API Interface

API Functions in Release 6.1

iTAC MES.Suite

iTAC SOFTWARE

Copyright

Document R 6.10 V2.0 EN state: RELEASED

Contents

Change history (since Release 4.20)

Current documentation for the "API Interface":

File name: API_Interface_R_6_1_EN.pdf
Document: R 6.10 V2.0 EN
Current as of: 08/2008

The following changes have been made with this document:

Document:	Inclusion of new API functions:	since release
R 6.10 V1.0 EN - 07/2008	»changeMatBinForSerialNumberRepair«	Rel. 6.10
	»checkMergedPartsForSnrComplete«	Rel. 6.10
	»createAttribute«	Rel. 6.10
	»customFunction«	Rel. 6.10
	»getMergedUnits«	Rel. 6.10
	»getStorage«	Rel. 6.10
	»mdataBomVerify«	Rel. 6.10
	»mdataGetBomData«	Rel. 6.10
	New API functions in the context of "container attributes":	
	»appendAttributesToMaterialBin«	Rel. 6.10
	»getAttributesForMaterialBin«	Rel. 6.10
	»getMaterialBinsForAttribute«	Rel. 6.10
	»removeAttributeFromMaterialBin«	Rel. 6.10
	New API functions in the context of "machine data collection":	
	»mdcCreateLog«	Rel. 6.10
	»mdcGetConditionCodes«	Rel. 6.10
	»mdcGetLog«	Rel. 6.10
	»mdcGetStationConditions«	Rel. 6.10
	»mdcGetStationMessages«	Rel. 6.10
	»mdcUploadStationCondition«	Rel. 6.10
	»registerUserAtLine«	Rel. 6.10
	»setProductionCycleTime«	Rel. 6.10
	»setSetupCycleTime«	Rel. 6.10
	New API functions in the context of "serial number lock":	
	»trGetLockedSerialNumbers«	Rel. 6.10
	»trGetSerialNumberLockHistory«	Rel. 6.10
	»trLockSerialNumbers«	Rel. 6.10
	»trUnlockSerialNumbers«	Rel. 6.10

Document:	Inclusion of new API functions:	since release
R 6.00 V1.0 EN - 06/2007	New API functions in the context of "work order attributes":	
	»appendAttributeToWorkorder«	Rel. 6.00
	»getAttributesForWorkorder«	Rel. 6.00
	»getWorkordersForAttribute«	Rel. 6.00
	»removeAttributeFromWorkorder«	Rel. 6.00
	New API functions in the context of "resource management":	
	»checkEquipmentData«	Rel. 6.00
	»getRequiredEquipmentData«	Rel. 6.00
	»getSetupEquipmentData«	Rel. 6.00
	»removeEquipmentForWorkorder«	Rel. 6.00
	»updateEquipmentData«	Rel. 6.00
	New API functions in the context of "bin-based traceability":	
	»assignBatchNoToWorkorder«	Rel. 6.00
	»getRegisteredBatch«	Rel. 6.00
	»mergeBatch«	Rel. 6.00
	»registerBatch«	Rel. 6.00
	»splitBatchNoToSerialNumber«	Rel. 6.00
	»unregisterBatch«	Rel. 6.00
	New API functions in the context of "failure data":	
	»getFailureCauseForStation«	Rel. 6.00
	»getFailureDataForSerialNumber«	Rel. 6.00
	»getFailureSlipDataForSerialNumber«	Rel. 6.00
	»getFailureTypForStation«	Rel. 6.00
	»getResultDataForSerialNumber«	Rel. 6.00
	»uploadFailureslip«	Rel. 6.00
	New API functions in the context of "material and setup data":	
	»changeFeederBank«	Rel. 6.00
	»createNewMaterialBinExt«	Rel. 6.00
	»getCompleteMaterialSetup«	Rel. 6.00
	»getMaterialBinData«	Rel. 6.00
	»getPlacementName«	Rel. 6.00
	»updateMaterialSetup«	Rel. 6.00
	»updateMaterialSetupHistory«	Rel. 6.00
	»uploadMaterialBinBookingsExt«	Rel. 6.00
	»uploadMaterialBookings«	Rel. 6.00
	»updateSetupDataBySnr«	Rel. 6.00
	New API functions in the context of "SMT Connector":	
	»smtConsumption«	Rel. 6.00
	»smtEventSetup«	Rel. 6.00

Document:	Inclusion of new API functions:	since release
	»smtSerialNumberSetup«	Rel. 6.00
	»smtSetupPreparation«	Rel. 6.00
	New API function from various areas:	
	»confirmAdvice«	Rel. 6.00
	»getAdvice«	Rel. 6.00
	»getNextBookingForStationAndLayer«	Rel. 6.00
	»getSnrForWorkorderAndWorkstep«	Rel. 6.00
	»getWorkorderForLine«	Rel. 6.00
R 5.00 V2.0 EN - 09/2006	»getMdaDocuments«	Rel. 5.00 SP01
	»getSetupDataBySerialNumber«	Rel. 5.00 SP01
R 5.00 V1.0 EN - 06/2006	»appendAttributeToSerialNumber«	Rel. 5.00
	»getAttributesForSerialNumber«	Rel. 5.00
	»getSerialNumbersForAttribute«	Rel. 5.00
	»getSetupDataBySerialNumber«	Rel. 5.00
	»getWorkplanItems«	Rel. 5.00
	»removeAttributeFromSerialNumber«	Rel. 5.00
	»shipGetLotFromSerialNo«	Rel. 5.00
	»testPaa«	Rel. 5.00
	»testSpa«	Rel. 5.00
R 4.20 V1.0 EN - 07/2005	»activateRecipe«	Rel. 4.20
	»activateWorkorder«	Rel. 4.20
	»assignSerialNumberForProductOrWorkorder«	Rel. 4.20
	»createRecipe«	Rel. 4.20
	»getCalendarData«	Rel. 4.20
	»getRecipeData«	Rel. 4.20
	»getRecipeHeaderAndVersion«	Rel. 4.20
	»getRegisteredUser«	Rel. 4.20
	»getSnrInfoData«	Rel. 4.20
	»registerUser«	Rel. 4.20
	»shipAddChildLotToParentLot«	Rel. 4.20
	»shipGetChildLotsForParentLot«	Rel. 4.20
	»shipMoveAllChildLotsToNewParentLot«	Rel. 4.20
	»shipMoveChildLotsFromLotToLot«	Rel. 4.20
	»shipMoveChildLotToNewParentLot«	Rel. 4.20
	»shipRemoveAllChildLotFromParentLot«	Rel. 4.20
	»shipRemoveChildLotFromLot«	Rel. 4.20
	»unregisterUser«	Rel. 4.20

Document:	Adaptation of existing function descriptions	
R 6.10 V2.0 EN - 08/2008	»mdataBomVerify«	Amended/adapted the function description
R 6.10 V1.0 EN - 07/2008	»appendAttributeToSerialNumber«	Description of the `attributeValue` variable amended; output value (`errorstring`) `-398` added
	»getSnrForWorkorderAndWorkstep«	Description of the `processLayer` variable amended
	»getWorkplanItems«	Description of the `qtyWOopen` variable in the `workplanData` array amended
	»setMaterialBinState«	Description of the `state` variable adapted
	»shipCheckSnrAddToShippingLot«	Description of the output values (`errorstring`) `0`, `1` and `2` adapted
	»shipGetChildLotsForParentLot« »shipGetLotFromSerialNo« »shipGetShippingLotForLotNo« »shipGetShippingLotsForPartNo« »shipReplaceShippingLot«	Description of the `meh` variable in the `shippingLot(s)` array adapted
	»testPaa« »testSpa«	Output value (`errorstring`) `-230` added
	»uploadFailureslip«	Description of the `testStepDescription` variable in the `failureslip` array amended
	»uploadMaterialBinBookingsExt« »uploadMaterialBookings«	Amended the function description
	»updateMaterialSetup« »updateSetupData« »updateSetupDataBySnr«	Advice added for station parameters
	»verifyMerge« »verifyMergeBySnrRef«	Output value (`errorstring`) `-394` added
	»verifyMergeProduct«	Amended the function description; description of the `booking` variable adapted; output value (`errorstring`) `-394` added
	»checkSerialNumberState« »checkSnrStateBySnrRef« »getFailureDataForSerialNumber« »getFailureSlipDataForSerialNumber« »getResultDataForSerialNumber« »getSerialNumberBySnrRef« »getSerialnumberInfo« »getSerialNumberHistoryData« »getSetupDataBySerialNumber« »getSnrHistoryData« »getSnrInfoData« »getSnrUploadInfo« »uploadResultDataAndRecipe« »uploadState« »uploadStateAndFailData« »uploadStateAndFailureData« »uploadStateAndResultData«	Output value (`errorstring`) `-436` added

Document:	Adaptation of existing function descriptions	
R 6.00 V3.0 EN - 12/2007	»assignBatchNoToWorkorder« »assignSerialNumberForProductOrWorkorder«	Description of the `processLayer` and `activateWorkOrder` variables adapted
	»assignSerialNumberToWorkOrder«	Description of the `processLayer` variable adapted
	»checkEquipmentData«	Amended the function description; description of the `checkStatus` variable adapted; output values (`errorstring`) `10` and `-309` adapted
	»getCompleteMaterialSetup«	Description of the `bookDate` variable adapted
	»setMaterialBinLocation«	Description of the `binLocation` variable adapted
	»shipAddSnrToShippingLot« »shipCheckSnrAddToShippingLot«	Advice in the function description amended
	»updateMaterialSetup«	Description of the `setupDate` and `endDate` variables in the `materialSetupData` array adapted
	»uploadMaterialBinBookingsExt« »uploadMaterialBookings«	Description of the `partNr` in the `materialBinBookings` array adapted
	»uploadFailureslip« »uploadResultDataAndRecipe«	Output values (`errorstring`) `10` and `-309` removed
	»uploadState« »uploadStateAndFailData« »uploadStateAndFailureData« »uploadStateAndResultData«	Output values (`errorstring`) `10` and `-309` removed, and `4` and `-375` added
R 6.00 V2.0 EN - 10/2007	»activateWorkorder«	Amended the function description; description of the `flag` variable adapted
	»assignSerialNumberForProductOrWorkorder«	Output value (`errorstring`) `-8` added
	»checkSnrStateBySnrRef«	Amended the function description
	»createNewMaterialBinExt«	Output values (`errorstring`) `-2` and `-5` added and `-3` adapted
	»createRecipe«	Amended the function description; description of the `partNr`, `bomVersion`, `processVersion` and `revisionIndex` variables adapted
	»getAttributesForSerialNumber«	Description of the `attributeCode` variable adapted
	»getFailureTypForStation«	Description of the `failureTypData` array adapted
	»getRequiredEquipmentData«	Data type for the `pmGroup` variable adapted in the function description
	»getSetupEquipmentData«	Description of the `equipmentSetupData` array adapted
	»getSnrForWorkorderAndWorkstep«	Output values (`errorstring`) `-205` and `-385` added; name of the `maxNumberOfRecords` variable changed to `numberOfRecords`
	»registerUser«	Amended the function description
	»shipRemoveSnrFromShippingLot«	Output value (`errorstring`) `0` adapted
	»unregisterBatch«	Output value (`errorstring`) `-205` added
	»uploadMaterialBinBookingsExt« »uploadMaterialBookings«	Description of the `quantity` variable in the `materialBinBookings` array adapted

Document:	Adaptation of existing function descriptions	
	»uploadResultDataAndRecipe« »uploadStateAndFailData« »uploadStateAndFailureData« »uploadStateAndResultData«	Output value (errorstring) -395 added
	»verifyMerge«	Amended the function description
R 6.00 V1.0 EN - 06/2007	»changeWorkOrder« »changeWorkOrderForLine«	Advice on order state added
	»checkSerialNumberState«	Description of the loopCounter variable adapted
	»getCurrentMaterialSetup«	Output values (errorstring) 1, 2 and 3 adapted
	»getMergeParts«	Amended the function description; description of the level variable adapted
	»getSetupDataBySerialNumber«	Amended the function description; description of the location and partNr variables adapted
	»setMaterialBinLocation«	Amended the function description; description of the binLocation variable adapted
	»uploadMaterialBookings«	Output value (errorstring) -392 added
	»uploadResultDataAndRecipe«	Output values (errorstring) 10, -62, -67, -210 and -309 added; description of the min, max, nom and toleranz variables adapted
	»uploadFailureslip« »uploadState« »uploadStateAndFailData« »uploadStateAndFailureData« »uploadStateAndResultData«	Output values (errorstring) 10, -210 and -309 added
	»uploadStationResult«	Description of the min, max, nom and toleranz variables adapted
	»verifyMerge« »verifyMergeProduct«	Output value (errorstring) -393 added
R 5.00 V3.0 EN - 11/2006	»assignSerialNumberMergeAndUploadState«	Description of the serialNumberRef, serialNumberRefPos, numberOfRecords and serialNumberSlave variables amended
	»checkSnrStateBySnrRef« »checkSerialNumberState«	Amended the function description; output value (errorstring) -1 adapted
	»getMdaDocuments«	Description of the KeyValue and mdaDataTypes variables amended; output values (errorstring) -75, -81 and -92 adapted
	»mergeParts«	Amended the function description
	»shipAddSnrToShippingLot«	Output values (errorstring) -9 added, -3 adapted, and -18 removed
	»shipCheckSnrAddToShippingLot«	Output values (errorstring) -30 added
	»verifyMergeProduct«	Description of the bomVersion and bomIndex variables amended
R 5.00 V2.0 EN - 09/2006	»assignSerialNumberForProductOrWorkorder« »assignSerialNumberMergeAndUploadState« »assignSerialNumberToWorkOrder«	Advice added for station parameter code no.: 13302

Document:	Adaptation of existing function descriptions	
	»bomCheckForWorkOrder« »setupActivation«	Advice added for station parameter code no.: 13120
	»getAttributesForSerialNumber«	Advice added to the description of the `snrId` and `attributeNumber` variables
R 5.00 V1.0 EN - 06/2006	»activateWorkorder«	Amended description of the `processLayer` variables
	»bomCheck« »bomCheckForWorkOrder«	Amended description of the `processLayer` variables
	»changeWorkOrder« »changeWorkOrderForLine« »getSerialnumberInfo« »getSnrInfoData« »getStationSetup« »setWorkOrderFromSerialNumber«	Adapted the description of the `attribut1` variables
	»createRecipe«	`numberOfRecords` variable added as an input value
	»getCalendarData«	`hourOfDay` variable added as an input value
	»getCurrentMaterialSetup«	Amended description of the `completeSetupData` variable
	»getNextSerialNumber«	Output values `-21`, `-22` and `-23` added
	»getNextSerialNumberForWorkOrder«	Adaptation of the function description; output values `-21`, `-22` and `-23` added
	»getProductInfo«	Adapted description of the `bomIndex` variable
	»getSnrUploadInfo«	Advice added to the description of the `processLayer` variable
	»getStationSetup«	Amended description of the `processLayer` variables
	»mergeParts«	Adapted output value (`errorstring`) `-5`
	»setupActivation«	Amended description of the `processLayer` variables
	»updateSetupData«	Output value (`errorstring`) `-50` added
	»updateSetupDataHistory«	Output values (`errorstring`) `2` adapted and `-50` added
	»uploadResultDataAndRecipe«	Advice added to the description of the `name` variable added (in the array `resultDataExtArray`); output values (`errorstring`) `-91`, `-93`, and `-94` added
	»uploadState«	Included output value `-50`
	»uploadStateAndFailData«	Included output value `-50`
	»uploadStateAndFailureData«	Included output value `-50`
	»uploadStateAndResultData«	Output values (`errorstring`) `-7` and `-8` adapted and `-50`, `-91`, `-93`, and `-94` added

Document:	Adaptation of existing function descriptions	
	various "upload" functions	The "upload" functions listed below can now pass a particular work step number directly via the `processLayer` variable, rather than just a layer; this behavior is controlled with the new station parameter code no. 8310; the following functions are affected: • »uploadResultDataAndRecipe« • »uploadState« • »uploadStateAndFailData« • »uploadStateAndFailureData« • »uploadStateAndResultData«
	various "mixed" functions	In all functions that pass a station as input value (`stationNr` variable) the following applies: for non-licensed stations, the output value (`errorstring`) `-78` (TR station) or `-79` (RM station) has been added
R 4.21 V2.0 EN - 02/2006	»bomCheck«	Amended the description of the `serialNumber` variable
	»changeMaterialBinQuantity«	Amended the description of the `materialBinNr` and `quantity` variables; data type for the `quantity` variable changed from integer to double; output value (`errorstring`) `-59` added
	»checkSerialNumberState« »getSerialnumberInfo« »getSnrInfoData« »getSnrUploadInfo«	Amended description of the `serialNumberPos` variable
	»createRecipe«	`recipeData` array added as output value
	»getSnrUploadInfo«	Included output value (`errorstring`) `-7`
	»shipGetSnrDataForShippingLot«	Variable name for output value changed (from `serialNrArray` to `serialNr`)
	various functions	"ArrayHolder" designation added
R 4.21 V1.0 DE - 12/2005	»assignSerialNumberForProductOrWorkorder«	Added advice notices on work order creation and the `partNr` and `bomVersion` variables; changed input values for the `activateWorkOrder` variable; output values `-11`, `-12` and `-13` added
	»createRecipe«	Output values `-67`, `-68` and `-69` added
	»getMaterialSetupData«	Adapted sequence of the variables in the `setupData` array
	»getSnrHistoryData«	Changed data type for the variables `state`, `bookDate`, and `createDate` from integer to string
	»setMaterialBinState«	Amended description of the `state` variable
	»shipMoveChildLotsFromLotToLot«	`numberOfRecords` variable added as an input value
	»shipReplaceShippingLot«	Adaptation of the function description; output value (`errorstring`) `-30` removed, output values `-33` and `-37` added
R 4.20 V2.0 EN - 08/2005	»assignSerialNumberForProductOrWorkorder«	Adaptation of the function description; output values (`errorstring`) `-18` and `-19` added

Document:	Adaptation of existing function descriptions	
	»assignSerialNumberMergeAndUploadState« »assignSerialNumberToWorkOrder«	Output value (`errorstring`) `-17` adapted; `-18` and `-19` added
	»checkSerialNumberState«	General description amended
	»getSerialNumberBySnrRef«	Added output value (`errorstring`) `1`
	»uploadResultDataAndRecipe« »uploadStateAndResultData«	Amended description of the `serialNumberState` variable (value passed **-1**); output value (`errorstring`) `-5` adapted
	»createRecipe« »getRecipeData«	Adapted description of the `type` variable
R 4.20 V1.0 EN - 07/2005	»assignSerialNumberMergeAndUploadState« »assignSerialNumberToWorkOrder«	Included output value (`errorstring`) `-17`
	»assignSerialNumberToWorkOrder«	Amended the description of the variables `serialNumberRef` and `serialNumberRefPos`
	»checkSnrStateBySnrRef« »checkSerialNumberState«	Included output value (`errorstring`) `9`, `-9` and `-17`
	»createNewMaterialBin«	Included output value (`errorstring`) `-1`, `-2`, `-3`, `-7`, and `-47`
	»getCurrentCalendar«	Included output value (`errorstring`) `-2`
	»getNextSerialNumber« »getNextSerialNumberForWorkOrder«	Adapted output value (`errorstring`) `-1` and included `-2`, `-3`, `-4`, `-6`, `-7`, and `-50`
	»getSerialnumberInfo«	Included advice into the function description
	»getSnrHistoryData«	Added output value (`errorstring`) `-39`; changed data type of the `quantity` variable from "double" to "string"; description of the `failcode` variable changed
	»getSnrUploadInfo«	Included output value (`errorstring`) `-4`, `-5` and `-6`
	»getStationQuantity«	Included output value (`errorstring`) `-4`, `-5`, `-6`, and `-7`
	»mergeParts«	Included output value (`errorstring`) `-17`, and `-18`
	»queryRecipeData«	Amended function description of the input values `numberOfRecords` and `name`
	»queryTestData«	Description of the `failcode` variable changed
	»shipCheckSnrFromShippingLot«	Adapted output value (`errorstring`) `-4`
	»shipGetShippingLotForLotNo«	Included output value (`errorstring`) `1`, deleted output value `-2`, and adapted `-1`
	»switchSerialNumberbySnrRef«	Included output value (`errorstring`) `-3`
	»uploadState«	Included output value (`errorstring`) `-7`, `-9`, `-10`, `-17`, `-70`, and `-71`
	»uploadStateAndFailData«	Included output value (`errorstring`) `-8`, `-9`, `-12`, and `-17`
	»uploadStateAndFailureData«	Added output value (`errorstring`) `-12` and `-17`

Document:	Adaptation of existing function descriptions	
	»uploadResultDataAndRecipe«	Added output value (`errorstring`) `-10`, `-11`, `-12`, `-16 and -17`; amended description of the `messtyp` variable; description of the `failcode` variable changed
	»uploadStateAndResultData«	Added output value (`errorstring`) `-10`, `-11`, `-12`, `-16 and -17`; description of the `failcode` variable changed
	»uploadStationResult«	Amended description of the `messtyp` variable; description of the `failcode` variable changed
	»verifyMerge« »verifyMergeBySnrRef«	Included output value (`errorstring`) `-17`
	»verifyMergeProduct«	Included output value (`errorstring`) `3`, `4` and `-17`

1 API for iTAC.MES.Suite

1.1 Introduction

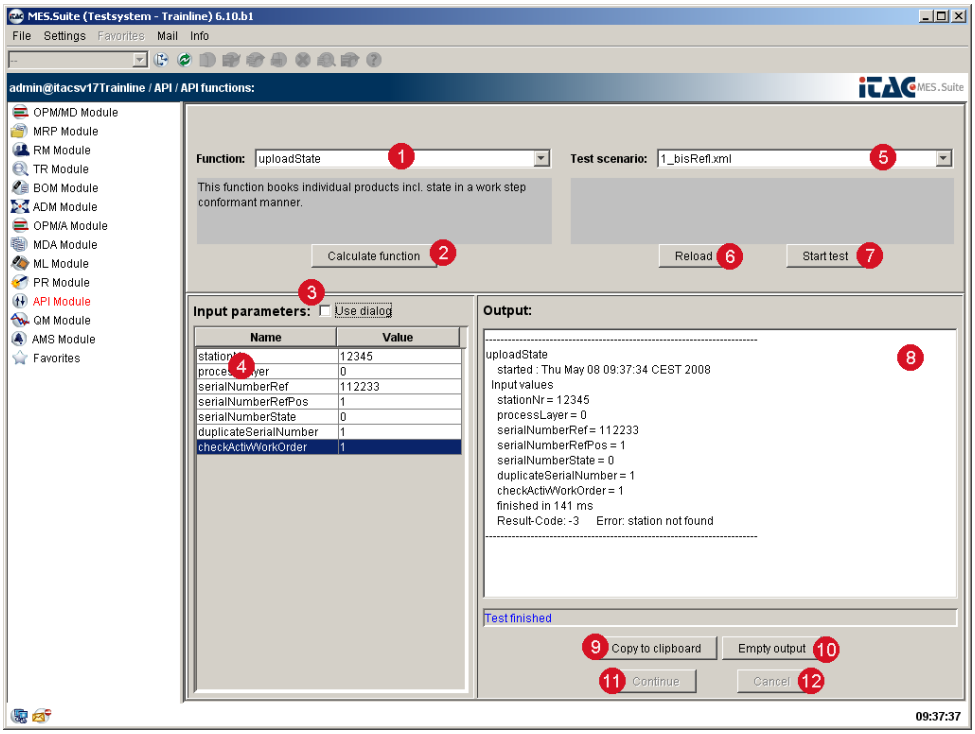
Various functionalities of iTAC.MES.Suite are available via an API (Application Programming Interface) for separate applications and integration in other system environments.

