long	•	•			
LastActivityDate	Last Activity Date	datetime	•				
LastModifiedDate	Last Modified Date	datetime	•				
LastName	Last Name	string (20)		•			
LinkedInURL	LinkedIn URL	string (200)					
MiddleInitial	Middle Initial	string (2)					
MobilePhone	Mobile Phone	string (25)					
NamePrefix	Name Prefix	integer				•	
NameSuffix	Name Suffix	integer				•	
Note	Note	string (20)					
Notification	Notification	boolean					
Phone	Phone	string (25)		•			

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
RoomNumber	Room Number	string (50)					
State	{LT:State}	string (40)					
SurveyOptOut	Survey Opt Out	boolean	•				
Title	Title	string (50)					
TwitterURL	Twitter URL	string (200)					
ZipCode	{LT:ZipCode}	string (16)					

Contract

The Contract entity describes an Autotask Contract. Contracts specify a billing arrangement with an Account. Autotask users manage contracts through the Contracts module. Autotask currently provides six contract types: Time and Materials, Fixed Price, Block Hours, Retainer, Incident, and Recurring Service. When creating a Contract, you must specify a Contract Type (Contract.ContractType).

Several contract types use one or more of the additional Contract related entities that are described in this document in order to create a functional billing arrangement; for example, a Block Hours contract requires the creation of one or more Contract Block entities in order to charge time against the contract. The API also provides two important but optional Contract related entities, "ContractNote" on page 94 and "ContractFactor" on page 91(BlockHour).

NOTE: Please review the Conditional Requirements listed under Field Details before creating or updating Contracts.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	Contract
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- create() and update() are allowed only for active Customer type Accounts (Account.AccountType = 1)
- On create() and update(), Contract.ContactName must be in Firstname Lastname format.
- On create() and update(), Contract.ContactName must match an existing Active Contact with AccountID that matches the Contract AccountID (Contact.AccountID = Contract.AccountID).
- Contract.StartDate must be < Contract.EndDate.
- TimeReportingRequiresStartStopTime allows only 0 or 1.
- BillingPreference field is Required for Time and Materials, Block, Retainer, and Incident Contract types.
- ContractPeriodType field is required for Recurring Service type contracts.
- SetupFee field is Required for Recurring Service type Contracts.

- Recurring Service type Contracts can NOT use the following three fields: EstimatedCost, EstimatedHours, and EstimatedRevenue.
- For all Contracts except Recurring Service, the following fields are Required: EstimatedCost, EstimatedHours, and EstimatedRevenue.
- An associated opportunity must belong to the contract's account or to any child account of the contract's account. That is, Contract.OpportunityID must reference an Opportunity where Opportunity.AccountID either equals Contract.AccountID or references an Account where Account.ParentAccountID equals Contract.AccountID

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountID	{LT:Account}	integer	●	●	Account		
BillingPreference	Billing Preference	integer				●	
Compliance	Contract Compliance	boolean					
ContactName	Contract Contact	string (100)					
ContractCategory	Category	integer				●	
ContractName	Contract Name	string (100)		●			
ContractNumber	Contract Number	string (50)					
ContractPeriodType	Contract Period Type	string (1)	●			●	
ContractType	Contract Type	integer	●	●		●	
Description	Description	string (2000)					
EndDate	End Date	datetime		●			
EstimatedCost	Estimated Cost	integer					
EstimatedHours	Estimated Hours	double					
EstimatedRevenue	Estimated Revenue	double					
id	id	long	●	●			
IsDefaultContract	Default Contract	boolean					

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
OpportunityID	opportunity_id	integer			Opportunity		
OverageBillingRate	Contract Overage Billing Rate	double					
PurchaseOrderNumber	Purchase Order Number	string (32)					
ServiceLevelAgreementID	Service Level Agreement ID	integer				•	
SetupFee	Contract Setup Fee	double					
SetupFeeAllocationCodeID	Contract Setup Fee Allocation Code ID	long					
StartDate	Start Date	datetime		•			
Status	Status	integer		•		•	
TimeReportingRequiresStartAndStopTimes	Time Reporting Requires Start and Stop Times	integer		•		•	

ContractBlock

The ContractBlock entity describes an Autotask Contract Block. The Contract Block represents a block of hours purchased for a Block Hours type Contract. With a Block Hours Contract, the customer pre-pays for a block of hours and then the pre-paid hours are reduced as billable work is performed. The contract maintains the balance of hours. You can create a Block Hours type Contract without creating an associated ContractBlock entity, but time cannot be charged against the contract.

NOTE: A Contract Block specifies a single per hour rate. Autotask Block Hour Factors accommodate work performed against the contract by resources with different role rates. The "ContractFactor" on page 91 entity describes a block hour factor.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ContractBlock
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Fields that Cannot Be Queried

The following fields from this entity will return an error when queried.

- HoursApproved

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

NOTE: The API does not currently support automatic cost creation when creating a contract block. To create contract costs automatically the contract blocks must be added through the User Interface.

Conditions and Requirements

- ContractBlock.ContractID must reference a Contract where ContractType is Blocks Hour type.
- ContractBlock.StartDate must be < ContractBlock.EndDate.
- On update(), ContractBlock.Hours must be >= ContractBlock.HoursApproved.
- ContractBlock.Hours and ContractBlock.HourlyRate must be >= 0.
- ContractBlock.StartDate and ContractBlock.EndDate must be within the StartDate and EndDate of the Contract referenced by ContractBlock.ContractID; that is, you cannot create a ContractBlock outside of

the Contract dates.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ContractID	Contract ID	integer		●	Contract		
DatePurchased	Date Purchased	datetime		●			
EndDate	End Date	datetime		●			
HourlyRate	Rate	double		●			
Hours	Hours	double		●			
HoursApproved	Hours Approved	double	●				
id	id	long	●	●			
InvoiceNumber	Invoice Number	string (50)					
IsPaid	Paid	string (10)		●			
PaymentNumber	Payment Number	string (50)					
PaymentType	Payment Type	integer				●	
StartDate	StartDate	datetime		●			

ContractCost

This entity describes a cost associated with an Autotask Contract. A cost is a billing item for products or materials. Cost items can be billable or non-billable. Billable cost items appear in Approve and Post.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ContractCost
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	• see conditions below
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

Conditions and Requirements

- create(), update(), delete(), and query() require Security Level permission to access the Contracts module.
- ContractCost must have either a ProductID or AllocationCodeID.
- ContractCost cannot be created, deleted, or queried if ContractID references a Block, Retainer, or Incident type Contract unless AllocationCodeID references a Block Purchase, Incident Purchase, or Retainer Purchase Material Cost Code; then, only the following fields can be updated: Name, Description, CostType, PurchaseOrderNumber, InternalPurchaseOrderNumber.
- After a ContractCost is created, ContractID cannot be updated.
- update() and delete() are allowed only when Billed = False (cost has not been approved and posted).
- Only query() is allowed if Billed = True (cost has been approved and posted).
- AllocationCodeID must reference a Material Cost Code type allocation code.
- On create(), or update() when ProductID has changed and is not Null, if no value is supplied for the AllocationCodeID, UnitCost, or UnitPrice fields, then those fields take their values from the Product.
- On create() or update() when AllocationCodeID has changed and is not Null:

  If ContractID references a Contract that has set a Contract Cost Default for the Material Cost Code associated with the AllocationCodeID, and no value was supplied for UnitPrice, UnitCost, or BillableToAccount, then these fields take their values from the Contract Cost Default.

If there is no Contract Cost Default set for the Material Cost Code associated with the AllocationCodeID, and if no value is supplied for the UnitCost and UnitPrice fields, then those fields take their values from the Material Cost Code.

- If no value is specified for BillableToAccount, the value is set to True
- Status is read only. On create(), Status is set as follows:

  If ProductID is null, then Status = Ready to Ship.

  If ProductID references a Product where DoesNotRequireProcurement = True, then Status = Ready to Ship.

  If ProductID references a Product where DoesNotRequireProcurement = False and the Product is not available in Inventory, then Status = Need To Order.

  If ProductID references a Product where DoesNotRequireProcurement = False and the Product is available in Inventory, then Status = Ready to Ship.

  If Status = Need To Order and ExtendedCost is > the value set in the workflow policy "Require approval before ordering..." then Status is automatically set to Waiting Approval.
- ExtendedCost = UnitQuantity * UnitCost. If UnitCost is Null, the value will be set to the value from the Material Cost Code associated with AllocationCodeID.
- BillableAmount = UnitQuantity * UnitPrice. If UnitPrice is Null, the value will be set to the value from the Material Cost Code associated with AllocationCodeID.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AllocationCodeID	Allocation Code	long			AllocationCode		
BillableAmount	Billable Amount	double	•				
BillableToAccount	Billable To Account	Boolean					
Billed	Billed	Boolean	•				
ContractID	Contract	long		•	Contract		
ContractServiceBundleID	Contract Service Bundle ID	long			ContractServiceBundle		
ContractServiceID	Contract Service ID	long			ContractService		

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
CostType	Cost Type	integer		●		●	
CreateDate	Create Date	datetime	●				
CreatorResourceID	Created By	long	●		Resource		
DatePurchased	Date Purchased	datetime		●			
Description	Product Description	string (2000)					
ExtendedCost	Extended Cost	double	●				
id	id	long	●	●			
InternalPurchaseOrderNumber	Internal Purchase Order Number	string (50)					
Name	Name	string (100)		●			
ProductID	Product	long			Product		
PurchaseOrderNumber	Purchase Order Number	string (32)					
Status	Status	long	●			●	
StatusLastModifiedBy	Last Modified By	long	●				
StatusLastModifiedDate	Last Modified Date	datetime	●				
UnitCost	Unit Cost	double					
UnitPrice	Unit Price	double					
UnitQuantity	Unit Quantity	double		●			

ContractFactor

The ContractFactor entity describes an Autotask Block Hour Factor, an option used with Block Hours Contracts. With a Block Hours Contract, the customer pre-pays for a block of hours at a fixed hourly rate. A Block Hour Factor can compensate for the fixed rate by applying a multiplier to specific role rates. For example, if the contract hourly rate is $60, but an Engineer's hourly billing rate is $120, applying a block hour factor of 2 will debit 2 hours of contract time for each hour worked by an Engineer. In Autotask, you create and manage Block Hour Factors from a Block Hours Contract's Summary page in the Contract module.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ContractFactor
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- Each RoleID/ContractID combination can have only one ContractFactor; for example, you cannot have TWO factors for the "Support Person" role on one contract.
- ContractFactor.ContractID must reference a Contract where Contract.ContractType is Block Hour type.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
BlockHourFactor	Hourly Offset	double		•			
ContractID	Contract ID	integer	•	•	Contract		
id	id	long	•	•			
RoleID	Role ID	integer		•	Role		

ContractMilestone

The ContractMilestone entity describes a billing milestone for an Autotask Fixed Price Contract. Billing milestones describe tangible work or measured progress that must be completed before billing can take place, for example, "user approval on design specifications", or "ten workstations installed". A fixed price contract can have multiple billing milestones. In Autotask, you add and manage contract milestones from the Fixed Price Contract's Summary page.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ContractMilestone
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

Conditions and Requirements

- ContractMilestone.AllocationCodeID must reference a Milestone type Allocation Code.
- ContractMilestone.ContractID must reference a Contract where ContractType is Fixed Price.
- If ContractMilestone.IsInitialPayment = False, ContractMilestone.Status cannot = Billed.
- If ContractMilestone.IsInitialPayment field = True, allow one Contract Milestone with status of "Billed" and one with status "Ready to Bill" for this Contract. That is, in Autotask, when creating a Fixed Price contract, you can specify whether the contract includes an initial payment. If yes, you can then optionally specify an amount received and/or an amount to be billed.

  For each amount specified, a ContractMilestone is automatically created. An amount received automatically creates a ContractMilestone with status "Billed"; an amount to be billed automatically creates a ContractMilestone with status "Ready to Bill".
- If ContractMilestone.Status = Billed, the field cannot be updated.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AllocationCodeID	Allocation Code ID	integer			AllocationCode		
Amount	Amount	double		●			
ContractID	Contract ID	integer	●	●	Contract		
CreateDate	Date Created	datetime	●				
CreatorResourceID	Created By ID	integer	●		Resource		
DateDue	Date Due	datetime		●			
Description	Description	string (250)					
id	id	long	●	●			
IsInitialPayment	Initial Payment	boolean		●			
Status	Status	integer		●		●	
Title	Name	string (50)		●			

ContractNote

This entity describes a note associated with an Autotask Contract. Notes are used to track information, update the status of the associated contract, and communicate with resources and customers. In Autotask, you create and manage Contract Notes from the Options or Notes features found on the Contract Summary page in the Contract module.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ContractNote
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ContractID	Contract ID	string (25)	•	•	Contract		
CreatorResourceID	Resource ID	integer	•		Resource		
Description	Journal	string (8000)		•			
id	id	integer	•	•			
LastActivityDate	Date Added	datetime	•				
Title	Title	string (250)		•			

ContractRate

This entity describes a billing rate for an Autotask Contract. A Contract Rate is associated with a Role and is specific to a contract, allowing you to override your company's standard role rates for labor tracked against the contract. Rates are used with Time and Materials, Fixed Price, and Retainer type contracts. In Autotask, you add and manage Contract Rates from the Contract Summary page.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ContractRate
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- Each RoleID/ContractID combination can have only one ContractRate; for example, you cannot have TWO rates for the "Support Person" role on one contract.
- ContractRate.ContractID cannot reference a Contract where Contract.ContractType is Recurring Service or Block Hour. It can reference any other Contract type, that is, Time and Materials, Fixed Price, or Retainer.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ContractHourlyRate	Rate Offset	double		•			
ContractID	Contract ID	integer	•	•	Contract		
id	id	long	•	•			
RoleID	Role ID	integer		•	Role		

ContractRetainer

This entity describes a payment amount applied to a Retainer type contract and sets the time period covered by the purchase. For example, a retainer purchase of $500 can be applied to three months of the Retainer contract.

The retainer balance is reduced by services performed at the contract rates, which can be the same or different from the standard billing rates. In addition, Project and Ticket Costs can be deducted from a Retainer contract. The contract maintains the balance of dollars.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ContractRetainer
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Fields that Cannot Be Queried

The following fields from this entity will return an error when queried.

- AmountApproved

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- For versions previous to v 1.5: when IsPaid = true (indicating that the retainer purchase has been paid for) the user program must include a ContractRetainer DatePaid.
- StartDate must be < EndDate
- ContractRetainer.StartDate and ContractRetainer.EndDate must be between the StartDate and EndDate of the contract referenced by ContractRetainer.ContractID; that is, in Autotask the Retainer purchase must start and end within the Contract start and end dates.
- ContractRetainer.ContractID must reference a Retainer type Contract.
- For update(), ContractRetainer.Amount must be >= Amount Approved.
- You cannot change IsPaid if ContractRetainer is associated to a ContractCost.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Amount	Amount	double		●			
AmountApproved	Amount Approved	double	●				
ContractID	Contract Object ID	integer	●	●	Contract		
DatePaid*	Date Paid	datetime					
DatePurchased	Date Purchased	datetime		●			
EndDate	End Date	datetime		●			
id	id	integer	●	●			
InvoiceNumber	Invoice Number	string (50)					
IsPaid	Paid	integer		●		●	
paymentID	Payment Type	integer				●	
PaymentNumber	Payment Number	string (50)					
StartDate	StartDate	datetime		●			
Status	Status	integer		●		●	

* DatePaid was removed in API Web Services version 1.5

Department

This entity describes an Autotask Department. A Department is an association used to manage resources, especially when assigning project tasks. Every resource must be associated with at least one department and have at least one assigned role in that department.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: Department
Can Create:
Can Update:
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Description	Description	string (1000)	•				
id	Department ID	long	•	•			
Name	Name	string (100)	•	•			
Number	Number	string (50)	•				
PrimaryLocationID	Primary Location ID	long	•	•	Internal Location		

About Recurring Service Contract Entity Relationships

A Recurring Service type contract can have one or more associated Services (ContractService) and/or Service Bundles (ContractServiceBundle). The Web Services API presents four additional entities related to ContractService or ContractServiceBundle entities: ContractServiceAdjustment, ContractServiceBundleAdjustment, ContractServiceUnit, and ContractServiceBundleUnit.

NOTE: The ContractService and ContractServiceBundle entities perform the same function for Services and Service Bundles associated with a contract. This is true for ContractServiceAdjustments and ContractServiceBundleAdjustments, and for ContractServiceUnits and ContractServiceBundleUnits.

The ContractService entity describes the association between an Autotask Service and a Recurring Service type Contract. It can specify an adjusted price for the service for that contract.

For example, the Autotask implementation has a service named Desktop Maintenance with a standard monthly cost of $20 per unit. A Desktop Maintenance service unit covers one workstation. When Desktop Maintenance is added to a Recurring Service Contract, the ContractService entity associates the Desktop Maintenance service with the contract and, if applicable, specifies an adjusted price for that contract. It does not specify how many units of the service are included in the contract.

NOTE: The ContractService entity is not required to add a Service to a Recurring Service contract. When the Web Services API creates a ContractServiceAdjustment, it looks for the appropriate ContractService and, if it does not find it, the API creates one.

Once the Desktop Maintenance service is associated with a Recurring Service contract, the ContractServiceAdjustment entity describes how many units of the Desktop Maintenance service should be added to or removed from the Recurring Service contract and specifies an effective date. There can be multiple ContractServiceAdjustments for a ContractService entity over the life of the contract as units are added or removed from the contract.

For example, if the Customer wants Desktop Maintenance for each of their ten workstations, a ContractServiceAdjustment sets the number of Desktop Maintenance service units for the Recurring Service contract to ten, effective at the contract start date. If the customer later wants to cover two more workstations, the ContractServiceAdjustment entity specifies an adjustment of plus two units, and specifies the date that the service becomes effective for the two additional workstations.

A ContractServiceUnit entity returns the information needed for billing for services added to a recurring service contract. It describes how many units of a Service were associated with a contract for a specific date range. The start and end dates for the date range represent one of two events: a contract period type break (for example, a Monthly type contract period has a period break date at the end of each month) or an adjustment to the number of units associated with the contract.

For example, the Customer's Recurring Service contract has a Period type of monthly and begins on January 1. The Desktop Monthly service is associated with the contract and has ten units of the service added with an effective date equal to the contract start date. One ServiceContractUnit with ten Desktop Maintenance service units is generated for January 1 through January 31, another for February 1 through February 28, and so on for the duration of the contract OR until the number of units changes.

If two additional units are added with an effective date of March 1, the only change for the ContractServiceUnit for March 1 through March 31 will be the number of units. If the effective date is within the month, for example, on March 10, one ContractServiceUnit is generated with a date range of March 1 through March 9 for ten units

of the service. Another ServiceContractUnit is generated for March 10 through March 31 for twelve units of the service. The next ServiceContractUnit will represent April 1 through April 30 with twelve units of service, unless the number of units changes in April.

ContractService

This entity describes an Autotask Service that has been added to a Recurring Service type contract. In Autotask, users add and manage Autotask Services from the Admin module: Products and Services > Services > Services. You add Services to a Contract when your create the contract or from the Contract Summary page.

The entity specifies an AdjustedPrice if applicable. It does not specify the number of units of the service that have been added to the contract. To add Contract Service units to the contract, or remove units, use the ContractServiceAdjustment entity.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ContractService
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- On Update(), all fields are Read Only except for AdjustedPrice and InvoiceDescription.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AdjustedPrice	Adjusted Price	double					
ContractID	Contract ID	integer	•	•	Contract		
id	id	long	•	•			
InvoiceDescription	Invoice Description (1000)	string					
ServiceID	Service ID	integer	•	•	Service	•	
UnitPrice	Unit Price	double	•				

ContractServiceAdjustment

This entity describes an adjustment to the quantity of units of a ContractService entity that are added to a Recurring Service type Contract. It can only be created; it CANNOT be queried or updated. Changes made to the Contract using the ContractServiceAdjustment entity affect only the quantity of units and not price/adjusted price of the Service.

In Autotask, users specify the number of Service Units associated with a Contract through the Contracts module when creating or managing a Recurring Service type Contract: Contracts module > Contracts > New Recurring Service Contract, or Contracts module > Contracts > Search Contracts > Open Contract > Services.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ContractServiceAdjustment
Can Create:	•
Can Update:	
Can Query:	
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- The ContractServiceAdjustment can be created without the specified ContractService entity in place. If the API creates a ContractServiceAdjustment entity and the API cannot find the correct ContractService entity, it will create it.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AdjustedUnitCost	Adjusted Unit Cost	double					
AdjustedUnitPrice	Adjusted Unit Price	double					
ContractID	Contract ID	integer		•	Contract		
EffectiveDate	Start Date	datetime		•			
id	id	long	•	•			
ServiceID	Service ID	integer		•	Service	•	
UnitChange	Unit Change	integer		•			

ContractServiceBundle

This entity describes an Autotask Service Bundle added to a Recurring Service type contract. Autotask Service Bundles group multiple Services for purchase. In Autotask, Administrators create and manage Autotask Service Bundles from the Admin module: Products and Services > Services > Service Bundles. Service Bundles are added to a Contract when the contract is create or through the Contract Summary view.