The production equipment is connected via the respective API functions. Together with the possibilities of the API functions, the basic functions permit the mapping of the specific processes within a production machine/ production line.

Thus, the API is characterized by its documentation and openness as well as the fact that it remains constant over very long periods of time. This means that external programs which were developed for a specific version of the iTAC.MES.Suite API will also function with subsequent versions. At the same time, however, each new version of the API will bring with it new functions, without having a negative effect on the existing functionality.

Test option

The API Module offers the option of testing all API functions. Individual API functions as well as entire scenarios can be tested in this way. Enter the appropriate input parameters for individual API functions directly or, for complex test scenarios, in a previously prepared XML file. Afterward, you can analyze the results in the output.



1. Selection of the desired API function; the input parameters are displayed under 4.
2. The function is calculated and the results are displayed in the output 8.
3. If the "Use dialog" option is set, the values for the required input parameters are queried via a dialog box; if the option is off, the values must be entered directly in the table 4.
4. Depending on the selected API function, the respective input parameters are listed here. The "Value" column can be edited directly.
   Advice: *Confirm the input of each value with* <ENTER>*!*
5. Enables the selection of an XML file in which the input parameters of multiple API functions can be specified for individual test scenarios; store under …`\workplace\Tests\*.xml`). In this way the system can calculate any API functions in a specific order using the desired values.
   Advice: *The structure and syntax of this XML file are made apparent by means of the examples listed below!*
6. The selected XML file is reloaded.
7. The XML file is imported and the API functions defined in the file are processed by the system; the results of the individual functions are displayed in output 8.
8. Display of the function results.
9. The entire output is copied to the clipboard.
10. The output is cleared.
11. If a pause is implemented in the test scenario, this can be used to continue the test.
    Advice: *Details on integrating pauses in your test scenario are available upon request!*
12. The running test is cancelled.

FIG. 1-1 *API Module in iTAC.MES.Suite*

XML example "variable declaration"

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<% int station=45012010; %>
<% int workorder=100757; %>
<Test>
  <Name/>
  <Comment/>
  <Date/>
  <Version/>
  <Functions>
    <Func name="queryRecipeData">
      <Parameter>
        <Param paramName="stationNr" default="<%=station%>"/>
        <Param paramName="workorder" default="<%=workorder%>"/>
        <Param paramName="numberOfRecords" defaultlt="1" />
        <Param paramName="RecipeDataArray" name="-" value="-"/>
      </Parameter>
    </Func>
  </Functions>
</Test>
```

XML example "loop"

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<% int serialstart=20; %>
<% int serialend=29; %>
<Test>
  <Name/>
  <Comment/>
  <Date/>
  <Version/>
    <% for (int i = serialstart; (i <= serialend); i++) { %>
    <Functions>
      <Func name="getSnrHistoryData">
        <Parameter>
          <Param paramName="serialNr" default="PAA-ZZY_<%=i%>"/>
        </Parameter>
      </Func>
    </Functions>
    <% } %>
</Test>
```

XML example "delay time"

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<Test>
  <Name/>
  <Comment/>
  <Date/>
  <Version/>
    <Functions>
      <Delay delayTime="1"/>
      <Func name="getSnrHistoryData">
        <Parameter>
          <Param paramName="serialNr" default="PAA-ZZY_20"/>
        </Parameter>
      </Func>
      <Delay delayTime="2"/>
      <Func name="getSnrHistoryData">
        <Parameter>
          <Param paramName="serialNr" default="PAA-ZZY_20"/>
        </Parameter>
      </Func>
      <Delay delayTime="3"/>
      <Func name="getSnrHistoryData">
        <Parameter>
          <Param paramName="serialNr" default="PAA-ZZY_20"/>
        </Parameter>
      </Func>
    </Functions>
</Test>
```

1.2 The API service / API server

The API functions are all implemented in a single service named **`EWApiServices`**. The implementation of the API service is developed 100% in Java and can be accessed via IIOP. The definition of the interface is specified in the CORBA-IDL notation (and, thus, with the CORBA-IDL data types). The following information is provided for each function:

- Return data type of the function (in CORBA-IDL notation)
- Name of the function
- Parameter list delimited with commas, each parameter with the following attributes
  - Direction of the parameter (in, out, inout)
  - Data type of the parameter (in CORBA IDL notation)
  - Name of the parameter

Example of a function declaration

```
long queryRecipeData(
     in string stationNr,
     in string workOrder,
     inout long numberOfRecords,
     inout RecipeDataArray recipeDataArray,
     out string errorString)
raises (ServerException);

struct RecipeData
     {
        string name;
        string value;
     };
typedef sequence<RecipeData> RecipeDataArray;
```

Data types can represent self-defined data types here. These can be structures and arrays. These are also defined in CORBA-IDL. In the example shown above, expression **`RecipeDataArray`** is a self-defined data type.

ADVICE

For the **`out`** *and* **`inout`** *parameters, note that the "Holder" data types of the respective parameter data type are used (CORBA standard).*

When passing floating point values within strings, it is ***imperative*** *that a "**.**" be used as decimal character; the use of a "**,**" results in misinterpretations of the data and may cause the software to behave incorrectly!*

CORBA IDL data types

The following native CORBA-IDL data types are possible [1]:

Data type IDL	Meaning
`short`	16-bit signed integer value
`unsigned short`	16-bit unsigned integer value
`long`	32-bit signed integer value
`unsigned long`	32-bit unsigned integer value
`long long`	64-bit signed integer value
`unsigned long long`	64-bit unsigned integer value
`float`	IEEE single precision floating point number (64-bit)
`double`	IEEE double precision floating point number (32-bit)
`boolean`	TRUE or FALSE
`octet`	8-bit numerical value which undergoes no character-set translation
`char`	8-bit value interpreted as character
`string`	ASCII-string of any length (8 bits per character)

TAB. 1-1 *Description of the IDL data types*

Because the API functions are all defined in CORBA-IDL, there are absolutely no inherent dependencies on specific operating systems, hardware details (big/little endian of the CPU), programming languages etc.

Any system which can communicate via CORBA is, in principle, able to use the API (nearly all platforms and systems are equipped with various ORB implementations and can, therefore, be used directly). It can be said here that CORBA represents the "most natural" of all methods of accessing the iTAC.MES.Suite API.

Any potential user who would like to use an API application via CORBA can immediately access the API. To do so, it is only necessary that the developer has access to the CORBA-IDL files of the API. These files are delivered with iTAC.MES.Suite.

The most elegant and convenient way to communicate with the API server is via a direct CORBA connection by means of Java and the iTAC middleware library. This method affords all of the advantages of the iTAC middleware ("LoadBalancing", "FailOver" etc.) and simultaneously facilitates simplified access to CORBA.

Various libraries are available for application programmers who would like to use the API (see chapter 1.3 »Overview of API libraries«).

1. Not all data types are used in the iTAC.MES.Suite API.

1.3 Overview of API libraries

By providing various libraries, the fact that not all developers wish to work directly with CORBA is taken into account. In fact, requests are often made for libraries which are implemented for a specific operating system and/or a specific programming language (or proprietary technology) and which should hide the CORBA layer. All of the possibilities mentioned in the following – with the exception of the "ORB Protocol" (see chapter 1.3.1) – automatically contain the iTAC middleware, which enables LoadBalancing and FailOver.

Communication model

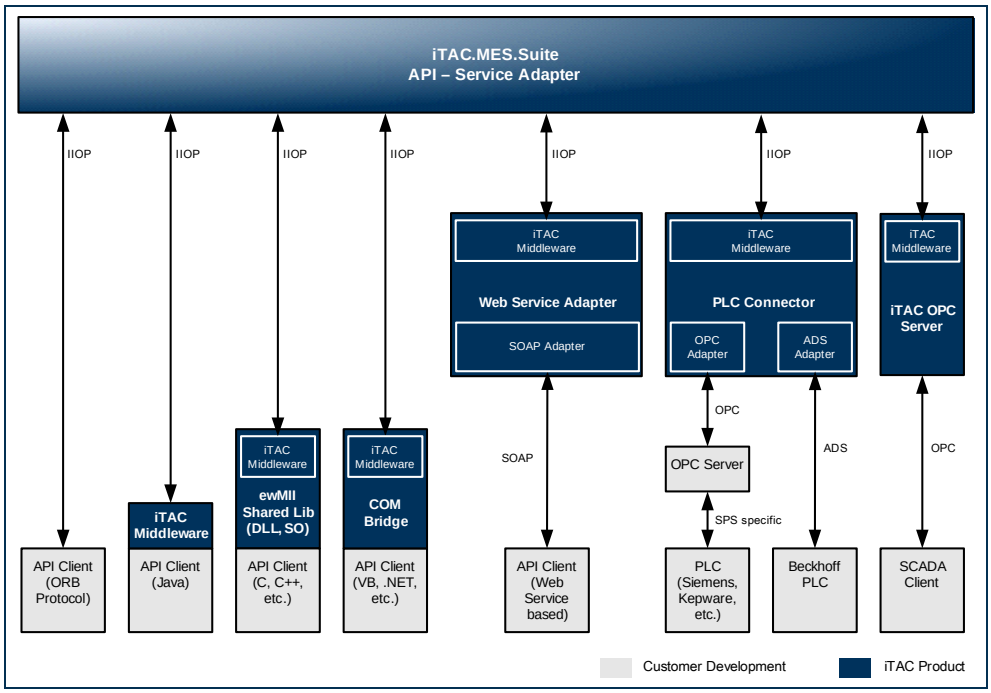


FIG. 1-2 *Communication model*

	Operating systems	Languages	Active comp.	Linkage (files)	3rd Party
ORB Protocol	All	various	no	idl files	no [a]
iTAC middleware	All [b]	Java	no	jar files	no
ewMII	Windows / Linux	C / C++	no	dll, lib, h / so, h	no
COM Bridge	Windows	VisualBasic	no	tlb file	yes [c]
Web Service Adapter	All	various	yes	wsdl file	no [d]
PLC Connector	PLC [e]	PLC, C, config	yes	config	yes [f]
iTAC OPC Server	SCADA	config	yes	config	no

TAB. 1-2 *Overview of the connection possibilities*

a. depending on ORB implementation
b. Systems with Java 6.0
c. JIntegra
d. depending on SOAP implementation
e. Siemens, Kepware, ADS Twincat
f. OPC server of the PLC

1.3.1 ORB Protocol: direct API access via IIOP

This access variant is the only variant WITHOUT a special library.

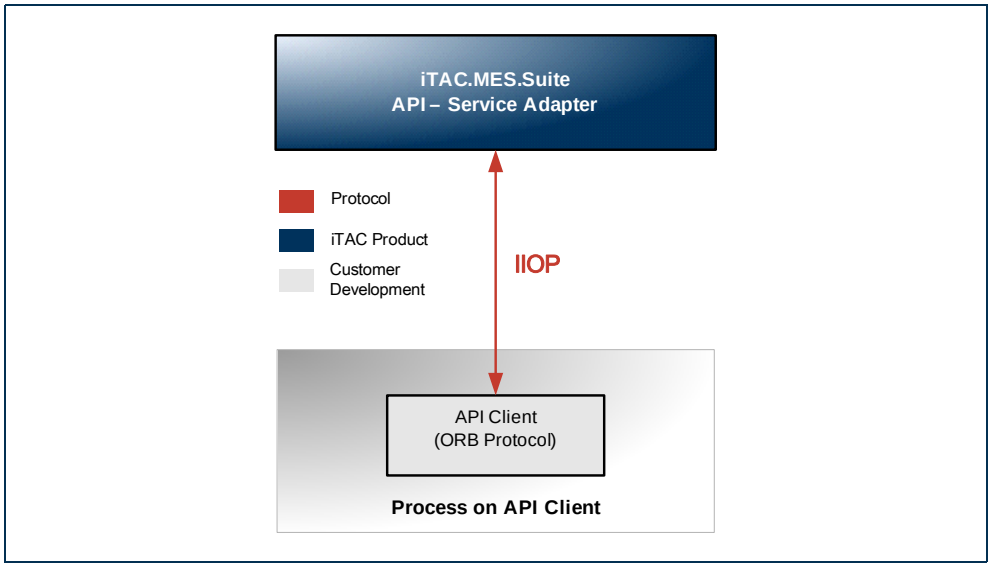


FIG. 1-3 *ORB Protocol: direct API access via IIOP*

With delivery of the IDL files for EWApiServices, it is possible to use the standard ORB mechanisms (IDL compiler, stub code, etc.) to directly access the API. The FailOver and LoadBalancing mechanisms are not automatically available in this case, however. These mechanisms may be implemented by the application developer himself if necessary. An exact specification of how to do this is available upon request. This method should only be used if none of the methods for establishing the API connection described below can be used. This should, however, only be necessary – if at all – for very exotic platforms or system environments.

Client applications generate the STUB code from the IDL files in this case. These generated files are then linked together with the API client application.

1.3.2 iTAC middleware: accessing the API by means of a Java application

For Java client applications, we make the iTAC middleware framework available; this can be used directly by Java applications. Compared to the ORB interface, very simplified programming is possible with this connection method. All features of the iTAC middleware are immediately available to the application (e.g. FailOver, LoadBalancing). The appropriate JAR files and an example Java client are provided to the API client developer for this purpose.

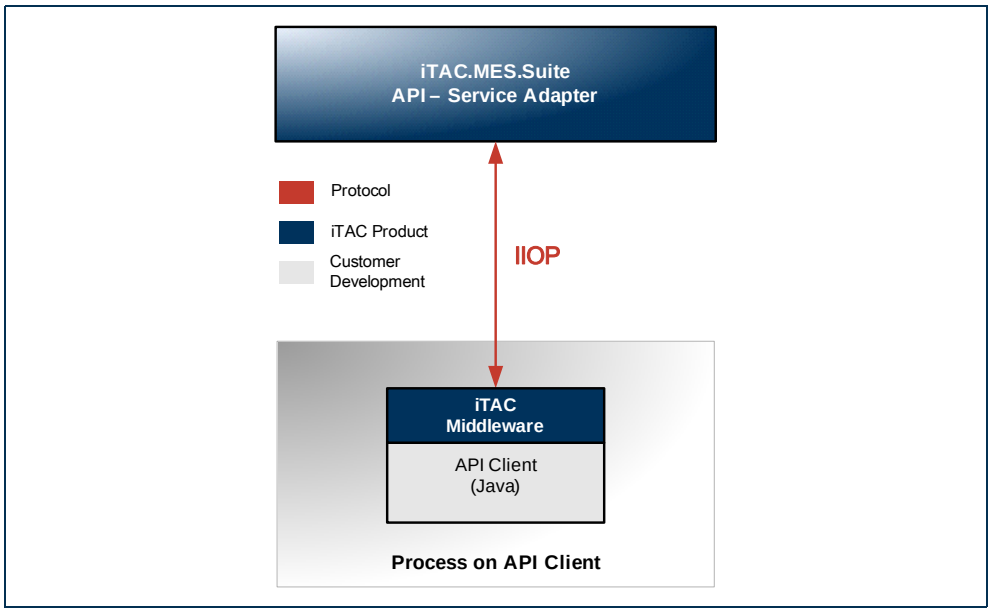


FIG. 1-4 *iTAC middleware: accessing the API by means of a Java application*

Internally, this connection uses the ORB implementation which is a component of the Java Runtime Environment. For this reason, no separate ORB implementation is necessary.

1.3.3 ewMII: accessing the API by means of shared libraries (DLL, SO)

Shared libraries are available for Windows and Linux. These are supplied in the form of "dll" (**d**ynamic **l**ink **l**ibrary for Windows) or "so" (**s**hared **o**bject for Linux) files. In addition, the required C header files, import library files etc. are also supplied. These contain all mapping of the functions and data types from C/C++ to CORBA-IDL; this information is absolutely necessary for developing an API client.

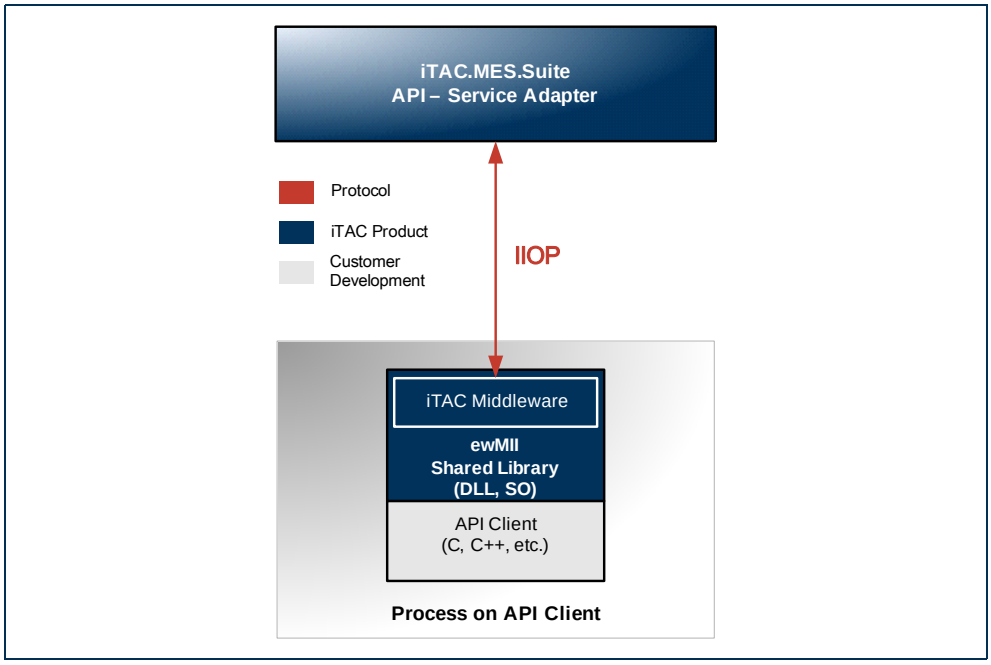


FIG. 1-5 *ewMII: accessing the API by means of shared libraries*

The shared libraries can be used by an application developer in the accustomed manner for the respective operating system or programming language. Development is usually performed in C or C++. Shared libraries can, in principle, be used by nearly any programming language or programming environment. iTAC only provides support for C/C++, however.

Example programs are supplied with the shared libraries to illustrate their use with the C programming language.

The connection between the shared library and the iTAC middleware is performed internally by means of JNI. The ORB implementation which is component of the Java Runtime Environment is used here. For this reason, no separate ORB implementation is necessary.

1.3.4 COM bridge: accessing the API by means of COM/DCOM objects

A library is available for use with the COM Bridge (Windows only) which presents the EWApiServices CORBA object as a COM/DCOM object – as is common under Windows. In this case, a TypeLibrary is supplied which Windows client programs can use directly by means of COM mechanisms.

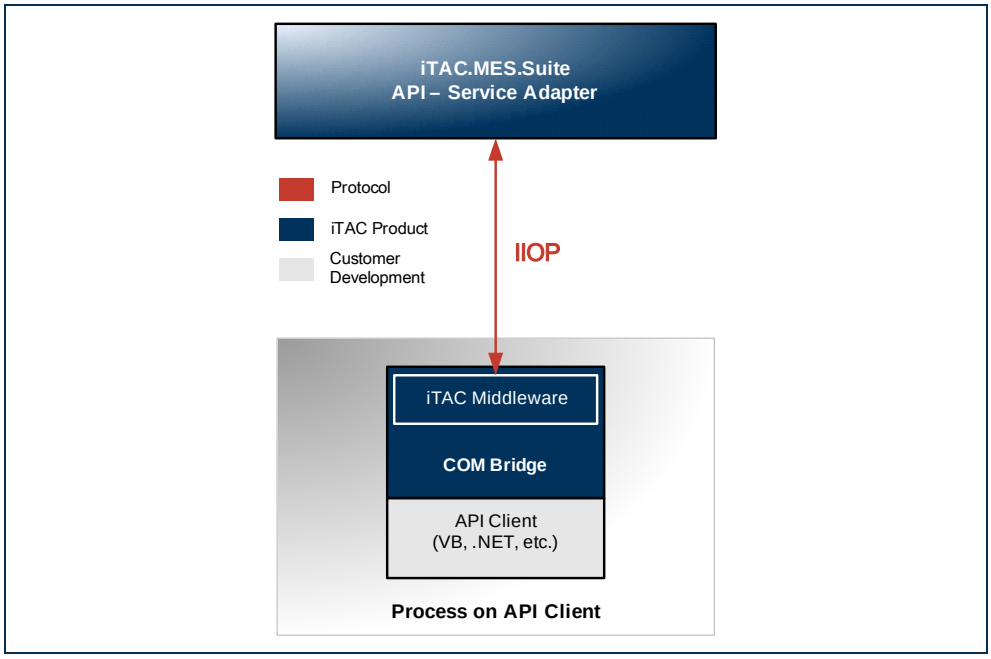


FIG. 1-6 *COM bridge: accessing the API by means of COM/DCOM objects*

A corresponding VisualBasic example program is supplied with the COM bridge; this example program accesses the EWApiService.

The connection between the COM bridge and the iTAC middleware is performed internally directly via Java. The ORB implementation which is component of the Java Runtime Environment is used here. For this reason, no separate ORB implementation is necessary. For the mapping between COM and Java, the "J-Integra for COM" product from the J-Integra company is used; this product must be licensed separately.

1.3.5 Web Service Adapter: accessing the API by means of SOAP/Web Service

As an active interface, the Web Service Adapter makes available all API functions by means of the SOAP/Web Service. In this case we supply a WSDL file which contains all functions and structures of the API.

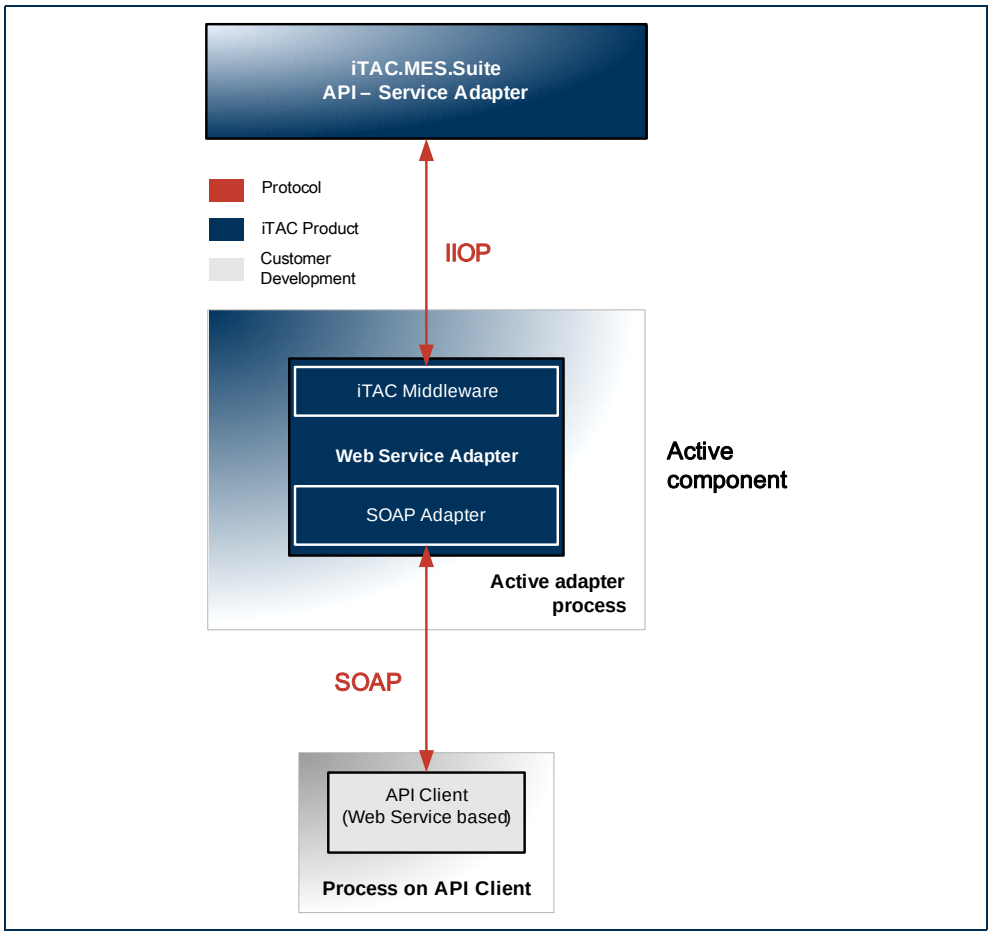


FIG. 1-7 *Web Service Adapter: accessing the API by means of SOAP/Web Service*

Client applications generate the STUB code from this WSDL file in this case. The generated files are then linked together with the API client application.

The exact procedure on the client side is dependent on the used SOAP/Web Service implementation and programming environment. The principle development path is, however, a component of the Web Service specification.

SOAP/Web Service is a protocol which always requires extensive resources. This affects both the CPU of the system on which the adapter runs as well as the network path via which SOAP communication occurs. Each SOAP message must be parsed at runtime, and the protocol has a very large amount of overhead.

In order to achieve the same level of performance with the Web Service Adapter as is achieved with the other API access variants described here, approx. 5-8 times more resources are required. **This applies to both CPU as well as bandwidth!** For this reason, the Web Service Adapter should NOT be run on an iTAC.MES.Suite application server; it should instead be installed on separate hardware. Because this adapter is implemented in Java, all of the operating systems approved by us for Java can be used (Linux or Windows).

1.3.6 PLC Connector: accessing the API within PLC systems

The PLC Connector is available for using the API directly from PLC systems. In this case, the corresponding OPC server is used on the PLC side. The OPC server is generally available from the PLC manufacturer.

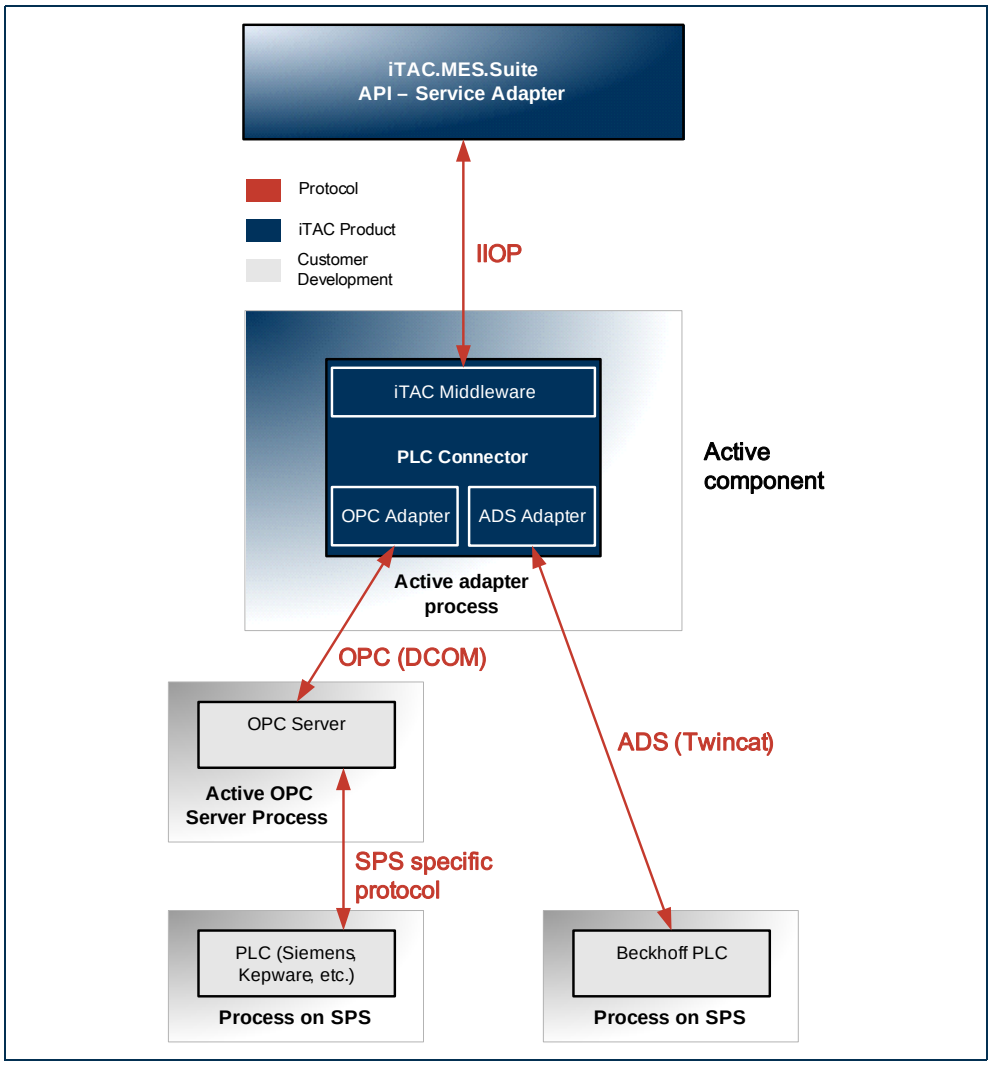


FIG. 1-8 *PLC Connector: accessing the API within PLC systems*

In this case, the PLC adapter communicates as an OPC client with the OPC server of the PLC and then transforms the calls into IIOP; these are then communicated to the API server by the iTAC middleware. Because the PLC adapter is an active process which communicates by means of OPC and because OPC is based on Microsoft COM/DCOM, this process must be installed on a Windows system. Microsoft does not supply a COM/DCOM implementation for other systems (such as e.g. Linux). This will, however, change with future OPC versions. OPC will then be based on a platform independent protocol (OPC UA by means of SOAP).

The development of a PLC program which communicates with the API server by means of the PLC Connector occurs by means of three methods:

- OPC variable definition in the OPC server of the PLC
- Configuration of the PLC Connector
- Optional program development on the PLC

In addition to communicating by means of OPC, with whose assistance the most common PLCs can be coupled, a connection to Beckhoff PLCs can be performed by means of TwinCAT ADS in the PLC adapter.

A concrete implementation of a PLC API client is usually performed in cooperation with PLC developers and iTAC Consulting.

e.g.: Siemens OPC server
Kepware OPC server

1.3.7 iTAC OPC server: accessing the API from SCADA systems

SCADA systems (**S**upervisory **C**ontrol **A**nd **D**ata **A**cquisition) are used for monitoring, control and data capture of technical processes.

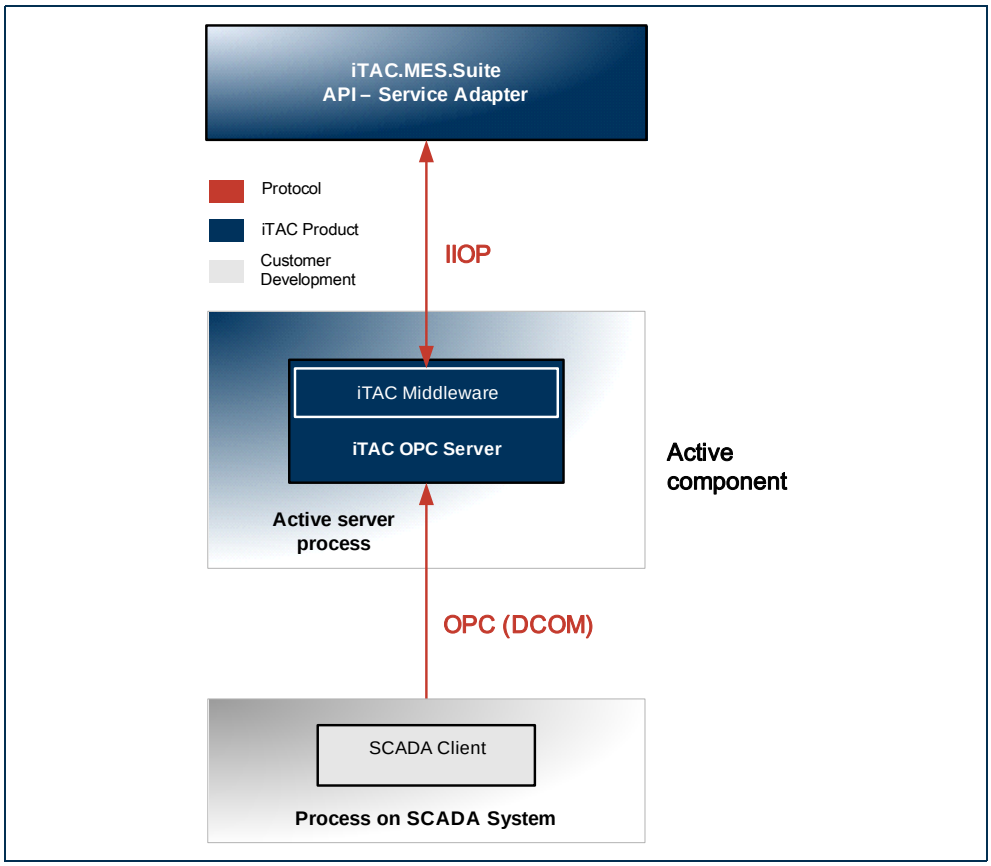


FIG. 1-9 *iTAC OPC server: accessing the API from SCADA systems*

The iTAC OPC server makes available access to the API. Due to extensive configuration possibilities on the iTAC OPC server, this access can be used to represent and/or influence various conditions in iTAC.MES.Suite. Conditional API calls can also be represented here by means of appropriate configurations.

Because OPC (**O**LE for **P**rocess **C**ontrol) is based on Microsoft COM/DCOM, this process must be installed on a Windows system. Microsoft does not supply a COM/DCOM implementation for other systems (such as e.g. Linux). This will, however, change with future OPC versions. OPC will then be based on a platform independent protocol (OPC UA [OPC Unified Architecture] by means of SOAP).

OPC-based SCADA systems can correspondingly access the configured variables in the iTAC OPC server and then optically display them either for display or edit purposes.

e.g. Siemens, Simatic WinCC
Wonderware, InTouch

2 Session handling

2.1 General

When using the "ewMII" (see chapter 1.3.3), "COM-Bridge" (see chapter 1.3.4) or "Web Service Adapter" (see chapter 1.3.5) API libraries, note the following:

- When calling the technical functions (see chapter 3 »API functions«), the first parameter is **ALWAYS** the **`sessionID`**; the `sessionID` must first be ascertained with the »apiLogin« function.

ADVICE

Note that the **`sessionID`** *is not listed as a parameter in the description of the technical functions!*

- Multiple sessions may be active simultaneously (multiple logins). The passed `sessionID` specifies the session for which the respective call is being executed.

2.2 apiHeartbeat

ADVICE

*This function is intended for future extensions and must **NOT** (yet) be used!*

Function declaration (CORBA)