The entity specifies an AdjustedPrice where applicable. It does not specify the number of units of the service bundle that have been added to the contract. To add ContractServiceBundle units to the contract, or remove units, use the ContractServiceBundleAdjustment entity.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ContractServiceBundle
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- On update(), all fields are Read Only except for AdjustedPrice and InvoiceDescription.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AdjustedPrice	Adjusted Price	double					
ContractID	Contract ID	integer	•	•	Contract		
id	Contract Service Bundle ID	long	•	•			
InvoiceDescription	Invoice Description (1000)	string					
ServiceBundleID	Service Bundle ID	integer	•	•	ServiceBundle	•	
UnitPrice	Unit Price	double					

ContractServiceBundleAdjustment

This entity describes an adjustment to the quantity of units of a ServiceBundle that are added to a Recurring Service type Contract. It can only be created; it cannot be queried or updated. Changes made to the Contract using the ContractServiceBundleAdjustment entity affect only the quantity of Service Bundle units and not the price/adjusted price of the bundle or services.

In Autotask, users specify the number of Service Bundle units that are associated with a Contract through the Contracts module when creating or managing a Recurring Service type Contract: Contracts module > Contracts > New Recurring Service Contract, or Contracts module > Contracts > Search Contracts > Open Contract > Services.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: ContractServiceBundleAdjustment
Can Create: •
Can Update:
Can Query:
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- ContractServiceBundleAdjustment can be created without the specified ContractServiceBundle entity in place. If the API creates a ContractServiceBundleAdjustment entity and the API cannot find the correct ContractServiceBundle, it will create the ContractServiceBundle entity.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AdjustedUnitPrice	Adjusted Unit Price	double					
ContractID	Contract ID	integer		•	Contract		
EffectiveDate	Start Date	datetime		•			
id	id	long	•	•			
ServiceBundleID	Service Bundle ID	integer		•	ServiceBundle	•	

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
UnitChange	Unit Change	integer		•			

ContractServiceBundleUnit

This entity describes the number of units of a specific service bundle that are associated with a Recurring Service contract for a specific date range. They provide information used in billing for the service bundle.

The start and end dates of the date range are determined by either the beginning or end date of a billing period (as specified by the Contract Billing Period type) or by an adjustment to the number of units of the service bundle. That is, each billing period generates one ContractServiceBundleUnit *unless* the number of service bundle units is adjusted within the billing period. When the number of ServiceBundle units for one ServiceBundle on a contract is adjusted, a new ContractServiceBundleUnit is created with a start date equal to the effective date of the adjustment. The end date for the ContractServiceBundleUnit will be the end date of the billing period in which the adjustment becomes effective or, if another adjustment is made in the same billing period, the effective date of that adjustment. Refer to "About Recurring Service Contract Entity Relationships" on page 99.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: ContractServiceBundleUnit
Can Create:
Can Update:
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ApproveAndPostDate	Approve and Post Date	datetime	•				
ContractID	Contract ID	integer	•	•	Contract		
Cost	Contract Period Cost	double	•				
EndDate	Contract Period End Date	datetime	•	•			

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
id	Contract Service Period ID	integer	●	●			
Price	Contract Period Price	double	●				
ServiceBundleID	Service ID	integer	●	●	ServiceBundle	●	
StartDate	Contract Period Date	datetime	●	●			
Units	Units	integer	●	●			

ContractServiceUnit

This entity describes the number of units of a specific service that are associated with a Recurring Service contract for a specific date range. They provide information used in billing for the service bundle.

The start and end dates of the date range are determined by either the beginning or end date of a billing period (as specified by the Contract Period Type) or by an adjustment to the number of units of the service. That is, each billing period generates a ContractServiceUnit unless the number of Service units is adjusted within the billing period. When the number of Service units for one service on a contract is adjusted, a new ContractServiceUnit is created with a start date equal to the effective date of the adjustment. The end date for the ContractServiceUnit will be the end date of the current billing period or, if another adjustment is made in the same billing period, the effective date of that adjustment. Refer to "About Recurring Service Contract Entity Relationships" on page 99.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: ContractServiceUnit
Can Create:
Can Update:
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ApproveAndPostDate	Approve and Post Date	datetime	•				
ContractID	Contract ID	integer	•	•	Contract		
Cost	Period Cost	float	•				
EndDate	End Date	datetime	•	•			
id	Contract Service Unique id	long	•	•			
Price	Price	double	•				
ServiceID	Service ID	integer	•	•	Service	•	

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
StartDate	Start Date	datetime	●	●			
Units	Units	integer	●	●			
VendorAccountID	Vendor Account ID	integer	●		Account		

ContractTicketPurchase

This entity describes a payment amount applied to the purchase of (or pre-payment for) one or more Service Desk Tickets through a Per Ticket Contract. With a Per Ticket contract, a customers pre-pays for a set of Service Desk tickets, and each ticket completed under that contract consumes one ticket purchase, no matter how much labor is required to resolve the issue. The contract maintains the balance of unconsumed ticket purchases.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ContractTicketPurchase
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Fields that Cannot Be Queried

The following fields from this entity will return an error when queried.

- TicketsUsed

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- ContractTicketPurchase.ContractID must reference a Per Ticket type Contract.
- ContractTicketPurchase.StartDate must be < ContractTicketPurchase.EndDate
- On update(), ContractTicketPurchase.TicketsPurchased must be >= ContractTicketPurchase.TicketsUsed
- ContractTicketPurchase.TicketsPurchased and ContractTicketPurchase.PerTicketRate must be >= 0.
- ContractTicketPurchase.StartDate and ContractTicketPurchase.EndDate must be within the StartDate and EndDate of the contract referenced by ContractTicketPurchase.ContractID; that is, in Autotask the Contract Ticket Purchase must start and end within the Contract start and end dates.
- Note that ContractTicketPurchase entity does not automatically create a new contract cost when a new contract ticket purchase is created.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ContractID	Contract ID	long		●	Contract		
DatePurchased	Date Purchased	datetime		●			
EndDate	End Date	datetime		●			
id	id	long	●	●			
InvoiceNumber	Invoice Number	string (50)					
IsPaid	Paid	boolean		●		●	
PaymentNumber	Payment Number	string (50)					
PaymentType	Payment Type	integer				●	
PerTicketRate	Rate	double		●			
StartDate	Start Date	datetime		●			
Status	Status	integer				●	
TicketsPurchased	Tickets Purchased	double		●			
TicketsUsed	Tickets Used	double	●				

Country

This entity describes a Country as defined in the Autotask CRM module. The Country entity is referenced by the Account, Contact, and Sales Order entities and specifies the preferred display name for the associated country as well as the default address format and invoice template to be used by accounts, contacts, and sales orders that reference the Country ID.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: Country
Can Create:
Can Update: •
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Active	Active	boolean					
AddressFormatID	Address Format ID	long		•		•	
CountryCode	Country Code	string (2)	•				
DisplayName	Display Name	string (100)		•			
id	Country ID	long	•	•			
IsDefaultCountry	Default Country	boolean					
Name	ISO Standard Name	string (50)	•				

ExpenseItem

This entity describes a line item associated with an Expense Report entity. It allows expense line items to be created, queried, and updated through the API.

In Autotask, Expense Items can be added to an Expense Report from the Timesheets or Home modules, or a ticket, project, or task.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ExpenseItem
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Fields that Cannot Be Queried

The following fields from this entity will return an error when queried.

- GLCode
- Reimbursable
- Rejected

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- create() and update() are allowed only when the associated ExpenseReport has a Status = In Progress or Rejected
- When a TicketID or ProjectID are set, the AccountID is calculated based on the ticket or project. If you set an AccountID that differs from the calculated AccountID, an error is returned.
- When a TaskID is set, the AccountID and ProjectID are calculated based on the task and associated project. If you set an AccountID or ProjectID that differs from the calculated AccountID and Project ID, an error is returned.
- EntertainmentLocation is required when Category = Entertainement Expense
- To, From, and Miles are all required when Category = Mileage
- If no AccountID is supplied, the expense is associated to the "0" account

- PaymentType picklist options = American Express, Cash, Company Check, Company Credit Card, Mastercard, Other, Personal Check, Visa. These picklist options are not configurable.
- PaymentCategory and WorkType picklist options are configurable in the Autotask Admin module. Use Web Services API getFieldInfo() call to retrieve the list for a specific implementation.

Behaviors

- Reimbursable status is updated from the corresponding attribute of the current PaymentType every time the ExpenseItem entity is touched.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountID	Account ID	integer			Account		
BillableToAccount	Billable To Account	boolean		•			
Description	Description	String (128)		•			
EntertainmentLocation	Entertainment Location	String (128)		See Conditions			
ExpenseAmount	Expense Amount	double		•			
ExpenseCategory	Expense Category	integer		•		•	
ExpenseDate	Expense Date	datetime		•			
ExpenseReportID	Expense Report ID	integer		•	ExpenseReport		
From	Origin	String (128)		See Conditions			
GLCode	GL Code	String (20)	•				
HaveReceipt	Have Receipt	boolean		•			
id	Expense Item ID	long	•	•			
Miles	Miles	double		See Conditions			
PaymentType	Payment Type	integer		•		•	

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ProjectID	Project ID	integer			Project		
PurchaseOrderNumber	purchase_ order_number	string (32)					
Reimbursable	Reimbursable	boolean	•				
Rejected	Rejected	boolean	•				
TaskID	Task ID	integer			Task		
TicketID	Ticket ID	integer			Ticket		
To	Destination	String (128)		See Conditions			
WorkType	Work Type	integer				•	

ExpenseReport

This entity describes Expense Reports created in Autotask and is used to submit expense line items for approval and reimbursement. This entity allows expense reports to be created, queried, and updated through the Web Services API.

In Autotask, Expense Reports are created and managed through the Timehsheets module or under My Expense Reports in the left navigation menu of the Home module.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ExpenseReport
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Fields that Cannot Be Queried

The following fields from this entity will return an error when queried.

- AmountDue
- DepartmentNumber
- Submit

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- Status picklist options = In Progress, Awaiting Approval, Approved for Payment, Rejected, Paid, Transferred to Quickbooks. They cannot be changed in Autotask.
- RejectionReason is valid when Status = Rejected; it is required on create() or update()
- Submit =1 (to effect submission) is only allowed if the current Status is "in Process" or "Rejected"
- SubmitterID must = a valid ResourceID.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AmountDue	Amount Due	double	●				
ApprovedDate	Approved Date	datetime	●				
ApproverID	Approver ID	integer	●		Resource		
CashAdvanceAmount	Cash Advance Amount	double					
DepartmentNumber	Department Number	String (50)	●				
ExpenseTotal	Expense Total	double	●				
id	Expense Report ID	long	●	●			
Name	Name	String (100)		●			
QuickBooksReferenceNumber	Quick Books Reference Number	String (100)					
RejectionReason	Rejection Reason	String (2048)	●				
Status	Status	integer	●			●	
Submit	Submit	boolean					
SubmitDate	Submit Date	datetime	●				
SubmitterID	Submitter ID	integer		●	Resource		
WeekEnding	Period Ending	datetime		●			

InstalledProduct

This entity describes Autotask Configuration Items (previously known as Installed Products). Configuration items are products that are associated with an Account entity. Autotask users manage configuration items through the CRM Module (CRM >Configuration Items, or CRM > Accounts > Configuration Items tab).

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: InstalledProduct

Can Create: •

Can Update: •

Can Query: •

Can Delete:

Can Have UDFs: •

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountID	{LT:Account}	integer	•	•	Account		
Active	Product Active	boolean		•			
ContactID	Contact Name	integer			Contact		
ContractID	Contract ID	integer			Contract		
CreateDate		datetime	•				
DailyCost	{LT:InstalledProduct} Daily Cost	double					
HourlyCost	{LT:InstalledProduct} Hourly Cost	double					

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
id	{LT:InstalledProduct} ID	long	●	●			
InstallDate	Install Date	datetime		●			
InstalledByContactID	Installed By Contact ID	integer	●		Contact		
InstalledByID	Installed By	integer	●		Resource		
Location	Location	string (100)					
MonthlyCost	{LT:InstalledProduct} Monthly Cost	double					
Notes	{LT:InstalledProduct} Notes	string (5000)					
NumberOfUsers	{LT:InstalledProduct} Number of Users	double					
ParentInstalledProductID	Parent {LT:In-stalledProduct}	integer			InstalledProduct		
PerUseCost	{LT:InstalledProduct} Per Use Cost	double					
ProductID	Product ID	integer		●			
ReferenceNumber	Reference Number	string (50)					
ReferenceTitle	Reference Title	string (200)					
SerialNumber	Serial Number	string (100)					
ServiceBundleID	Service Bundle ID	integer			Service Bundle		
ServiceID	Service ID	integer			Service		
SetupFee	{LT:InstalledProduct} Setup Fee	double					
Type	{LT:InstalledProduct} Type	integer				●	
VendorID	Vendor Name	integer			Account		
WarrantyExpirationDate	Warranty Expiration Date	datetime					

InstalledProductType

This entity describes a Type, for example, printer, server, or workstation, assigned to a Configuration Item in Autotask. The InstalledProductType associates one or more User-defined Fields with configuration items of the same type. The User-defined Fields allow the user to capture key information for configuration items of the specified type, for example, make and model, or replacement part numbers.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: InstalledProductType
Can Create: •
Can Update: •
Can Query: •
Can Delete: •
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- All actions on this entity require System Administrator or Full Access security level.
- InstalledProductType.Name must be unique.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Active	Active	boolean		•			
id	ID	long	•	•			
Name	Name	string (100)		•			

InstalledProductTypeUdfAssociation

This entity associates User-defined Fields with an InstalledProductType. An InstalledProductType is assigned to a Configuration Item in Autotask. The user defined fields describe key data that the user wants to capture for configuration items of the specified type, for example, make and model, or replacement part numbers. .

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: InstalledProductTypeUdfAssociation

Can Create: •

Can Update: •

Can Query: •

Can Delete: •

Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- All actions on this entity require a System Administrator or Full Access security level.
- UserDefinedFieldDefinitionId must reference a User-defined Field of type 6 (Installed Product)
- There cannot be two InstalledProductTypeUdfAssociations with the same InstalledProductTypeId and UserDefinedFieldDefinitionId.
- InstalledProductTypeId and UserDefinedFieldDefinitionId cannot be updated.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
id	Id	long	•	•			
InstalledProductTypeId	Installed Product Type Id	long	•	•	InstalledProductType		

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Required	Required	boolean		●			
SortOrder	Sort Order	integer		●			
UserDefinedFieldDefinitionId	User Defined Field Defin-ition Id	long	●	●		●	

InternalLocation

This entity describes a Location defined in Company Setup in the Autotask Admin module. Locations represent the various sites where the Autotask user's company has a physical presence. Every resource is associated with a location. The time zone and holiday set of the associated location are applied to the resource's time entries and schedules.

Do not confuse InternalLocation with "AccountLocation" on page 58, the entity that holds site configuration information for a specified Account entity.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	InternalLocation
Can Create:	
Can Update:	
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

NOTE: TimeZone returns the Microsoft Time Zone Index value associated with the internal Location.

Conditions and Requirements

An internal location's Country is mapped to either a Country from the Autotask Countries list or to "Other". This impacts API queries as follows:

- On query():

  If the Country field is mapped to a country in Autotask, then the country display value will be returned.

  If the Country value is mapped to “Other” in Autotask, the stored text value is returned.

- On query() by Country:

  The system searches only the Country display name.

  If the entity’s Country value is not mapped to an Autotask country, then you must query for Country = “Other”; that is, the string “Other”, not the stored text value. This will return all entities where the country value is non-standard.

  For details on how the Country value is mapped in Autotask, refer to "Account" on page 54.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AdditionalAddressInfo	Additional Address Info	string (100)	●				
Address1	Address 1	string (100)	●				
Address2	Address 2	string (100)	●				
City	City	string (50)	●				
Country	Country	string (100)	●				
HolidaySetId	Holiday Set	long	●			●	
id	id	long	●	●			
IsDefault	Is Default	boolean	●				
Name	Name	string (100)	●	●			
PostalCode string 20	{LT:ZipCode}	string (20)	●				
State	{LT:State}	string (25)	●				
TimeZone	Time Zone Code	string (100)	●				

InventoryItem

This entity describes an Autotask product that is associated with an Inventory Location in the Autotask Inventory module. Once an InventoryItem entity has been created, you can track quantities for that item (quantity on hand, quantity on order) and provide a value for minimum and maximum quantity for use with the Auto-Fill Order feature. You can assign serial numbers to instances of InventoryItems, add them to purchase orders, and "receive" them. You can also transfer them between inventory locations or associate them with an account. Inventory items are added and managed in Autotask through the Inventory module.

NOTE: Please review the Conditions and Requirements listed below before creating or updating InventoryItems.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	InventoryItem
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- In Autotask, Inventory Add/Edit Item permission is required to Add or Edit an Inventory Item.
- Each InventoryItem requires a unique combination of InventoryItem.ProductID and InventoryItem.InventoryLocationID.
- InventoryLocationID must reference an Active inventory location.
- QuantityMinimum must be >= 0.
- QuantityMaximum must be >= QuantityMinimum.
- For serialized products (that is, products that track a serial number):

  On create(), QuantityOnHand must = 0 (quantity for these inventory items is increased by creating InventoryItemSerialNumber entities).

  On update(), QuantityOnHand is Read Only

- QuantityOnHand must be >= the quantity Picked + quantity Reserved.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
BackOrder	Back Order	integer	●				
Bin	Bin	string (50)					
id	Inventory Item ID	long	●	●			
InventoryLocationID	Inventory Location ID	integer	●	●	InventoryLocation		
OnOrder	On Order	integer	●				
Picked	Picked	integer	●				
ProductID	Product ID	integer	●	●	Product		
QuantityMaximum	Quantity Maximum	integer		●			
QuantityMinimum	Quantity Minimum	integer		●			
QuantityOnHand	Quantity On Hand	integer		●			
ReferenceNumber	Reference Number	string (50)					
Reserved	Reserved	integer	●				

InventoryItemSerialNumber

This entity describes a serial number associated with an Inventory Item. It allows users to track and manage Inventory Items created from serialized Products, that is, Autotask Products that require a unique serial number. An InventoryItemSerialNumber entity can be associated with only one InventoryItem.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	InventoryItemSerialNumber
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- In Autotask, Inventory Add/Edit Items permission is required to create or update an InventoryItemSerialNumber.
- InventoryItemID must reference an InventoryItem based on a serialized Product; that is, InventoryItem.ProductID must reference a Product where Serialized = True.
- For create() and update(), SerialNumber must be unique.
- SerialNumber cannot be updated if the InventoryItemID is associated with an InventoryItem that has been picked or delivered/shipped.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
id	Inventory Item Serial Number ID	long	•	•			
InventoryItemID	Inventory Item ID	long	•	•	InventoryItem		
SerialNumber	Serial Number	string (100)		•			

InventoryLocation

This entity describes an Autotask Inventory Location, that is, a physical or virtual place where your company stores or assigns inventory items. For example, an Inventory location can be an actual warehouse or retail outlet, or a virtual holding place such as "Lost Items" or "Returned Items". Every Inventory Item must be associated with an Inventory Location. You assign a Location to Inventory Items when you add the items to inventory or to a purchase order.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	InventoryLocation
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	No
Can Have UDFs:	No

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- LocationName must be unique.
- In Autotask, the Inventory Manage Locations permission is required to create, update, or query an Inventory Location.
- The InventoryLocation where IsDefault = True cannot be inactivated.
- If ResourceID has a value IsDefault cannot = True.

In the following table:

> For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Active	Active	boolean		•			
id	LocationID	long	•	•			
IsDefault	IsDefault	boolean	•				

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
LocationName	Location Name	string (50)		•			
ResourceID	Resource ID	integer	•		Resource		

InventoryTransfer

This entity describes a transaction where a specified quantity of one InventoryItem entity is transferred from the item's currently assigned InventoryLocation to another InventoryLocation. Note that the InventoryTransfer entity does not describe transactions where an InventoryItem is associated with an Account as a Configuration Item.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	InventoryTransfer
Can Create:	•
Can Update:	
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

In Autotask, Transfer Inventory Items permission is required to create or update an InventoryTransfer.

- FromLocationID and ProductID must = InventoryLocationID and ProductID of a valid InventoryItem.
- QuantityTransferred must be <= QuantityOnHand.
- QuantityTransferred must be > 0.
- ToLocationID cannot be = to FromLocationID.
- ToLocationID must reference an active Inventory Location.
- For products that track a serial number:

  SerialNumber is required.

  SerialNumber must reference a valid Serial Number (InventoryItemSerialNumber.SerialNumber).

  InventoryItemSerialNumber.InventoryItemID must reference a valid Inventory Item with InventoryLocationID and ProductID = InventoryTransfer.FromLocationID and InventoryTransfer.ProductID.

  QuantityTransferred must = 1.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
FromLocationID	Transfer From Inventory Location ID	long	●	●	InventoryLocation		
id	Inventory Transfer ID	long	●	●			
Notes	Notes	string (4000)	●				
ProductID	Product ID	long	●	●	Product		
QuantityTransferred	Quantity Transferred	integer	●	●			
SerialNumber	Serial Number	string (100)	●				
ToLocationID	Transfer To Inventory Location ID	long	●	●	InventoryLocation		
TransferByResourceID	Transfer By Resource ID	integer	●		Resource		
TransferDate	Transfer Date	datetime	●				

Invoice

The Invoice entity describes an Autotask Invoice. Invoices include Billing Items that have been approved and posted and are being billed to a customer or presented for information purposes only. They are processed through the Autotask Contracts module (Contracts > Invoices).

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	Invoice
Can Create:	
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Fields that Cannot Be Queried

The following fields from this entity will return an error when queried.

- DueDate
- TaxGroup

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

NOTE: This entity includes the WebServiceDate field. It allows the external application to flag the date and time that the object is processed. It does not appear in the Autotask interface. The API provides this field as a convenience for applications that work specifically with the Invoice entity via the Web Services API.