```
long apiHeartbeat(
     in long sessionId,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
sessionId	Unique designator (ID) of the active session

TAB. 2-1 *Parameter (in) »apiHeartbeat«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 2-2 *Parameter (out) »apiHeartbeat«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: Invalid/unknown session id
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 2-3 *Output values for »apiHeartbeat«*

2.3 apiLogin

This function must always be called before actually calling the technical function (see chapter 3 »API functions«). User or machine authentication is performed by means of this function. Following successful login, the server assigns a unique designator for the session (`sessionID`); this designator is retained until the »apiLogout« API function is called.

ADVICE

Unlike machine authentication, for user authentication a password must be passed in addition to the login name!

Function declaration (CORBA)

```
long apiLogin(
     in ApiSessionValidationStruct sessValData,
     out string errorString)
raises (ServerException);

struct ApiSessionValidationStruct
     {
        string user;
        string password;
        string client;
        boolean isVip;
        boolean isStation;
        string systemIdentifier;
     };
```

Parameter (in)

Parameter name	Explanation
`sessValData`	Structure with the following variables:
`user`	Login name: user name or station number of the work station in iTAC.MES.Suite
`password`	Password of the user (person) ***Advice:*** *No password is required for stations; it is, however, necessary to pass at least an empty string, as NULL is invalid!*
`client`	Abbreviation for the client
`isVip`	TRUE = User/station is identified as VIP FALSE = User/station is not identified as VIP ***Advice:*** *User logins should always set the value to FALSE; machine logins should always set the value to TRUE!*
`isStation`	TRUE = `user` is a station FALSE = `user` is a user (person) ***Advice:*** *User logins should always set the value to FALSE; machine logins should always set the value to TRUE!*
`systemIdentifier`	Unique designator (ID) of the client ***Advice:*** *"Hostname/IP address" is generally used as the designator. This designator is output in the log file and thereby serves to simplify assignment of log information!*

TAB. 2-4 *Parameter (in) »apiLogin«*

Parameter (out)

Parameter name	Explanation
`errorString`	Output text according to output value (see table)

TAB. 2-5 *Parameter (out) »apiLogin«*

Output values

Value	Explanations (errorString)
`> 0`	OK: Value is the unique designator (ID) of the active session (`sessionID`)
`-1`	Error: Invalid/user/capacity/password
`-2`	Error: Unknown client
`-3`	Error: User/capacity locked
`-4`	Error: Licenses exhausted, login denied
`-99`	Error: Server error
`-100`	Error: Corba no Service
`-101`	Error: Corba Exception

TAB. 2-6 *Output values for »apiLogin«*

2.4 apiLogout

This function is used to deactivate the specified session. This function should be called before the client application is exited so that all processes of the session can be deconstructed in a defined manner.

Function declaration (CORBA)

```
void apiLogout(
     in long sessionId,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
sessionId	Unique designator (ID) of the active session

TAB. 2-7 *Parameter (in) »apiLogout«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 2-8 *Parameter (out) »apiLogout«*

Output values This function provides returns no result.

3 API functions

This section explains all functions, including their variables and input and output values.

ADVICE

*When calling the functions, the first parameter is **ALWAYS** the* `sessionID`*; the* `sessionID` *must first be ascertained with the »apiLogin« function!*

3.1 activateRecipe

This function activates a specified recipe version. All other recipe versions are deactivated. To identify a recipe uniquely, the recipe header structure and the recipe version to be activated must be passed.

Function declaration (CORBA)

```
long activateRecipe(
     in long recipeId,
     in long recipeVersionId,
     in string stationNr,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
`recipeId`	Identification number of the recipe header structure
`recipeVersionId`	Identification number of the recipe version
`stationNr`	Station no. of the work station in iTAC.MES.Suite ***Special case:** When passing [0], the user registered at the station is passed, otherwise, the default API user is assigned!*

TAB. 3-1 *Parameter (in) »activateRecipe«*

Parameter (out)

Parameter name	Explanation
`errorString`	Output text according to output value (see table)

TAB. 3-2 *Parameter (out) »activateRecipe«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-61	Error: Database error while fetching recipe data
-63	Error: Invalid session
-65	Error: Recipe not activate
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-3 *Output values for »activateRecipe«*

3.2 activateWorkorder

This function activates or deactivates the passed work order on the station or line.

Function declaration (CORBA)

```
long activateWorkorder(
     in string station,
     in string workorder,
     in long processLayer,
     in long flag,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
station	Station no. of the work station in iTAC.MES.Suite
workorder	Number of a production order
processLayer	Orientation of the PCB during the work step [0; 1; 2; 3]: **0** = Component side **1** = Solder side **2** = Position independent **3** = Double-sided panel ***Advice:*** *If the value [3] is passed, the possibly differing work steps for component and solder side are combined for this work order and are considered as a whole (e.g. for the setup check)!*
flag	Mode "Work order activation" [1; 2; 3; 4]: **1** = Activate work order for the station only **2** = Activate work order for entire line **3** = Deactivate work order for station/line **4** = Deactivate and finish work order for station/line

TAB. 3-4 *Parameter (in) »activateWorkorder«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-5 *Parameter (out) »activateWorkorder«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: activation failed
-2	Error: workorder not found
-3	Error: station not found
-5	Error: `Flag` has an invalid value
-6	Error: `processLayer` has an invalid value
-7	Error: Station is not valid for this workorder
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station

TAB. 3-6 *Output values for »activateWorkorder«*

Value	Explanations (errorString)
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-6 *Output values for »activateWorkorder«*

3.3 appendAttributesToMaterialBin

This function enables the assignment of attributes to a container number. This allows individual information, such as a customer-specific material number and a specific manufacturing characteristic, to be assigned to a container number. Multiple, different attribute codes can be assigned to a container number (successively); an attribute code may only be used once per container number, however.

ADVICE

Please note that this function only assigns the actual value for the attribute (`attributeValue`). The attribute (`attributeCode`) must already exist in the system; the attributes are created in the TR Module under the "Unit information" function or with »createAttribute«.

In the context of serial number attributes, the »getAttributesForMaterialBin«, »getMaterialBinsForAttribute«, and »removeAttributeFromSerialNumber« functions must also be taken into account!

Function declaration (CORBA)

```
long appendAttributesToMaterialBin(
     in string stationNr,
     in string materialBinNr,
     in AttributeArray infos,
     out string errorString)
raises (ServerException);

struct AttributeInfo
     {
        string attributeCode;
        string value;
     };
typedef sequence<AttributeInfo> AttributeArray;
```

Parameter (in)

Parameter name	Explanation
`stationNr`	Station no. of the work station in iTAC.MES.Suite
`materialBinNr`	Number of the material bin (container no.)
`infos`	Array with the following variables:
`attributeCode`	Attribute code
`value`	Value of the attribute

TAB. 3-7 *Parameter (in) »appendAttributesToMaterialBin«*

Parameter (out)

Parameter name	Explanation
`errorString`	Output text according to output value (see table)

TAB. 3-8 *Parameter (out) »appendAttributesToMaterialBin«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-76	Error: attribute code must be at least one character
-78	Error: no TR license for this station
-92	Error: attribute code does not exist
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-401	Error: attribute code is already assigned to material bin
-434	Error: material bin was not found

TAB. 3-9 *Output values for »appendAttributesToMaterialBin«*

3.4 appendAttributeToSerialNumber

This function enables the assignment of attributes to a serial number. This allows individual information, such as a customer-specific material number and specific manufacturing characteristic, to be assigned to a serial number. Multiple, different attribute codes can be assigned to a serial number (successively); an attribute code may only be used once per serial number, however.

ADVICE

Please note that this function only assigns the actual value for the attribute (`attributeValue`). The attribute (`attributeCode`) must already exist in the system; the attributes are created in the TR Module under the "Unit information" function or with »createAttribute«.

In the context of serial number attributes, the »getAttributesForSerialNumber«, »getSerialNumbersForAttribute«, and »removeAttributeFromSerialNumber« functions must also be taken into account!

Function declaration (CORBA)

```
long appendAttributeToSerialNumber(
     in string stationNr,
     in string serialNumber,
     in string attributeCode,
     in string attributeValue,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
`stationNr`	Station no. of the work station in iTAC.MES.Suite
`serialNumber`	Serial no. of the panel
`attributeCode`	Attribute code
`attributeValue`	Value of the attribute ***Advice:*** *If "unique" was selected as the key type for this attribute during creation in the TR Module, the value passed here must not yet exist in the site (see output value `-398`)!*

TAB. 3-10 *Parameter (in) »appendAttributeToSerialNumber«*

Parameter (out)

Parameter name	Explanation
`errorString`	Output text according to output value (see table)

TAB. 3-11 *Parameter (out) »appendAttributeToSerialNumber«*

Output values

Value	Explanations (errorString)
`0`	OK: [empty]
`-3`	Error: station not found
`-39`	Error: serialnumber not found
`-76`	Error: attribute code must be at least one character
`-78`	Error: no TR license for this station

TAB. 3-12 *Output values for »appendAttributeToSerialNumber«*

Value	Explanations (errorString)
-92	Error: attribute code does not exist
-96	Error: attribute code is already assigned to serialnumber
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-398	Error: key attribute value not unique

TAB. 3-12 *Output values for »appendAttributeToSerialNumber«*

3.5 appendAttributeToWorkorder

This function enables the assignment of attributes to an order number. This allows individual information, such as a customer-specific material number and specific manufacturing characteristic, to be assigned to an order. If the order number is not known, a serial number can be used instead to ascertain the work order.

Multiple attributes can be assigned to a work order (successively); an attribute code may only be used more than once for a given work order if the `attributeValue-Desc` parameter is used.

If, in addition to the work order, a part number is passed – this part number must be contained in the order BOM – a reference is established between the work order attribute and this component. This can be used e.g. to assign a concrete IC from the BOM to the "software version" work order attribute with characteristic "V1.2" as attribute value. This IC can also be queried again at a later time.

ADVICE

Please note that this function only assigns the actual value for the attribute (`attributeValue`). The attribute (`attributeCode`) must already exist in the system; the attributes are created in the TR Module under the "Unit information" function or with »createAttribute«.

In the context of work order attributes, the »getAttributesForWorkorder«, »getWorkordersForAttribute«, and »removeAttributeFromWorkorder« functions must also be taken into account!

Function declaration (CORBA)

```
long appendAttributeToWorkorder(
     in string stationNr,
     in string chargeExt,
     in string serialNumber,
     in string partNo,
     in string attributeCode,
     in string attributeValue,
     in string attributeValueDesc,
     in string attributeType,
     in string objectKey,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
`stationNr`	Station no. of the work station in iTAC.MES.Suite
`chargeExt`	Production order number ***Advice:** If the value [-1] is passed, the parameter is not evaluated; in this case, it is absolutely necessary that a serial number be passed (see `serialNumber`)!*
`serialNumber`	Serial no. of the panel ***Advice:** If the value [-1] is passed, the parameter is not evaluated; in this case, it is absolutely necessary that an order number be passed (see `chargeExt`)!*
`partNo`	Part no. in iTAC.MES.Suite ***Advice:** This parameter is optional and enables the creation of work order attributes with a part reference; the part number must be contained in the order BOM. If no part reference is to be established, the value [-1] must be passed here!*

TAB. 3-13 *Parameter (in) »appendAttributeToWorkorder«*

Parameter name	Explanation
attributeCode	Attribute code
attributeValue	Value of the attribute
attributeValueDesc	Description of the attribute value (optional) ***Advice:*** *If this parameter is used, it is also possible to assign multiple attribute values (*`attributeValue`*) to an attribute (*`attributeCode`*)!*
attributeType	Attribute type ***Advice:*** *This parameter is optional and enables the classification of work order attributes into individual types. If no organization by attribute type is to be performed, the value [-1] must be passed here!*
objectKey	Key term for the attribute ***Advice:*** *This parameter is optional and enables, in addition to organization by type (see* `attributeType`*), another grouping option for the work order attributes. If no key term is assigned, the value [-1] must be passed here!* ***Special case:*** *If an empty string is passed here and the* `partNo` *parameter used, the passed part number is automatically used as* `objectKey`*!*

TAB. 3-13 *Parameter (in) »appendAttributeToWorkorder«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-14 *Parameter (out) »appendAttributeToWorkorder«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-76	Error: attribute code must be at least one character
-78	Error: no TR license for this station
-81	Error: workorder not found
-92	Error: attribute code does not exist
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-215	Error: attribute code is already assigned to workorder
-220	Error: Part not found

TAB. 3-15 *Output values for »appendAttributeToWorkorder«*

3.6 assignBatchNoToWorkorder

This function is used to create an empty material bin for "bin-based traceability" and assign it to a work order. The part number of the unit which is to be manufactured with the work order is automatically assigned to the material bin. Here, the work order does not need to be active on the station. To determine the work order the bin should be assigned to, and to determine which tests are to be carried out for this, the input values **`partNumber`** (part), **`bomVersion`** (BOM version), and **`workOrder`** (work order) may be combined.

ADVICE

If no work order is passed via `workOrder`*, a search for a suitable work order is carried out based on the passed product data (*`partNumber` *and* `bomVersion`*). If no work order is found, a new work order is created. It is a condition for automatic work order creation that the work plan uses work order type "01". This API function is influenced by station parameter code nos.: 13301 and 13302. Further information on this topic can be found in the documentation for the ADM Module!*
In the context of bin-based traceability, the »getRegisteredBatch«, »mergeBatch«, »registerBatch«, »splitBatchNoToSerialNumber« and »unregisterBatch« functions must also be taken into account!

Combinations if the work order number is known:

- **`partNumber`**, **`bomVersion`** and **`workorder`**
  all dependencies are checked for plausibility, the work order passed must match the part and BOM version
- **`partNumber`** and **`workorder`**
  the work order passed must match the part, it is not checked whether the BOM version is correct
- **`workorder`**
  the bin is assigned to the passed work order without further checking

Combinations if the WO number is unknown:

- **`partNumber`** and **`bomVersion`**
  searches for a suitable WO for the bin based on the passed part and on the BOM version; if several WOs are found, the bin is assigned to the most recent WO; if no WO is found, a new WO is created
- **`partNumber`**
  searches for a suitable WO for the bin based on the passed part; if several WOs are found, the bin is assigned to the most recent WO; if no WO is found, a new WO is created

Function declaration (CORBA)

```
long assignBatchNoToWorkorder(
     in string stationNr,
     in string workorder,
     in string partNumber,
     in string bomVersion,
     in long processLayer,
     in string batchNumber,
     in double quantity,
     in long activateWorkorder,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
workorder	Number of a production order ***Advice:*** *If an empty string is passed, a search for a suitable WO is carried out based on the product data (*`partNumber` *and* `bomVersion`*)!*
partNumber	Part no. in iTAC.MES.Suite ***Advice:*** *If this field is not to be evaluated (*`workorder` *is used), the value [-1] must be passed!*
bomVersion	Bill-of-material version from iTAC.MES.Suite ***Advice:*** *If this field is not to be evaluated (*`workOrder` *and/ or* `partNumber` *are used), the value [-1] must be passed!*
processLayer	Orientation of the PCB during the work step [-1; 0; 1; 2]: **-1** = if **activateWorkOrder** = 0 **0** = Component side **1** = Solder side **2** = Position independent ***Advice:*** *The orientation of the PCB only needs to be passed if work order activation is used (***activateWorkOrder** *= 1)!*
batchNumber	Number of the material bin (container no.) ***Advice:*** *The part number of the unit which is to be manufactured with the work order is automatically assigned to the material bin!*
quantity	Quantity of material container ***Advice:*** *The value passed here determines the maximum possible number of units contained in the bin!*
activateWorkOrder	Mode "Work order activation" [0; 1]: **1** = Activate work order **0** = Do not activate work order ***Advice:*** *If work order activation is used (***activateWorkOrder** *= 1)) the orientation of the PCB of the current work step must be passed (see* **processLayer***)!*

TAB. 3-16 *Parameter (in) »assignBatchNoToWorkorder«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-17 *Parameter (out) »assignBatchNoToWorkorder«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-39	Error: serialnumber not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-67	Error: no valid bom version
-78	Error: no TR license for this station
-81	Error: workorder not found
-82	Error: serialnumber belongs to another workorder
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-220	Error: Part not found
-301	Error: Creating an already existing container no. is only allowed if the old one is empty
-302	Error: Error while deactivating container
-303	Error: Container has a binLocation
-356	Error: Error while creating materialBin
-362	Error: Partnumber not found
-363	Error: Quantity not allowed
-364	Error: serialnumber already assigned to this workorder
-365	Error: Part does not fit to workorders part
-366	Error: Unable to create workorder
-367	Error: State of workorder is not valid ***Advice:*** *Value is only returned, if the station parameter code no. 13301 is active!*
-368	Error: Workorder quantity is exceeded ***Advice:*** *Value is only returned, if the station parameter code no. 13301 is active!*
-369	Error: materialBin is locked

TAB. 3-18 *Output values for »assignBatchNoToWorkorder«*

3.7 assignSerialNumberForProductOrWorkorder

This function assigns a serial number to a work order. Here, the work order does not need to be active on the station. To determine the work order the serial number should be assigned to, and to determine which tests are to be carried out for this, the input values **`partNr`** (part), **`bomVersion`** (BOM version), and **`workOrder`** (work order) may be combined.

ADVICE

If no WO is passed via `workOrder`*, a search for a suitable WO is carried out based on the passed product data (*`partNr` *and* `bomVersion`*). If no WO is found, a new work order is created (cf. API function »assignSerialNumberToWorkOrder«). It is a condition for automatic work order creation that the work plan uses work order type "01".*
This API function is influenced by station parameter code numbers: 4610, 7208, 13301 and 13302. Further information on this topic can be found in the documentation for the ADM Module!

Combinations if the work order number is known:

- **`partNr`**, **`bomVersion`** and **`workOrder`**
  all dependencies are checked for plausibility, the work order passed must match the part and BOM version
- **`partNr`** and **`workOrder`**
  the work order passed must match the part, it is not checked whether the BOM version is correct
- **`workOrder`**
  the serial number is assigned to the work order passed without further checking

Combinations if the WO number is unknown:

- **`partNr`** and **`bomVersion`**
  searches for a suitable WO for the serial number based on the passed part and on the BOM version; if several WOs are found, the serial number is assigned to the most recent WO; if no WO is found, a new WO is created
- **`partNr`**
  searches for a suitable WO for the serial number based on the passed part; if several WOs are found, the serial number is assigned to the most recent WO; if no WO is found, a new WO is created

Function declaration (CORBA)

```
long assignSerialNumberForProductOrWorkorder(
      in string stationNr,
      in string workOrder,
      in string partNr,
      in string bomVersion,
      in string serialNumberRef,
      in string serialNumberRefPos,
      in long processLayer,
      in long numberOfRecords,
      in SerialNumberArray serialNumberArray,
      in long activateWorkOrder,
      out string errorString)
raises (ServerException);

struct SerialNumberData
```

```
        {
           string serialNumber;
           string serialNumberPos;
        };
    typedef sequence<SerialNumberData> SerialNumberArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
workOrder	Number of a production order ***Advice:*** *If an empty string is passed, a search for a suitable WO is carried out based on the product data (*`partNr` *and* `bomVersion`*)!*
partNr	Part no. in iTAC.MES.Suite ***Advice:*** *If this field is not to be evaluated (*`workOrder` *is used), the value [-1] must be passed!*
bomVersion	Bill-of-material version from iTAC.MES.Suite ***Advice:*** *If this field is not to be evaluated (*`workOrder` *and/ or* `partNr` *are used), the value [-1] must be passed!*
serialNumberRef	Serial no. of the first single panel or unit-array; this no. is used as reference serial no. ***Special case:*** *If the value [-1] is passed here and for* `serialNumberRefPos`*, all serial numbers passed in the array are booked as single panels!*
serialNumberRefPos	Position of the serial no. of the first single panel or unit-array ***Special case:*** *If the value [-1] is passed here and for* `serialNumberRef`*, all serial numbers passed in the array are booked as single panels!*
processLayer	Orientation of the PCB during the work step [-1; 0; 1; 2]: **-1** = if **activateWorkOrder** = 0 **0** = Component side **1** = Solder side **2** = Position independent ***Advice:*** *The orientation of the PCB only needs to be passed if work order activation is used (***activateWorkOrder** *= 1)!*
numberOfRecords	Default value for the number of entries in the array
serialNumberArray	Array with the following variables:
`serialNumber`	Serial no. of the panel
`serialNumberPos`	Position of the single panel in the unit-array
activateWorkOrder	Mode "Work order activation" [0; 1]: **1** = Activate work order **0** = Do not activate work order ***Advice:*** *If work order activation is used (***activateWorkOrder** *= 1)) the orientation of the PCB of the current work step must be passed (see* **processLayer***)!*

TAB. 3-19 *Parameter (in) »assignSerialNumberForProductOrWorkorder«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-20 *Parameter (out) »assignSerialNumberForProductOrWorkorder«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: workorder not found
-2	Error: Serialnumber already assigned to another workorder
-3	Error: station not found
-4	Error: Serialnumber already assigned to this workorder
-5	Error: bomVersion does not fit workorders articleVersion
-6	Error: Part not found
-7	Error: Part does not fit to workorders part
-8	Error: unable to create workorder
-11	Error: merge is invalid
-12	Error: Slave is not finished
-13	Error: Serialnumber is part of another workorder and locked
-18	Error: State of workorder is not valid ***Advice:*** *Value is only returned, if the station parameter code no. 13301 is active!*
-19	Error: Workorder quantity is exceeded ***Advice:*** *Value is only returned, if the station parameter code no. 13301 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-21 *Output values for »assignSerialNumberForProductOrWorkorder«*

3.8 assignSerialNumberMergeAndUploadState

This function combines several API functions that are also available as individual functions. The functions that are combined are »assignSerialNumberToWorkOrder«, »mergeParts« and »uploadState«. The direct association between the assignment of a serial no. to a WO (assign) and the product merging with another serial no. (merge) prevents a serial no. from being assigned to several WOs at the same time. As a result, this function ensures a higher process reliability compared to the separate function calls.

ADVICE

This API function is influenced by station parameter code numbers: 4610, 7208, 13301 and 13302. Further information on this topic can be found in the documentation for the ADM Module!

The function operates as follows:

1. During a transaction, serial numbers that are returned are assigned to a product version and work order that have been configured on the line; independent of whether this is a multiple or single panel, the serial numbers are passed via the array `serialNumberArray`.
   Subsequently, a product merging is carried out (merge slave product into master product), during which both the serial no. passed via `serialNumberSlave` and the referenced serial nos. (in case of multiple printed panels) are discarded.

   Before the assignment, a fundamental plausibility check is carried out to ensure that the serial no. may be used. If the product merging is erroneous, the assignment to the WO is undone as well.
2. After successful assignment and merging, an optional »uploadState« may be carried out. This causes the serial nos. and their states to be booked in a work step conformant manner. If an unexpected abort takes place, only the »uploadState« must be repeated; Point 1 is no longer affected.

Function declaration (CORBA)

```
long assignSerialNumberMergeAndUploadState(
      in string stationNr,
      in long processLayer,
      in string serialNumberRef,
      in string serialNumberRefPos,
      in long numberOfRecords,
      in SerialNumberArray serialNumberArray,
      in string serialNumberSlave,
      in long doUploadState,
      in long serialNumberState,
      out string errorString)
raises (ServerException);

struct SerialNumberData
      {
         string serialNumber;
         string serialNumberPos;
      };
typedef sequence<SerialNumberData> SerialNumberArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
serialNumberRef	Serial no. of the first single panel or unit-array; this no. is used as reference serial no. ***Advice:*** *Serial no. which becomes active during a product merging (see* `serialNumberSlave`*)!* ***Special case:*** *If the value [-1] is passed here and for* `serialNumberRefPos`*, all serial numbers passed in the array are booked as single panels!*
serialNumberRefPos	Position of the serial no. of the first single panel or unit-array ***Special case:*** *If the value [-1] is passed here and for* `serialNumberRef`*, all serial numbers passed in the array are booked as single panels!*
numberOfRecords	Default value for the number of entries in the array ***Special case:*** *If the value [-1] (single panel) is passed for* `serialNumberRef` *and* `serialNumberRefPos`*, the value for* `numberOfRecords` *must be [1]!*
serialNumberArray	Array with the following variables:
`serialNumber`	Serial no. of the panel ***Advice:*** *Serial no. which becomes active during a product merging (see* `serialNumberSlave`*)!*
`serialNumberPos`	Position of the single panel in the unit-array ***Special case:*** *If the value [-1] is passed for* `serialNumberRef` *and* `serialNumberRefPos`*, this field is not evaluated (single panel)!*
serialNumberSlave	Serial no. which becomes inactive during product merging (see `serialNumberRef` or `serialNumber`) ***Advice:*** *The serial number can still be used as a new* `serialNumberRef` *(multiple panel) or* `serialNumber` *(single panel)!*
doUploadState	Activation/deactivation [0; 1]: **0** = Function »uploadState« is not executed **1** = Function »uploadState« is executed
serialNumberState	State of the product associated with the serial no. [0; 1; 2]: **0** = Pass **1** = Fail **2** = Scrap

TAB. 3-22 *Parameter (in) »assignSerialNumberMergeAndUploadState«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-23 *Parameter (out) »assignSerialNumberMergeAndUploadState«*

Output values

Value	Explanations (errorString)
`0`	OK: [empty]
`-1`	Error: no active workOrder found
`-2`	Error: serialNumber has already been given to another workOrder
`-3`	Error: station not found
`-4`	Error: serialNumber has already been given to the actual workOrder
`-5`	Error: stationNr is not valid for adding serialNumbers
`-6`	Error: No serialNumbers added because they have always been given to the actual workOrder
`-10`	Error: Merge error
`-11`	Error: merge is invalid
`-12`	Error: no workOrder found for serialNumber
`-13`	Error: amount of master and slave-serialNumbers is not equal
`-14`	Error: serialNumber slave is not finished
`-15`	OK: … but serialNumber slave is FAIL at last workstep
`-16`	OK: … but serialNumber slave is SCRAP at last workstep
`-17`	Error: serialnumber is locked
`-18`	Error: State of workorder is not valid ***Advice:*** *Value is only returned, if the station parameter code no. 13301 is active!*
`-19`	Error: Workorder quantity is exceeded ***Advice:*** *The value is only returned if the station parameter code no. 13301 is active and* **`doUploadState`** *"0" is passed as input parameter!*
`-21`	Error: `serialNumberState` not valid [0; 1; 2]
`-22`	Error: station is not valid for this serialNumber
`-23`	Error: serialNumber is not a serialNumberRef
`-24`	Error: Upload database error
`-25`	Error: serialNumber not unique
`-78`	Error: no TR license for this station
`-99`	Error: Server error
`-100`	Error: Corba no Service
`-101`	Error: Corba Exception

TAB. 3-24 *Output values for »assignSerialNumberMergeAndUploadState«*

3.9 assignSerialNumberToWorkOrder

With this function, serial numbers that are returned are assigned to a product version and order that have been configured at the station. Before the assignment, a plausibility check is carried out to ensure that the serial no. may be used.

ADVICE

This API function is influenced by station parameter code numbers: 4610, 7208, 13301 and 13302. Further information on this topic can be found in the documentation for the ADM Module!

Function declaration (CORBA)

```
long assignSerialNumberToWorkOrder(
     in string stationNr,
     in long processLayer,
     in string serialNumberRef,
     in string serialNumberRefPos,
     in long numberOfRecords,
     in SerialNumberArray serialNumberArray,
     out string errorString)
raises (ServerException);

struct SerialNumberData
     {
        string serialNumber;
        string serialNumberPos;
     };
typedef sequence<SerialNumberData> SerialNumberArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent ***Advice:*** *The orientation of the PCB of the current work step according to the active work order must be passed here!*
serialNumberRef	Serial no. of the first single panel or unit-array; this no. is used as reference serial no. ***Special case:*** *If the value [-1] is passed here and for* `serialNumberRefPos`*, all serial numbers passed in the array are booked as single panels!*
serialNumberRefPos	Position of the serial no. of the first single panel or unit-array ***Special case:*** *If the value [-1] is passed here and for* `serialNumberRef`*, all serial numbers passed in the array are booked as single panels!*
numberOfRecords	Default value for the number of entries in the array
serialNumberArray	Array with the following variables:
`serialNumber`	Serial no. of the panel
`serialNumberPos`	Position of the single panel in the unit-array

TAB. 3-25 *Parameter (in) »assignSerialNumberToWorkOrder«*

Parameter (out)

Parameter name	Explanation
`errorString`	Output text according to output value (see table)

TAB. 3-26 *Parameter (out) »assignSerialNumberToWorkOrder«*

Output values

Value	Explanations (errorString)
`0`	OK: [empty]
`-1`	Error: no active workOrder found
`-2`	Error: serialNumber has already been given to another workOrder
`-3`	Error: station not found
`-4`	Error: serialNumber has already been given to the actual workOrder
`-5`	Error: stationNr is not valid for adding serialNumbers
`-6`	Error: No serialNumbers added because they have always been given to the actual workOrder
`-17`	Error: serialnumber is locked
`-18`	Error: State of workorder is not valid ***Advice:*** *Value is only returned, if the station parameter code no. 13301 is active!*
`-19`	Error: Workorder quantity is exceeded ***Advice:*** *Value is only returned, if the station parameter code no. 13301 is active!*
`-78`	Error: no TR license for this station
`-99`	Error: Server error
`-100`	Error: Corba no Service
`-101`	Error: Corba Exception

TAB. 3-27 *Output values for »assignSerialNumberToWorkOrder«*

3.10 bomCheck

This function compares the current setup data of a station with the BOM data for the PCB for which the component placement is to be undertaken. This avoids incorrect component placement as a result of incorrect setup.

ADVICE

Compared to »bomCheckForWorkOrder«, this function is based on the input of a serial no.

Function declaration (CORBA)

```
long bomCheck(
      in string stationNr,
      in long processLayer,
      in string serialNumber,
      in long checkActivPreparation,
      out long numberOfRecords,
      out BomCompFailDataArray aBomCompFailData,
      out string errorString)
raises (ServerException);

struct BomCompFailData
      {
         string compName;
         string partNr;
         string comment;
      };
typedef sequence<BomCompFailData> BomCompFailDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2; 3]: **0** = Component side **1** = Solder side **2** = Position independent **3** = Double-sided panel ***Advice:** If the value [3] is passed, the setup data for the component and solder side are combined and considered as a whole!*
serialNumber	Serial no. of the panel ***Special case:** For input <= 1 character, this variable is not evaluated; instead, the setup of the active WO is used!*
checkActivPreparation	Comparison of the setup data [0; 1]: **0** = Preset materials are compared with current order data (BOM) **1** = Current setup and pre-setup are compared with current order data (BOM)

TAB. 3-28 *Parameter (in) »bomCheck«*

Parameter (out)

Parameter name	Explanation
numberOfRecords	Number of entries in the array returned
aBomCompFailData	Array with the following variables:

TAB. 3-29 *Parameter (out) »bomCheck«*

Parameter name	Explanation
compName	The mounting place conforms to the iTAC.MES.Suite BOM
partNr	Part no. in iTAC.MES.Suite
comment	Comment
errorString	Output text according to output value (see table)

TAB. 3-29 *Parameter (out) »bomCheck«*

Output values

Value	Explanations (errorString)
5	Warning:No station for setup activated
4	Warning:No bom-position set for setup
3	Warning:No process step defined for setup
2	Warning:Setup not completely activated
1	Warning:Setup not activated
0	OK: [empty]
-1	Error: stationNr not found
-2	Error: workorder not found
-3	Error: serialNumber not found
-4	Error: wrong workorder - setup prepared for another workorder
-5	Error: supplier not found
-6	Error: BOM not found
-7	Error: no parts setup
-8	Error: preparation not valid, material is missing
-9	Error: station is not valid for preparation in workplan
-10	Error: serialNumber is assigned to another workorder
-11	Error: BOM is assigned to another workorder than serial number
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-30 *Output values for »bomCheck«*

3.11 bomCheckForWorkOrder

This function compares the current setup data of a station with the BOM data for the PCB for which the component placement is to be undertaken. This avoids incorrect component placement as a result of incorrect setup.

ADVICE

Compared to the function »bomCheck« already described, this function is based on the input of a work order no.!
This API function is influenced by station parameter code no.: 13120. Further information on this topic can be found in the documentation for the ADM Module!

Function declaration (CORBA)

```
long bomCheckForWorkorder(
     in string stationNr,
     in long processLayer,
     in string workorder,
     in long checkActivPreparation,
     out long numberOfRecords,
     out BomCompFailDataArray aBomCompFailData,
     out string errorString)
raises (ServerException);

struct BomCompFailData
     {
        string compName;
        string partNr;
        string comment;
     };
typedef sequence<BomCompFailData> BomCompFailDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2; 3]: **0** = Component side **1** = Solder side **2** = Position independent **3** = Double-sided panel ***Advice:** If the value [3] is passed, the setup data for the component and solder side are combined and considered as a whole!*
workorder	Number of a production order
checkActivPreparation	Comparison of the setup data [0; 1]: **0** = Preset materials are compared with current order data (BOM) **1** = Current setup and pre-setup are compared with current order data (BOM)

TAB. 3-31 *Parameter (in) »bomCheckForWorkOrder«*

Parameter (out)

Parameter name	Explanation
numberOfRecords	Number of entries in the array returned
aBomCompFailData	Array with the following variables:

TAB. 3-32 *Parameter (out) »bomCheckForWorkOrder«*

Parameter name	Explanation
compName	The mounting place conforms to the iTAC.MES.Suite BOM
partNr	Part no. in iTAC.MES.Suite
comment	Comment
errorString	Output text according to output value (see table)

TAB. 3-32 *Parameter (out) »bomCheckForWorkOrder«*

Output values

Value	Explanations (errorString)
5	Warning:No station for setup activated
4	Warning:No bom-position set for setup
3	Warning:No process step defined for setup
2	Warning:Setup not completely activated
1	Warning:Setup not activated
0	OK: [empty]
-1	Error: stationNr not found
-2	Error: workorder not found
-3	Error: serialNumber not found
-4	Error: wrong workorder - setup prepared for another workorder
-5	Error: supplier not found
-6	Error: BOM not found
-7	Error: no parts setup
-8	Error: preparation not valid, material is missing
-9	Error: station is not valid for preparation in workplan
-10	Error: serialNumber is assigned to another workorder
-11	Error: BOM is assigned to another workorder than serial number
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-33 *Output values for »bomCheckForWorkOrder«*

3.12 changeFeederBank

This function can be used to exchange a feeder bank on a mounting device. Because individual feeders cannot be exchanged during operation, two feeder banks are often loaded identically. When a roll of one feeder of the feeder bank becomes empty, the entire bank is retracted for reloading. As this occurs, mounting continues with the other feeder bank.

ADVICE

*To exchange a feeder bank, the function must be executed twice: on the first call, the current feeder bank must be deactivated (`flag` = **0**), on the second call, the new feeder bank must be activated (`flag` = **1**)!*

Function declaration (CORBA)

```
long changeFeederBank(
     in string stationNr,
     in string feederBank,
     in long flag,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
feederBank	Labelling of the feeder bank
flag	Automatic feeder bank activation [0; 1]: **0** = Current feeder bank is deactivated **1** = New feeder bank is activated

TAB. 3-34 *Parameter (in) »changeFeederBank«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-35 *Parameter (out) »changeFeederBank«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:** Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-36 *Output values for »changeFeederBank«*

3.13 changeMatBinForSerialNumberRepair

This function is used to exchange the container for a specific component of a unit (individual item); e.g. component exchange during the course of a repair.