Conditions and Requirements

- To maintain consistency between data submitted via the API and via the UI, Invoice.PaidDate ignores time stamp information and stores the date only.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountID	{LT:Account} ID	integer	●		Account		
BatchID	Batch ID	integer	●				
Comments	Comments	string (4000)	●				
CreateDateTime	Create Date Time	datetime	●				
CreatorResourceID	Creator Resource ID	integer	●		Resource		
DueDate	Due Date	datetime	●				
FromDate	From Date	datetime	●				
id	Invoice ID	long	●	●			
InvoiceDateTime	Invoice Date Time	datetime	●	●			
InvoiceEditorTemplateID	Invoice Editor Template ID	integer	●		Invoice Template		
InvoiceNumber	Invoice Number	string (100)					
InvoiceTotal	Invoice Total	double	●				
IsVoided	Is Voided	boolean	●				
OrderNumber	Order Number	string (20)	●				
PaidDate	Paid Date	datetime					
PaymentTerm	Payment Term ID	integer	●			●	
TaxGroup	Tax Group ID	integer	●			●	
TaxRegionName	Tax Region Name	string (200)	●				
ToDate	To Date	datetime	●				
TotalTaxValue	Total Tax Value	double	●				
VoidedByResourceID	Voided By Resource ID	integer	●		Resource		
VoidedDate	Voided Date	datetime	●				
WebServiceDate	Web Service Use Date	datetime					

InvoiceTemplate

This entity describes an Autotask Invoice Template that defines the content and appearance of an Autotask Invoice. Invoices include Billing Items that have been approved and posted and are being billed to a customer or presented for information purposes only. The template determines what account, billing item, tax and financial related details the Autotask user wants to include on the invoice, and how the output will be grouped and presented. Invoice templates are added through the UI in the Admin module: Contracts (and Billing) > Invoices > Templates.

NOTE: With the InvoiceTemplate entity and additional new entities and fields, the API provides the information needed for an external application to reproduce all information and content on an Autotask invoice, but not necessarily create an exact visual reproduction of the invoice.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	InvoiceTemplate
Can Create:	
Can Update:	
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

NOTE: This entity includes the WebServiceDate field. It allows the external application to flag the date and time that the object is processed. It does not appear in the Autotask interface. The API provides this field as a convenience for applications that work specifically with the Invoice entity via the Web Services API.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Data-type	Read Only	Is Required	Reference Name	Pick-list	Picklist Parent Column Name
CoveredByBlockRetainerContractLabel	Covered By Block Retainer Contract Label	string (50)	●				
CoveredByRecurringServiceFixedPricePerTicketContractLabel	Covered By Recurring Service Fixed Price Per Ticket Contract Label	string (50)	●			●	
DateFormat	Date Format	integer	●	●			
DisplayFixedPriceContractLabor	Display Labor Associated With Fixed Price Contracts	boolean	●	●			
DisplayRecurringServiceContractLabor	Display Labor Associated With Recurring Service Contracts	boolean	●	●			
DisplaySeparateLineItemForEachTax	Display Separate Line Item For Each Tax	boolean	●	●			
DisplayTaxCategory	Display Tax Category						
DisplayTaxCategorySuperscripts	Display Tax Category Superscripts	boolean	●	●			
DisplayZeroAmountRecurringServicesAndBundles	Display Zero Amount Recurring Services And Bundles		●	●		●	
GroupBy	Group By	integer	●	●			

Field Name	Label	Data-type	Read Only	Is Required	Reference Name	Pick-list	Picklist Parent Column Name
id	Invoice ID	long	●	●			
ItemizeItemsInEachGroup	Itemize Items In Each Group	integer	●	●		●	
ItemizeServicesAndBundles	Itemize Services And Bundles	boolean	●	●			
Name	Name	string (50)	●	●			
NonBillableLaborLabel	Non Billable Labor Label	string (50)	●	●			
NumberFormat	Number Format	integer	●	●		●	
PageLayout	Page Layout	integer	●	●		●	
PageNumberFormat	Display Page Number Format	integer	●	●		●	
PaymentTerms	Payment Terms	integer	●	●	PaymentTerm		
RateCostUnitOfMeasure	Rate Cost Unit Of Measure	string (50)	●	●			
SortBy	Sort By	integer	●	●		●	
TimeFormat	Time Format	integer	●	●		●	

Opportunity

This entity describes an Autotask Opportunity. An opportunity is a forecasted piece of business: that is, an identifiable prospect that needs a product or service and offers a potential sale, project, or contract. Autotask Opportunities allow you to describe the amount, due date, and probability of expected sales revenue from an opportunity, track the progress of the opportunity, and generate sales forecasts.

You can track Opportunities for all account types and track multiple opportunities for each account. Opportunities can also be associated with a Quote or eQuote generated in Autotask. You add and manage Opportunities in Autotask in the CRM module.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	Opportunity
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	No
Can Have UDFs:	•

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

Conditions and Requirements

- Probability must be >= 0 and <= 100.
- UseQuoteTotals can only = "True" if an Autotask Quote entity references this Opportunity.
- ProjectedCloseDate must be >= CreateDate.
- ContactID must reference an Active Autotask Contact associated with the Account referenced by the AccountID field.
- The AccountID reference by Opportunity.ContactID must = Opportunity.AccountID, that is, the Opportunity Contact must be associated with the same Account as the Opportunity.
- OwnerResourceID must reference an Active Autotask Resource with security level access to the CRM module.
- Invoice.Amount (Revenue) value for Opportunities created via Web Services are written to the One-Time Amount column in the database. If an Opportunity is later updated via Web Services, any new value provided for Amount will be written to the One-Time Amount column and any values in the Monthly Amount, Quarterly Amount, Semi-Annual Amount, and Yearly Amount database columns will be cleared.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountID	Account Object ID	integer		•	Account		
AdvancedField1	Number Of Users	double					
AdvancedField2	Setup Fee	double					
AdvancedField3	Hourly Cost	double					
AdvancedField4	Daily Cost	double					
AdvancedField5	Monthly Cost	double					
Amount	Amount	double		•			
Barriers	Barriers	string (500)					
ClosedDate	Closed Date	datetime					
ContactID	Contact Object ID	integer			Contact		
Cost	Cost	double		•			
CreateDate	Create Date	datetime		•			
HelpNeeded	Help Needed	string (500)					
id	Object ID	long	•	•			
LeadReferral	Lead Referral Object ID	integer				•	
Market	Market	string (500)					
NextStep	Next Step	string (500)					
OwnerResourceID	Creator Object ID	integer		•	Resource		
Probability	Probability	integer		•			
ProductID	Product Object ID	integer			Product		
ProjectedCloseDate	Projected Close	datetime		•			
ProjectedLiveDate	Start Date	datetime					
PromotionName	Promotion Name	string (50)					

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Rating	opportunity_rating_id	integer				●	
RevenueSpread	Spread Revenue Recog-nition Value	integer					
RevenueSpreadUnit	Spread Revenue Recog-nition Unit	string (10)				●	
SalesOrderID	Sales Order ID	integer	●		Sales Order		
Stage	Stage Object ID	integer		●		●	
Status	Status	integer		●		●	
ThroughDate	Through Date	datetime					
Title	Description	string (128)		●			
TotalAmountMonths	number_months_for_ estimating_total_profit	integer					
UseQuoteTotals	Use Quote Total Amount	boolean		●			

PaymentTerm

This entity describes an Autotask Payment Term. A payment term specifies the conditions and requirements for payment due on an Autotask invoice; for example, Net 30 days.

In Autotask, payment terms are set up in the Admin module and added to a Quote or invoice template.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: ShippingType
Can Create:
Can Update:
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Active	Active	boolean					
Description	Description	string (2000)					
id	Payment Term ID	long	•	•			
Name	Name	string (100)		•			
PaymentDueInDays	Payment Due In Days	integer					

Phase

This entity describes an Autotask project Phase. Phases allow users to break projects into sub-groups of project tasks. They can be a sub-phase to a Parent phase. Users manage phases through the Project's Schedule view in the Projects module.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: Phase

Can Create: •

Can Update: •

Can Query: •

Can Delete:

Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- On update(), ProjectID cannot be changed. An error will be returned.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
CreateDate	Phase Creation Date	datetime	•				
CreatorResourceID	Phase Creator	integer	•		Resource		
Description	Phase Description	string (8000)					
DueDate	Phase End Date	datetime					
EstimatedHours	Phase Estimated Hours	double	•				

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ExternalID	Phase External ID	string (50)					
id	Phase ID	long	●	●			
LastActivityDateTime	Phase Last Activity Date	datetime	●				
ParentPhaseID	Parent Phase	integer			Task		
PhaseNumber	Phase Number	string (50)	●				
ProjectID	Project	integer		●	Project		
Scheduled	Is Scheduled	boolean	●				
StartDate	Phase Start Date	datetime					
Title	Phase Title	string (255)		●			

Product

This entity describes an instance of hardware, software, or a material item in Autotask that a company sells or supports for customers. You can "install" Products to customer Accounts and once installed, users can support Products through Service Desk tickets, bill for products, and manage the products through the customer Account. With the Autotask Inventory module, Products can be added to inventory as Inventory Items.

In Autotask, Administrators manage Products through the Admin module: Products and Services > Products > Products. If the Inventory module is enabled, users with the correct permission can manage Products through Inventory > Manage Products.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	Product
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	•

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

NOTE: To accommodate changes in the Autotask database and UI while maintaining compatibility within the API, Product allocation code, type 7, now maps to Material (cost) allocation code, type 4. Pre-existing type 7 codes have been added to type 4 codes because Product allocation codes have been eliminated from the database. Existing queries for type 7 will be redirected to type 4.

To maintain compatibility with existing integrations, Product.ProductAllocationCodeID is still required and will accept either type 7 or type 4 codes.

Conditions and Requirements

- Default value for DoesNotRequireProcurement = False.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Active	Active	boolean		•			
CostAllocationCodeID	Cost Allocation Code ID	integer			AllocationCode		
DefaultVendorID	Vendor Account ID	integer			Account		
Description	Product Description	string (200)					
DoesNotRequireProcurement	Does Not Require Procurement	boolean					
ExternalProductID	External ID	string (50)					
id	ProductID	long		•			
Link	Product Link	string (500)					
ManufacturerName	Manufacturer Account Name	string (100)					
ManufacturerProductName	Manufacturer Product Number	string (50)					
MSRP	MSRP	double					
Name	Product Name	string (100)		•			
PeriodType	Period Type	string (10)				•	
ProductAllocationCodeID	Allocation Code ID	integer		•	AllocationCode		
ProductCategory	Product Category	integer				•	
Serialized	Is Serialized	boolean		•			
SKU	Product SKU	string (50)					
UnitCost	Unit Cost	double					
UnitPrice	Unit Price	double					
VendorProductNumber	Vendor Product Number	string (50)					

ProductVendor

This entity describes a Vendor type Account that is associated with an Autotask Product. Products can be associated with more than one Product Vendor. A Product Vendor is not required unless the user intends to create an Inventory Item from the Product and add the Inventory Item to one or more purchase orders.

In Autotask, users associate Vendors with Products and manage the Product Vendors through the Add/Edit Product window accessed through the Product Search list in the Admin or Inventory modules.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: ProductVendor
Can Create: •
Can Update: •
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

The Web Services API ProductVendor entity displays the following behaviors.

- On create(), if no default ProductVendor exists, ProductVendor.IsDefault is set to True for the newly created ProductVendor. If multiple Product Vendors are created in a single batch, and no default ProductVendor exists, IsDefault is set to True for the first ProductVendor created via the batch.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Active	Is Active	boolean		•			
id	id	long	•	•			
IsDefault	Is Default	boolean		•			
ProductID	Product ID	integer	•	•	Product		

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
VendorCost	Vendor Cost	double					
VendorID	Vendor Account ID	integer		•	Account		
VendorPartNumber	Vendor Part Number	string (50)					

Project

This entity describes an Autotask Project. A project defines and organizes a group of related tasks, events, and documents. Each Project is specific to one Account and can include phases. Autotask users manage Projects through the Projects module.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	Project
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	•

Fields that Cannot Be Queried

The following fields from this entity will return an error when queried.

- ChangeOrdersBudget
- ExtProjectType

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

Conditions and Requirements

- Autotask no longer supports projects of Type = Business Objective, Archived, or Inactive. Existing Projects of these types have been reassigned as follows:

  Business Objective type have been reassigned to Internal.

  Archived type have been reassigned to their original type (Client, Internal, Proposal, or Template) with Project.Status = Complete.

  Inactive type have been reassigned to their original type with Project.Status = Inactive. To reactivate an inactive project, the Project.Status must be assigned a status other than Inactive.
- Projects are now inactivated by assigning the Inactive status; that is, Project.Status = Inactive.

New Behavior

- With the June 2013 redesign of the Autotask Project schedule, updates to Project.StartDate will impact all tasks in the project. To remain compatible with the new schedule features, all dates for the project tasks/phases will be pushed out or moved back the same number of days that the project start date is moved. The adjustments will respect Project Settings for non-business days and holidays.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountID	{LT:Account}	integer		●	Account		
ActualBilledHours	Actual Billed Hours	double	●				
ActualHours	Actual Hours	double	●				
ChangeOrdersBudget	Changed orders	double	●				
ChangeOrdersRevenue	Change Orders Revenue	double	●				
CompanyOwnerResourceID	{LT:Account} Owner	integer			Resource		
CompletedDateTime	Completed date	datetime					
CompletedPercentage	Completed Percentage	integer	●				
ContractID	Contract	integer			Contract		
CreateDateTime	Create DateTime	datetime	●				
CreatorResourceID	Created By	integer	●		Resource		
Department	Department	integer				●	
Description	Description	string (2000)					
Duration	Duration	integer	●				
EndDateTime	End Date	datetime		●			
EstimatedSalesCost	Estimated Sales Cost	double					
EstimatedTime	Estimated Time	double	●				
ExtPNumber	Ext Project Number	string (50)					

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ExtProjectType	Ext Project Type	integer				●	
id	id	long	●	●			
LaborEstimatedCosts	{LT:Labor} Estimated Costs	double					
LaborEstimatedMarginPercentage	{LT:Labor} Estimated Margin Percentage	double	●				
LaborEstimatedRevenue	{LT:Labor}Estimated Revenue	double					
LineOfBusiness	{LT:LineOfBusiness}	integer				●	
OriginalEstimatedRevenue	Original Estimated Revenue	double					
ProjectCostEstimatedMarginPercentage	Project Cost Estimated Margin Percentage	double	●				
ProjectCostsBudget	Project Estimated costs	double					
ProjectCostsRevenue	Project Cost Revenue	double					
ProjectLeadResourceID	Project Lead	integer		●	Resource		
ProjectName	Project Name	string (100)		●			
ProjectNumber	Project Number	string (50)	●				
PurchaseOrderNumber	Purchase Order Number	string (32)					
SGDA	SG&A	double					
StartDateTime	Start Date	datetime		●			
Status	Status	integer				●	
StatusDateTime	Status Date	datetime					
StatusDetail	Status Detail	string (2000)					
Type	Type	integer		●		●	

ProjectCost

This entity describes a cost associated with an Autotask Project. A cost is a billing item for products or materials. Cost items can be billable or non-billable. Billable cost items appear in Approve and Post.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ProjectCost
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	• see conditions below
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

Conditions and Requirements

- create(), update(), delete(), and query() require Security Level permission to access the Projects module and the user must have access to the specific project.
- ProjectCost must have either a ProductID or AllocationCodeID.
- ProjectID cannot be updated.
- update() and delete() are allowed only when Billed = False (cost has not been approved and posted).
- Only query() is allowed if Billed = True (cost has been approved and posted).
- AllocationCodeID must reference a Material Cost Code type allocation code.
- On create(), or update() when ProductID has changed and is not Null, if no value is supplied for the AllocationCodeID, UnitCost, or UnitPrice fields, then those fields take their values from the Product.
- On create() or update() when AllocationCodeID has changed and is not Null, and if no value is supplied for the UnitCost and UnitPrice fields, then those fields take their values from the Material Cost Code associated with AllocationCodeID.
- If no value is supplied for BillableToAccount, the value is set to True.
- Status is read only. On create(), Status is set as follows:

  If ProductID is null, then Status = Ready to Ship.

  If ProductID references a Product where DoesNotRequireProcurement = True, then Status = Ready to Ship.

If ProductID references a Product where DoesNotRequireProcurement = False and the Product is not available in Inventory, then Status = Need To Order.

If ProductID references a Product where DoesNotRequireProcurement = False and the Product is available in Inventory, then Status = Ready to Ship.

If Status = Need To Order and ExtendedCost is > the value set in the workflow policy "Require approval before ordering..." then Status is automatically set to Waiting Approval.

- ExtendedCost = UnitQuantity * UnitCost. If UnitCost is Null, the value will be set to the value from the Material Cost Code associated with AllocationCodeID.
- BillableAmount = UnitQuantity * UnitPrice. If UnitPrice is Null, the value will be set to the value from the Material Cost Code associated with AllocationCodeID.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AllocationCodeID	Allocation Code	long			AllocationCode		
BillableAmount	Billable Amount	double	•				
BillableToAccount	Billable To Account	Boolean					
Billed	Billed	Boolean	•				
ContractServiceBundleID	Contract Service Bundle ID	long			ContractServiceBundle		
ContractServiceID	Contract Service ID	long			ContractService		
CostType	Cost Type	integer		•		•	
CreateDate	Create Date	datetime	•				
CreatorResourceID	Created By	long	•		Resource		
DatePurchased	Date Purchased	datetime		•			
Description	Product Description	string (2000)					

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
EstimatedCost	Estimated Cost	double	●				
ExtendedCost	Extended Cost	double	●				
id	id	long	●	●			
InternalPurchaseOrderNumber	Internal Purchase Order Number	string (50)					
Name	Name	string (100)		●			
ProductID	Product	long			Product		
ProjectID	Project	long		●	Project		
PurchaseOrderNumber	Purchase Order Number	string (32)					
Status	Status	long	●			●	
StatusLastModifiedBy	Last Modified By	long	●				
StatusLastModifiedDate	Last Modified Date	datetime	●				
UnitCost	Unit Cost	double					
UnitPrice	Unit Price	double					
UnitQuantity	Unit Quantity	double		●			

ProjectNote

This entity describes notes created by an Autotask user and associated with a Project entity. Autotask users manage Project Notes through the Notes option accessed through the Project Menu on the Project Summary page.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: ProjectNote
Can Create: •
Can Update: •
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

NOTE: API queries for ProjectNote entities with Publish = 1 now include all System Workflow Notes. If you do not want system workflow notes returned, you must modify the query to include a condition that excludes ProjectNote.NoteType = 13.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Announce	Announce	boolean		•			
CreatorResourceID	Creator Resource	integer	•		Resource		
Description	Description	String (3200)		•			
id	Task Note ID	long	•	•			
LastActivityDate	LastActivityDate	datetime	•				
NoteType	Note Type	integer		•		•	
ProjectID	Project	integer		•	Project		
Publish	Publish	integer		•		•	

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Title	Title	string (250)		•			

PurchaseOrder

This entity describes an Autotask Inventory module Purchase Order. Purchase orders allow users to track one or more products ordered and received from Vendor type accounts. Purchase orders must be submitted (Status = Submitted) before items can be received. Autotask users create and manage purchase orders from the Inventory module.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	PurchaseOrder
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- In Autotask, Inventory Add/Edit Orders permission is required to create or update a purchase order .
- VendorID must reference an active Account of AccountType = Vendor
- On create(), Status = New
- On update()

  Current Status must = New

  Can only update Status to 'Submitted' when the current Status = 'New'

  Cannot change Status to 'New' or 'Received in Full/Part'.

  Cannot update Status to 'Submitted', unless Purchase Order Items have been created for this Purchase Order.

  Cannot change Status to 'Canceled' when the current Status = 'Cancelled' or 'Received in Full/Part'.
- In Autotask, Tax Regions have replaced Tax Groups. To preserve existing API code, PurchaseOrder-.TaxGroup now passes the TaxRegionID.
- A Purchase Order search conducted through the Autotask UI will not recognize Purchase Orders that do not have an associated Purchase Order Item.

When working in the Autotask UI, a purchase order item must be associated with a purchase order at the time the purchase order is created. When working with the API, a PurchaseOrder entity is created before any PurchaseOrderItem entities are associated with it.These Purchase Orders , however, are not recognized in the UI as valid Purchase Orders until they are associated with one or more Purchase Order Items.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
CancelDateTime	Cancel Date	datetime	●				
CreateDateTime	Create Date	datetime	●				
CreatorResourceID	Creator Resource ID	integer	●		Resource		
ExternalPONumber	External Purchase Order Number	string (50)					
Fax	Fax	string (25)					
Freight	Freight Cost	double					
GeneralMemo	General Memo	string (4000)					
id	Order ID	long	●	●			
PaymentTerm	Payment Term ID	integer				●	
Phone	Phone	string (25)					
PurchaseForAccountID	Purchase For Account ID	integer			Account		
ShippingDate	Expected Ship Date	datetime					
ShippingType	Shipping Type	integer			ShippingType		
ShipToAddress1	Address Line 1	string (128)		●			
ShipToAddress2	Address Line 2	string (128)					
ShipToCity	City	string (30)		●			
ShipToName	Addressee Name	string (100)		●			

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ShipToPostalCode	Postal Code	string (10)		●			
ShipToState	State	string (25)		●			
Status	Order Status ID	integer		●		●	
SubmitDateTime	Submit Date	datetime	●				
TaxGroup	Tax Region ID	integer				●	
VendorID	Vendor Account ID	integer	●	●	Account		
VendorInvoiceNumber	Vendor Invoice Number	string (50)					

PurchaseOrderItem

This entity associates a Product entity with a PurchaseOrder entity. Purchase Orders are associated with a specific vendor Account. Products added to a purchase order as a PurchaseOrderItem can be "received" into Inventory, that is, added to an InventoryLocation as an InventoryItem. This allows users to track and manage the ordering and receipt of Inventory Items, for example, hardware, software, and supplies. You add products to purchase orders in the Inventory module.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	PurchaseOrderItem
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- In Autotask, Inventory Add/Edit Items permission is required to create or update a purchase order item.
- The combination of ProductId, InventoryLocationId, and OrderID must be unique.
- InventoryLocationID must reference an active InventoryLocation.
- Quantity must be >= 1.
- UnitCost must be >= 0.00.
- On create() and update(), OrderID must reference a PurchaseOrder with Status = New.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
id	Inventory Item ID	long	●	●			
InventoryLocationID	Inventory Location ID	integer		●	InventoryLocation		
Memo	Memo	string (4000)					
OrderID	Inventory Order ID	integer	●	●	PurchaseOrder		
ProductID	Product ID	integer		●	Product		
Quantity	Quantity Ordered	integer		●			
SalesOrderID	Sales Order ID	long	●		Sales Order		
UnitCost	Product Unit Cost	double		●			

PurchaseOrderReceive

This entity describes a transaction where a specified quantity of a PurchaseOrderItem is "received", that is, debited from the Quantity value of the associated PurchaseOrderItem and added to the QuantityOnHand value of the associated InventoryItem entity. The InventoryItem must share the same InventoryLocationID and ProductID as the Purchase Order Item. A PurchaseOrderReceive transaction can debit all or part of the PurchaseOrderItem.Quanity value and tracks the quantity previously received and the quantity back ordered (QuantityBackOrdered = PurchaseOrderItem.Quantity - QuantityPreviouslyReceived + QuantityNowReceived.)

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	PurchaseOrderReceive
Can Create:	•
Can Update:	
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- In Autotask, Inventory Receive Items permission is required to create or update the receipt of a purchase order item.
- PurchaseOrderItemID must be valid.
- The Status of the OrderID associated with the PurchaseOrderItemID must = "Submitted" or "Received in Part".
- QuantityNowReceiving must be > 0

  NOTE: The Autotask Inventory module interface allows Quantity = 0; the API does not.
- The sum of QuantityNowReceiving + QuantityPreviouslyReceived must be <= the Quantity for the associated Purchase Order Item (PurchaseOrderItemID).
- For serialized products:

  QuantityNowReceiving must = 1.

  SerialNumber is required.

  SerialNumber must be unique.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
id	Inventory Item ID	long	●	●			
PurchaseOrderItemID	Purchase Order Item ID	long	●	●	PurchaseOrderItem		
QuantityBackOrdered	Quantity Back Ordered	integer	●				
QuantityNowReceiving	Quantity Now Receiving	integer	●	●			
QuantityPreviouslyReceived	Quantity Previously Received	integer	●				
ReceiveDate	Receive Date	datetime	●				
ReceivedByResourceID	Transfer By Resource ID	integer	●		Resource		
SerialNumber	Serial Number	string (50)	●				

Quote

This entity describes a Quote in Autotask. The quote allows users to specify and track multiple products, services, labor items, etc., to further define an Opportunity. A Quote must be associated with an Autotask Opportunity entity and an Opportunity can have only one associated quote. The quote total can update the Opportunity's forecasted amount. Quotes can be used internally or presented to the customer as an eQuote.

Quotes are created and managed in Autotask through the CRM module.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	Quote
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- ExpirationDate must be >= EffectiveDate.
- OpportunityID must be an Active Autotask Opportunity that does not already have an associated Quote.
- ContactID must be an Active Autotask Contact from the Account referenced by the Quote AccountID.
- On update(), AccountID and OpportunityID are Read Only.
- ProposalProjectID can accept only Proposals of Type = 2 (Project).
- In Autotask, Tax Regions have replaced Tax Groups. To preserve existing API code, Quote.TaxGroup now passes the TaxRegionID.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountID	AccountID	integer			Account		
BillToLocationID	Bill To Location ID	integer		●	QuoteLocation		
CalculateTaxSeparately	calculate_tax_separately	boolean					
Comment	Quote Comment	string (1000)					
ContactID	Contact ID	integer		●	Contact		
CreateDate	CreateDate	datetime	●				
CreatorResourceID	Creator Resource ID	integer	●		Resource		
Description	Quote Description	string (2000)					
EffectiveDate	Effective Date	datetime		●			
eQuoteActive	eQuote Active	boolean					
ExpirationDate	Expiration Date	datetime		●			
ExternalQuoteNumber	External Quote Number	string (50)					
GroupByProductCategory	group_by_product_category	boolean					
id	Quote ID	integer	●	●			
Name	Quote Name	string (100)		●			
OpportunityID	Opportunity ID	integer	●	●	Opportunity		
PaymentTerm	Payment Term	integer				●	
PaymentType	Payment Type	integer				●	
ProposalProjectID	Project ID	integer			Project		
PurchaseOrderNumber	Purchase Order Number	string (50)					
ShippingType	Shipping Type ID	integer			ShippingType	●	
ShipToLocationID	Ship To Location ID	integer		●	QuoteLocation		
ShowEachTaxInGroup	show_each_tax_in_tax_group	boolean					
SoldToLocationID	Sold To Location ID	integer		●	QuoteLocation		
TaxGroup	Tax Region ID	integer				●	

QuoteItem

This entity describes an Autotask Quote Item. Quote Items define a line item added to an Autotask Quote. Users can select quote items from labor rates (roles), products, services, material costs and expense lists, or manually enter one time quote items.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	QuoteItem
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	•
Can Have UDFs:	

Fields that Cannot Be Queried

The following fields from this entity will return an error when queried.

- AverageCost
- HighestCost

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements:

- QuoteItem.Type is required for all QuoteItem entities.
- Name is required and must be present except where QuoteItem.Type = Service or ServiceBundle (Type 11 or 12).
- Name is Read Only if the QuoteItem.Type = Service or ServiceBundle (Type 11 or 12). You cannot change the Name of a Service or ServiceBundle type Quote.
- Each QuoteItem entity allows only one "Item" field; that is, only one of the following fields: ProductID, CostID, LaborID, ExpenseID, ShippingID, ServiceID, ServiceBundleID, is allowed for each QuoteItem entity that you create.
- If Type = 1 (Product)

  ProductID value is required.

  PeriodType value is required.

The PeriodType CANNOT be semi-annual. Only month, quarter, year, and one-time period types are allowed for Product.

- If Type = 2 (Cost), CostID value is required.
- If Type = 3 (Labor), LaborID value is required.
- If Type = 4 (Expense), ExpenseID value is required.
- If Type = 6 (Shipping), ShippingID value is required.
- If Type = 10 (Discount) then you CANNOT HAVE a QuoteItem.LineDiscount value, that is, only QuoteItem.UnitDiscount and QuoteItem.PercentageDiscount are valid.
- If Type = 11 (Service), then

  ServiceID value is required.

  UnitCost will default to the unit cost set for that Service in the Admin module.
- If Type = 12 (ServiceBundle), then

  ServiceBundleID value is required.

  UnitCost will default to the sum of the unit costs for all Services within that bundle.
- If Type = 13 (ContractSetup), then

  QuoteItem.UnitCost must be ZERO (or not present), that is, there are no costs allowed on Setup Fees.

  QuoteItem.UnitDiscount, QuoteItem.PercentageDiscount, and QuoteItem.LineDiscount must be ZERO (or not present), that is, there are no discounts allowed on Setup Fees.
- A QuoteItem can have no discount or only one type of discount. If no discount is applied, UnitDiscount, PercentageDiscount, and LineDiscount must = 0. When a discount is applied, only one of these fields can have a value > 0. The other two must = 0.
- Any AllocationCode entity referenced by QuoteItem.CostID must be of Type = Material Cost.
- Any AllocationCode referenced by QuoteItem.ExpenseID must be of Type = Expenses.
- A Quote entity cannot have more than one QuoteItem of the same Service.
- A Quote entity cannot have more than one QuoteItem of the same ServiceBundle.
- The TaxCategoryID value for ProductID, CostID, and ExpenseID is pulled from the associated billing code; ShippingID or LaborID use the value passed in. If there is no value for TaxCategoryID, the item is not subject to tax.
- IsTaxable = True if the QuoteItem is assigned a TaxCategoryID, the associated Quote is assigned a TaxGroupID (Tax Region ID), and TotalEffectiveTax > 0.

  TotalEffectiveTax = the value of TaxID where Tax.TaxCategoryID = QuoteItem.TaxCategoryID and Tax.TaxRegionID = Quote.TaxGroupID.
- QuoteItems **can be deleted**.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AverageCost	average_cost	double	●				
CostID	Cost ID	integer			AllocationCode		
Description	Quote Item Description	string (2000)					
ExpenseID	Expense ID	integer			AllocationCode		
HighestCost	highest_cost	double	●				
id	Quote Item ID	long	●	●			
IsOptional	Is Optional	boolean		●			
IsTaxable	Taxable	boolean					
LaborID	Labor ID	integer			Role		
LineDiscount	Line Discount	double		●			
Name	Quote Item Name	string (100)					
PercentageDiscount	Discount Per-centage	double		●			
PeriodType	Period Type	string (50)				●	
ProductID	Product ID	integer			Product		
Quantity	Quantity	double		●			
QuoteID	Quote ID	integer	●	●	Quote		
ServiceBundleID	Service Bundle ID	integer			ServiceBundle		
ServiceID	Service ID	integer			Service		
ShippingID	Shipping ID	integer			ShippingType		
TaxCategoryID	Tax Category ID	integer			Tax Category		
TotalEffectiveTax	Tax Rate Applied	double	●				
Type	Quote Item Type	integer		●		●	
UnitCost	Unit Type	double					
UnitDiscount	Unit Discount	double		●			
UnitPrice	Unit Price	double					

QuoteLocation

This entity describes a location associated with an Autotask Quote. It defines address information for a Quote entity ShipToLocationID and/or BillToLocationID. Quotes are created and managed in Autotask through the CRM module.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: QuoteLocation
Can Create: •
Can Update: •
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Address1	Address Line 1	string (50)					
Address2	Address Line 2	string (50)					
City	City	string (50)					
id	Quote Location ID	long	•	•			
PostalCode	Postal Code	string (20)					
State	State/Province	string (50)					

Resource

This entity describes an Autotask Resource. Resources are employees, contractors, or consultants with access to a company's Autotask system. Autotask administrators manage Resources through the Admin Module (Admin >Site Setup > Resource Setup > Resources).

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	Resource
Can Create:	
Can Update:	System Administrators only
Can Query:	•
Can Delete:	
Can Have UDFs:	

Fields that Cannot Be Queried

The following fields from this entity will return an error when queried.

- DefaultServiceDeskRoleID
- Password

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- Resource update() is only available when logged into the Web Services with Autotask System Administrator level access.
- On query() or update(), if the date range for Internal Cost does not include the current date, the Internal Cost is not active and InternalCost value is set to 0.
- For Password field

  Not required on update() performed on existing Resource. If password is set, then we will store in DB.

  Query will never return the Password field as part of the entity returned (it will always be null).

  To update for an existing entity, set the Password field to the new password to be stored on update().

  If strong password is specified for the Autotask application, must enforce the following restrictions:
  1)Must be at least 7 characters
  2) Must have at least one special character in the second through sixth position. Special characters: ` ~

! @ # $ % ^ & * () _ + - = {} | \ [] : ; " ' < > ? , . /
3) Must contain characters from at least 2 of the following 3 groups: English uppercase letters A, B, C, ... Z; English lowercase letters a, b, c, ... z; Westernized Arabic numerals 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
4) Must be different from the resource's previous 5 passwords

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountingReferenceID	Accounting Reference ID	string (100)					
Active	Status	boolean		●		●	
DateFormat	Date Format	string (20)	●			●	
DefaultServiceDeskRoleID	Default Service Desk Role	long			Role		
Email	Email	string (50)		●			
Email2	Add Email 1	string (50)					
Email3	Add Email 2	string (50)					
EmailTypeCode	Email Type	string (20)		●			
EmailTypeCode2	Add Email 1 Type	string (20)					
EmailTypeCode3	Add Email 2 Type	string (20)					
FirstName	First Name	string (50)		●			
Gender	Gender	string (1)				●	
Greeting	Greeting	integer				●	
HomePhone	Home Phone	string (25)					
id	Resource ID	long	●	●			`
Initials	Pay Roll Iden-tifier	string (32)					

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
InternalCost	Internal Cost	double	●				
LastName	Last Name	string (50)		●			
LocationID	Pimary Location	integer	●	●		●	
MiddleName	Middle Initial	string (50)					
MobilePhone	Mobile Phone	string (25)					
NumberFormat	Number Format	string (20)		●		●	
OfficeExtension	Office Extension	string (10)					
OfficePhone	Office Phone	string (25)					
Password	Password	string (50)					
PayrollType	Payroll Type	integer		●		●	
ResourceType	Resource Type	string (15)		●		●	
Suffix	Suffix	string (10)				●	
TimeFormat	Time Format	string (20)	●			●	
Title	Title	string (50)					
TravelAvailabilityPct	Travel Avail-ability Pct	string (15)				●	
UserName	UserName	string (15)		●			
UserType	User Type	integer	●	●		●	

ResourceRole

This entity describes a Resource - Role relationship. A resource must be assigned to at least one role. Resources are assigned to one or more roles through their department and queue associations.

A role is required when a resource is assigned to a ticket or task. The role rate dictates the base rate for work the resource performs on that task or ticket.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ResourceRole
Can Create:	
Can Update:	
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
DepartmentID	Department	long	•		Department		
id	ID	long	•	•			
QueueID	Queue	long	•			Yes	
ResourceID	Resource	long	•	•	Resource		
RoleID	Role	long	•	•	Role		

Role

This entity describes an Autotask Role. Roles are associated with a department and have a standard billing rate. Resources are associated with one or more department/role combinations. Resources must specify a Role when entering time. When billing for that time, the default rate for the role determines the billable rate *unless* it is overridden, for example, by a Contract or WorkType.

Roles are managed through the Admin module. They are created through Site Setup > Company Setup > Roles tab and assigned to a resource when adding or editing the resource (Admin > Site Setup > Resource Setup > Resources > New or Edit) or adding the resource to a department (Admin > Site Setup > Company Setup > Departments tab).

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: Role
Can Create:
Can Update:
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Description	Description	string (200)	•				
HourlyFactor	Hourly Factor	double	•				
HourlyRate	Hourly Rate	double	•				
id	Role ID	long	•	•			

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Name	Name	string (200)	•				
SystemRole	System Role	boolean	•				

SalesOrder

This entity describes an Autotask Sales Order. In Autotask, a sales order is associated with an Opportunity and provides an option to track cost items generated from an Autotask Quote.

Sales orders are created automatically in Autotask when an Opportunity is closed using the Won Opportunity Wizard. Procurement must be enabled. Users can edit a sales order but cannot not manually create one.

NOTE: In Autotask, Sales Orders are only available to users when the Procurement module is enabled. The SalesOrder entity is always available in the API.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	SalesOrder
Can Create:	
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	•

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

Conditions and Requirements

In Autotask , the Ship To Country and Bill To Country fields now provide a countries pick list. The following conditions apply to the API for SalesOrder.ShipToCountry and SalesOrder.BillToCountry.

On create() and update():

If the value provided for Country matches either a standard country abbreviation, an ISO standard country name, or an Autotask country display name, the value is mapped to that country.

If no match is found, the value is mapped to "Other". The text that was passed in is stored.

In the UI, "Other" appears in the Country field as "Other [stored text value]". These "Other" values are not available for selection in the UI.

On query():

If the entity is mapped to a country, then the country display value will be returned.

If the country value is mapped to "Other", the stored text value is returned.

On query() by BillToCountry or ShipToCountry:

The system searches only the country display name.

If the entity's Country value is not mapped to an Autotask country, then you must query for Country = "Other"; that is, the string "Other", not the stored text value. This will return all entities where the country value is non-standard.

The system looks at both fields for each value passed in. For example:

If you query on BillToCountry = China and ShipToCountry = Australia, the system returns all SalesOrders where (BillToCountry = China or Australia) + (ShipToCountry = China or Australia).

If you query on BillToCountry = China and do not provide a value for ShipToCountry, the system returns all SalesOrders where (BillToCountry = China) + (ShipToCountry = China).

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountID	{LT:Account} Name	integer	●	●	Account		
AdditionalAddressInformation	Additional Address Information	string (100)					
AdditionalAddressInformation	Additional Address Information	string (100)					
BillToAddress1	Bill to Address1	string (150)					
BillToAddress2	Bill to Address2	string (150)					
BillToCity	Bill to City	string (50)					
BillToCountry	Bill to Country	string (100)					
BillToCountryID	Bill To Country ID	integer			Country		
BillToPostalCode	Bill to {LT:ZipCode}	string (50)					
BillToState	Bill to {LT:State}	string (50)					
Contact	Contact ID	integer		●	Contact		
id	Sales Order ID	integer	●	●			
OpportunityID	Opportunity ID	integer	●	●	Opportunity		

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
OwnerResourceID	Owner	integer		●	Resource		
PromisedDueDate	Promised Due Date	datetime					
SalesOrderDate	Sales Order Date	datetime		●			
ShipToAddress1	Ship To Address1	string (150)					
ShipToAddress2	Ship To Address2	string (150)					
ShipToCity	Ship To City	string (50)					
ShipToCountry	Ship To Country	string (100)					
ShipToCountryID	Ship To Country ID	integer			Country		
ShipToPostalCode	Ship To {LT:ZipCode}	string (50)					
ShipToState	Ship To {LT:State}	string (50)					
Status	Status	integer		●		●	
Title	Title	string (128)	●	●			

Service

This entity describes a deliverable item that represents a pre-defined unit of work performed for a set price, for example, a "Disk Backup" or "Virus Check" performed for one computer. Services are billed at regular intervals. In Autotask, a Service can be added as a component of a recurring service type contract. The customer is billed for each unit of service associated with the contract instead of billing separately for labor and parts each time the service is provided.

Services are created in the Autotask Admin module > Products and Services > Services page.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	Service
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- On create(), AllocationCodeID, Name, PeriodType, and UnitPrice are required.
- A newly created service is Active by default.
- On create(), an active Recurring Service contract Allocation Code is required.
- If a request is made to update a Service that is associated with an inactive Recurring Service Contract code, you can pass in the inactive code. You cannot, however, set AllocationCodeID for an existing Service to an inactive Recurring Service Contract code.
- create() or update() require System Administrator level security. There are no restrictions on query().

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AllocationCodeID	AllocationCode ID	integer		●	AllocationCode		
CreateDate	Create Date	datetime	●				
CreatorResourceID	Creator Resource ID	integer	●		Resource		
Description	Description	string (200)					
id	Service ID	long	●	●			
InvoiceDescription	Invoice Description	string (1000)					
IsActive	Is Active	boolean					
LastModifiedDate	Update Date	datetime	●				
Name	Name	string (100)		●			
PeriodType	Period Type	string (1)		●		●	
ServiceLevelAgreementID	Service Level Agreement Id	long	●			●	
UnitCost	Unit Cost	double					
UnitPrice	Unit Price	double		●			
UpdateResourceID	Update By ID	integer	●		Resource		
VendorAccountID	Vendor Account ID	integer			Account		

ServiceBundle

This entity describes a group of Service items that are priced and billed as one component of Recurring Service type contract. The group usually consists of Services that are performed together or at the same interval. The customer is billed for each service bundle associated with the contract instead of billing separately for each service in the bundle. The price for the bundle may offer a discount.

Service Bundles are created in the Autotask Admin module > Products and Services > Service Bundles page.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: ServiceBundle
Can Create:
Can Update:
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AllocationCodeID	AllocationCode ID	integer	•		AllocationCode		
CreateDate	Create Date	datetime	•				
CreatorResourceID	Creator Resource ID	integer	•		Resource		
Description	Description	string (200)	•				
id	Service Bundle ID	long	•	•			
InvoiceDescription	Invoice Description (1000)	string	•				

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
IsActive	Is Active	boolean	●				
LastModifiedDate	Update Date	datetime	●				
Name	Name	string (100)	●				
PercentageDiscount	Discount Percent	double	●				
PeriodType	Period Type	string (1)	●			●	
ServiceLevelAgreementID	Service Level Agreement Id	long	●			●	
UnitCost	Unit Cost	double	●				
UnitDiscount	Discount Dollars	double	●				
UnitPrice	Unit Price	double	●				
UpdateResourceID	Update By ID	integer	●		Resource		

ServiceBundleService

This entity describes a Service added to a ServiceBundle. A ServiceBundle describes Service items that are priced and billed as one component of Recurring Service type contract. Refer to "ServiceBundle" on page 179.