Function declaration (CORBA)

```
long changeMatBinForSerialNumberRepair(
      in string stationNo
      in long processLayer
      in string serialNumber
      in long serialNumberPos
      in string materialNo
      in string compName
      in string materialBinNoOld
      in string materialBinNoNew
      out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNo	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
serialNumber	Serial no. of the panel
serialNumberPos	Position of the single panel in the unit-array ***Advice:*** *If the value [-1] is passed, the position is not evaluated; in this case, each single panel must be specified by means of a unique* `serialNumber` *!*
materialNo	Part no. in iTAC.MES.Suite for the component
compName	The mounting place conforms to the iTAC.MES.Suite BOM
materialBinNoOld	Number of the old material bin (container no.)
materialBinNoNew	Number of the new material bin (container no.)

TAB. 3-37 *Parameter (in) »changeMatBinForSerialNumberRepair«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-38 *Parameter (out) »changeMatBinForSerialNumberRepair«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-81	Error: workorder not found
-99	Error: Server error
-100	Error: Corba no Service

TAB. 3-39 *Output values for »changeMatBinForSerialNumberRepair«*

Value	Explanations (errorString)
-101	Error: Corba Exception
-216	Error: at least one material bin was not found
-220	Error: Part not found

TAB. 3-39 *Output values for »changeMatBinForSerialNumberRepair«*

3.14 changeMaterialBinAttributes

This function is used to change the attributes of a material bin (container). The values are passed via the array `configDataArray`.

This permits, e.g., the price (average purchase price) and price base (number of parts on which the purchase price is based) for the material on a material bin to be adjusted if there are deviations from the default values in the part master. By passing the attribute names (`cost` = price and `cost_base` = price base in the example) and the associated "new" values, the attributes for the material bin passed are adjusted.

ADVICE

This function permits every attribute of a material bin to be changed. The imprudent adjustment of individual attributes may profoundly affect process reliability and stability. This function should, therefore, only be used in conjunction with close consultation with the iTAC employee responsible for the project!

Function declaration (CORBA)

```
long changeMaterialBinAttributes(
     in string stationNr,
     in string MaterialBinNr,
     in long numberOfRecords,
     in ConfigDataArray configDataArray,
     out string errorString)
raises (ServerException);

struct ConfigData
     {
        string name;
        string value;
     };
typedef sequence<ConfigData> ConfigDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
MaterialBinNr	Number of the material bin (container no.)
numberOfRecords	Default value for the number of entries in the array
configDataArray	Array with the following variables:
name	Name of the attribute to be changed
value	Value for the attribute selected via `name`

TAB. 3-40 *Parameter (in) »changeMaterialBinAttributes«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-41 *Parameter (out) »changeMaterialBinAttributes«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: Parameter Error
-2	Error: Session Error
-3	Error: Material Server Error
-4	Error: materialBinNr not found
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-42 *Output values for »changeMaterialBinAttributes«*

3.15 changeMaterialBinQuantity

This function changes the residual quantity in a material bin. The quantity passed is subtracted from or added to the quantity already specified on the bin. In addition, an order no. can be passed under which the material is booked. If this value is empty, it is booked without an order.

Function declaration (CORBA)

```
long changeMaterialBinQuantity(
     in string stationNr,
     in string materialBinNr,
     in double quantity,
     in long transactionType,
     in string workOrder,
     in string textInfo,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
materialBinNr	Number of the material bin (container no.) ***Advice:*** *This container must be stored in iTAC.MES.Suite!*
quantity	Quantity of material container ***Advice:*** *The current container quantity is changed by the quantity specified here; depending on the sign, the quantity is deducted or added!*
transactionType	Code for the movement type from iTAC.MES.Suite (see movement types ML Module)
workOrder	Number of a production order
textInfo	Arbitrary information

TAB. 3-43 *Parameter (in) »changeMaterialBinQuantity«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-44 *Parameter (out) »changeMaterialBinQuantity«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-2	Error: stationNr not found
-3	Error: materialBinNr not found
-4	Error: wrong quantity
-5	Error: workOrder not found
-6	Error: `transactionType` not valid
-59	Error: Container not in stock
-78	Error: no TR license for this station

TAB. 3-45 *Output values for »changeMaterialBinQuantity«*

Value	Explanations (errorString)
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-45 *Output values for »changeMaterialBinQuantity«*

3.16 changeWorkOrder

This function configures a new WO at a station. The WO is the one whose setup has advanced most.

ADVICE

Only released WOs can be set (`state` = R)!

Function declaration (CORBA)

```
long changeWorkOrder(
     in string stationNr,
     out string partNr,
     out string partDesc,
     out long bomVersion,
     out string bomIndex,
     out string cadPartNr,
     out long processLayer,
     out string workOrder,
     out long quantity,
     out string state,
     out string customerName,
     out string customerPartNr,
     out string attribut1,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite

TAB. 3-46 *Parameter (in) »changeWorkOrder«*

Parameter (out)

Parameter name	Explanation
partNr	Part no. in iTAC.MES.Suite
partDesc	Part description in iTAC.MES.Suite
bomVersion	Bill-of-material version from iTAC.MES.Suite
bomIndex	Bill-of-material version (index) that is passed to the system by means of changes in BOMs; these are manual changes or changes that have been initiated by higher-level ERP systems
cadPartNr	Part no. of the fundamental bearer, e.g., number of the "bare" printed circuit board
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
workOrder	Number of a production order
quantity	Quantity of production order

TAB. 3-47 *Parameter (out) »changeWorkOrder«*

Parameter name	Explanation
state	State of the production order [N; A; I; F; R; S; P; D; E]: **N** = new **A** = planned **I** = initialized **F** = created **R** = released **S** = started **P** = stopped **D** = deleted **E** = finished
customerName	Customer name in iTAC.MES.Suite
customerPartNr	Customer material no. in iTAC.MES.Suite
attribut1	Specific parameter from the part master (e.g., **H** for High, **L** for Low)
errorString	Output text according to output value (see table)

TAB. 3-47 *Parameter (out) »changeWorkOrder«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-2	Error: wrong station
-3	Error: wrong workOrder
-4	Error: wrong process
-5	Error: no recipe-data available
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-48 *Output values for »changeWorkOrder«*

3.17 changeWorkOrderForLine

The function configures a new order on all stations of one line. The order used is the one that has advanced most.

ADVICE

Only released WOs can be set (`state` = R)!

Function declaration (CORBA)

```
long changeWorkOrderForLine(
     in string stationNr,
     out string partNr,
     out string partDesc,
     out long bomVersion,
     out string bomIndex,
     out string cadPartNr,
     out long processLayer,
     out string workOrder,
     out long quantity,
     out string state,
     out string customerName,
     out string customerPartNr,
     out string attribut1,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite

TAB. 3-49 *Parameter (in) »changeWorkOrderForLine«*

Parameter (out)

Parameter name	Explanation
partNr	Part no. in iTAC.MES.Suite
partDesc	Part description in iTAC.MES.Suite
bomVersion	Bill-of-material version from iTAC.MES.Suite
bomIndex	Bill-of-material version (index) that is passed to the system by means of changes in BOMs; these are manual changes or changes that have been initiated by higher-level ERP systems
cadPartNr	Part no. of the fundamental bearer, e.g., number of the "bare" printed circuit board
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
workOrder	Number of a production order
quantity	Quantity of production order

TAB. 3-50 *Parameter (out) »changeWorkOrderForLine«*

Parameter name	Explanation
state	State of the production order [N; A; I; F; R; S; P; D; E]: **N** = new **A** = planned **I** = initialized **F** = created **R** = released **S** = started **P** = stopped **D** = deleted **E** = finished
customerName	Customer name in iTAC.MES.Suite
customerPartNr	Customer material no. in iTAC.MES.Suite
attribut1	Specific parameter from the part master (e.g., **H** for High, **L** for Low)
errorString	Output text according to output value (see table)

TAB. 3-50 *Parameter (out) »changeWorkOrderForLine«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-2	Error: wrong station
-3	Error: wrong workOrder
-4	Error: wrong process
-5	Error: no recipe-data available
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-51 *Output values for »changeWorkOrderForLine«*

3.18 checkEquipmentData

This function checks whether the resources required for the work order are valid and rigged at the station.

ADVICE

This API function is influenced by site parameter code no.: 13030. Further information on this topic can be found in the documentation for the ADM Module. In the context of resource management, the »getRequiredEquipmentData« »getSetupEquipmentData«, »removeEquipmentForWorkorder« and »updateEquipmentData« functions must also be taken into account! The various **upload** *functions that increase the usage counter for the maintenance interval do not check the validity of the equipment; to perform this check, this API function must be used explicitly!*

Function declaration (CORBA)

```
long checkEquipmentData(
     in string stationNr,
     in string workOrder,
     in string partNo,
     in long processLayer,
     in long returnFailuresOnly,
     out EquipmentCheckDataArray equipmentCheckData,
     out string errorString)
raises (ServerException);

struct EquipmentCheckData
     {
        string equipmentNo;
        string equipmentIndex;
        long checkStatus;
        string equipmentSetupGroup;
        string partNo;
        string partGrp;
        string equipmentExt;
        string equipmentDescription;
        long groupItemType;
        string info1;
        string info2;
        string info3;
     };
typedef sequence<EquipmentCheckData> EquipmentCheckDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
workOrder	Number of a production order ***Special case:** If [-1] is passed here, the work order which is active on the station is used!*
partNo	Part no. in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
returnFailuresOnly	Specifies which resources are output [0; 1]: **0** = All resources are output **1** = Only missing or invalid resources are output

TAB. 3-52 *Parameter (in) »checkEquipmentData«*

Parameter (out)

Parameter name	Explanation
equipmentCheckData	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
equipmentNo	Number of the resource in iTAC.MES.Suite
equipmentIndex	Individual designator for the resource
checkStatus	Check status for the resource [0; 1; 2; 3; 4; 5; 6]: **0** = **OK:** Resource is valid and rigged **1** = **Error:** Resource must either be maintained or the point in time of the last permitted use has been exceeded. ***Advice:*** *[1] is only output if site parameter code no. 13030 was set to TRUE; otherwise the value [2] is output!* **2** = **Warning:** Resource must either be maintained or the point in time of the last permitted use has been exceeded. ***Advice:*** *[2] is only output if site parameter code no. 13030 was set to FALSE; otherwise, the value [1] is output!* **3** = **Error:** Resource state is invalid; the resource must have the "available" state **4** = **Error:** Resource is not rigged ***Advice:*** *The affected resource group is output with parameter* equipmentSetupGroup*!* **5** = **Error:** Resource is invalid; period between "valid from" and "valid to" has been exceeded ***Advice:*** *[5] is only output if site parameter code no. 13030 is set to TRUE; otherwise the value [6] is output!* **6** = **Warning:** Resource is invalid; period between "valid from" and "valid to" was exceeded ***Advice:*** *[6] is only output if site parameter code no. 13030 was set to FALSE; otherwise the value [5] is output!*
equipmentSetup-Group	Name of the resource group in iTAC.MES.Suite; resource groups must be defined as work steps when assigning resources. ***Advice:*** *This parameter is only output if the value [4] is output for* checkStatus*!*
partNo	Part number (device class) on the basis of which the resource was created
partGrp	Part group for the part (device class)
equipmentExt	External resource number
equipmentDescrip-tion	Description of the resource

TAB. 3-53 *Parameter (out) »checkEquipmentData«*

Parameter name	Explanation
groupItemType	Assignment type for the resource [0; 1]: **0** = Resource was assigned directly **1** = Resource was assigned indirectly via a part (device class) (see partNo)
info1	
info2	Individual additional information 1 - 3
info3	
errorString	Output text according to output value (see table)

TAB. 3-53 *Parameter (out) »checkEquipmentData«*

Output values

Value	Explanations (errorString)
10	Warning:Equipment setup-check failed ***Advice:*** *Value is obtained from the resources passed in the array returned (the variable* checkStatus *is evaluated on the basis of a "worst case" assumption) and is only output if site parameter code no. 13030 was set to FALSE; otherwise the value "-309" is output!*
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-81	Error: workorder not found
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-308	Error: workplan not found
-309	Error: Equipment setup-check failed ***Advice:*** *Value is obtained from the resources passed in the array returned (the variable* checkStatus *is evaluated on the basis of a "worst case" assumption) and is only output if site parameter code no. 13030 was set to TRUE; otherwise the value "10" is output!*

TAB. 3-54 *Output values for »checkEquipmentData«*

3.19 checkMergedPartsForSnrComplete

This function checks whether all necessary components (semifinished products) were mounted in the finished product. In the array returned, either all or only the missing components can be output (see also parameter `allMergePartsVisible`). Components are returned according to their mounting place, i.e. components that are present multiple times are also output separately.

Function declaration (CORBA)

```
long checkMergedPartsForSnrComplete(
     in string stationNr,
     in string serialNumberRef,
     in long serialNumberPos,
     in long allMergePartsVisible,
     in long checkMultibleBoard,
     out MergePartArray mergePartArray,
     out string errorString)
raises (ServerException);

struct MergePartData
     {
        string serialNumber;
        long serialNumberPos;
        string compPartNo;
        string compName;
        double compBomQty;
        double mergedInSnrQty;
        string info1;
        string info2;
        string info3;
     };
typedef sequence<MergePartData> MergePartArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
serialNumberRef	Reference serial number: serial number of the 1st single panel or of the unit-array
serialNumberPos	Position of the reference serial number ***Advice:*** *If the value [-1] is passed, the position is not evaluated; in this case the passed* reference serial number *may not be part of a multiple panel!*
allMergePartsVisible	Specifies which components are output [0; 1]: **0** = All components are output **1** = Only the components that have not yet been mounted are output
checkMultibleBoard	If the finished product(`serialNumberRef`) is part of a multiple panel, this specifies whether only the passed panels (`serialNumberPos`) or all panels are checked [0; 1]: **0** = All panels are checked **1** = Only the passed panel is checked

TAB. 3-55 *Parameter (in) »checkMergedPartsForSnrComplete«*

Parameter (out)

Parameter name	Explanation
mergePartArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
serialNumber	Serial no. of the panel
serialNumber-Pos	Position of the single panel in the unit-array
compPartNo	Part no. of the component (semifinished product) in iTAC.MES.Suite
compName	The mounting place conforms to the iTAC.MES.Suite BOM
compBomQty	Number of this component in the BOM
mergedInSnrQty	Number of components already mounted in the finished product
info1	
info2	Individual additional information 1 - 3
info3	
errorString	Output text according to output value (see table)

TAB. 3-56 *Parameter (out) »checkMergedPartsForSnrComplete«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-39	Error: serialnumber not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-394	Error: Serialnumber is not complete ***Advice:*** *In combination with this value, the array returned is always also output with the missing components!*

TAB. 3-57 *Output values for »checkMergedPartsForSnrComplete«*

3.20 checkSnrStateBySnrRef

This function checks the state of a complete multiple printed panel separately for all single panels it contains. It is checked whether the transmitted serial no. is valid for the subsequent station and its associated work step. A plausibility check is carried out to determine whether the state of progress is OK and what current state (PASS, FAIL, SCRAP) the contained single panels possess.

With activation of station parameter code no. 13300, a check is also performed to determine whether the work order to which the serial number is assigned matches the work order currently active on the station.

ADVICE

The **checkSnrStateBySnrRef** *API function is influenced by station parameter code nos.: 7203, 7204, 7209, 13010, 13300 and 16000. In addition, this function is controlled by the station settings for "Upload Settings". Further information on this topic can be found in the documentation for the ADM Module!*
This API function is intended for checking the state of all single panels of a printed circuit board, whereby generally only the last work step obligated to supply confirmation is checked. If the check is to be done for all work steps obligated to supply confirmation (code no. 7203 must be set to "J"), an array of serial numbers is only returned if the state bookings of all work steps obligated to supply confirmation are completed and have the state "PASS". Otherwise, only the corresponding error code is returned. For a detailed analysis of the state bookings across all work steps of the respective single panels of a printed circuit board, it is recommended that the »checkSerialNumberState« API function be used for each single panel!

Function declaration (CORBA)

```
long checkSnrStateBySnrRef(
     in string stationNr,
     in long processLayer,
     in string serialNumber,
     out long numberOfRecords,
     out StateDataArray testDataArray,
     out string errorString)
raises (ServerException);

struct StateData
     {
        string serialNumber;
        string serialNumberPos;
        long loopCounter;
        long state;
     };
typedef sequence<StateData> StateDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
serialNumber	Serial no. of the panel

TAB. 3-58 *Parameter (in) »checkSnrStateBySnrRef«*

Parameter (out)

Parameter name	Explanation
numberOfRecords	Number of entries in the array returned
testDataArray	Array with the following variables:
serialNumber	Serial no. of the panel
serialNumberPos	Position of the single panel in the unit-array
loopCounter	Number of bookings against this serial no. and this work step (stations of the associated ERP group) ***Advice:*** *Only bookings of stations of the type "tester" are counted; bookings of repair and diagnosis stations are* ***not*** *counted!*
state	State of the serial no. (uploadState) [0; 1; 2]: **0** = Pass **1** = Fail **2** = Scrap
errorString	Output text according to output value (see table)

TAB. 3-59 *Parameter (out) »checkSnrStateBySnrRef«*

Output values

Value	Explanations (errorString)
Advice: *The value for the* errorString *is calculated from all single panels! Here, the worst single panel state determines the total panel state (worst case).*	
9	station valid but serialnumber not booked before
8	not valid for this station, is fail at one of the previous stations
7	not valid for this station, is scrap at one of the previous stations
6	State FAIL but MAX-Count reached (only active for Code no. 7204 TEST_RETRY_COUNTER_AKTIV)
5	State PASS but MAX-Count reached (only active for Code no. 7204 TEST_RETRY_COUNTER_AKTIV)
4	State SCRAP or Serialnumber not active
3	Snr not valid for this station, needs to be seen at one of the previous stations
2	Snr not valid for this station, needs to be seen at previous station
1	State "FAIL"
0	OK: State "PASS"
-1	Error: wrong productId ***Advice:*** *This value is also output if the work order of the serial number does not match the work order currently active at the station; in order for this to occur, however, station parameter code no. 13300 must be active!*
-2	Error: Snr not found
-3	Error: station not found
-4	Error: process step not found
-5	Error: serialNumber not unique
-9	Error: Max sequence reached ***Advice:*** *This value depends on the defaults for "max. recursion" in the upload settings and is only output if the station parameter code no. 7204 for the station is deactivated!*
-11	Error: Database Error

TAB. 3-60 *Output values for »checkSnrStateBySnrRef«*

Value	Explanations (errorString)
-17	Error: serialnumber is locked
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-436	Error: serialnumber is archived

TAB. 3-60 *Output values for »checkSnrStateBySnrRef«*

3.21 checkSerialNumberState

This function checks the state of a serial number. It checks whether the serial no.passed is permitted for the station passed and the associated work step. A plausibility check is carried out. It checks whether the work progress is OK and which state (PASS, FAIL, SCRAP) is associated with the serial no.

With activation of station parameter code no. 13300, a check is also performed to determine whether the work order to which the serial number is assigned matches the work order currently active on the station.

ADVICE

For the first station of a WO, the API function **`checkSerialNumberState`** *can only be used to query the "test retry counter". This API function is influenced by station parameter code nos.: 7203, 7204, 7209, 13010, 13300 and 16000. In addition, this function is controlled by the station settings for "Upload Settings". Further information on this topic can be found in the documentation for the ADM Module!*

Function declaration (CORBA)

```
long checkSerialNumberState(
     in string stationNr,
     in long processLayer,
     in string serialNumber,
     in string serialNumberPos,
     out long loopCounter,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
`stationNr`	Station no. of the work station in iTAC.MES.Suite
`processLayer`	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
`serialNumber`	Serial no. of the panel
`serialNumberPos`	Position of the single panel in the unit-array ***Advice:*** *If the value [-1] is passed, the position is not evaluated; in this case, each single panel must be specified by means of a unique* `serialNumber` *!*

TAB. 3-61 *Parameter (in) »checkSerialNumberState«*

Parameter (out)

Parameter name	Explanation
`loopCounter`	Number of bookings against this serial no. and this work step (stations of the associated ERP group) ***Special case:*** *If the function returns an output value <>* `0`*, the value [1] is always returned here!* ***Advice:*** *Only bookings of stations of the type "tester" are counted; bookings of repair and diagnosis stations are* ***not*** *counted!*
`errorString`	Output text according to output value (see table)

TAB. 3-62 *Parameter (out) »checkSerialNumberState«*

Output values

Value	Explanations (errorString)
9	station valid but serialnumber not booked before
8	not valid for this station, is fail at one of the previous stations
7	not valid for this station, is scrap at one of the previous stations
6	State FAIL but Max - Test Retry Counter reached (only active for TEST_RETRY_COUNTER_AKTIV)
5	State PASS but Max - Test Retry Counter reached (only active for TEST_RETRY_COUNTER_AKTIV)
4	State REJECT or serialno. not active
3	Not valid for this Station. Part needs to be seen at one of the previous Stations
2	Not valid for this Station. Part needs to be seen at the previous Station first
1	State "FAIL"
0	OK: State "PASS"
-1	Error: wrong productId ***Advice:*** *This value is also output if the work order of the serial number does not match the work order currently active at the station; in order for this to occur, however, station parameter code no. 13300 must be active!*
-2	Error: Snr not found
-3	Error: station not found
-4	Error: process step not found
-5	Error: serialNumber not unique
-9	Error: Max sequence reached ***Advice:*** *This value depends on the defaults for "max. recursion" in the upload settings and is only output if the station parameter code no. 7204 for the station is deactivated!*
-11	Error: Database Error
-17	Error: serialnumber is locked
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-436	Error: serialnumber is archived

TAB. 3-63 *Output values for »checkSerialNumberState«*

3.22 confirmAdvice

This function is used to collect the data necessary for confirming a customer-specific advice dialog. Here, the data for user authentication and advice confirmation are passed to the server.

ADVICE

Advice notices managed in iTAC.MES.Suite can be displayed in external applications as dialog boxes. The appearance of the advice notices is event-driven and can be configured for various objects. This function is always used in combination with the »getAdvice« function!

Function declaration (CORBA)

```
long confirmAdvice(
     in string userName,
     in string password,
     in long adviceId,
     in long confirmationStatus,
     in StringArray adviceEditText,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
userName	Login name of the user (user identification)
password	Password of the user
adviceId	Unique designator (ID) for the advice
confirmationstatus	Confirmation status [0; 1]: **0** = Advice not confirmed **1** = Advice confirmed
adviceEditText	Supplemental text for the advice (optional)

TAB. 3-64 *Parameter (in) »confirmAdvice«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-65 *Parameter (out) »confirmAdvice«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-51	Error: password is invalid
-63	Error: Invalid session
-75	Error: a required parameter is missing
-98	Error: unknown user
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-66 *Output values for »confirmAdvice«*

3.23 createAttribute

This function enables the creation of attributes for containers, serial numbers and work orders.

ADVICE

Alternatively, attributes can also be created in the TR Module under the "Unit information" function.

The assignment of new attributes to containers, serial numbers and work orders is performed with the »appendAttributesToMaterialBin«, »appendAttributeToSerialNumber« and »appendAttributeToWorkorder« functions, respectively!