Service Bundles are created in the Autotask Admin module > Products and Services > Service Bundles page.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ServiceBundleService
Can Create:	•
Can Update:	
Can Query:	•
Can Delete:	•
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Conditions and Requirements

- A ServiceBundle cannot have duplicate ServiceBundleServices assigned; that is, you can assign an individual ServiceBundleService to a ServiceBundle only once. If you attempt to add a ServiceBundleService to the same ServiceBundle more than once, an error message opens.
- You cannot delete the last ServiceBundleService assigned to a ServiceBundle.
- Access to the Admin module is required to add a ServiceBundleService to a Service Bundle.
- The API respects the setting "Use sum of selected service unit prices". When this ServiceBundle setting is selected in the Autotask UI , the ServiceBundle extended price is automatically readjusted when a ServiceBundleService is assigned to or removed from the ServiceBundle.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
id	Service Bundle Service ID	long	•	•			

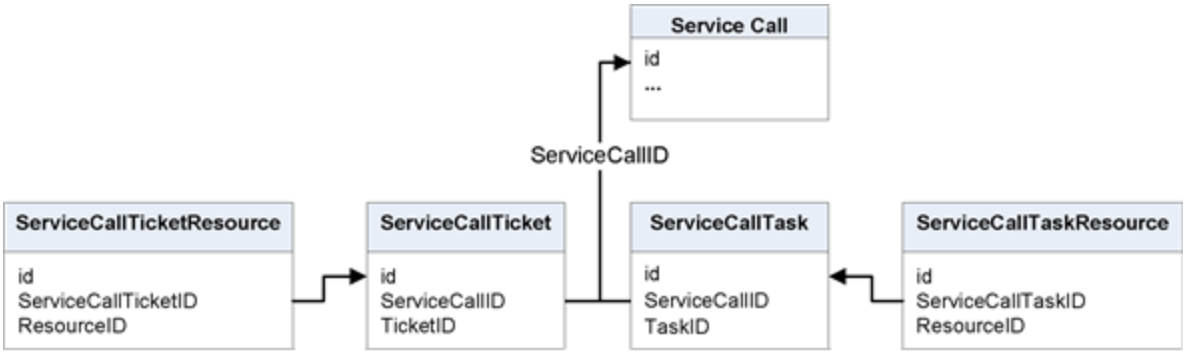
Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ServiceBundleID	Service Bundle ID	long	●	●			
ServiceID	Service ID	long	●	●			

ServiceCall

ServiceCall Entity Relationships

The Web Services API presents five service call related entities: ServiceCall, ServiceCallTicket, ServiceCallTask, ServiceCallTicketResource, and ServiceCallTaskResource. The following diagram illustrates the relationship between these entities.

NOTE: When using the service call entities, the external application must accommodate these relationships.



- Both "ServiceCallTicket" on page 188 and "ServiceCallTask" on page 185 are associated with a ServiceCall entity and require a ServiceCallID value.
- A "ServiceCallTicketResource" on page 189 is associated with a "ServiceCallTicket" on page 188 and requires a ServiceCallTicketID value. The same relationship exists between a "ServiceCallTaskResource" on page 186 and "ServiceCallTask" on page 185.

To view sample code demonstrating the relationship between the ServiceCall entities, see Appendix A, Sample Code for ServiceCall Entities.

ServiceCall Entity

This entity describes an Autotask service call. Service calls are instances of time, with specified start and stop times, that are scheduled to perform work for an Account. Tasks and/or tickets can be assigned to a service call and the call can be associated with one or more resources.

You manage service calls through the Service Desk module (Service Calls or Dispatcher's Workshop). You can also edit service calls through the Projects module if a task is assigned to the call, and through Service Desk > Tickets if a ticket is assigned to the call.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ServiceCall
Can Create:	•
Can Update:	•
Can Query:	•

Can Delete:

Can Have UDFs:

Field Details

The following table describes the Autotask standard ServiceCallentity fields. The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- There is a workflow rule event for "Service Call Scheduled". If an action performed through the UI creates a ServiceCallScheduled workflow rule event, then that action must trigger the workflow rule event when performed through the API.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountID	{LT:Account} ID	integer		●	Account		
Complete	Complete	short					
CreateDateTime	Create Date	datetime	●				
CreatorResourceID	Created By	integer	●				
Description	Description	string (2000)					
EndDateTime	End Date	datetime		●			
id	Service Call ID	long	●	●			
LastModifiedDateTime	Last Modified Date Time	datetime	●				
StartDateTime	Start Date	datetime		●			

ServiceCallTask

NOTE: This entity works in combination with the Service Call and other Service Call related entities. Before working with this entity, review the topic "ServiceCall" on page 183.

This entity describes an Autotask project task assigned to a service call. Service calls are instances of time scheduled to perform work for an account. A project task associated with a service call describes work that must be performed during the scheduled time.

Users create tasks through the Projects module and can assign tasks to a service call through the Projects module, the Service Desk module, or Dispatcher's Workshop.

A ServiceCallTask entity is associated with a ServiceCall entity and requires a ServiceCallID value; for additional information, see "ServiceCall" on page 183.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ServiceCallTask
Can Create:	•
Can Update:	
Can Query:	•
Can Delete:	•
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements:

- All tasks and tickets associated with a service call must be for the same account. There are no parent/child account loopholes.
- A single task or ticket can be associated with multiple service calls at the same time.
- Any resource can create service calls and associate/disassociate tasks/tickets to service calls.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
id	Service Call Task ID	long	•	•			
ServiceCallID	Service Call ID	integer		•	ServiceCall		
TaskID	Task ID	integer		•	Task		

ServiceCallTaskResource

NOTE: This entity works in combination with the Service Call and Service Call related entities. Before working with this entity, review the topic "ServiceCall" on page 183.

This entity describes an Autotask resource assigned to a task that is assigned to a service call. The service call task resource is the resource assigned to perform the task.

Users assign resources to a service call task through the Projects module or Dispatcher's Workshop. You can assign tasks to a service call through the Projects module, the Service Desk module, or Dispatcher's Workshop.

A ServiceCallTaskResource entity is associated with a ServiceCallTask entity and requires a ServiceCallTaskID value; for additional information, see "ServiceCall" on page 183.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ServiceCallTaskResource
Can Create:	•
Can Update:	
Can Query:	•
Can Delete:	•
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements:

- Only a resource that is a primary or secondary resource on ticket or task associated with a service call can be assigned to that service call.
- If a resource is disassociated from a ticket or task and that ticket or task was the resource's only association to the service call, then the resource must be disassociated from the service call.
- A resources can be disassociated from a service call. This will not disassociate the resource from the task or ticket on the service call.

  The resource can be re-associated with the service call at any time, as long as the resource remains associated with a task or ticket on the service call.
- A service call with no associated tasks or tickets will have no associated resources.
- Any resource can create service calls and associate/disassociate tasks/tickets to service calls (there does not appear to be any restriction enforced in the UI).

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
id		long	●	●			
ResourceID	Resource ID	integer		●	Resource		
ServiceCallTaskID	Service Call Task ID	integer		●	ServiceCallTaskID		

ServiceCallTicket

NOTE: This entity works in combination with the Service Call and Service Call related entities. Before working with this entity, review the topic "ServiceCall" on page 183.

The ServiceCallTicket entity describes an Autotask ticket assigned to a service call. Service calls are instances of time scheduled to perform work for a service request. A ticket associated with a service call describes work that must be performed during the scheduled time.

You create Tickets through the Service Desk module or by clicking the New Service Request button in the sub-navigation menu. You associate tickets to a service call when you create the service call, through the Service Desk module, or through Dispatcher's Workshop.

A ServiceCallTicket entity is associated with a ServiceCall entity and requires a ServiceCallID value; for additional information, see "ServiceCall" on page 183.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ServiceCallTicket
Can Create:	•
Can Update:	
Can Query:	•
Can Delete:	•
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements:

- All tasks and tickets associated with a service call must be for the same account. There are no parent/child account loopholes.
- A single task or ticket can be associated with multiple service calls at the same time.
- Any resource can create service calls and associate/disassociate tasks/tickets to service calls.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
id	Service Call Ticket ID	long	•	•			
ServiceCallID	Service Call ID	integer		•	ServiceCall		
TicketID	Ticket ID	integer		•	Ticket		

ServiceCallTicketResource

NOTE: This entity works in combination with the Service Call and Service Call related entities. Before working with this entity, review the topic "ServiceCall" on page 183.

This entity describes an Autotask resource assigned to a ticket that is assigned to a service call. The service call ticket resource is the resource assigned to perform the work described on the ticket. Users assign resources to a service call ticket when they add the ticket, through the Service Desk module, or through Dispatcher's Workshop.

A ServiceCallTicketResource entity is associated with a ServiceCallTicket entity and requires a ServiceCallTicketID value; for additional information, see "ServiceCall" on page 183.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	ServiceCallTicketResource
Can Create:	•
Can Update:	
Can Query:	•
Can Delete:	•
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements:

- Only a resource that is a primary or secondary resource on ticket or task associated with a service call can be assigned to that service call.
- If a resource is disassociated from a ticket or task and that ticket or task was the resource's only association to the service call, then the resource must be disassociated from the service call.
- A resources can be disassociated from a service call. This will not disassociate the resource from the task or ticket on the service call.

  The resource can be re-associated with the service call at any time, as long as the resource remains associated with a task or ticket on the service call.
- A service call with no associated tasks or tickets will have no associated resources.
- Any resource can create service calls and associate/disassociate tasks/tickets to service calls (there does not appear to be any restriction enforced in the UI).

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
id		long	●	●			
ResourceID	Resource ID	integer		●	Resource		
ServiceCallTicketID	Service Call Ticket ID	integer		●	ServiceCallTicketID		

ShippingType

This entity describes an Autotask Shipping Type. A Shipping Type defines a carrier for a product shipment and can be associated with a Quote entity.

In Autotask, Shipping Types are set up in the Admin module and added to a Quote in the CRM module.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: ShippingType
Can Create:
Can Update:
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AllocationCodeID	AllocationCodeID	integer	•		AllocationCode		
Description	Description	string (2000)	•				
id	Shipping Type ID	long	•	•			
IsActive	Is Active	boolean	•				
Name	Name	string (100)	•				

Task

This entity describes an Autotask Task. Tasks are associated with a Project and define work that must be done. Tasks can have one or more Resource assigned to them and can be scheduled for Service Calls. Autotask users manage Tasks through the Projects module and, when associated with a Service Call, through the Service Desk module or Dispatcher's Workshop.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	Task
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	•

Fields that Cannot Be Queried

The following fields from this entity will return an error when queried.

- RemainingHours

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

Conditions and Requirement

- AllocationCodeID and DepartmentID are not required unless the task has a primary or secondary Resource assigned. AllocationCodeID and Department ID are only required as follows:

  If the task has a Primary Resource (AssignedResourceID) assigned, an AllocationCodeID must be provided. DepartmentID is set to the Primary Resource's Department.

  If the task has only a secondary resource assigned, then both AllocationCodeID and DepartmentID must be provided.

  NOTE: Although the secondary resource is not exposed via the API, the API must still respect the AllocationCodeID and DepartmentID requirement.

- On update(), ProjectID cannot be changed. An error will be returned.
- AllocationCodeID must reference a Work Type allocation code.

- Autotask allows a Role to be inactivated. An attempt to create a Task using a Resource + Role combination with an inactive Role will trigger an error.

  You can update an existing Task if the Resource + Role combination uses an inactive Role.
- Priority is not required. If no value is provided on create() or update(), the value defaults to 0.
- If no value is provided for IsVisibleInClientPortal and CanClientPortalUserCompleteTask, they both default to False.
- If IsVisibleInClientPortal is False, then CanClientPortalUserCompleteTask will override to False.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AllocationCodeID	Allocation Code Name	integer			AllocationCode		
AssignedResourceID	Resource	integer			Resource		
AssignedResourceRoleID	Resource Role Name	integer			Role		
CanClientPortalUserCompleteTask	Can Client Portal User Complete Task	boolean					
CompletedDateTime	Task Complete Date	datetime	•				
CreateDateTime	Task Creation Date	datetime	•				
CreatorResourceID	Task Creator	integer	•		Resource		
DepartmentID	Task Department Name	integer				•	
Description	Task Description	string (8000)					
EndDateTime	Task End Datetime	datetime					
EstimatedHours	Task Estimated Hours	double					
ExternalID	Task External ID	string (50)					
id	Task ID	long	•	•			

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
IsVisibleInClientPortal	Is Visible In Client Portal	boolean					
LastActivityDateTime	Task Last Activity Date Time	datetime	●				
PhaseID	Phase ID	integer			Phase		
Priority	Task Priority	integer					
ProjectID	Project	integer		●	Project		
PurchaseOrderNumber	Purchase Order Number	string (32)					
RemainingHours	Remaining Hours	double					
StartDateTime	Task Start Date	datetime					
Status	Ticket Status	integer		●		●	
TaskIsBillable	Task Billable	boolean	●				
TaskNumber	Task Number	string (50)	●				
TaskType	Task Type	integer		●		●	
Title	Task Title	string (255)		●			

TaskNote

This entity describes notes created by an Autotask user and associated with a Task entity. Autotask users manage Task Notes on Project Tasks. Users can add notes to a new or existing task.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: TaskNote
Can Create: •
Can Update: •
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

NOTE: API queries for TaskNote entities with Publish = 1 include all System Workflow Notes. If you do not want system workflow notes returned, you must modify the query to include a condition that excludes TaskNote.NoteType = 13.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
CreatorResourceID	Creator Resource	integer	•		Resource		
Description	Description	String (3200)		•			
id	Task Note ID	long	•	•			
LastActivityDate	LastActivityDate	datetime	•				
NoteType	Note Type	integer		•		•	
Publish	Publish	integer		•		•	
TaskID	Task	integer		•	Task		
Title	Title	string (250)		•			

TaskPredecessor

This entity describes a predecessor/successor arrangement between two project schedule items. A predecessor item comes before one or more tasks (successor tasks) in the Project schedule. A successor task is scheduled to begin at the predecessor task's end date plus the number of "lag" days specified. for the predecessor.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	TaskPredecessor
Can Create:	•
Can Update:	• (LagDays only)
Can Query:	•
Can Delete:	•
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements:

- Currently, the API does **not** support project Phases as predecessors.
- Only LagDays can be updated.
- The start date of the Successor Task (SuccessorTaskID) will shift based on the Predecessor Task's (PredecessorTaskID) end date plus the number of Lag Days (LagDays) where specified.
- The Predecessor Task and Successor Task must be associated with the same Project.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
id	ID	long	•	•			
LagDays	Lag Days	Integer					
Predecessor TaskID	Predecessor Task ID	long		•	Task		
SuccessorTaskID	Successor Task ID	long		•	Task		

TaskSecondaryResource

This entity describes a secondary resource assigned to a project Task. Secondary resources are different from the primary resource. A Task can have more than one Secondary Resource assigned, and a task can have secondary resources without a primary resource assigned.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	TaskSecondaryResource
Can Create:	•
Can Update:	
Can Query:	•
Can Delete:	•
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements:

- The secondary resource cannot be the primary resource on the task (TaskSecondaryResource.Resource ID cannot equal Task.AssignedResourceID).
- ResourceID and RoleID must be an existing Resource/Role pair in Autotask.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
id	ID	long	•	•			
ResourceID	Resource ID	long		•	Resource		
RoleID	Role	long		•	Role		
TaskID	Task	long		•	Task		

Tax

The Tax entity describes the tax rate charged to a customer for specific goods or services purchased in a specified geographic area. The goods and services are represented by a TaxCategory. The geographic area is represented by the a TaxRegion. There can be multiple TaxCategories per TaxRegion. Tax entities

NOTE: Any change made to the Tax entity will impact all items that are billed after the change is made.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: Tax
Can Create: •
Can Update: •
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

Conditions and Requirements:

- Only one tax rate can be compounded per TaxRegion and TaxCategory combination.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
id	Tax ID	integer	•	•			
IsCompounded	Compounded	boolean					
TaxCategoryID	Tax Category ID	integer	•	•	TaxCategory		
TaxName	Tax Name	string (100)		•			
TaxRate	Tax Rate	double		•			

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
TaxRegionID	Tax Region ID	integer	●	●	TaxRegion		

TaxCategory

The TaxCategory entity describes the tax rate for a specific billing item. A TaxCategory associated with a TaxRegion determines the tax charged to customers.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: TaxCategory
Can Create: •
Can Update: •
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Active	Active	boolean					
Description	Tax Category Description	string (200)					
id	Tax Category ID	integer	•	•			
Name	Tax Category Name	string (200)		•			

TaxRegion

The TaxRegion entity describes a geographic area where billing items have the same tax rate. The TaxRegion together with the TaxCategory determine the total tax charged to customers.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: TaxRegion
Can Create: •
Can Update: •
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Active	Active	boolean					
id	Tax Region ID	integer	•	•			
Name	Tax Region Name	string (200)		•			

Ticket

This entity describes an Autotask Ticket. Tickets define service requests within the Autotask system. Autotask users manage Tickets through a number of modules including Service Desk, Home, Directory, CRM, and Contracts. They can click New Ticket on the Autotask interface sub-navigation menu to open the New Ticket window.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	Ticket
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	•

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

NOTE: Although the Autotask interface now allows multiple resources on a ticket, the current versions of the Web Services API **do not** support this feature. Web Services is, however, aware of and will check existing multiple resources on a ticket and will not allow any resource to be assigned as primary resource if that resource is already a secondary resource.

Conditions and Requirements:

- On create(), if InstalledProductID is populated, the InstalledProduct.AccountID must = Ticket.AccountID
- On update(), InstalledProduct.ID cannot be updated to an InstalledProduct where InstalledProduct.AccountID ≠ Ticket.AccountID. Note: If the InstalledProduct value is not being updated, and for some reason it is already associated with an Account that is different from the Ticket Account, the update() will not fail.
- For the ContactID field, Contact.AccountID must = Ticket.AccountID or the ParentAccountID of Ticket.AccountID.
- update() is allowed on a Ticket with an inactive ContactID value if that value is not being changed, or if a new active value is assigned. A new inactive ContactID value cannot be assigned on create() or update().
- On create(), Priority must be an active priority.
- If the current priority is inactive, update() is allowed if the Priority value is not changed, or if Priority is

changed to an active value.

- The AllocationCodeID field must reference a Work Type allocation code.
- AllocationCodeID is required on create() and update() if your company has enabled the Autotask Workflow Policy that requires a Work Type on a Ticket.
- Autotask now allows Role to be inactivated. An attempt to create a Ticket using a Resource + Role combination with an inactive Role will trigger an error.

  You can update an existing Ticket that has a Resource + Role combination that uses an inactive role.
- update() is allowed on a Ticket with an inactive attribute value if that value is not being changed. A new inactive attribute value cannot be assigned.
- For the OpportunityID field, Opportunity.AccountID must = Ticket.AccountID.
- TicketType must = Incident before the ticket can be associated with a ProblemTicketID.
- ProblemTicketID cannot = TicketID of a ticket that is already associated with a ProblemTicketID; that is, an incident ticket already associated with a problem ticket cannot become a problem ticket.
- You cannot create a ticket with TicketType = Problem and specify a ProblemTicket ID or specify ProblemTicketID for an existing Ticket with TicketType=Problem; that is, a ticket that is already a problem ticket cannot become an incident to another problem ticket.
- Tickets where Type = Service Request cannot be associated with a ProblemTicket ID. If TicketType = Service Request and the ticket also specifies a ProblemTicketID, the ticket type is updated to Service Request. An error message indicates that Service Request tickets cannot be associated with a problem ticket.
- Tickets with no TicketType specified are set automatically to Service Request.
- If TicketType = Problem and incidents are associated with the ticket, TicketType cannot be changed to Incident or Service Request until the incidents are disassociated from the ticket.
- ChangeApprovalBoard must reference an active Change Approval Board.
- ChangeApprovalType equals the default value if no value is set.
- ChangeApprovalStatus can only be set to Requested or Assigned. All other statuses, Not Assigned, Partially Approved, Approved, or Rejected can only be set by the system.

  If ChangeApprovalStatus = Assigned, user can change it to Requested (only).

  If ChangeApprovalStatus = Requested, user can change it to Assigned (only)
- ChangeInfoFields are available regardless of whether they are Active or Inactive.
- If a ticket has TicketType not equal to "Change Request" and it has data in one or more the fields that are exclusive to Change Request tickets, then the ticket can be saved. Although the data will remain intact and will be reportable, it will not be viewable in the ticket in Autotask. This includes the following fields: ChangeApprovalBoard, ChangeApprovalType, ChangeApprovalStatus, ChangeInfoField1, ChangeInfoField2, ChangeInfoField3, ChangeInfoField4, ChangeInfoField5.
- If TicketType = ChangeRequest, ProblemTicketID cannot have a value.
- The ChangeManagement module must be enabled to create a new ticket with TicketType = Change Request. Any existing Change Request tickets can be edited.

Other fields related to change request will accept values when Change Management is not enabled, but that data will not be available through the UI.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Behaviors

- The Web Services API stores and returns all time data in Eastern Standard Time (EST).

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AccountID	{LT:Account}	integer		•	Account		
AllocationCodeID	Allocation Code Name	integer			AllocationCode		
AssignedResourceID	Resource	integer			Resource		
AssignedResourceRoleID	Resource Role Name	integer			Role		
ChangeApprovalBoard	Change Approval Board ID	integer				•	
ChangeApprovalStatus	Change Approval Status	integer				•	
ChangeApprovalType	Change Approval Type	integer				•	
ChangeInfoField1	Change Info Field 1	string (8000)					
ChangeInfoField2	Change Info Field 2	string (8000)					
ChangeInfoField3	Change Info Field 3	string (8000)					
ChangeInfoField4	Change Info Field 4	string (8000)					
ChangeInfoField5	Change Info Field 5	string (8000)					
CompletedDate	Ticket Date Completed by Complete Project Wizard	datetime	•				
ContactID	Ticket Contact	integer			Contact		

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ContractID	Contract	integer			Contract		
CreateDate	Ticket Creation Date	date	•				
CreatorResourceID	Ticket Creator	integer	•		Resource		
Description	Ticket Description	string (8000)					
DueDateTime	Ticket End Date	datetime		•			
EstimatedHours	Ticket Estimated Hours	double					
FirstResponseDateTime	First Response Date Time	datetime	•				
FirstResponseDueDateTime	First Response Due Date Time	datetime	•				
id	Ticket ID	long	•	•			
InstalledProductID	{LT:InstalledProduct}	integer			InstalledProduct		
IssueType	Ticket Issue	integer				•	
LastActivityDate	Ticket Last Activity Date	datetime	•				
LastCustomerNotificationDateTime	Last Customer Notification DateTime	datetime	•				
LastCustomerVisibleActivityDateTime	Last Customer Visible Activity DateTime	datetime	•				
OpportunityId	Opportunity ID	integer			Opportunity		
Priority	Ticket Priority	integer		•		•	
ProblemTicketId	Problem Ticket ID	integer			Ticket		
PurchaseOrderNumber	Purchase Order Number	string (32)					
QueueID	Ticket Department Name OR Ticket Queue Name	integer				•	
Resolution	Resolution	string (32000)					

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ResolutionPlanDateTime	Resolution Plan Date Time	datetime	●				
ResolutionPlanDueDateTime	Resolution Plan Due Date Time	datetime	●				
ResolvedDateTime	Resolved Date Time	datetime	●				
ResolvedDueDateTime	Resolved Due Date Time	datetime	●				
ServiceLevelAgreement HasBeenMet	Has Met SLA	boolean	●				
ServiceLevelAgreementID	Service Level Agreement ID	integer				●	
Source	Ticket Source	integer				●	
Status	Ticket Status	integer		●		●	
SubIssueType	Ticket Subissue Type	integer				●	
TicketNumber	Ticket Number	string (50)	●				
TicketType	Ticket Type	integer				●	
Title	Ticket Title	string (255)		●			

TicketChangeRequestApproval

This entity describes a record of approval for a ticket change request. The change request approval process is part of the Autotask Change Management feature set. Change Management features are only available in the Autotask UI when the Change Management module is enabled.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	TicketChangeRequestApproval
Can Create:	•
Can Update:	
Can Query:	•
Can Delete:	•
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

Conditions and Requirements

- TicketID must be a valid ID for a ticket where TicketType = ChangeRequest.
- Users can update data on behalf of another user.
- ResourceID must be a valid ID for an Active resource.
- A single ChangeRequestApproval record must have 1 ResourceID or 1ContactID. The record cannot have both a resource and contact.
- When a new ResourceID is added by the user, the new resource will be added as an Other Resource Approver.

  If the resource is an Other Resource Approver and is not a member of the Change Approval Board, then that Resource Approver can be deleted from the ticket. Deleting the resource will also delete all associated vote data.
- If the Resource indicated by the Resource ID is a member of the Change Approval Board assigned to the associated ticket (TicketID), that Resource cannot be deleted from the ticket.
- ContactID must be a valid ID for an Active contact.
- If the ContactID is not the ticket contact, primary account contact, or parent account primary contact, then the contact will be added as an Other Contact Approver.

If the contact is an Other Contact Approver and is not a member of the Change Approval Board (ticket contact, primary account contact, or parent account primary contact), then that Contact Approver can be deleted from the ticket. Deleting the contact will also delete all associated vote data.

- If the ContactID is associated a Contact who is member of the Change Approval Board assigned to the associated ticket (TicketID), the ContactID cannot be deleted from the ticket.
- ApproveRejectDateTime is read only and set when IsApproved is set to approve or reject.

  This Field cannot have a value if IsApproved is null.
- ApproveRejectNote cannot have a value if IsApproved is null.
- IsApproved can equal one of the following options:

  null = has not yet been approved or rejected (default state)

  true = approved

  false = rejected

  NOTE: The only time IsApproved can be null is when a Resource or Contact is being added to the ticket as an approver. If the Resource or Contact is already an approver, then IsApproved cannot be null.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ApproveRejectDateTime	Approve Reject DateTime	datetime	●				
ApproveRejectNote	Approve Reject Note	string (2000)					
ContactID	Contact ID	integer			Contact		
id	Change Request Ticket Vote ID	integer	●	●			
IsApproved	Is Approved	boolean					
ResourceID	Resource ID	integer			Resource		
TicketID	Ticket ID	integer		●	Ticket		

TicketCost

This entity describes a cost associated with an Autotask Ticket. A cost is a billing item for products or materials. Cost items can be billable or non-billable. Billable cost items appear in Approve and Post.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	TicketCost
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	• see conditions below
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

Conditions and Requirements

- To create(), update(), delete(), and query() TicketCost, the logged in user must have Security Level permission to access the Service Desk module or must be assigned to the ticket as a primary or secondary resource.
- TicketCost must have either a ProductID or AllocationCodeID.
- TicketID cannot be updated.
- update() and delete() are allowed only when Billed = False (cost has not been approved and posted).
- Only query() is allowed if Billed = True (cost has been approved and posted).
- AllocationCodeID must reference a Material Cost Code type allocation code.
- On create() or update() when ProductID has changed and is not Null, if no value is supplied for the AllocationCodeID, UnitCost, or UnitPrice fields, then those fields take their values from the Product.
- On create() or update() when AllocationCodeID has changed and is not Null, and if no value is supplied for the UnitCost and UnitPrice fields, then those fields take their values from the Material Cost Code associated with AllocationCodeID.
- If no value is supplied for BillableToAccount, the value is set to True.
- Status is read only. On create(), Status is set as follows:

  If ProductID is null, then Status = Ready to Ship.

If ProductID references a Product where DoesNotRequireProcurement = True, then Status = Ready to Ship.

If ProductID references a Product where DoesNotRequireProcurement = False and the Product is not available in Inventory, then Status = Need To Order.

If ProductID references a Product where DoesNotRequireProcurement = False and the Product is available in Inventory, then Status = Ready to Ship.

If Status = Need To Order and ExtendedCost is > the value set in the workflow policy "Require approval before ordering..." then Status is automatically set to Waiting Approval.

- ExtendedCost = UnitQuantity * UnitCost. If UnitCost is Null, the value will be set to the value from the Material Cost Code associated with AllocationCodeID.
- BillableAmount = UnitQuantity * UnitPrice. If UnitPrice is Null, the value will be set to the value from the Material Cost Code associated with AllocationCodeID.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AllocationCodeID	Allocation Code	long			AllocationCode		
BillableAmount	Billable Amount	double	•				
BillableToAccount	Billable To Account	Boolean					
Billed	Billed	Boolean	•				
ContractServiceBundleID	Contract Service Bundle ID	long			ContractServiceBundle		
ContractServiceID	Contract Service ID	long			ContractService		
CostType	Cost Type	integer		•		•	
CreateDate	Create Date	datetime	•				
CreatorResourceID	Created By	long	•		Resource		
DatePurchased	Date Purchased	datetime		•			

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
Description	Product Description	string (2000)					
ExtendedCost	Extended Cost	double	●				
id	id	long	●	●			
InternalPurchaseOrderNumber	Internal Purchase Order Number	string (50)					
Name	Name	string (100)		●			
ProductID	Product	long			Product		
PurchaseOrderNumber	Purchase Order Number	string (32)					
Status	Status	long	●			●	
StatusLastModifiedBy	Last Modified By	long	●				
StatusLastModifiedDate	Last Modified Date	datetime	●				
TicketID	Ticket	long		●	Ticket		
UnitCost	Unit Cost	double					
UnitPrice	Unit Price	double					
UnitQuantity	Unit Quantity	double		●			

TicketNote

This entity describes notes created by an Autotask user and associated with a Ticket entity. Autotask users manage ticket notes on Service Desk tickets. Users can add notes to a new or existing ticket.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: TicketNote
Can Create: •
Can Update: •
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

NOTE: With the current redesign of Autotask's workflow engine, API queries for TicketNote entities with Publish = 1 now include all System Workflow Notes. If your query currently includes code that specifies TicketNote.Publish = 1 and you do not want system workflow notes returned, you must modify the query to include a condition that excludes TicketNote.NoteType = 13.

Conditions and Requirements:

- The TicketNote entity must respect the edit ticket note permissions implemented with Autotask release 2013.2 (October 22,). In Autotask these permissions are assigned to the user's Security Level. For details on Security Level permissions, refer to https://ww1.autotask.net/Help/Content/AdminSetup/SiteSetup/Security/CustomizingSecurityLevels.htm.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
CreatorResourceID	Creator Resource	integer	•		Resource		
Description	Description	String (3200)		•			
id	Ticket Note ID	long	•	•			

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
LastActivityDate	LastActivityDate	datetime	●				
NoteType	Note Type	integer		●		●	
Publish	Publish	integer		●		●	
TicketID	Ticket	integer		●	Ticket		
Title	Title	string (250)		●			

TicketSecondaryResource

This entity describes a secondary resource assigned to a Ticket. Secondary resources are different from the primary resource. A Ticket can have more than one Secondary Resource assigned, and can have secondary resources without a primary resource assigned.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	TicketSecondaryResource
Can Create:	•
Can Update:	
Can Query:	•
Can Delete:	•
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements:

- The secondary resource cannot be the primary resource on the ticket (TicketSecondaryResource.Resource ID cannot equal Ticket.AssignedResourceID).
- ResourceID and RoleID must be an existing Resource/Role pair in Autotask.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
id	ID	long	•	•			
ResourceID	Resource ID	long		•	Resource		
RoleID	Role	long		•	Role		
TicketID	Task	long		•	Ticket		

TimeEntry

This entity describes an Autotask Time Entry. A time entry allows an Autotask resource to enter time against a Ticket or Task and to enter General or Regular time, for example, in-house meeting, travel, or training time. Users enter time through a number of modules including Home, Contracts, CRM, Directory, Projects, Service Desk, and Timesheets.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	TimeEntry
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

NOTE: The AllocationCodeID field represents the value displayed as "Work Type" in the Autotask interface

Conditions and Requirements:

For All TimeEntries

- TimeEntry now respects Autotask Proxy Time Entry; that is, when Proxy Time Entry is enabled, TimeEntry create() and update() are no longer limited to the TimeEntry resource.

  When the Proxy Time Entry workflow policy is set to Enabled for timesheet approvers, a timesheet approver for the TimeEntry Resource can create and update a TimeEntry.

  When the workflow policy is set to Enabled for timesheet approvers and Administrators, timesheet approvers can create and update Time Entries for those resources whose timesheets they approve, and Administrators can create and update Time Entries for any active resource.

  When the Proxy Time Entry policy is disabled, only TimeEntry.ResourceID may edit his or her TimeEntry.
- TimeEntry.HoursWorked must be >0 and <= 24.
- TimeEntry.StartDateTime must be < TimeEntry.EndDateTime.
- A TimeEntry may only be added to a timesheet with the status of New or Rejected (not Submitted, Approved. etc.).

- A TimeEntry cannot be updated if it has been approved and posted.
- A TimeEntry will be rounded if the workflow policies governing rounding are enabled.
- A False value for TimeEntry.NonBillable requires a TimeEntry.TicketID or TimeEntry.TaskID value. Time entries that are not associated with a Task or Ticket must be non-billable.
- The TimeEntry fields BillingApprovalDateTime, BillingApprovalLevelMostRecent, and BillingApprovalResourceIDare updated automatically on creation of a new BillingItemApprovalLevel entity row that references TimeEntry.id.
- BillingApprovalResourceID and BillingApprovalDateTime must both either be null or contain a value at the same time. One cannot be populated without the other having a value.

For Task and Ticket TimeEntry

- AllocationCodeID must reference a Work Type (General) allocation code.
- AllocationCodeId must reference an active AllocationCode.
- User cannot pass through AllocationCodeID, Contract, NonBillable, or ShowOnInvoice values without the required Security Level permissions. No permissions are required to leave any of the four fields blank.
- An AllocationCodeID and/or ContractID value requires a TicketID or TaskID value. Allocation Codes (Work Type) and Contracts must be associated with a task or ticket.
- Ticket and Task Time Entries must include a RoleID that is valid for the ResourceID. The RoleID must be the same as TicketID.AssignedResourceRoleID or TaskID.AssignedResourceRoleID except when the workflow policy to "Allow Resource to Modify Role" is enabled.
- An attempt to create a TimeEntry using a Resource + Role combination with an inactive Role will trigger an error.

  You can update an existing TimeEntry that has a Resource + Role combination that uses an inactive Role.
- A True value for ShowOnInvoice requires a TicketID or TaskID value. Time entries that are not associated with a Task or Ticket cannot display on an invoice.
- ShowOnInvoice cannot = True if the ContractID references a Flat Rate or Recurring Service type contract.
- ContractID must reference a contract that is active for the associated account or its parent.
- A Ticket TimeEntry must have SummaryNotes.
- A Ticket TimeEntry must have TimeEntry.StartDateTime and TimeEntry.EndDateTime values, that is, they must use start and stop time.
- TimeEntry may not be created for a TicketID whereTicket.Status = Complete if the Service Desk workflow policy to prohibit time entry on a completed ticket is enabled.
- Resources without access to the Service Desk module can only create or update TimeEntries for the tickets assigned to them.
- A Task TimeEntry must have a valid RoleID for the Resource and Task.
- Task TimeEntries must have TimeEntry.StartDateTime and TimeEntry.EndDateTime values, that is, they must use start and stop time, when the associated Project is set to "Requires Start/Stop Times".

- A Task TimeEntry must have TimeEntry.StartDateTime and TimeEntry.EndDateTime values, that is, they must use start and stop time, when the Project Task Start/End time workflow policy is enabled.
- The SummaryNotes field is required for Task TimeEntry when the Time Summary Requirements workflow policy is enabled.
- Task Time Entry by a Resource that is not assigned as primary or secondary resource on the task is allowed only when the "Enable My Tasks Only checkbox when entering time" workflow policy is enabled.
- A TimeEntry cannot be created for a task associated with a Project where Project.Type = Archived (1), Template (3), Baseline (8).

For General TimeEntry

- On a General TimeEntry (not for task/tickets) the InternalAllocationCode value must be present and it must be of type "InternalActivity" or "GeneralActivity".
- "Floating Holiday" time is allowed only if the Floating Holidays workflow policy is enabled and the Floating Holiday internal allocation code is set to Display in Regular Time (Admin > Site Setup > Company Setup > Internal Allocation Codes tab > Display in Regular Time is selected for Floating Holiday)
- "Personal Time" time is allowed only if the Personal Time workflow policy is enabled and the Personal Time internal allocation code is set to Display in Regular Time (Admin > Site Setup > Company Setup > Internal Allocation Codes tab > Display in Regular Time is selected for Personal Time)
- "Sick Time" time is allowed only if the Sick Time workflow policy is enabled and the Sick Time internal allocation code is set to Display in Regular Time (Admin > Site Setup > Company Setup > Internal Allocation Codes tab > Display in Regular Time is selected for Sick Time)
- "Vacation Time" time is allowed only if the Vacation Time workflow policy is enabled and the Vacation Time internal allocation code is set to Display in Regular Time (Admin > Site Setup > Company Setup > Internal Allocation Codes tab > Display in Regular Time is selected Vacation Time)
- For General TimeEntry (not for task/tickets) the RoleID defaults to the resource's default RoleID.
- The SummaryNotes field is required for General time entries when the Time Summary Requirements workflow policy is enabled.

When querying TimeEntry entities, the following condition applies.

- To limit the return to only ticket time entries or only task time entries, you must specify a Type value. Picklist values for Type equal 2 (ITServiceRequest) for use with TicketID or 6 (ProjectTask) for use with TaskID.

Behaviors:

The Web Services API TimeEntry entity displays the following behaviors on create() and update().

For All TimeEntries

- All TimeEntries made through the API are in Eastern Standard Time (EST).

For Task and Ticket TimeEntry

- On create(), if no value is provided for the AllocationCodeID, ContractID, NonBillable, or ShowOnInvoice fields, the field will be populated by the value inherited from the associated task, issue, or ticket.
- On update(), if no value is provided for AllocationCodeID, ContractID, NonBillable, or ShowOnInvoice fields, the fields *do not* inherit the value from the associated task, issue, or ticket. It is presumed that they were changed to show no value.

NOTE: A TimeEntry cannot be created without an AllocationCodeID and/or ContractID if its associated task, issue or ticket has one; the TimeEntry will automatically inherit the values. If the intention is to have a TimeEntry without an AllocationCodeID or ContractID, it must be updated after it is created.

- If TimeEntry.HoursWorked is left empty, the API calculates a value from the start & end times; that is, the value of TimeEntry.HoursWorked = TimeEntry.EndDateTime – TimeEntry.StartDateTime.
- The Web Services TimeEntry entity does not support the Autotask interface feature "Bill Immediately". All time entries created and updated through the Web Services API must be approved and posted.
- In the Autotask interface, you cannot enter time on Projects of Type = Archived, Template, Baseline. Although currently the Web Services API allows this, it will be prohibited in the future. We strongly recommend that API users not allow time entry on Archived, Template, or Baseline project types.
- If the Contract specified on the TimeEntry excludes a Role or AllocationCodeID (Work Type) specified on the TimeEntry, the default exclusion contract is applied. If the original contract does not specify a default exclusion contract, no contract will be applied.

For General TimeEntry

- General time (not customer facing) will always be Non-Billable and will never show on invoice.
- On General TimeEntry (not for task/tickets) the RoleID will always be the default role for the resource, regardless of what value is provided.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term.

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
AllocationCodeID	Allocation Code ID	integer			AllocationCode		
BillingApprovalDateTime	Billing Approval Date Time	datetime					
BillingAp-provalLevelMostRecent	Billing Approval Level Most Recent	integer	•				
BillingApprovalResourceID	Billing Approval Resource ID	integer			Resource		
ContractID	Contract ID	integer			Contract		

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ContractServiceBundleID	Contract Service Bundle ID	long			ContractServiceBundle		
ContractServiceID	Contract Service ID	long			ContractService		
CreateDateTime	Create Date Time	datetime	•				
CreatorUserID	Creator User ID	integer	•				
DateWorked	Date	datetime		•			
EndDateTime	End Date Time	datetime					
HoursToBill	Hours To Bill	double	•				
HoursWorked	Hours Worked	double					
id	Time Entry ID	long	•	•			
InternalAllocationCodeID	Internal Allocation Code ID	integer			AllocationCode		
InternalNotes	Internal Notes	string (8000)					
LastModifiedDateTime	Last Modified Datetime	DateTime	•				
LastModifiedUserID	Last Modified User ID	integer	•				
NonBillable	Non-Billable	boolean					
OffsetHours	Offset Hours	double					
ResourceID	Resource ID	integer		•	Resource		
RoleID	Role ID	integer		•	Role		

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ShowOnInvoice	Show On Invoice	boolean					
StartDateTime	Start Date Time	datetime					
SummaryNotes	Summary Notes	string (8000)					
TaskID	Task ID	integer			Task		
TicketID	Ticket ID	integer			Ticket		
Type	Task Type Link	integer		•		•	

UserDefinedFieldDefinition

This entity defines a User-defined Field (UDF) in Autotask. User-defined Fields are custom fields that each Autotask customer can add to their Account, Contact, Opportunity, Sales Order, Projects, Products, Configuration Items, Ticket, and Task tables. These fields hold data that is unique to the Autotask customer's company and cannot be mapped to the standard Autotask field sets.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name:	UserDefinedFieldDefinition
Can Create:	•
Can Update:	•
Can Query:	•
Can Delete:	
Can Have UDFs:	

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244.

Conditions and Requirements

- A User-defined Field (UDF) cannot be deleted via the API.
- After creation, UDFType cannot be updated via the API.
- After creation, DataType cannot be updated via the API.
- A value for the field MergeVariableName must be unique. It can contain only alpha characters (letters) and must begin with "var", for example, varCustomData.
- DefaultValue must be a valid format for the UserDefinedFieldDefinition DataType:

  String (single or multi-line) - Accepts an alphanumeric string.

  Date - Accepts any valid date format.

  Numeric - Accepts numbers. UserDefinedFieldDefinition.NumberOfDecimalPlaces defines how many decimal places are allowed.

  List - List values are defined by the UserDefinedFieldListItem entity; default value must be a UserDefinedFieldListItem.ValueForDisplay where UserDefinedFieldListItem.UdfFieldID = UserDefinedFieldDefinition.id
- For a UserDefinedFieldDefintion with DataType = List, DefaultValue cannot be set during creation because DefaultValue is reference to a UserDefinedFieldListItem. The UserDefinedFieldListItem cannot

be created before the associated UserDefinedFieldDefinition. Refer to "UserDefinedFieldListItem" on page 224.

- CrmToProjectUdfId requires that UserDefinedFieldDefinition.UdfType = Project; otherwise, an error is thrown.

  CrmToProjectUdfId must reference a UserDefinedFieldDefinition.id. The referenced UserDefinedFieldDefinition must be an Opportunity type. Both UserDefinedFieldDefinition entities (the Project type and the Opportunity type) must be the same DataType.

- If DataType = string, DisplayFormat determines if the field is single line or multi-line. If DataType = string and DisplayFormat = 0, DisplayFormat will default to 1 (single line).

  If DataType has a value other than string, DisplayFormat will =0 (undefined).