Function declaration (CORBA)

```
long createAttribute(
     in string stationNo,
     in long attributeMode,
     in string attributeCode,
     in string attributeDesc,
     in string uniquenesType,
     out string errorString
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNo	Station no. of the work station in iTAC.MES.Suite
attributeMode	Specifies the attribute type [0; 1; 2] **0** = Serial number attribute **1** = Work order attribute **2** = Container attribute
attributeCode	Attribute code
attributeDesc	Attribute designator
uniquenesType	Specifies whether the attribute values for this attribute must be unique [0; 1]: **0** = Attribute values do **not** need to be unique **1** = Attribute values must be unique ***Advice:*** *The check for value uniqueness is currently only supported for serial number attributes; if **1** or **2** is selected for* `attributeMode`*, this parameter is not evaluated!*

TAB. 3-67 *Parameter (in) »createAttribute«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-68 *Parameter (out) »createAttribute«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-78	Error: no TR license for this station

TAB. 3-69 *Output values for »createAttribute«*

Value	Explanations (errorString)
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-400	Error: attribute already exists
-433	Error: attribute mode not exists

TAB. 3-69 *Output values for »createAttribute«*

3.24 createNewMaterialBin

This function creates a new material bin from the parameters passed. It checks whether the `MaterialBinNr` already exists. If it exists, is empty, and is not stored at a storage position, the "old" container is deactivated an a new one is created.

ADVICE

If detailed container information is to be passed (e.g. manufacturer specifications, receiving number, container state etc.), the »createNewMaterialBinExt« function can be used!

Function declaration (CORBA)

```
long createNewMaterialBin(
     in string stationNr,
     in string materialBinNr,
     in string productNr,
     in string chargeNr,
     in string dateCode,
     in double quantity,
     out string errorString
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
materialBinNr	Number of the material bin (container no.)
productNr	Part no. of the container material in iTAC.MES.Suite
chargeNr	Storage unit or supplier charge
dateCode	DateCode for the storage unit
quantity	Quantity of material container

TAB. 3-70 *Parameter (in) »createNewMaterialBin«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-71 *Parameter (out) »createNewMaterialBin«*

Output values

Value	Explanations (errorString)
>0	OK: [Identification number of the material bin (ContainerID)] *»Values > 0 mean: execution OK with specification of the new ContainerID«*
0	OK: [empty]
-1	Error: [empty]
-2	Error: StationNr not found
-3	Error: productNr not found
-5	Error: Container is not empty
-6	Error: Container has a binLocation
-7	Error: Error while deactivating Container
-47	Error: wrong character [?; %; _] in materialBinNr
-78	Error: no TR license for this station

TAB. 3-72 *Output values for »createNewMaterialBin«*

Value	Explanations (errorString)
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-72 *Output values for »createNewMaterialBin«*

3.25 createNewMaterialBinExt

This function creates a new material bin from the parameters passed. It checks whether the `MaterialBinNr` already exists. If it exists, is empty, and is not stored at a storage position, the "old" container is deactivated an a new one is created.

ADVICE

Unlike the »createNewMaterialBin« function, detailed container information can be passed here (e.g. manufacturer specifications, receiving number, container state etc.)!

Function declaration (CORBA)

```
long createNewMaterialBinExt(
     in string stationNr,
     in string materialBinNr,
     in string productNr,
     in string chargeNr,
     in string dateCode,
     in double quantity,
     in string supplierNo,
     in string supplierName,
     in long expirationDate,
     in char materialBinState,
     in string weNumber,
     in StringArray classification,
     out string errorString
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
materialBinNr	Number of the material bin (container no.)
productNr	Part no. of the container material in iTAC.MES.Suite
chargeNr	Storage unit or supplier charge
dateCode	DateCode for the storage unit
quantity	Quantity of material container
supplierNo	Supplier number
supplierName	Supplier name
expirationDate	Point in time from which on the container must no longer be used; the total time is specified as the number of seconds since 01-01-1970
materialBinState	Container state [B; C; E; F; L; Q; R; S]: **B** = LOCKED (Container locked) **C** = COMPLETE (container completed) **E** = FINISHED (Container finished) **F** = INIT (new container created) **L** = MRP LOCKED (Container not available for allocation) **Q** = QA LOCKED (Container with QA lock) **R** = READY (Container ready) **S** = IN PROCESS (Container in process)
weNumber	Receiving number
classification	Classification of the container

TAB. 3-73 *Parameter (in) »createNewMaterialBinExt«*

Parameter (out)

Parameter name	Explanation
`errorString`	Output text according to output value (see table)

TAB. 3-74 *Parameter (out) »createNewMaterialBinExt«*

Output values

Value	Explanations (errorString)
>0	OK: [Identification number of the material bin (ContainerID)] *»Values > 0 mean: execution OK with specification of the new ContainerID«*
0	OK: [empty]
-2	Error: StationNr not found
-3	Error: productNr not found
-5	Error: Container is not empty
-47	Error: wrong character [?; %; _] in materialBinNr
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-213	Error: Container state not valid
-301	Error: Creating an already existing container no. is only allowed if the old one is empty
-302	Error: Error while deactivating Container
-303	Error: Container has a binLocation

TAB. 3-75 *Output values for »createNewMaterialBinExt«*

3.26 createRecipe

This function creates a new recipe. If a recipe for the station and the passed part already exists, only a new recipe version is created based on the header structure defined in the existing recipe.

With this function it is possible to, among other things, directly assign the new recipe to the part; in this case the recipe can be used for all product versions of the part. If the assignment is to be more detailed, the recipe can also be assigned directly to a specific BOM version, a specific process version if applicable or a specific change index of a part. The possible combination of input parameters can be found in the following table:

Recipe for:	Part	BOM version	Process version	Index of changes
partNr	x	x	x	x
bomVersion	[-1]	x	x	[-1] (x)
processVersion	[-1]	[-1]	x	[-1] (X)
revisionIndex	[empty]	[empty]	[empty]	x

The "x" identifies input parameters which must absolutely be passed; if e.g. the recipe is to be assigned to a process version, the parameters `partNr`, `bomVersion` and `processVersion` must be filled with concrete product data.

Function declaration (CORBA)

```
long createRecipe(
      in string stationNr,
      in string partNr,
      in long bomVersion,
      in long processVersion,
      in string revisionIndex,
      inout long numberOfRecords,
      inout RecipeDataExtendedStructArray recipeData,
      in boolean activate,
      out string errorString)
raises (ServerException);

struct RecipeDataExtendedStruct
      {
         long recipeVersionId;
         double sequentialNumber;
         string name;
         string type;
         string remark;
         string lowerLimit;
         string upperLimit;
         string nominal;
         string tolerance;
         string unit;
         string measureType;
         string lowerActionLimit;
         string upperActionLimit;
         string lowerScrapLimit;
         string upperScrapLimit;
         long weighting;
         long distributionType;
         long confidenceInterval;
      };
typedef sequence<RecipeDataExtendedStruct> RecipeDataExtended-
StructArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite ***Special case:*** *When passing [0], the user registered at the station is passed, otherwise, the default API user is assigned!*
partNr	Part no. in iTAC.MES.Suite ***Advice:*** *This field must absolutely be filled (see assignment table above)!*
bomVersion	Bill-of-material version from iTAC.MES.Suite ***Special case:*** *[-1] specifies that this variable will not be analyzed!*
processVersion	Process version of the BOM from iTAC.MES.Suite ***Special case:*** *[-1] specifies that this variable will not be analyzed!*
revisionIndex	Index of changes of the BOM from an external ERP system (SAP) ***Advice:*** *This variable identifies a product version and may be used as an alternative to* **bomVersion** *and* **processVersion***!* ***Special case:*** *[empty] specifies that this variable will not be analyzed!*
activate	Automatically activates the recipe after it has been created [0; 1]: **1** = Recipe is activated (TRUE) **0** = Recipe is not activated (FALSE)

TAB. 3-76 *Parameter (in) »createRecipe«*

Parameter (inout)

Parameter name	Explanation
numberOfRecords	**in**: Default value for the number of entries in the array **out**: Number of entries in the array returned
recipeData	Array with the following variables:
recipeVersionId	Identification number of the recipe version ***Advice:*** *In this function, this field is not evaluated!*
sequentialNumber	Testing step no. in the recipe
name	Name of the recipe parameter
type	Specification of the parameter type [V; R; B; D]: **V** = Config **R** = Result **B** = Config-Result **D** = Dynamic
remark	Comment
lowerLimit	Lower limit for measurement step ***Advice:*** *Alternatively, the limit value can also be defined with the* nominal *and* tolerance *parameters!*
upperLimit	Upper limit for measurement step ***Advice:*** *Alternatively, the limit value can also be defined with the* nominal *and* tolerance *parameters!*

TAB. 3-77 *Parameter (inout) »createRecipe«*

Parameter name	Explanation
nominal	Nominal value ***Advice:*** *Alternatively, the limit value can also be defined with the* `lowerLimit` *and* `upperLimit` *parameters; the* `lowerLimit` *and* `upperLimit` *parameters override this setting!*
tolerance	Tolerance value for measurement step ***Advice:*** *Alternatively, the limit value can also be defined with the* `lowerLimit` *and* `upperLimit` *parameters; the* `lowerLimit` *and* `upperLimit` *parameters override this setting!*
unit	Unit for the value passed
measureType	Specification of the measurement value type [U; T; B; O; H; N; D]: **U** = unknown **T** = text **B** = binary **O** = octal **H** = hexadecimal **N** = decimal EN "." **D** = decimal DE ","
lowerActionLimit	Lower intervention limit
upperActionLimit	Upper intervention limit
lowerScrapLimit	Lower scrap limit
upperScrapLimit	Upper scrap limit
weighting	Weighting for the PAA [0 - 9]: ***Advice:*** *This field is a mandatory field!*
distributionType	Distribution function [0; 1]: **0** = normal distribution **1** = log. normal distribution ***Advice:*** *This field is a mandatory field!*
confidenceInterval	Confidence region [0; 1; 2; 3]: **0** = 3 Sigma **1** = 4 Sigma **2** = 6 Sigma **3** = 8 Sigma ***Advice:*** *This field is a mandatory field!*

TAB. 3-77 *Parameter (inout) »createRecipe«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-78 *Parameter (out) »createRecipe«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-3	Error: station not found

TAB. 3-79 *Output values for »createRecipe«*

Value	Explanations (errorString)
-62	Error: Product not found
-63	Error: Invalid session
-65	Error: Recipe not activate
-66	Error: Recipe not created
-67	Error: no valid bom version
-68	Error: no valid process version
-69	Error: Wrong process version
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-79 *Output values for »createRecipe«*

3.27 customFunction

This function enables the grouping together of various API functions into a "customer method"; in this way, complex, customer-specific API workflows can be represented and called centrally via one method.

ADVICE

Which API workflows can be meaningfully grouped into a method in this way depends on the individual project requirements. These methods must be explicitly developed for each customer-specific application type. Therefore, only the functional framework is described here!

Function declaration (CORBA)

```
long customFunction(
     in string methodName,
     in StringArray inArgs,
     out StringArray outArgs,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
methodName	Name of the customer-specific method
inArgs	Array with the variables that are required depending on the customer method and the API functions that are grouped together there

TAB. 3-80 *Parameter (in) »customFunction«*

Parameter (out)

Parameter name	Explanation
outArgs	Array with the variables that are output depending on the customer method and the API functions that are grouped together there
errorString	Output text according to output value (see table)

TAB. 3-81 *Parameter (out) »customFunction«*

ADVICE

Depending on API workflow and the API functions used there, additional output values – that are not listed here – may be output!

Output values

Value	Explanations (errorString)
0	OK: [empty]
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-82 *Output values for »customFunction«*

Value	Explanations (errorString)
-424	Error: class not found
-425	Error: function not found
-426	Error: illegal arguments for function

TAB. 3-82 *Output values for »customFunction«*

3.28 getAdvice

This function is used to query customer-specific advice data. On the basis of the entered reference data, the function returns an array of non-confirmed advice notices.

ADVICE

Advice notices managed in iTAC.MES.Suite can be displayed in external applications as dialog boxes. The appearance of the advice notices is event-driven and can be configured for various objects. This function is always used in combination with the »confirmAdvice« function!

Function declaration (CORBA)

```
long getAdvice(
     in string stationNr,
     in boolean ignoreStationNr,
     in string workOrder,
     in string partNr,
     in string serialNumber,
     in long workstepNo,
     in string erpGrpNr,
     in string userName,
     in boolean checkStationGroup,
     in boolean checkPartGroup,
     out AdviceArray advices,
     out string errorString)
raises (ServerException);

struct Advice
     {
        long id;
        string text;
        string name;
        long mdaRootKey;
        long adviceType;
     };
typedef sequence<Advice> AdviceArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
ignoreStationNr	Controls whether the station number is taken into account when ascertaining the advice data [0; 1] **1** = `stationNr` is not evaluated (TRUE) **0** = `stationNr` is evaluated (FALSE)
workOrder	Number of a production order; ***Advice:*** *The input is optional; if [-1] or [empty] is passed, the field is not evaluated!*
partNr	Part no. in iTAC.MES.Suite ***Advice:*** *The input is optional; if [-1] or [empty] is passed, the field is not evaluated!*
serialNumber	Serial no. of the panel ***Advice:*** *The input is optional; if [-1] or [empty] is passed, the field is not evaluated!*

TAB. 3-83 *Parameter (in) »getAdvice«*

Parameter name	Explanation
workstepNo	Number of the work step ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!*
erpGrpNr	Number of the ERP group ***Advice:*** *The input is optional; if [-1] or [empty] is passed, the field is not evaluated!*
userName	Login name of the user (user identification) ***Advice:*** *The input is optional; if [-1] or [empty] is passed, the field is not evaluated!*
checkStationGroup	Controls whether the machine group of the station is also taken into account when ascertaining the advice data (`stationNr`) [0; 1] **1** = Machine group is taken into account (TRUE) **0** = Machine group is not taken into account (FALSE)
checkPartGroup	Controls whether the part group of the part is also taken into account when ascertaining the advice data (`partNr`) [0; 1] **1** = Part group is taken into account (TRUE) **0** = Part group is not taken into account (FALSE)

TAB. 3-83 *Parameter (in) »getAdvice«*

Parameter (out)

Parameter name	Explanation
advices	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
`id`	Unique designator (ID) for the advice
`text`	Supplementary text for the advice; multiple lines must be separated using a semicolon
`name`	Advice name
`mdaRootKey`	Root directory to the referenced MDA items
`adviceType`	Designator for the advice type [0; 1; 2]: **0** = Advice is only displayed to each user once during the validity period **1** = Advice is displayed repeatedly independent of the validity period **2** = Advice is displayed repeatedly during the validity period
errorString	Output text according to output value (see table)

TAB. 3-84 *Parameter (out) »getAdvice«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-62	Error: Product not found
-63	Error: Invalid session
-75	Error: a required parameter is missing ***Advice:*** *This value is only returned if an empty string is passed for the* `stationNr` *parameter!*
-78	Error: no TR license for this station
-98	Error: unknown user
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-201	Error: erp-group not found

TAB. 3-85 *Output values for »getAdvice«*

3.29 getAttributesForMaterialBin

This function returns a specific attribute or all existing attributes of a container number.

ADVICE

In the context of container attributes, the »appendAttributesToMaterialBin«, »createAttribute«, »getMaterialBinsForAttribute« and »removeAttributeFromMaterialBin« functions must also be taken into account!

Function declaration (CORBA)

```
long getAttributesForMaterialBin(
     in string stationNr,
     in string materialBinNr,
     in string attributeCode,
     out MaterialBinAttributeInfoArray matBinAttributeInfos,
     out string errorString)
raises (ServerException);

struct MaterialBinAttributeInfo
     {
        long snrMatId;
        string materialBinNr;
        long attributeNumber;
        string attributeDesc;
        string attributeValue;
        string attributeCode;
     };
typedef sequence<MaterialBinAttributeInfo> MaterialBinAttrib-
uteInfoArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
materialBinNr	Number of the material bin (container no.)
attributeCode	Attribute code ***Advice:*** *The wildcard character "*****" can be used during input; pay attention to upper- and lower-case letters!* ***Special case:*** *If the value [-1] is passed, all attributes of the container are returned!*

TAB. 3-86 *Parameter (in) »getAttributesForMaterialBin«*

Parameter (out)

Parameter name	Explanation
matBinAttributeInfos	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
snrMatId	Internal identification number for the container number ***Advice:*** *This parameter is not output at the test client (API Module)!*
materialBinNr	Number of the material bin (container no.)
attributeNumber	Attribute number ***Advice:*** *This parameter is not output at the test client (API Module)!*
attributeDesc	Attribute designator
attributeValue	Value of the attribute
attributeCode	Attribute code
errorString	Output text according to output value (see table)

TAB. 3-87 *Parameter (out) »getAttributesForMaterialBin«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-76	Error: attribute code must be at least one character
-78	Error: no TR license for this station
-92	Error: attribute code does not exist
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-402	Error: no attributes found for material bin
-434	Error: material bin was not found

TAB. 3-88 *Output values for »getAttributesForMaterialBin«*

3.30 getAttributesForSerialNumber

This function returns a specific attribute or all existing attributes of a serial number.

ADVICE

In the context of serial number attributes, the »appendAttributeToSerialNumber«, »createAttribute«, »getSerialNumbersForAttribute« and »removeAttributeFromSerialNumber« functions must also be taken into account!

Function declaration (CORBA)

```
long getAttributesForSerialNumber(
     in string stationNr,
     in string serialNumber,
     in string attributeCode,
     out SnrAttributeInfoArray snrAttributeInfos,
     out string errorString)
raises (ServerException);

struct SnrAttributeInfo
     {
        long snrId;
        string serialNumber;
        long attributeNumber;
        string attributeValue;
        string attributeCode;
     };
typedef sequence<SnrAttributeInfo> SnrAttributeInfoArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
serialNumber	Serial no. of the panel
attributeCode	Attribute code ***Advice:*** *If the value [*] is passed, all attributes of the serial number are returned! The input of a specific attribute code or [*] without any additional information ("|A" or "|S") only returns attributes found in the current integration level.* ***Special cases "|A" or "|S":*** *If the character string* ***|A*** *is added to the attribute code or [*] during input (e.g. "1122|A" or "*|A"), the attributes of all superordinate and subordinate integration levels are also output. In this context, it is not important whether the serial number in the respective integration levels is identical or not. If the additional information* ***|S*** *is used, the search is carried out for all integration levels, however, only those attributes are output for which the serial numbers are identical.*

TAB. 3-89 *Parameter (in) »getAttributesForSerialNumber«*

Parameter (out)

Parameter name	Explanation
snrAttributeInfos	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
snrId	Internal identification number for the serial number ***Advice:*** *This parameter is not output at the test client (API Module)!*
serialNumber	Serial no. of the panel
attributeNumber	Attribute number ***Advice:*** *This parameter is not output at the test client (API Module)!*
attributeValue	Value of the attribute
attributeCode	Attribute code
errorString	Output text according to output value (see table)

TAB. 3-90 *Parameter (out) »getAttributesForSerialNumber«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-39	Error: serialnumber not found
-72	Error: attribute not found for serialnumber
-73	Error: No attributes found for serialnumber
-76	Error: attribute code must be at least one character
-78	Error: no TR license for this station
-92	Error: attribute code does not exist
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-91 *Output values for »getAttributesForSerialNumber«*

3.31 getAttributesForWorkorder

This function returns a specific attribute or all existing attributes of a work order. If the order number is not known, a serial number can be used instead to ascertain the work order.
If, in addition to the work order, a part number is passed – this part number must be contained in the order BOM – the work order attributes with a certain part reference can be specifically queried.

ADVICE

In the context of work order attributes, the »appendAttributeToWorkorder«, »createAttribute«, »getWorkordersForAttribute« and »removeAttributeFromWorkorder« functions must also be taken into account!

Function declaration (CORBA)

```
long getAttributesForWorkorder(
     in string stationNr,
     in string chargeExt,
     in string serialNumber,
     in string partNo,
     in string attributeCode,
     in string attributeType,
     out ChargeAttributeInfoArray chargeAttributeInfos,
     out string errorString)
raises (ServerException);

struct ChargeAttributeInfo
     {
        long long chargeNr;
        string chargeExt;
        long attributeNumber;
        string attributeValue;
        string attributeCode;
        string attributeValueDesc;
        string attributeType;
        string objectKey;
        string partNo;
     };
typedef sequence<ChargeAttributeInfo> ChargeAttributeInfoArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
ChargeExt	Production order number ***Advice:*** *If the value [-1] is passed, the parameter is not evaluated; in this case, it is absolutely necessary that a serial number be passed (see* `serialNumber`*)!*
serialNumber	Serial no. of the panel ***Advice:*** *If the value [-1] is passed, the parameter is not evaluated; in this case, it is absolutely necessary that an order number be passed (see* `chargeExt`*)!*
partNo	Part no. in iTAC.MES.Suite ***Advice:*** *This parameter is optional and enables the querying of work order attributes with a part reference; the part number must be contained in the order BOM. If no part reference is to be established, the value [-1] must be passed here!*

TAB. 3-92 *Parameter (in) »getAttributesForWorkorder«*

Parameter name	Explanation
attributeCode	Attribute code ***Advice:*** *The wildcard character "*****" can be used during input; pay attention to upper- and lower-case letters!* ***Special case:*** *If the value [-1] is passed, all attributes of the work order are returned; pay attention here to the association with the* `attributeType` *parameter!*
attributeType	Attribute type ***Advice:*** *The wildcard character "*****" can be used during input; pay attention to upper- and lower-case letters!* ***Special case:*** *If the value [-1] is passed, all attributes of the work order are returned independent of the order type; pay attention here to the association with the* `attributeCode` *parameter!*

TAB. 3-92 *Parameter (in) »getAttributesForWorkorder«*

Parameter (out)

Parameter name	Explanation
chargeAttributeInfos	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
chargeNr	Internal identification number for the work order ***Advice:*** *This parameter is not output at the test client (API Module)!*
chargeExt	Production order number
attributeNumber	Attribute number ***Advice:*** *This parameter is not output at the test client (API Module)!*
attributeValue	Value of the attribute
attributeCode	Attribute code
attributeValue-Desc	Description of the attribute value (optional) ***Advice:*** *A value is output here only if a description for the attribute value was passed for the work order attribute during creation!*
attributeType	Attribute type ***Advice:*** *A value is output here only if an attribute type value was defined for the work order attribute during creation!*
objectKey	Key term for the attribute ***Advice:*** *A value is output here only if a key term was defined for the work order attribute during creation!*
partNo	Part no. in iTAC.MES.Suite ***Advice:*** *As long as a part reference was defined for the work order attribute during creation, the corresponding part number of the component appears here; otherwise, a [-1] is returned here!*
errorString	Output text according to output value (see table)

TAB. 3-93 *Parameter (out) »getAttributesForWorkorder«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-76	Error: attribute code must be at least one character
-78	Error: no TR license for this station
-81	Error: workorder not found
-92	Error: attribute code does not exist
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-218	Error: attribute not found for workorder
-219	Error: no attributes found for workorder
-220	Error: Part not found
-222	Error: attribute type not exists

TAB. 3-94 *Output values for »getAttributesForWorkorder«*

3.32 getBomItems

This function returns an order-based BOM. The order number or, optionally, the serial number is used as input value; if both values are passed, the order number is used. The extent of the returned BOM information may be controlled via three control parameters.

- The input value `alternative` specifies whether the alternative BOM components are to be output, too.
- The input value `processBased` specifies whether the process-related BOM (with station no. passed) or the general BOM – valid for all processes – is output.
- The input value `bomType` specifies whether the construction BOM or the scheduling BOM is output:
  Construction BOM = all components are listed according to their mounting place
  Scheduling BOM = identical components are shown in a consolidated manner

Function declaration (CORBA)

```
long getBomItems(
      in string stationNr,
      in string workorder,
      in string serialNumber,
      in long alternative,
      in long processBased,
      in long bomType,
      out long numberOfRecords,
      out BomItemDataArray bomItemData,
      out string errorString)
raises (ServerException);

struct BomItemData
      {
         string partNo;
         string partDesc;
         string secPartNo;
         string compName;
         string process;
         long processLayer;
         string processInfo;
         string bomInfo;
         long quantity;
         string unit;
         long setup;
         long isProduct;
         long isAlternative;
      };
typedef sequence<BomItemData> BomItemDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
workorder	Number of a production order
serialNumber	Serial no. of the panel
alternative	Controls the output for the BOM components [0; 1]: **0** = Without alternatives **1** = With alternatives

TAB. 3-95 *Parameter (in) »getBomItems«*

Parameter name	Explanation
processBased	Controls the output with respect to the process [0; 1]: **0** = General BOM **1** = Process-related BOM
bomType	Controls the output for the type of BOM [0; 1]: **0** = Scheduling BOM **1** = Construction BOM

TAB. 3-95 *Parameter (in) »getBomItems«*

Parameter (out)

Parameter name	Explanation
numberOfRecords	Number of entries in the array returned
bomItemData	Array with the following variables:
partNo	Part no. in iTAC.MES.Suite (component)
partDesc	Part description in iTAC.MES.Suite (component)
secPartNr	Customer-specific part no. (customer mat. no.)
compName	The mounting place conforms to the iTAC.MES.Suite BOM
process	Machine group, on which the mounting was carried out
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
processInfo	Output of the stored process information
bomInfo	Output of the stored BOM information
quantity	Quantity of the part ***Advice:*** *Decimal values are rounded to the nearest whole number!*
unit	Unit for the value passed
setup	Designator for the setup requirement [0; 1]: **0** = No setup required **1** = Setup required
isProduct	Designator for raw material / product [0; 1]: **0** = Raw material (component) **1** = Product (unit)
isAlternative	Designator for regular or alternative component in the BOM [0; 1]: **0** = Regular component **1** = Alternative component
errorString	Output text according to output value (see table)

TAB. 3-96 *Parameter (out) »getBomItems«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: general Error
-2	Error: stationNr not found
-3	Error: workOrder not found

TAB. 3-97 *Output values for »getBomItems«*

Value	Explanations (errorString)
-4	Error: serialNumber not found
-5	Error: `alternative` not valid [0; 1]
-6	Error: `processBased` not valid [0; 1]
-7	Error: `bomType` not valid [0; 1]
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-97 *Output values for »getBomItems«*

3.33 getCalendarData

This function provides information about the current date and time (server). No input values need to be passed to the function.