- SortOrder must be > 0. If no value is provided or the value is <0, it will default to 1.
- NumberOfDecimalPlaces must be >0 and <= 4. If no value is provided it will default to 2.
- IsProtected will = false unless the UdfType = Product or Site Configuration in which case it takes the value provided.
- IsVisibleToClientPortal will always = false unless the UDFType is Installed Product or Ticket in which case it will take the provided value.
- IsFieldMapping = True indicates a System UDF created by Autotask. System UDFs cannot be created, edited, or deleted. This field defaults to False for all user created User-Defined Fields.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
CreateDate	Create Date	datetime	●				
CRMToProjectUDF	CRM to Project UDF id	long					
DataType	Data Type	integer		●		●	
DefaultValue	Default Value	string (1024)					
Description	Description	string (128)					
DisplayFormat	Display Format	integer				●	
id	id	long	●	●			
IsActive	Active	boolean					
IsFieldMapping	Field Mapping	boolean	●				
IsProtected	Protected	boolean					

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
IsRequired	Required	boolean					
IsVisibleToClientPortal	Visible To Client Portal	boolean					
MergeVariableName	Merge Variable Name	string (100)					
Name	Name	string (45)		•			
NumberOfDecimalPlaces	Number Of Decimal Places	integer					
SortOrder	Sort Order	integer					
UDFType	UDF Type	integer		•		•	

UserDefinedFieldListItem

This entity describes a list item associated with a UserDefinedFieldDefinition entity that has DataType = List.

Entity Details

You can also retrieve this information with the Web Services API call "getEntityInfo()" on page 242.

Entity Name: UserDefinedFieldListItem
Can Create: •
Can Update: •
Can Query: •
Can Delete:
Can Have UDFs:

Field Details

The following table describes the standard Autotask field objects for this entity. To retrieve more detailed information specific to a particular Autotask implementation, use the Web Services API call "getFieldInfo()" on page 244. For information on entity UDFs, use "getUDFInfo()" on page 248.

Conditions and Requirements

- A UserDefinedListItem requires a UdfFieldId that references an existing UserDefinedFieldDefinition.
- ValueForDisplay cannot be changed if the UserDefinedFieldListItem is the default value for a UserDefinedFieldDefinition.
- ValueForDisplay and ValueForExport are both required and must be unique within the collection of UserDefinedFieldListItems associated to a UserDefinedFieldDefinition.

In the following table:

For String Datatypes, the number in parentheses ( ) indicates the maximum number of characters allowed.

LT indicates Local Term

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
CreateDate	Create Date	datetime	•				
id	id	long	•	•			
UdfFieldId	User Defined Field Definition	long		•	UserDefinedFieldDefintion		
ValueForDisplay	Value For Display	string (128)		•			

Field Name	Label	Datatype	Read Only	Is Required	Reference Name	Picklist	Picklist Parent Column Name
ValueForExport	Value For Export	string (128)		•			

User-defined Fields (UDFs)

A User-defined Field (UDF) is a custom field created in Autotask by customers to collect and store data that is not included in the default Autotask entity fields. Autotask administrators create UDFs in the Administration Settings of the Autotask system.

Not all entities use UDFs. The following entities currently use UDFs: Account, AccountLocation, Contact, InstalledProduct (Configuration Items), Opportunity, Product, Project, Task, Ticket.

To determine if an Entity can use UDFs, locate the Entity in the list of "Entities" on page 48 and click to open the entity's topic . Alternately, use the getEntityInfo() and getUDFInfo().

- "getEntityInfo()" on page 242 will show any UDFs being used.
- "getUDFInfo()" on page 248 returns a list of all UDFs for a specified entity and descriptive information about the fields.

The Web Services API permits you to retrieve values for all entity UDFs via the query method. You may also update UDFs for entity types that allow update() and create().

Considerations

Permissions	Client must be logged in with a user account that has access to the appropriate Autotask module for the entity type and with sufficient rights to perform the specified API call.

Conditions and Requirements

- UDF fields required in the UI are not required for the API.

  If no value is provided for a required UDF on create(), and a default exists in Autotask, the default will populate the field.

  If no value is provided for a required UDF on create(), and there is no Autotask default, the field remains unpopulated.

  If no value is provided on update() and the field is not currently populated, the field remains unpopulated.

  If no value is provided on update() and the field is currently populated,the current value remains in place.

  On update(), if a required field is currently populated, the current value cannot be removed without providing a new value.

UDF update() Sample Code

The following code updates a UDF with the UDF name LicenseNumber.

```
Dim ticketNew As New WebReference.Ticket

Dim udf As New WebReference.UserDefinedField

udf.Name = "LicenseNumber"

udf.Value = "123-ABC"
```

```
ticketNew.UserDefinedFields = New WebReference.UserDefinedField() {udf}

Dim sResponse As WebReference.ATWSResponse

sResponse = myService.update(New WebReference.Entity() {CType(ticketNew, WebRe-
ference.Entity)})
```

API Calls

The Web Services API calls allow you to create, query, and update the Autotask entities available in the API. Not all API calls are available for each entity type.

Additional calls, getEntityInfo(), getFieldInfo(), and getUDFInfo(), allow you to retrieve information on all currently available entities, entity fields, and entity user-defined fields.

getZoneInfo() and getThresholdAndUsageInfo() return information about your access to the API, for example, which zone your Autotask application uses and your threshold level and the number of requests made within the previous hour.

NOTE: Autotask Web Services API stores and returns all time data in Eastern Standard Time (EST).

The **Attachment** API calls allow you to create, delete, and retrieve attachments. They are unique to managing Autotask Attachments through the API.

API Call Descriptions

The API currently includes the following calls. For additional details on any call, including sample code, see to that call's topic's topic.

"Attachment API Calls" on page 230 - "CreateAttachment()" on page 230, "GetAttachment()" on page 231, "DeleteAttachment()" on page 231	Exclusive to the AttachmentInfo entity, you must use these API calls to create, retrieve, and delete Autotask attachments using the Web Services API.
"create()" on page 233	Use the create() API call to add new objects in the Autotask system.
"delete()" on page 236	Use the delete() API call to remove objects in the Autotask system.
"query()" on page 237	Executes a query against Autotask and returns an array of matching entities.
"update()" on page 240	Use this call to update objects in the Autotask system.
"getEntityInfo()" on page 242	Returns an array of EntityInfo objects that contain descriptive information about each available Autotask Web Service entity.
"getFieldInfo()" on page 244	Returns an array of the standard Autotask fields for a specified entity with descriptive information about each field.
"getThresholdAndUsageInfo()" on page 247	Returns the number of external requests allocated for a specific database, the timeframe measured, and the current number of external requests
"getUDFInfo()" on page 248	Returns an array of all User-defined fields (UDF), that is, non-standard fields, for a specified entity.
"GetWsdlVersion()" on page 250	Returns the WSDL version being accessed by the logged in end user.
"getZoneInfo()" on page 251	Returns the proper zone URL to be used for all subsequent API calls.

For information on the return values for **ATWSResponse** and **ATWSError**, see "API Return Values" on page 253.

Attachment API Calls

The following three API calls, GetAttachment(), CreateAttachment(), and DeleteAttachment() are specific to working with attachments in the API. Attachments are external documents that are associated with an Autotask Account, Ticket, Project, or Opportunity entity.

Use these three API calls along with the "AttachmentInfo" on page 70 entity to create and manage attachments. Attachments do not use the standard create() and update() API calls.

You can use query() on the AttachmentInfo entity to determine the AttachmentId for a specific attachment.

CreateAttachment()

Use the CreateAttachment() API call to add an attachment to an account, ticket, project, or opportunity. The attachment can be a specific document uploaded to the Autotask server, a link to a file, a link to a folder, or a URL.

- The API size limit for individual attachment files is 6-7 MB. This is less than the 10 MB limit for individual attachment files when working through the Autotask UI.
- File types are limited to the file types allowed in Autotask. For information, see "Adding Attachments" in the online Help. The maximum file size indicated in the Help does not apply to the API.

Sample Code CreateAttachment()

```
Dim attachment as New atws.Attachment

attachment.Info = New atws.AttachmentInfo

attachment.Info.FullPath = "C:\My Attachments\file.jpg"

attachment.Info.ParentID = 12345

attachment.Info.ParentType = 4

attachment.Info.Publish = 1

attachment.Info.Title = "My Attachment"

attachment.Info.Type = 4


attachment.Data = System.IO.File.ReadAllBytes("C:\My Attachments\file.jpg")


Dim attachmentId As Long = myService.CreateAttachment(attachment)


If attachmentId > 0 Then

     'success

Else
```

```
    'failure
End If
```

GetAttachment()

Use query() to determine the attachmentId. Then use the GetAttachment() API call to get attachment information from Autotask. Finally, add the necessary code to download the attachment to Autotask. See the sample code below.

NOTE: To correct a defect, the GetAttachment call now returns only the attachment filename and extension for Autotask attachments. This change does **not** apply to Client Access Portal attachments.

The new code sample below retrieves and downloads an Autotask attachment.

Sample Code GetAttachment()

The updated code sample below retrieves and downloads an Autotask attachment.

```
Dim apiResponse As atws.ATWSResponse
Dim attachmentInfoEntity As atws.AttachmentInfo = New atws.AttachmentInfo()
Dim attachmentEntity As atws.Attachment = New atws.Attachment()

    ' Query the API for the attachment info.
    apiResponse = apiService.query(xmlQuery)

If apiResponse.EntityResults.Count > 0 Then
    attachmentInfoEntity = apiResponse.EntityResults(0)
End If

' Inflate the attachment entity from the database.
attachmentEntity = apiService.GetAttachment(attachmentInfoEntity.id)

' Download the attachment from Autotask.
System.IO.File.WriteAllBytes(folderPath & attachmentEntity.Info.FullPath, attachmentEntity.Data)
```

DeleteAttachment()

Use the DeleteAttachment() API call to delete an attachment from an account, ticket, project, or opportunity entity. Use query() to determine the attachmentId.

Sample Code DeleteAttachment()

```
Dim errorMessage As String = myService.DeleteAttachment(attachmentId)

If String.IsNullOrEmpty(errorMessage) Then

     'success

Else

     'failure
```

create()

Use the create() API call to add new objects in the Autotask system. Not all entities accept the create() call. To find out if an entity accepts create(), find the Entity in the "Entities" on page 48 list, click the link, and check the Entity Details. Alternately, use the getEntityInfo() call.

This create() call is like an SQL INSERT statement. See Considerations after the code samples.

NOTE: Use the Entity Returned by create() to Obtain Id Only.
You can use the entity returned by create() and update() to obtain the Id field content, but do not rely on the rest of the content. Some entity fields cannot be calculated immediately during create and update. In the returned entity , those fields may not be accurate.

If you need to access the entity for information other than the Id, query for the entity after the create() or update() succeeds.

Considerations

The following considerations apply to the Web Services create() call.

Permissions	Client must be logged in with a user account that has access to the appropriate Autotask module for the entity type and with sufficient rights to perform the specified API call.
id	The id is auto-generated by Autotask and uniquely identifies a particular entity instance. It is also used with the update() call to identify the target record. You cannot assign a value to this field.
Read Only Fields	Some fields are set by Autotask and cannot be changed via the API. These are marked as Read Only in the tables for each entity type. New values assigned to these fields are ignored for any entity object passed to the create() or update() API calls.
Required Fields	For update() and create() calls, you must specify values for required fields. If no value is specified, an error will occur and no data will be changed in Autotask.
System Integrity	Some fields on Entities require an id to a parent Entity. These are referred to as Reference fields. For example, when creating a Ticket Note you must supply the id for the Ticket that the note will be associated with.
Update/Create Maximum Entities	The Web Service API calls can update or create a maximum of 200 objects with one call. If you send more then 200 elements in your array, the call will fail and no changes will occur in Autotask.
White Spacing	API functions will trim all white spaces in the beginning and end of each string type entry.
Exists?	We suggest that when using the create() call you incorporate a query to check whether the object being created already exists. A sample query that provides this check appears below.

Sample create() Code

The following sample code creates a new Ticket.

```
Function AddTicket() As Boolean
```

```
    Dim sResponse As WebReference.ATWSResponse

    Dim TicketToCreate As New WebReference.Ticket

    Dim TicketArray() As WebReference.Ticket

    TicketToCreate.AccountID = 123  'an existing account id

    TicketToCreate.DueDateTime = CDate("2/03/2008")

    TicketToCreate.Priority = 3 ' this corresponds to a picklist value

    TicketToCreate.QueueID = 23 ' this corresponds to a picklist value

    TicketToCreate.Status = 4 'this corresponds to a picklist value

    TicketToCreate.Title = "My Ticket Title"

    TicketArray(0) = TicketToCreate

      Dim entityArray() As WebReference.Entity = CType(TicketArray, WebReference.Entity
    ())

    sResponse = myService.create(entityArray)

    If sResponse.ReturnCode = 1 Then

        Return True

    Else

        Return False

    End If

End Function
```

Sample Code to Check if the Object Exists

Autotask suggests that before creating a new object, you include a query to check that the object being created does not already exist. The following sample query checks for a contact.

```
Function CheckContactExists(ByVal sLastName As String, ByVal sEmail As String, ByVal
sPhone As String) As Boolean

    Dim sResponse As WebReference.ATWSResponse

    Dim sBuilder As New System.Text.StringBuilder

    sBuilder.Append("<queryxml version=""1.0"">")

    sBuilder.Append("<entity>Contact</entity>")

    sBuilder.Append("<query>")

    sBuilder.Append("<field>LastName<expression op=""equals"">" & sLastName & "</ex-
    pression></field>")

    sBuilder.Append("<field>EMailAddress<expression op=""equals"">" & sEmail & "</ex-
    pression></field>")

    sBuilder.Append("<field>Phone<expression op=""equals"">sPhone</expression></field>")
```

```
    sBuilder.Append("</query></queryxml>")
    sResponse = myService.query(sBuilder.ToString, 0)
    If sResponse.ReturnCode > 0 AndAlso sResponse.EntityResults.length > 0 Then
        Return True
    Else
        Return False
    End If
End Function
```

delete()

Use the delete() API call to remove objects in the Autotask system. Not all entities accept the delete() call. To find out if an entity accepts delete(), find the Entity in the "Entities" on page 48 list, click the link, and check the Entity Details. Alternately, use the getEntityInfo() call.

Considerations

The following considerations apply to this Web Services call.

Permissions	Client must be logged in with a user account that has access to the appropriate Autotask module for the entity type and with sufficient rights to perform the specified API call.
id	The id is auto-generated by Autotask and uniquely identifies a particular entity instance. Use this value to identify the target record when using the delete() API call. Use query() to determine the id for an entity.
Update/Create/Delete Maximum Entities	The Web Service API calls can update, create or delete a maximum of 200 objects with one call. If you send more then 200 elements in your array, the call will fail and no changes will occur in Autotask.

Sample delete() Code

The following sample code removes a QuoteItem from the Autotask system.

```
Dim quoteItem As New WebReference.QuoteItem

          quoteItem.id = 123456


          Dim response As WebReference.ATWSResponse = myService.delete(New WebRe-
          ference.Entity() {quoteItem})

          If response.ReturnCode = 1 Then

               Console.WriteLine("Quote Item has been deleted.")

          Else

               For Each ex As WebReference.ATWSError In response.Errors

                    Console.WriteLine(ex.Message)

               Next

          End If
```

query()

This API call executes a query against Autotask and returns an array of matching entities. The queries are built using the QueryXML format and will return a maximum of 500 records at once, sorted by their id value.

To query for additional records over the 500 maximum for a given set of search criteria, repeat the query and filter by id value > the previous maximum id value retrieved.

Currently, all entities and all fields are supported by the query() call. For additional information on the QueryXML format, see "QueryXML" on page 254.

NOTES:

- All API queries will be in Eastern Standard Time (EST).
- Valid query() XML must comply with the following or an error message is generated:

  Any field tag must contain an expression tag.

  An expression tag must contain an op attribute

  Example from Sample Code 2 -

```
sBuilder.Append
("<field>Name<expression op=""startswith"">sValue</expression></field>")
```

- Querying on protected UDFs is case sensitive.

  Considerations

The following list outlines some general considerations for using the Web Services query() API call.

Permissions	Client must be logged in with a user account that has access to the appropriate Autotask module for the entity type(s) queried and with sufficient rights to perform the specified API call.
Query Maximum Entities	The Web Service API query() call will return a maximum of 500 objects per call, sorted by the id field. If you would like to retrieve more than 500 items, you should resubmit your query with a condition on the id field.
White Spacing	API functions will trim all white spaces in the beginning and end of each string type entry.

Sample query() Code 1

```
Dim boolQueryFinished = False

Dim strCurrentID As String = "0"

While Not (boolQueryFinished)

     Dim strQuery As String = "<queryxml><entity>Account</entity>" & _

     "<query><field>id<expression op=""greaterthan"">" & strCurrentID & "</ex-
     pression></field></query>" & _

     "</queryxml>"

     Dim r As myWS.ATWSResponse
```

```
     r = myService.query(strQuery)
     If r.EntityResults.Length > 0 Then
          For Each ent As myWS.Entity In r.EntityResults
               ' execute some code on the current account
               Console.WriteLine(CType(ent, myWS.Account).AccountName)
               Console.WriteLine(ent.id)
               strCurrentID = ent.id
          Next
     Else
          boolQueryFinished = True
     End If
End While
```

Sample query() Code 2

The following sample code submits a simple query() API call.

```
Dim sResponse As WebReference.ATWSResponse
Dim sBuilder As New System.Text.StringBuilder
sBuilder.Append("<queryxml version=""1.0"">")
sBuilder.Append("<entity>Contact</entity>")
sBuilder.Append("<query>")
sBuilder.Append("<field>Name<expression op=""startswith"">sValue</expression></field>")
sBuilder.Append("</query></queryxml>")
sResponse = myService.query(sBuilder.ToString)
If sResponse.ReturnCode > 0 Then
     For Each field As WebReference.TicketNote In sResponse.EntityResults
          Console.Write(field.CreatorResourceID)
          Console.Write(field.Description)
          Console.Write(field.id)
          Console.Write(field.LastActivityDate)
          Console.Write(field.NoteType)
          Console.Write(field.Publish)
          Console.Write(field.TicketID)
          Console.Write(field.Title)
```

```
    Next
End If
```

update()

Use this call to update objects in the Autotask system. Not all entities accept the update() call. To determine if an entity can be updated, locate the entity in the "Entities" on page 48 list, click and check the Entity Details in the topic. Alternately, use the "getEntityInfo()" on page 242 call.

NOTE: When you use the update() call, all entity fields are updated unless they are ReadOnly. On update(), if no value is provided for **non**- ReadOnly fields, those fields are cleared.

NOTE: Use the Entity Returned by update() to Obtain Id Only.
You can use the entity returned by create() and update() to obtain the Id field content, but do not rely on the rest of the content. Some entity fields cannot be calculated immediately during create and update. In the returned entity , those fields may not be accurate.

If you need to access the entity for information other than the Id, query for the entity after the create() or update() succeeds.

Considerations

The following general considerations apply to the Web Services update() call.

Permissions	Client must be logged in with a user account that has access to the appropriate Autotask module for the entity type and with sufficient rights to perform the specified API call.
id	The id is auto-generated by Autotask and uniquely identifies a particular entity instance. It is also used with the update() call to identify the target record. You cannot assign a value to this field.
ReadOnly Fields	Some fields are set by Autotask and cannot be changed via the API. These are marked as Read Only in the tables for each entity type. New values assigned to these fields are ignored for any entity object passed to the create() or update() API calls.
Required Fields	For update() and create() calls, you must specify values for required fields. If no value is specified, an error will occur and no data will be changed in Autotask.
System Integ-rity	Some fields on Entities require an id to a parent Entity. These are referred to as Reference fields. For example, when creating a Ticket Note you must supply the id for the Ticket that the note will be associated with.
Update/Create Maximum Entities	The Web Service API calls can update or create a maximum of 200 objects with one call. If you send more then 200 elements in your array, the call will fail and no changes will occur in Autotask.
White Spacing	API functions will trim all white spaces in the beginning and end of each string type entry.

Sample update() Code

```
Public Function UpdateAccount(ByVal aID As Integer) As Boolean

     Dim sResponse As WebReference.ATWSResponse

     Dim AccountToUpdate As New WebReference.Account

     Dim AccountArray() As WebReference.Account

     AccountToUpdate.id = aID
```

```
    AccountToUpdate.AccountName = txtAccountName.text
    AccountToUpdate.AccountNumber = txtAccountNumber.text
    AccountToUpdate.AccountType = txtAccountType.text
    AccountToUpdate.Address1 = txtAddress1.text
    AccountToUpdate.Address2 = txtAddress2.text
    AccountToUpdate.AlternatePhone1 = txtAlternatePhone1.text
    AccountToUpdate.AssetValue = txtAssetValue.text
    AccountToUpdate.City = txtCity.text
    AccountToUpdate.CompetitorID = txtCompetitorID.text
    AccountToUpdate.Country = txtCountry.text
    AccountToUpdate.CreateDate = txtCreateDate.text
    AccountToUpdate.Fax = txtFax.text
    AccountToUpdate.KeyAccountIcon = txtKeyAccountIcon.text
    AccountToUpdate.MarketSegmentID = txtMarketSegmentID.text
    AccountToUpdate.OwnerResourceID = txtOwnerResourceID.text
    AccountToUpdate.ParentAccountID = txtParentAccountID.text
    AccountToUpdate.Phone = txtPhone.text
    AccountToUpdate.PostalCode = txtPostalCode.text
    AccountToUpdate.SICCode = txtSICCode.text
    AccountToUpdate.State = txtState.text
    AccountToUpdate.StockMarket = txtStockMarket.text
    AccountToUpdate.StockSymbol = txtStockSymbol.text
    AccountToUpdate.TerritoryID = txtTerritoryID.text
    AccountToUpdate.WebAddress = txtWebAddress.text
    AccountArray(0) = AccountToUpdate
    Dim entityArray() As WebReference.Entity = CType(AccountArray, WebReference.Entity
    ())
    sResponse = myService.update(entityArray)
    If sResponse.ReturnCode = 1 Then
        Return True
    Else
        Return False
    End If
End Function
```

getEntityInfo()

The getEntityInfo() API call returns an array of EntityInfo objects that contain descriptive information about each available Autotask Web Service entity including which actions the Web Services API can perform on each entity, whether or not the entity supports User-defined fields, and what actions the logged in end user has permission to perform.

getEntityInfo() returns the following information for each entity.

Data	Description
EntityName as String	The Name of the entity, for example, Ticket, Note, Account.
CanCreate as Boolean	Indicates whether objects can be created on the specified entity using the API. The access varies by Autotask end user permission level. See CanICreate, below.
CanUpdate as Boolean	Indicates whether objects on the specified entity can be updated using the API. The access varies by Autotask end user permission level. See CanIUpdate, below.
CanQuery as Boolean	Indicates whether objects on the specified entity can be queried using the API. Note that currently all entities can be queried; however, the access varies by Autotask end user permission level. See CanIQuery, below.
CanDelete as Boolean	Indicates whether the specified entity can be deleted using the API. The access varies by Autotask end user permission level. See CanIDelete, below.
Has User Defined Fields	Indicates whether the specified entity for your organization contains user defined fields (UDFs). Currently the, Account, AccountLocation, Contact, InstalledProduct (Configuration Items), Opportunity, Product, Project, Task, and Ticket entities can contain UDFs. If your organization uses UDFs with an entity, the getFieldInfo() call does not return the UDFs. You must use the getUDFInfo() call to return a listing and descriptions of the UDFs.
UserAccessForCreate as Boolean	Indicates the logged in user's permission to Create objects on the specified entity: 0 (None), 1 (All), 2 (Restricted). See Autotask Security, below.
UserAccessForUpdate as Boolean	Indicates the logged in user's permission to Update objects on the specified entity: 0 (None), 1 (All), 2 (Restricted). See Autotask Security, below.
UserAccessForQuery as Boolean	Indicates the logged in user's permission to Query objects on the specified entity: 0 (None), 1 (All), 2 (Restricted). See Autotask Security, below.
UserAccessForDelete as Boolean	Indicates the logged in user's permission to Delete objects on the specified entity: 0 (None), 1 (All), 2 (Restricted). See Autotask Security, below.

Autotask Security

In Autotask, the logged in User's security level controls access to features and data. Autotask provides basic security levels, for example, System Administrator or Manager, that are pre-set to provide access to Autotask modules like Service Desk and Projects. Autotask administrators can then modify or copy the basic security levels to create custom security levels. The custom security levels set additional controls over access to features and data within the assigned modules .

The data returned by getEntityInfo() "UserAccessFor" fields corresponds to the levels of permissions assigned through basic and custom security levels:

- 0 (None) indicates the user can never perform the action (even when the entity allows the action).
- 1 (All) indicates the user can always perform the action.
- 2 (Restricted) indicates that the user can perform the action under specific conditions. These conditions usually correspond to security level permission settings like "Mine" or "My Territory", which allow access to only those items with which the user is associated. For example, a Mine permission on the Account entity requires that the logged in user must be the account owner or a member of the account team. A My Territory permission includes the Mine permissions and also includes users that are associated with the same territory as the account.

TIP: The Full Access security level ensures access to all modules and features.

For additional details on Autotask custom security levels, refer to the online help topic Managing Security Levels.

Site-Wide Restrict Deletion Installed Module

For legal or accounting reasons, some companies choose to enable the Autotask "Site-Wide Restrict Deletion" installed module. This module prevents any user from deleting Accounts, Tickets, or TimeEntries. Site-Wide Restrict Deletion applies to the Web Services API.

Sample getEntityInfo() Code

The following getEntityInfo() code sample prints out the properties of each entity.

```
Dim ei() As WebReference.EntityInfo = myService.getEntityInfo

For Each ent As WebReference.EntityInfo In ei

     Console.Write("Field Name=" & ent.Name)

     Console.Write("Field CanCreate=" & ent.CanCreate)

     Console.Write("Field CanDelete=" & ent.CanDelete)

     Console.Write("Field CanQuery=" & ent.CanQuery)

     Console.Write("Field CanUpdate=" & ent.CanUpdate)

     Console.Write("Field HasUDF=" & ent.HasUserDefinedFields)

Next
```

getFieldInfo()

This call is an informational Web Service that returns an array of the standard Autotask fields for a specified entity with descriptive information about each field. The information includes the field label and type; whether the field is required, read only, can be queried; whether it references another field and the reference data type; or whether the field is a pick list.

If the field is a pick list, the information includes the available values, their active/inactive status, and whether or not they are system values. In addition, where the picklist is a child, the name of the picklist parent is provided.

getFieldInfo() returns the following information for each field.

Data	Description
Name as String	Name of the field element, this is not what you see on the UI.
Label as String	Name of the field on the UI.
Type as String	The field element data type **Integer**: Whole number with no decimal places **String**: Can contain alpha-numeric and most special characters (note: some fields have specific lengths) **Double**: Like Integer but can contain decimal places **dateTime**: Contains the date and time as one value like a time stamp **Date**: Date only, no time **Boolean**: True (Yes) (1) or False (No) (0)
Description as String	The long Description of the field element.
IsRequired as Boolean	True or False. Returns True (Yes) if the element is required to update or create.
IsReadOnly as Boolean	True or False. Returns True (Yes) if the element cannot be updated or created.
IsQueryable as Boolean	True or False. Returns True (Yes) if the element can be searched on.
IsReference as Boolean	True or False. Returns True (Yes) if the field contains the id value of another entity type.
ReferenceEntityType as String	If the element IsReference is True (Yes), indicates the Entity Type of the reference.
IsPicklist as Boolean	True or False. Returns True (Yes) if the element is a picklist, specifically a drop down list.

Data	Description
PickListValues() as PickListValue	If IsPicklist = True, then this will hold the values of the Drop Down List. The picklist values elements include the following: Value as String — The picklist value of the item. Label as String — The name of the item as it appears in the drop down list. IsDefault as Boolean — Indicates whether the item is the default value for the field. SortOrder as Boolean — Indicates the order of the items as they appear in the list. ParentValue as String — If the picklist is a child of another field, identifies the parent field. IsActive — Indicates the active/inactive status of the value. IsSystem — Indicates if the value is a system value.
PicklistParentFieldName	If IsPicklist=True for the current field, and if the value for the current field depends upon another field for this entity, then PicklistParentFieldName holds the name of the field. Use the known value for this field to filter the picklistValues for the current field.

Sample getFieldInfo() Code

The following sample code returns an array containing all standard Autotask fields for a specified entity and selected descriptive information for each field.

```
Dim MetaDataSet As New DataSet

Dim EntityDataTable As DataTable

Dim FieldDataRow As DataRow

Dim EntityInfo() As WebReference.EntityInfo

EntityInfo = myService.getEntityInfo

For Each ent As WebReference.EntityInfo In

     EntityDataTable = MetaDataSet.Tables.Add(ent.Name)

     EntityDataTable.Columns.Add("name")

     EntityDataTable.Columns.Add("type")

     EntityDataTable.Columns.Add("isRequired")

     EntityDataTable.Columns.Add("isUDF")

     EntityDataTable.Columns.Add("datatype")

     Dim FieldInfo() As WebReference.Field

     FieldInfo = myService.getFieldInfo(ent.Name)

     For Each fld As WebReference.Field In FieldInfo

          FieldDataRow = EntityDataTable.NewRow()
```

```
            FieldDataRow("Name") = fld.Name.ToString
            FieldDataRow("type") = fld.Type.ToString
            FieldDataRow("isRequired") = fld.IsRequired
            FieldDataRow("isUDF") = String.Empty
            FieldDataRow("datatype") = fld.Type
            EntityDataTable.Rows.Add(FieldDataRow)
        Next
Next
```

getThresholdAndUsageInfo()

This call is an informational Web Service that returns the number of external requests allocated for a specific database, the timeframe measured, and the current number of external requests. Use this call to determine the threshold for the database or the database and to monitor your usage to determine if and when you approach the threshold. For more information on the Web Services threshold, see "API Call Threshold" on page 40.

The return string format is "thresholdOfExternalRequest: {0}; TimeframeOfLimitation: {1}; numberOfExternalRequest: {2};".

Sample getThresholdAndUsageInfo() Code

```
Try

     Dim response As WebReference.ATWSResponse = myService.getThresholdAndUsageInfo()

     If response.ReturnCode = 1 Then

          For Each er As WebReference.EntityReturnInfo In response.EntityReturnInfoResults

               Console.WriteLine(er.Message)

               ' Sample Output:

               ' thresholdOfExternalRequest: 100; TimeframeOfLimitation: 60;numberOfExternalRequest: 99;

          Next

     Else

          For Each ex As WebReference.ATWSError In response.Errors

               Console.WriteLine(ex.Message)

          Next

     End If

Catch ex As System.Web.Services.Protocols.SoapException

     Console.WriteLine(ex.Message)

     ' System.Web.Services.Protocols.SoapException: The Autotask API request threshold of 100 requests per 60 minutes has been

     ' exceeded. Future requests will not be processed until the number of requests is within this threshold.

End Try
```

getUDFInfo()

This API call is an informational Web Service that returns an array of all User-defined fields (UDF), that is, non-standard fields, for a specified entity. Currently, the following Autotask entities can include UDFs: Account, AccountLocation, Contact, InstalledProduct, Opportunity, Product, Project, Task, Ticket.

getUDFInfo() returns the following information for each field.

Data	Description
Name as String	Name of the field element, this is not what you see on the UI.
Label as String	Name of the field on the UI.
Type as String	The field element data type **String**: Can contain alpha-numeric and most special characters (note: some fields have specific lengths) **Double**: Like Integer but can contain decimal places **dateTime**: Contains the date and time as one value like a time stamp **Date**: Date only, no time **Boolean**: True (Yes) (1) or False (No) (0)
Description as String	The long Description of the field element.
IsRequired as Boolean	True or False. Returns True (Yes) if the element is required to update or create.
IsReadOnly as Boolean	True or False. Returns True (Yes) if the element cannot be updated or created.
IsQueryable as Boolean	True or False. Returns True (Yes) if the element can be searched on.
IsReference as Boolean	True or False. Returns True (Yes) if the field contains the ID value of another entity type.
ReferenceEntityType as String	If the element IsReference is True (Yes), indicates the Entity Type of the reference.
IsPicklist as Boolean	True or False. Returns True (Yes) if the element is a picklist, specifically a drop down list.
PickListValues() as PickListValue	If IsPicklist = True, then this will hold the values of the Drop Down List. The picklist values elements include the following: Value as String — The picklist value of the item. Label as String — The name of the item as it appears in the drop down list. IsDefault as Boolean — Indicates whether the item is the default value for the field. SortOrder as Boolean — Indicates the order of the items as they appear in the list. ParentValue as String — If the picklist is a child of another field, identifies the parent field.

Sample getUDFInfo() Code

The following sample code returns an array containing all UDFs for a specified entity and selected descriptive information for each field.

```
Dim MetaDataSet As New DataSet

Dim EntityDataTable As DataTable

Dim FieldDataRow As DataRow

Dim EntityInfo() As WebReference.EntityInfo

EntityInfo = myService.getEntityInfo

For Each ent As WebReference.EntityInfo In EntityInfo

     EntityDataTable = MetaDataSet.Tables.Add(ent.Name)

     EntityDataTable.Columns.Add("name")

     EntityDataTable.Columns.Add("type")

     EntityDataTable.Columns.Add("isRequired")

     EntityDataTable.Columns.Add("isUDF")

     EntityDataTable.Columns.Add("datatype")

     Dim FieldInfo() As WebReference.Field

     FieldInfo = myService.getFieldInfo(ent.Name)

     For Each UDF As WebReference.Field In myService.getUDFInfo(ent.Name)

          FieldDataRow = EntityDataTable.NewRow()

          FieldDataRow("Name") = UDF.Name.ToString

          FieldDataRow("Type") = String.Empty

          FieldDataRow("isRequired") = UDF.IsRequired.ToString

          FieldDataRow("isUDF") = "UDF"

          FieldDataRow("datatype") = UDF.Type

          EntityDataTable.Rows.Add(FieldDataRow)

     Next

Next
```

GetWsdlVersion()

This API call is an informational Web Service that returns the current version of the WSDL the user is logged into. The value returned will be in the standard software versions format of X>Y>Z.

WSDL Versions

The Web Services API uses a three decimal numbering system to indicate its most current version. The API is currently in version 1.X.X. The second place number increases by 1 when a significant update occurs that will likely impact most existing integrations.

The third place number increases whenever a change is made to the Web Services WSDL. These changes will not impact existing integrations.

It is possible that, at some point, the WSDL used for integration development will differ from the WSDL available in different Autotask zones, for example, when an Autotask release is rolled out over several weeks. In addition, when working with international customers, there is always a delay between an Autotask English language release and the same release in the localized zones. Developers can use this call to confirm which WSDL is being accessed by the logged in user.

getZoneInfo()

This API call is an informational Web Service that returns the proper zone URL to be used for all subsequent API calls.

Autotask has multiple zones that are generally allocated geographically. Because an Autotask client can be located in any of Autotask's zones, we recommend that this call be made first to determine the proper zone (and accompanying URL) to use for all API interactions. Although each Autotask zone is aware of all other zones and is able to forward the user agent to the proper zone and URL, to get the most efficient response you must use the correct zone and URL information.

NOTE: Use the base URL https://webservices.autotask.net with the getZoneInfo() call. This base URL has been replaced for most activities by a zone specific URL, but it is still available for this call to determine your zone.

getZoneInfo() returns the following information.

Data	Description
URL as String	This field contains the proper URL for zone of this user. This URL must be used for all subsequent calls to the Web Services.
Database Type	This field contains the value that indicates the database type you are accessing.
CI as integer	The CI number is a unique identifier for the database you are accessing.
ErrorCode as Integer	If the value of ErrorCode is negative, then there has been an error in determining the proper zone for the Autotask user.

Sample getZoneInfo() Code

The following sample code returns the proper URL to use for all subsequent calls to the API.

```
Dim myService As New atws.ATWS

'// this call can be issued to any zone, we have chosen

'// webservices.autotask.net (North American zone)

myService.Url = "https://webservices.autotask.net/atservices/1.5/atws.asmx/"

Dim cred As New System.Net.NetworkCredential(userName, password)

Dim credCache As New System.Net.CredentialCache

credCache.Add(New Uri(myService.Url), "Basic", cred)

myService.Credentials = credCache

Dim URLZone As String

Try

     Dim resZoneInfo As atws.ATWSZoneInfo

     resZoneInfo = myService.getZoneInfo(userName)
```

```
    If resZoneInfo.ErrorCode >= 0 Then
        Log("ZoneInfo addr:" & resZoneInfo.URL)
        URLZone = resZoneInfo.URL
    Else
        Log("ZoneInfo error! [" & resZoneInfo.ErrorCode.ToString & "]:" &
        resZoneInfo.URL)
        Exit Sub
    End If
Catch ex As Exception
    Conlog("Connection Test (ZoneInfo) Failed:" & ex.Message)
    Exit Sub
End Try
'// Now reassign the proper URL to the service object.
myService = New atws.ATWS
myService.Url = URLZone
cred = New System.Net.NetworkCredential(userName, password)
credCache = New System.Net.CredentialCache
credCache.Add(New Uri(myService.Url), "Basic", cred)
myService.Credentials = credCache
```

API Return Values

ATWSResponse

Use this code to retrieve information about a request that was made. The following API calls return an ATWSResponse object: query(), update(), create().

ATWSResponse will return information as follows.

ReturnCode	1 = Success Otherwise, failure.
QueryResults	An array of the related entities for the request. • For a query() call, this is an array of entities that qualified for the search criteria. • For update() and create() calls, these are the entities that were either updated or created.
QueryResultType	A string representing the entity type being returned.
Errors	This retrieves and displays the errors (See ATWSError)

ATWSError

Use this code to retrieve the error code for an error that has occurred.

Message()	An array of string objects representing the API error message.

QueryXML

This section describes the Autotask Web Services API QueryXML schema for building queries using the "query ()" on page 237 API call. This XML schema allows you to build complex queries utilizing multiple expressions and conditions.

NOTE: query() will return a maximum of 500 records at once, sorted by their id value. To query records over the 500 maximum for a given set of search criteria, repeat the query and filter by id value > the previous maximum id value retrieved.

Autotask Web Services API QueryXML requires well formed XML. It must adhere to the basic XML standards and XML special characters must be escaped. For a list of XML special characters, see "XML Special Characters" on page 258, below.

QueryXML Elements and Attributes

The following table describes the Autotask Web Services API QueryXML elements.

Element	Description	Values (where appropriate)
<queryxml>	The <queryxml> element is the root element. The child elements of the <queryxml> element are <entity> and <query> elements.	
<entity>	Defines the Autotask element being queried.	Permissible <entity> values:The name of any entity exposed by the Web Services API
<query>	Defines the criteria for the data you wish to retrieve. The allowable child elements of the <query> element are <field> elements and <condition> elements.	
<field>	Defines a field to be queried. You can query on multiple entity fields by specifying more than one <field> element. Multiple <field> elements are always combined using logical AND. The child elements of the <field> element include the field name as a text value and one or more <expression> elements. To specify that a field is a UDF, you must add udf=”true” within the field tag. You can only specify **one UDF field per query**.	

Element	Description	Values (where appropriate)
<expression>	The <expression> element defines the test to apply to the field. You can include multiple expressions for a given field. Multiple expressions within a single field will be combined using a logical OR. Use the 'op' attribute of the <expression> element to specify the type of comparison you are testing for. The available operator attribute values for an expression element are listed to the right. The value for the comparison is specified as a text value between the start and end tags for the expression element. **Example using Expression Element**: The following example would match all records where the first-name field begins with 'F'. <field>FirstName <expression op="beginswith">F</expression> </field>	Available operator attribute values: Equals NotEqual GreaterThan LessThan GreaterThanorEquals LessThanOrEquals BeginsWith EndsWith Contains IsNotNull IsNull IsThisDay Like NotLike SoundsLike
<condition>	The <condition> element allows you to define a set of criteria to be evaluated as a single expression. You may use it to specify an OR condition between different fields and also to build more complex criteria sets by combining and nesting multiple <condition> elements. When combining multiple <condition> elements, you use the operator attribute to specify whether the expression will be evaluated as an AND or OR condition. The default value for the operator attribute is AND. The operator attribute will be ignored for a <condition> element when it is the first element within its parent element.	

QueryXML Examples

Example 1: Simple field test

When querying on a single field, the <query> element would look like the following:

SQL:

firstname = 'Joe'

QueryXML:

```
<queryxml>
    <entity>contact</entity>
    <query>
        <field>firstname
            <expression op="equals">Joe</expression>
```

```
            </field>
        </query>
</queryxml>
```

Example 2: Multiple Field Test

When querying on multiple fields, the <query> element would look like the following:

SQL:

firstname = 'Joe' AND lastname = 'Smith'

QueryXML:

```
<queryxml>
        <entity>contact</entity>
        <query>
            <condition>
                <field>firstname
                    <expression op="equals">Joe</expression>
                </field>
            </condition>
            <condition>
                <field>lastname
                    <expression op="equals">Smith</expression>
                </field>
            </condition>
        </query>
</queryxml>
```

Example 3: Multiple Fields combined with OR

To create an OR condition between two field elements, wrap each field element in a <condition> element, and specify operator="OR" attribute for the second condition element:

SQL:

firstname = 'Joe' OR lastname = 'Brown'

QueryXML:

```
<queryxml>
```

```
    <entity>contact</entity>
    <query>
        <condition>
            <field>firstname
            <expression op="equals">Joe</expression>
            </field>
        </condition>
        <condition operator="OR">
            <field>lastname
            <expression op="equals">Brown</expression>
            </field>
        </condition>
    </query>
</queryxml>
```

Example 4: Nested Conditions

Use nested conditions to specify more complex search criteria:

SQL:

```
(
    firstname = 'Joe'
    OR (
        (firstname = 'Larry' and lastname = 'Brown')
        OR
        (firstname = 'Mary' and lastname = 'Smith')
    )
)
AND city <> 'Albany'
```

QueryXML:

```
<queryxml>
    <entity>contact</entity>
    <query>
        <field>firstname
```

```
            <expression op="equals">Joe</expression>
        </field>
        <condition operator="OR">
            <condition>
                <field>firstname
                    <expression op="equals">Larry</expression>
                </field>
                <field>lastname
                    <expression op="equals">Brown</expression>
                </field>
            </condition>
            <condition operator="OR">
                <field>firstname
                    <expression op="equals">Mary</expression>
                </field>
                <field>lastname
                    <expression op="equals">Smith</expression>
                </field>
            </condition>
        </condition>
        <field>city
            <expression op="notequal">Albany</expression>
        </field>
    </query>
</queryxml
```

XML Special Characters

XML predefines entity references for the following five for special characters that would otherwise be interpreted as part of markup language.

Character Name	Character Reference	Entity Reference	Numeric Reference
Ampersand	&	&amp;	&
Left angle bracket	<	&lt;	<

Character Name	Character Reference	Entity Reference	Numeric Reference
Right angle bracket	>	&gt;	&#62;
Straight quotation mark	"	&quot;	&#39;
Apostrophe	'	&apos;	&#34;

In order to use any of these special characters in context other than markup, you must escape them using the specified Entity Reference.

Appendix A Sample Code

The following link opens a PDF document containing Sample Code for Service Call entities.

Appendix A