Function declaration (CORBA)

```
long getCalendarData(
      out long year,
      out long month,
      out long weekOfYear,
      out long weekOfMonth,
      out long dayOfMonth,
      out long dayOfYear,
      out long dayOfWeek,
      out long dayOfWeekInMonth,
      out long amPm,
      out long hour,
      out long hourOfDay,
      out long minute,
      out long second,
      out long zoneOffset,
      out long dstOffset,
      out string errorString)
raises (ServerException);
```

Parameter (out)

Parameter name	Explanation
year	Current year (e.g. 2003)
month	Current month [1 - 12]
weekOfYear	Current week of the year [1 - 52]
weekOfMonth	Current week of the month
dayOfMonth	Current day of the month [1 - 31]
dayOfYear	Current day of the year [1 - 365]
dayOfWeek	Current day of the week [1 - 7] (1 = Sunday; 2 = Monday; etc.)
dayOfWeekInMonth	Display of a continuous integer number of a seven-day period of the month, beginning with the first day of the month [1 - 5] (1st to 7th = 1; 8th to 14th = 2; etc.)
amPm	[0; 1] 0 = am; 1 = pm
hour	Current hour [0 - 23]
hourOfDay	Current hour of the day (24 hrs.) [0 - 23]
minute	Current minute [0 - 59]
second	Current second [0 - 59]
zoneOffset	Deviations from the GMT time zone (in milliseconds) [1 - 7]
dstOffset	Deviations when switching to daylight savings time (in milli-seconds)
errorString	Output text according to output value (see table)

TAB. 3-98 *Parameter (out) »getCalendarData«*

Output values

Value	Explanations (errorString)
0	OK: [empty]

TAB. 3-99 *Output values for »getCalendarData«*

3.34 getCompleteMaterialSetup

This function ascertains the complete setup for a specific date for a station and PCB orientation during the work step.

ADVICE

This API function is influenced by station parameter code no.: 8310. Further information on this topic can be found in the documentation for the ADM Module!

Function declaration (CORBA)

```
long getCompleteMaterialSetup(
     in string stationNr,
     in long processLayer,
     in long bookDate,
     out string workOrder,
     out MaterialSetupDataArray materialSetupData,
     out CompPositionArray compPositionArray,
     out string errorString)
raises (ServerException);

struct MaterialSetupData
     {
        string position;
        string materialBinNr;
        string stationNr;
        string productNr;
        string supplierCode;
        string supplierName;
        string workOrder;
        string placementName;
        string partNr;
        long compReference;
        long setupDate;
        long endDate;
        long returnCode;
        string returnComment;
        string dateCode;
        boolean removeSetup;
     };
typedef sequence<MaterialSetupData> MaterialSetupDataArray;

struct CompPosition
     {
        long compReference;
        string compName;
        long panelPosition;
     };
typedef sequence<CompPosition> CompPositionArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite

TAB. 3-100 *Parameter (in) »getCompleteMaterialSetup«*

Parameter name	Explanation
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent ***Special case:*** *Depending on the setting of configuration parameter code no. 8310, the work step number (AVO) used in an ERP system can be passed directly instead of the PCB orientation!*
bookDate	Point in time for which the setup is to be ascertained (typically, this is the point in time of a state booking), the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20* ***Special case:*** *[-1] passes the current date and the current time!*

TAB. 3-100 *Parameter (in) »getCompleteMaterialSetup«*

Parameter (out)

Parameter name	Explanation
workOrder	Number of a production order ***Advice:*** *This work order was active on the passed station at the* `bookDate` *point in time!*
materialSetupData	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **`Size`***!*
`position`	Setup location (e.g. track) of the material that has been set up
`materialBinNr`	Number of the material bin (container no.)
`stationNr`	Station no. of the work station in iTAC.MES.Suite
`productNr`	Part no. of the product in iTAC.MES.Suite
`supplierCode`	Supplier number
`supplierName`	Supplier name
`workOrder`	Production order number ***Advice:*** *This work order was active on the passed station at the* `bookDate` *point in time!*
`placementName`	Name of the mounting program
`partNr`	Part no. of the container material in iTAC.MES.Suite
`compReference`	Reference value to array `compPositionArray`
`setupDate`	Start date for the setup (start of production); the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*

TAB. 3-101 *Parameter (out) »getCompleteMaterialSetup«*

Parameter name	Explanation
endDate	Point in time at which the container was stripped down; the total time is specified as the number of seconds since 01-01-1970
returnCode	"errorString" for the array returned **2**; **-1**; **-2**; **-7**; **-10**; **-11**; **-101** ***Advice:*** *See parameter* `returnComments`*!*
returnComment	Explanation "errorString": **2** = KompNames not set **-1** = StationId [...] not found **-2** = ChargeExt [...] not found **-7** = LabelId [...] not found" **-10** = Invalid Daterange **-11** = Invalid Position **-101** = ServerExceptionMessage
dateCode	DateCode for the storage unit
removeSetup	Identifies whether the container was stripped down [0; 1]: **1** = Container was already stripped down (TRUE) **0** = Container is still rigged (FALSE)
compPositionArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
compReference	Reference value to array `materialSetupData`
compName	The mounting place conforms to the iTAC.MES.Suite BOM
panelPosition	Panel position
errorString	Output text according to output value (see table)

TAB. 3-101 *Parameter (out) »getCompleteMaterialSetup«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-102 *Output values for »getCompleteMaterialSetup«*

3.35 getCurrentCalendar

This function computes the current calendar data and the shift that belongs to the current station.

Function declaration (CORBA)

```
long getCurrentCalendar(
     in string stationNr,
     out long shift,
     out CalendarData calendarData,
     out string errorString)
raises (ServerException);

struct CalendarData
     {
        string localizedDate;
        long date;
        long calendar_week;
        long day_of_week;
        long year;
        long month;
        long day;
        long hour;
        long minute;
        long second;
     };
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite

TAB. 3-103 *Parameter (in) »getCurrentCalendar«*

Parameter (out)

Parameter name	Explanation
shift	Value for shift type [0; 1; 2; 3]: **0** = Shift undefined **1** = Morning shift **2** = Late shift **3** = Night shift
calendarData	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
localizedDate	Date string, formatted according to the locale ***Advice:*** *For C-implementations, 126 bytes of dynamic memory must be requested for this variable!*
date	Current date; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
calendar_week	Current number of week in the year. [1 - 52]
day_of_week	Current day of the week [1 - 7] (1 = Sunday; 2 = Monday; etc.)

TAB. 3-104 *Parameter (out) »getCurrentCalendar«*

Parameter name	Explanation
year	Current year (e.g. 2003)
month	Current month [1 - 12]
day	Current day [1 - 31]
hour	Current hour [0 - 23]
minute	Current minute [0 - 59]
second	Current second [0 - 59]
errorString	Output text according to output value (see table)

TAB. 3-104 *Parameter (out) »getCurrentCalendar«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: stationNr not found
-2	Error: shift not found
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-105 *Output values for »getCurrentCalendar«*

3.36 getCurrentMaterialSetup

This function checks whether there is an active setup at the station passed. Depending on the parameter `completeSetupData` [0; 1], the function can return the materials that are fully set up.

ADVICE

The `getCurrentMaterialSetup` *API function is influenced by the "CHECK_EXPIRATION" station parameter (code no.: 7134), which activates and deactivates monitoring of the expiration date at the manual setup station. Further information on this topic can be found in the documentation for the ADM Module!*

Function declaration (CORBA)

```
long getCurrentMaterialSetup(
     in string stationNr,
     in long completeSetupData,
     out string setupWorkOrder,
     out long setupActive,
     out long numberOfRecords,
     out SetupDataArray setupData,
     out string errorString)
raises (ServerException);

struct SetupData
     {
        string workorder;
        string productNo;
        string productDesc;
        string partOrder;
        string partNo;
        string partDesc;
        string partName;
        string partDateCode;
        string customerNo;
        string customerName;
        string position;
        double quantity;
        long startDate;
        long endDate;
        long creationDate;
     };
typedef sequence<SetupData> SetupDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
completeSetupData	Extent of the setup data [0; 1]: **0** = Only the header data of the active setup are transferred **1** = All rigged material of the active setup is transferred ***Advice:*** *Only if* ***0*** *is passed for this parameter and if the "CHECK_EXPIRATION" config parameter is set are the output values (*`errorstring`*)* ***1****,* ***2*** *or* ***3*** *returned!*

TAB. 3-106 *Parameter (in) »getCurrentMaterialSetup«*

Parameter (out)

Parameter name	Explanation
setupWorkOrder	Order no. for which the station is currently set up
setupActive	Setup exists [0; 1]: **0** = No **1** = Yes
numberOfRecords	Number of entries in the array returned
setupData	Array with the following variables:
workorder	Number of a production order
productNo	Part no. of the product in iTAC.MES.Suite
productDesc	Part desc. of the product in iTAC.MES.Suite
partOrder	Storage unit or supplier charge
partNo	Part no. in iTAC.MES.Suite
partDesc	Part description in iTAC.MES.Suite
partName	Part label in iTAC.MES.Suite for individual component
partDateCode	DateCode for the article
customerNo	Specific part no. in iTAC.MES.Suite
customerName	Customer name in iTAC.MES.Suite
position	Setup location (e.g. track) of the material that has been set up
quantity	Quantity of material container
startDate	Start date for the setup (start of production); the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
endDate	Last booking for the setup; the total time is specified as the number of seconds since 01-01-1970
creationDate	Date of creation for the setup; the total time is specified as the number of seconds since 01-01-1970, 0:00:00
errorString	Output text according to output value (see table)

TAB. 3-107 *Parameter (out) »getCurrentMaterialSetup«*

Output values

Value	Explanations (errorString)
Advice: *Only in combination with activated "CHECK_EXPIRATION" and* **completeSetupData** = ***0*** *config parameters are the values* **1**; **2** *or* **3** *returned!*	
3	Warning: One or more container(s) are empty and expired.
2	Warning: One or more container(s) are expired
1	Warning: One or more container(s) are empty.
0	OK: [empty]
-1	Error: stationNr not found
-2	Error: completeSetupData not valid [0; 1]
-78	Error: no TR license for this station

TAB. 3-108 *Output values for »getCurrentMaterialSetup«*

Value	Explanations (errorString)
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-108 *Output values for »getCurrentMaterialSetup«*

3.37 getFailureCauseForStation

This function outputs all failure causes stored on the passed station.

Function declaration (CORBA)

```
long getFailureCauseForStation(
     in string stationNr,
     out FailureCauseDataArray failureCauseData,
     out string errorString)
raises (ServerException);

struct FailureCauseData
     {
        string failureGroupCode;
        string failureGroupBez;
        string failureGroupInfo;
        string failureCauseCode;
        string failureCauseBez;
        string failureCauseInfo;
        long errorcode;
        string errortxt;
     };
typedef sequence<FailureCauseData> FailureCauseDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite

TAB. 3-109 *Parameter (in) »getFailureCauseForStation«*

Parameter (out)

Parameter name	Explanation
failureCauseData	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
failureGroupCode	[empty]: Unique designator for the failure cause group ***Advice:*** *Function is not currently supported; an empty string is returned!*
failureGroupBez	[empty]: Description for the failure cause group ***Advice:*** *Function is not currently supported; an empty string is returned!*
failureGroupInfo	[empty]: Information text for the failure cause group ***Advice:*** *Function is not currently supported; an empty string is returned!*
failureCauseCode	Unique designator for the failure cause
failureCauseBez	Description of the failure cause
failureCauseInfo	[empty]: Information text for the failure cause ***Advice:*** *Function is not currently supported; an empty string is returned!*
errorString	Output text according to output value (see table)

TAB. 3-110 *Parameter (out) »getFailureCauseForStation«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-111 *Output values for »getFailureCauseForStation«*

3.38 getFailureDataForSerialNumber

This function outputs the booked attributive failure data for the passed serial number. Here, it is possible to select for the serial number whether the failure data for a specific work step, for all work steps of the current integration level or for all work steps of the current and the parent integration levels are to be output (`allProductEntries`).

Function declaration (CORBA)

```
long getFailureDataForSerialNumber(
     in string stationNr,
     in long processLayer,
     in string serialNumber,
     in long serialNumberPos,
     in long allProductEntries,
     out FailureDataArray failureData,
     out string errorString)
raises (ServerException);

struct FailureData
     {
        string serialNumber;
        long serialNumberPos;
        string workorder;
        long workstepNo;
        string stationNr;
        long processLayer;
        long seqNumber;
        string compName;
        string compPartNumber;
        string failureType;
        string failureCause;
        long failureBookDate;
        string repair;
        double cycleTime;
        string infoText;
     };
typedef sequence<FailureData> FailureDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent ***Advice:*** *The layer is only evaluated, if the variable* `allProductEntries` *has the value* ***0****!*
serialNumber	Serial no. of the panel ***Advice:*** *All relevant order data are ascertained on the basis of the serial number!*
serialNumberPos	Position of the single panel in the unit-array ***Advice:*** *If the value [-1] is passed, the position is not evaluated; in this case, each single panel must be specified by means of a unique* `serialNumber` *!*

TAB. 3-112 *Parameter (in) »getFailureDataForSerialNumber«*

Parameter name	Explanation
allProductEntries	Selection of the failure data to be output [0; 1; 2]: **0** = Failure data of a work step; the work step is ascertained on the basis of `stationNr` and `processLayer` **1** = Failure data of the current integration level (work order) **2** = Failure data of the current and of all parent integration levels in which the `serialNumber` was installed

TAB. 3-112 *Parameter (in) »getFailureDataForSerialNumber«*

Parameter (out)

Parameter name	Explanation
failureData	Array with the following variables: ***Advice:** The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension **Size**!*
`serialNumber`	Serial no. of the panel
`serialNumberPos`	Position of the single panel in the unit-array
`workorder`	Number of a production order
`workstepNo`	Number of the work step
`stationNr`	Station no. of the work station in iTAC.MES.Suite
`processLayer`	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
`seqNumber`	Display of the test run
`compName`	Fault location (mounting place) is consistent with the iTAC.MES.Suite BOM
`compPartNumber`	Part number of the faulty component is consistent with the iTAC.MES.Suite BOM
`failureType`	Failure type ***Advice:** The failure types must be created in advance in iTAC.MES.Suite and must be assigned to the station (see ADM Module)!*
`failureCause`	Failure cause; reason that led to the failure ***Advice:** The failure causes must be created in advance in iTAC.MES.Suite and must be assigned to the station (see ADM Module)!*
`failureBookDate`	Date of the failure booking; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:** 1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
`repair`	Repair state [0; 1]: **0** = Not repaired **1** = Repaired

TAB. 3-113 *Parameter (out) »getFailureDataForSerialNumber«*

Parameter name	Explanation
`cycleTime`	Duration of the check in seconds
`infoText`	Output of an information text
`errorString`	Output text according to output value (see table)

TAB. 3-113 *Parameter (out) »getFailureDataForSerialNumber«*

Output values

Value	Explanations (errorString)
`0`	OK: [empty]
`-3`	Error: station not found
`-39`	Error: serialnumber not found
`-50`	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
`-78`	Error: no TR license for this station
`-99`	Error: Server error
`-100`	Error: Corba no Service
`-101`	Error: Corba Exception
`-304`	Error: Invalid value for `processLayer` [0,1,2]
`-314`	Error: Invalid value for `allProductEntries`
`-436`	Error: serialnumber is archived

TAB. 3-114 *Output values for »getFailureDataForSerialNumber«*

3.39 getFailureSlipDataForSerialNumber

This function outputs the booked fault note data for the passed serial number. Here, it is possible to select for the serial number whether the data for a specific work step or for all work steps of the current integration level are to be output (`allProductEntries`). In addition, it is possible to output the fault note data for all test runs of a work step or only for the last test run (`onlyLastEntry`).

Function declaration (CORBA)

```
long getFailureSlipDataForSerialNumber(
     in string stationNr,
     in long processLayer,
     in string serialNumber,
     in long serialNumberPos,
     in long allProductEntries,
     in long onlyLastEntry
     out FailureSlipDataArray failureSlipData,
     out string errorString)
raises (ServerException);

struct FailureSlipData
     {
        string workorder;
        long workstepNo;
        string stationNr;
        long processLayer;
        long seqNumber;
        string infoText;
        long failureBookDate;
     };
typedef sequence<FailureSlipData> FailureSlipDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent ***Advice:*** *The layer is only evaluated, if the variable* `allProductEntries` *has the value* ***0****!*
serialNumber	Serial no. of the panel ***Advice:*** *All relevant order data are ascertained on the basis of the serial number!*
serialNumberPos	Position of the single panel in the unit-array ***Advice:*** *If the value [-1] is passed, the position is not evaluated; in this case, each single panel must be specified by means of a unique* `serialNumber` *!*
allProductEntries	Selection of the failure data to be output [0; 1]: **0** = Failure data of a work step; the work step is ascertained on the basis of `stationNr` and `processLayer` **1** = Failure data of the current integration level (work order)
onlyLastEntry	Selection of the test runs to be output [0; 1]: **0** = Fault note for all test runs **1** = Fault note of the last test run

TAB. 3-115 *Parameter (in) »getFailureSlipDataForSerialNumber«*

Parameter (out)

Parameter name	Explanation
failureSlipData	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
workorder	Number of a production order
workstepNo	Number of the work step
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
seqNumber	Display of the test run
infoText	Content of the fault note
failureBookDate	Date of the failure booking; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
errorString	Output text according to output value (see table)

TAB. 3-116 *Parameter (out) »getFailureSlipDataForSerialNumber«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-39	Error: serialnumber not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-304	Error: Invalid value for processLayer [0,1,2]
-314	Error: Invalid value for allProductEntries
-315	Error: Invalid value for onlyLastEntry [0,1]
-436	Error: serialnumber is archived

TAB. 3-117 *Output values for »getFailureSlipDataForSerialNumber«*

3.40 getFailureTypForStation

This function outputs all failure types stored on the passed station.

Function declaration (CORBA)

```
long getFailureTypForStation(
     in string stationNr,
     out FailureTypDataArray failureTypData,
     out string errorString)
raises (ServerException);

struct FailureTypData
     {
        string failureGroupCode;
        string failureGroupBez;
        string failureGroupInfo;
        string failureTypeCode;
        string failureTypeBez;
        string failureTypeInfo;
        long failureTypePseudo;
        char failureTypeCategory;
        long errorcode;
        string errortxt;
     };
typedef sequence<FailureTypData> FailureTypDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite

TAB. 3-118 *Parameter (in) »getFailureTypForStation«*

Parameter (out)

Parameter name	Explanation
failureTypData	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
failureGroupCode	Unique designator for the failure type group
failureGroupBez	Description for the failure type group
failureGroupInfo	Information text for the failure cause group ***Advice:*** *Function is not currently supported; an empty string is returned!*
failureTypeCode	Unique designator for the failure cause
failureTypeBez	Description of the failure cause
failureTypeInfo	Information text for the failure type
failureTypePseudo	Identifies the failure type as "pseudo failure" [0; 1]: **1** = Pseudo failure **0** = Not a pseudo failure
failureTypeCategory	Identifies the failure type category [C; S]: **C** = Component failure **S** = Soldering point failure

TAB. 3-119 *Parameter (out) »getFailureTypForStation«*

Parameter name	Explanation
errorcode	- currently has no use -
errortxt	- currently has no use -
errorString	Output text according to output value (see table)

TAB. 3-119 *Parameter (out) »getFailureTypForStation«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-120 *Output values for »getFailureTypForStation«*

3.41 getKapSetupDataForMaterial

This function calculates where and when a container was set up and returns respective setup information.

Function declaration (CORBA)

```
long getKapSetupDataForMaterial(
     in string materialBinNr,
     out long numberOfRecords,
     out SetupDataInfArray setupData,
     out string errorString)
raises (ServerException);

struct SetupDataInf
     {
        string stationNr;
        string position;
        long startDate;
        long endDate;
        string materialBinNr;
        string partOrder;
        string partNo;
        string partDateCode;
     };
typedef sequence<SetupDataInf> SetupDataInfArray;
```

Parameter (in)

Parameter name	Explanation
materialBinNr	Number of the material bin (container no.)

TAB. 3-121 *Parameter (in) »getKapSetupDataForMaterial«*

Parameter (out)

Parameter name	Explanation
numberOfRecords	Number of entries in the array returned
setupData	Array with the following variables:
stationNr	Station no. of the work station in iTAC.MES.Suite
position	Setup location (e.g. track) of the material that has been set up
startDate	Start date of the setup; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
endDate	Last booking for the setup; the total time is specified as the number of seconds since 01-01-1970
materialBinNr	Number of the material bin (container no.)
partOrder	Storage unit or supplier charge
partNo	Part no. of the container material in iTAC.MES.Suite
partDateCode	DateCode for the article
errorString	Output text according to output value (see table)

TAB. 3-122 *Parameter (out) »getKapSetupDataForMaterial«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: materialBinNr not found
-10	Error: general Error
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-123 *Output values for »getKapSetupDataForMaterial«*

3.42 getMaterialBinData

This function uses specific search criteria to ascertain corresponding container data. Available search criteria are part number, part group, container state, change date and the container number itself. All search criteria are "AND" coupled such that only the container data that meet all used filter conditions are returned.

Function declaration (CORBA)

```
long getMaterialBinData(
     in string stationNr,
     in string materialBinNr,
     in string materialBinState,
     in string partNr,
     in string partGrp,
     in long changeDate,
     out MaterialBinDataExtArray materialBinDataExt,
     out string errorString)
raises (ServerException);

struct MaterialBinDataExt
     {
     string materialBinNr;
     string partNr;
     double totalQuantity;
     double quantity;
     string supplierNo;
     string supplierName;
     string chargeNr;
     string dateCode;
     long expirationDate;
     string materialBinState;
     string binLocationNr;
     string binLocationDescr;
     double cost;
     long costBase;
     string weNr;
     long createDate;
     long stamp;
     long changeDate;
     string classification1;
     string classification2;
     string classification3;
     string classification4;
     string classification5;
     };
typedef sequence<MaterialBinDataExt> MaterialBinDataExtArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
materialBinNr	Number of the material bin (container no.) ***Advice:*** *The input is optional; if [-1] or [empty] is passed, the field is not evaluated!*

TAB. 3-124 *Parameter (in) »getMaterialBinData«*

Parameter name	Explanation
materialBinState	Container state [B; C; E; F; L; Q; R; S]: **B** = LOCKED (Container locked) **C** = COMPLETE (container completed) **E** = FINISHED (Container finished) **F** = INIT (new container created) **L** = MRP LOCKED (Container not available for allocation) **Q** = QA LOCKED (Container with QA lock) **R** = READY (Container ready) **S** = IN PROCESS (Container in process) ***Advice:*** *The input is optional; if [-1] or [empty] is passed, there is no restriction. If only containers with specific states are to be output, individual values can be separated with ";"!*
partNr	Part no. of the container material in iTAC.MES.Suite ***Advice:*** *The input is optional; if [-1] or [empty] is passed, the field is not evaluated!*
partGrp	Part group of the container material in iTAC.MES.Suite ***Advice:*** *The input is optional; if [-1] or [empty] is passed, the field is not evaluated!*
changeDate	Change date: All containers are ascertained which were created or for which the container state (`state`) or original container quantity (`totalQuantity`) was changed since this point in time. ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!*

TAB. 3-124 *Parameter (in) »getMaterialBinData«*

Parameter (out)

Parameter name	Explanation
materialBinDataExt	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **`Size`***!*
`materialBinNr`	Number of the material bin (container no.)
`partNr`	Part no. of the container material in iTAC.MES.Suite
`totalQuantity`	Original storage unit quantity
`quantity`	Quantity of material container
`supplierNo`	Supplier number
`supplierName`	Supplier name
`chargeNr`	Storage unit or supplier charge
`dateCode`	DateCode for the storage unit
`expirationDate`	Point in time from which on the container must no longer be used; the total time is specified as the number of seconds since 01-01-1970

TAB. 3-125 *Parameter (out) »getMaterialBinData«*

Parameter name	Explanation
materialBinState	Container state [B; C; E; F; L; Q; R; S]: **B** = LOCKED (Container locked) **C** = COMPLETE (container completed) **E** = FINISHED (Container finished) **F** = INIT (new container created) **L** = MRP LOCKED (Container not available for allocation) **Q** = QA LOCKED (Container with QA lock) **R** = READY (Container ready) **S** = IN PROCESS (Container in process)
binLocationNr	Storage location no. in iTAC.MES.Suite
binLocationDescr	Storage location description in iTAC.MES.Suite
cost	Average purchase price [€] of the material (see costBase)
costBase	Material quantity on which the purchase price is based (see cost)
weNr	Receiving number
createDate	Date of creation for the container; the total time is specified as the number of seconds since 01-01-1970
stamp	Date of the last quantity change on the container; the total time is specified as the number of seconds since 01-01-1970
changeDate	Date of the last state change for the container; the total time is specified as the number of seconds since 01-01-1970
classification1 classification2 classification3 classification4 classification5	Classification of the container (classification1) ***Advice:*** *The* classification [2,3,4,5] *parameters are intended for future program extensions and are not currently evaluated!*
errorString	Output text according to output value (see table)

TAB. 3-125 *Parameter (out) »getMaterialBinData«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-75	Error: a required parameter is missing ***Advice:*** *This value is only returned if an empty string is passed for the* stationNr *parameter!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-126 *Output values for »getMaterialBinData«*

3.43 getMaterialBinDataByProductNo

This function retrieves all associated containers based on a part. For this purpose, there are two modes:

1. `includeEmptyBin` = **0**; only those containers are listed that are currently registered in the system as having a part stock > 0
2. `includeEmptyBin` = **1**; in addition, the "old" containers that are registered in the system with stock "0" are also listed (history)

Function declaration (CORBA)

```
long getMaterialBinDataByProductNo(
     in string stationNr,
     in string productNo,
     in long includeEmptyBin,
     out long numberOfRecords,
     out MaterialBinDataArray materialBinDataArray,
     out string errorString)
raises (ServerException);

struct MaterialBinData
     {
        string materialBinNr;
        string productNr;
        double quantity;
        string chargeNr;
        string binLocationNr;
        string binLocationDescr;
        string state;
        long expirationDate;
        string dateCode;
        double totalQuantity;
        double cost;
        long costBase;
     };
typedef sequence<MaterialBinData> MaterialBinDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
productNo	Part no. of the product in iTAC.MES.Suite
includeEmptyBin	Selection criterion for the containers [0; 1]: **0** = Current containers (with stock) **1** = Current and old containers

TAB. 3-127 *Parameter (in) »getMaterialBinDataByProductNo«*

Parameter (out)

Parameter name	Explanation
numberOfRecords	Number of entries in the array returned
materialBinDataArray	Array with the following variables:
materialBinNr	Number of the material bin (container no.)
productNr	Part no. of the container material in iTAC.MES.Suite
quantity	Quantity of material container
chargeNr	Storage unit or supplier charge
binLocationNr	Storage location no. in iTAC.MES.Suite

TAB. 3-128 *Parameter (out) »getMaterialBinDataByProductNo«*

Parameter name	Explanation
binLocationDescr	Storage location description in iTAC.MES.Suite
state	Container state [B; C; E; F; L; Q; R; S]: **B** = LOCKED (Container locked) **C** = COMPLETE (container completed) **E** = FINISHED (Container finished) **F** = INIT (new container created) **L** = MRP LOCKED (Container not available for allocation) **Q** = QA LOCKED (Container with QA lock) **R** = READY (Container ready) **S** = IN PROCESS (Container in process)
expirationDate	Point in time from which on the container must no longer be used; the total time is specified as the number of seconds since 01-01-1970
dateCode	DateCode for the storage unit
totalQuantity	Original storage unit quantity
cost	Average purchase price [€] of the material (see costBase)
costBase	Material quantity on which the purchase price is based (see cost)
errorString	Output text according to output value (see table)

TAB. 3-128 *Parameter (out) »getMaterialBinDataByProductNo«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: stationNr not found
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-129 *Output values for »getMaterialBinDataByProductNo«*

3.44 getMaterialBinInfo

This function provides information about a material container.

Function declaration (CORBA)

```
long getMaterialBinInfo(
     in string stationNr,
     in string materialBinNr,
     out MaterialBinData materialBinData,
     out string errorString)
raises (ServerException);

struct MaterialBinData
     {
        string materialBinNr;
        string productNr;
        double quantity;
        string chargeNr;
        string binLocationNr;
        string binLocationDescr;
        string state;
        long expirationDate;
        string dateCode;
        double totalQuantity;
        double cost;
        long costBase;
     };
typedef sequence<MaterialBinData> MaterialBinDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
materialBinNr	Number of the material bin (container no.)

TAB. 3-130 *Parameter (in) »getMaterialBinInfo«*

Parameter (out)

Parameter name	Explanation
materialBinData	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
materialBinNr	Number of the material bin (container no.)
productNr	Part no. of the container material in iTAC.MES.Suite
quantity	Quantity of material container
chargeNr	Storage unit or supplier charge
binLocationNr	Storage location no. in iTAC.MES.Suite
binLocationDescr	Storage location description in iTAC.MES.Suite

TAB. 3-131 *Parameter (out) »getMaterialBinInfo«*

Parameter name	Explanation
`state`	Container state [B; C; E; F; L; Q; R; S]: **B** = LOCKED (Container locked) **C** = COMPLETE (container completed) **E** = FINISHED (Container finished) **F** = INIT (new container created) **L** = MRP LOCKED (Container not available for allocation) **Q** = QA LOCKED (Container with QA lock) **R** = READY (Container ready) **S** = IN PROCESS (Container in process)
`expirationDate`	Point in time from which on the container must no longer be used; the total time is specified as the number of seconds since 01-01-1970
`dateCode`	DateCode for the storage unit
`totalQuantity`	Original storage unit quantity
`cost`	Average purchase price [€] of the material (see `costBase`)
`costBase`	Material quantity on which the purchase price is based (see `cost`)
`errorString`	Output text according to output value (see table)

TAB. 3-131 *Parameter (out) »getMaterialBinInfo«*

Output values

Value	Explanations (errorString)
`0`	OK: [empty]
`-1`	Error: general Error
`-2`	Error: stationNr not found
`-3`	Error: materialBinNr not found
`-78`	Error: no TR license for this station
`-99`	Error: Server error
`-100`	Error: Corba no Service
`-101`	Error: Corba Exception

TAB. 3-132 *Output values for »getMaterialBinInfo«*

3.45 getMaterialBinsForAttribute

This function returns all container numbers for a specific attribute.

ADVICE

In the context of container attributes, the »appendAttributesToMaterialBin«, »createAttribute«, »getAttributesForMaterialBin« and »removeAttributeFromMaterialBin« functions must also be taken into account!

Function declaration (CORBA)

```
long getMaterialBinsForAttribute(
     in string stationNr,
     in string attributeCode,
     in string attributeValue,
     out MaterialBinAttributeInfoArray materialBinArray,
     out string errorString)
raises (ServerException);

struct MaterialBinAttributeInfo
     {
        long snrMatId;
        string materialBinNr;
        long attributeNumber;
        string attributeDesc;
        string attributeValue;
        string attributeCode;
        };
typedef sequence<MaterialBinAttributeInfo> MaterialBinAttrib-
uteInfoArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
attributeCode	Attribute code
attributeValue	Value of the attribute

TAB. 3-133 *Parameter (in) »getMaterialBinsForAttribute«*

Parameter (out)

Parameter name	Explanation
materialBinArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
snrMatId	Internal identification number for the container number ***Advice:*** *This parameter is not output at the test client (API Module)!*
materialBinNr	Number of the material bin (container no.)
attributeNumber	Attribute number ***Advice:*** *This parameter is not output at the test client (API Module)!*
attributeDesc	Attribute designator

TAB. 3-134 *Parameter (out) »getMaterialBinsForAttribute«*

Parameter name	Explanation
attributeValue	Value of the attribute
attributeCode	Attribute code
errorString	Output text according to output value (see table)

TAB. 3-134 *Parameter (out) »getMaterialBinsForAttribute«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-76	Error: attribute code must be at least one character
-78	Error: no TR license for this station
-92	Error: attribute code does not exist
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-435	Error: no material bins found for attribute

TAB. 3-135 *Output values for »getMaterialBinsForAttribute«*

3.46 getMaterialSetupData

This function checks whether there is an active setup for the date specified at the station passed. Depending on the parameter `setupState`, the function can supply various setup data.

ADVICE

The `getMaterialSetupData` *API function is influenced by the "CHECK_EXPIRATION" station parameter (code no.: 7134), which activates and deactivates monitoring of the expiration date at the manual setup station. Further information on this topic can be found in the documentation for the ADM Module!*

Function declaration (CORBA)

```
long getMaterialSetupData(
      in string stationNr,
      in long setupState,
      in long date,
      out string setupWorkOrder,
      out long setupActive,
      out long numberOfRecords,
      out SetupDataExtArray setupData,
      out string errorString)
raises (ServerException);

struct SetupDataExt
      {
         string workorder;
         string productNo;
         string productDesc;
         string partOrder;
         string partNo;
         string partDesc;
         string partName;
         string partDateCode;
         string customerNo;
         string customerName;
         string position;
         double quantity;
         long startDate;
         long endDate;
         long creationDate;
         string materialBinNr;
         string creationUser;
         double totalQuantity;
         long state;
      };
typedef sequence<SetupDataExt> SetupDataExtArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
setupState	Extent of the setup data [0; 1; 2]: **0** = Activated setup data only **1** = Pre-setup only **2** = Complete setup
date	Start date for the setup (start of production); the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*

TAB. 3-136 *Parameter (in) »getMaterialSetupData«*

Parameter (out)

Parameter name	Explanation
setupWorkOrder	Order no. for which the station is currently set up
setupActive	Setup exists [0; 1]: **0** = No **1** = Yes
numberOfRecords	Number of entries in the array returned
setupData	Array with the following variables:
workorder	Number of a production order
productNo	Part no. of the product in iTAC.MES.Suite
productDesc	Part desc. of the product in iTAC.MES.Suite
partOrder	Storage unit or supplier charge
partNo	Part no. in iTAC.MES.Suite
partDesc	Part description in iTAC.MES.Suite
partName	Part label in iTAC.MES.Suite for individual component
partDateCode	DateCode for the article
customerNo	Specific part no. in iTAC.MES.Suite
customerName	Customer name in iTAC.MES.Suite
position	Setup location (e.g. track) of the material that has been set up
quantity	Quantity of material container
startDate	Start date for the setup (start of production); the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
endDate	Point in time at which the container was stripped down; the total time is specified as the number of seconds since 01-01-1970
creationDate	Date of creation for the setup; the total time is specified as the number of seconds since 01-01-1970
materialBinNr	Number of the material bin (container no.)
creationUser	User that created the container

TAB. 3-137 *Parameter (out) »getMaterialSetupData«*

Parameter name	Explanation
`totalQuantity`	Original storage unit quantity
`state`	Container state [0; 1; 10; 11]: **0** = Container set up **1** = Container set up and activated in conjunction with activated config parameter "CHECK_EXPIRATION": **10** = Container set up and expired **11** = Container set up, activated, and expired
`errorString`	Output text according to output value (see table)

TAB. 3-137 *Parameter (out) »getMaterialSetupData«*

Output values

Value	Explanations (errorString)
***Advice:** The values `1`; `2` or `3` are only returned in conjunction with an activated config parameter "CHECK_EXPIRATION"!*	
`3`	Warning: One or more container(s) are empty and expired.
`2`	Warning: One or more container(s) are expired
`1`	Warning: One or more container(s) are empty.
`0`	OK: [empty]
`-1`	Error: stationNr not found
`-2`	Error: `setupState` not valid [0; 1; 2]
`-78`	Error: no TR license for this station
`-99`	Error: Server error
`-100`	Error: Corba no Service
`-101`	Error: Corba Exception

TAB. 3-138 *Output values for »getMaterialSetupData«*

3.47 getMaterialSetupDataForKapAndPeriod

This function retrieves the setups that were used at the passed station within the period specified.

Function declaration (CORBA)

```
long getMaterialSetupDataForKapAndPeriod(
     in string stationNr,
     in long dateFrom,
     in long dateTo,
     out long numberOfRecords,
     out SetupDataInfArray setupData,
     out string errorString)
raises (ServerException);

struct SetupDataInf
     {
        string stationNr;
        string position;
        long startDate;
        long endDate;
        string materialBinNr;
        string partOrder;
        string partNo;
        string partDateCode;
     };
typedef sequence<SetupDataInf> SetupDataInfArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
dateFrom	Start time of the start-up; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
dateTo	End time for the expiry, the total time is specified as the number of seconds since 01-01-1970

TAB. 3-139 *Parameter (in) »getMaterialSetupDataForKapAndPeriod«*

Parameter (out)

Parameter name	Explanation
numberOfRecords	Number of entries in the array returned
setupData	Array with the following variables:
stationNr	Station no. of the work station in iTAC.MES.Suite
position	Setup location (e.g. track) of the material that has been set up
startDate	Start date of the setup; the total time is specified as the number of seconds since 01-01-1970, 0:00:00
endDate	Last booking for the setup; the total time is specified as the number of seconds since 01-01-1970
materialBinNr	Number of the material bin (container no.)
partOrder	Storage unit or supplier charge

TAB. 3-140 *Parameter (out) »getMaterialSetupDataForKapAndPeriod«*

Parameter name	Explanation
partNo	Part no. in iTAC.MES.Suite
partDateCode	DateCode for the article
errorString	Output text according to output value (see table)

TAB. 3-140 *Parameter (out) »getMaterialSetupDataForKapAndPeriod«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: stationNr not found
-2	Error: Wrong Time
-10	Error: general Error
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-141 *Output values for »getMaterialSetupDataForKapAndPeriod«*

3.48 getMdaDocuments

This function returns the assigned MDA items for the following objects: part number, part group, work plan, work step, work order and (indirectly) serial number. With this function, it is possible to restrict the result set to specific file extensions or item states. Note that for the objects work order and serial number, it is possible to specify here whether and which (sub-)objects which are referenced to MDA items should be output as well.

ADVICE

This API function is influenced by iTAC.MES.Suite configuration parameter code no.: 2080; all details necessary for outputting the MDA items are defined there (output path, URL etc.). Further information on this topic can be found in the documentation for the ADM Module!

Function declaration (CORBA)

```
long getMdaDocuments(
     in string stationNr,
     in KeyValuePairArray keyValues,
     in KeyValuePairArray mdaDataTypes,
     out MDAObjectArray mdaObjects,
     out string errorString)
raises (ServerException);

struct KeyValuePair
     {
        string name;
        string value;
     };
typedef sequence<KeyValuePair> KeyValuePairArray;

struct MDAObject
     {
        string uniqueFileName;
        string mdaFileName;
        string uniqueFilePath;
        string urlName;
        string mdaDataType;
        string docType;
        string mdaName;
        string mdaDesc;
        string mdaVersion;
        string mdaVersName;
        string mdaVersDesc;
        string mdaStatus;
        string mdaActive;
     };
typedef sequence<MDAObject> MDAObjectArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
keyValues	Array with the following variables:
name	This parameter is used to specify the object for which MDA-items are to be output. The following objects are available: • **serialNo** = serial number; only in combination with the parameter **mdaDataTypes** ("mdaDataType") • **chargeExt** = WO number; optionally also in combination with the parameter **mdaDataTypes** ("mdaDataType") • **partNo** = part number • **partGroupNo** = part group • **workPlanNo** = work plan number; only in connection with "workPlanVers" and "workPlanAlt" • **workPlanVers** = work plan version; only in connection with "workPlanNo" and "workPlanAlt" • **workPlanAlt** = alternative work order type which applies for the work plan; only in connection with "workPlanNo" and "workPlanVers" • **workStepNo** = work step number; only in connection with "workPlanNo", "workPlanVers" and "workPlanAlt" • **workStepAg** = number for work step alternative; only in connection with "workStepNo" (as restriction) • **workStepAvo** = work step number in the ERP system; only in connection with "workPlanNo" and "workStepNo" (as restriction) • **bomVersion** = BOM version; only in connection with "partNo", "serialNo" or "chargeExt" • **bomIndex** = Change index of the BOM; only in connection with "partNo", "serialNo" or "chargeExt" • **processLayer** = Orientation of the PCB during the work step; only in connection with "stationNo" • **errorCode** = Failure code • **stationNo** = Station number
value	Value for the parameter **name** ***Example:** If the value "partNo" is passed for* name*, the corresponding part number must be specified here.*
mdaDataTypes	Array with the following variables:
name	This parameter can be used to filter the results; the following filter mechanisms can be freely combined (OR link): • **mdaDataType** = Is only evaluated by serial number and work order objects (see "serialNo" and "chargeExt"). It is possible to specify whether and which (sub-)objects which are referenced to MDA items should be output as well (see value). • **fileExtension** = Filters the results on basis of the specified file extensions (see value) • **mdaStatus** = Filters the results on basis of the specified element states (see value)

TAB. 3-142 *Parameter (in) »getMdaDocuments«*

Parameter name	Explanation
value	Value for the parameter **name** with "mdaDataType": • **part** - MDA items of the part • **partGroup** - MDA items of the part group • **workPlan** - MDA items of the work plan • **workStep** - MDA items of the work step • **charge** - MDA items of the work order • **bomHead** - MDA items of the BOM version • **processVersion** - MDA items of the process version Value for the parameter **name** with "fileExtension": e.g. "**.doc**" or "**.pdf**" Value for the parameter **name** with "mdaStatus": • **UNUSED** - Unused MDA items • **PROCESS** - MDA items in the release process • **RELEASED** - Released MDA items • **CANCELED** - Unreleased MDA items • **ACTIVE** - Active MDA items

TAB. 3-142 *Parameter (in) »getMdaDocuments«*

Parameter (out)

Parameter name	Explanation
mdaObjects	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
uniqueFileName	Unique designator (ID) for the attached file
mdaFileName	File name of the attached file or the URL of the link
uniqueFilePath	Path of the returned file on the server pool
urlName	- currently has no use -
mdaDataType	MDA type (document or link)
docType	- currently has no use -
mdaName	Name of the MDA item
mdaDesc	Description of the MDA item
mdaVersion	Version number of the MDA item
mdaVersName	Alternative version name
mdaVersDesc	Supplementary description of the current version
mdaStatus	State of the MDA item
mdaActive	Code for the current version of an MDA item
errorString	Output text according to output value (see table)

TAB. 3-143 *Parameter (out) »getMdaDocuments«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-75	***Advice:*** *Depending on the current constellation and the used input parameters, the parameters missing for the function call are output!* Error: werkId Error: workPlanAlt requires workPlanId or workPlanNo Error: workPlanVers requires workPlanAlt Error: workStepNr requires workPlanId or workPlanNo Error: processLayer requires stationNo Error: bomVersion requires part or reference to it Error: bomIndex requires part or reference to it Error: processVersion requires bomVersion or bomIndex
-81	***Advice:*** *Depending on the current constellation and used input parameters, the parameters missing for order ascertainment are output!* Error: Charge not found for serialNo=[serialNo] Error: Charge not found for chargeExt=[chargeExt] Error: VAuft not found for charge=[chargeNr]
-92	Error: unknown keyValue.name [name]
-97	Error: partGrp not found
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-144 *Output values for »getMdaDocuments«*

3.49 getMergeParts

This function is used to output the mounted semifinished products of the **subordinate** level in an array depending on the current integration level of the passed serial number. If the passed serial number is in the lowest integration level, the **parent** integration level is output.

ADVICE

*This API function is influenced by server parameter code no.: 2030. Further information on this topic can be found in the documentation for the ADM Module! The API function should **not** be used for new installations. It may only be used if iTAC.MES.Suite is used within a site; see also »getMergedUnits«!*

Function declaration (CORBA)

```
long getMergeParts(
     in string serialNumber,
     out long numberOfRecords,
     out MergePartsArray mergePartsArray,
     out string errorString)
raises (ServerException);

struct MergePartsData
     {
        string stationNr;
        string stationDesc;
        string serialNumber;
        string serialNumberPos;
        string parentSerialNumber;
        string slaveSerialNumber;
        string level;
        string workOrder;
        string partNr;
        string partDesc;
        long bomVersion;
        string bomIndex;
        string cadPartNr;
     };
typedef sequence<MergePartsData> MergePartsArray;
```

Parameter (in)

Parameter name	Explanation
serialNumber	Serial no. of the panel

TAB. 3-145 *Parameter (in) »getMergeParts«*

Parameter (out)

Parameter name	Explanation
numberOfRecords	Number of entries in the array returned
mergePartsArray	Array with the following variables:
stationNr	Station no. of the work station in iTAC.MES.Suite
stationDesc	Station label of the work station in iTAC.MES.Suite
serialNumber	Serial no. of the panel
serialNumberPos	Position of the single panel in the unit-array

TAB. 3-146 *Parameter (out) »getMergeParts«*

Parameter name	Explanation
parentSerialNumber	Higher order serial no. according to BOM structure.
slaveSerialNumber	Ancillary serial no. (built into part) according to BOM structure.
level	Parameter is not currently used; a "1" is always returned here
workOrder	Number of a production order
partNr	Part no. in iTAC.MES.Suite
partDesc	Part description in iTAC.MES.Suite
bomVersion	Bill-of-material version from iTAC.MES.Suite
bomIndex	Bill-of-material version (index) that is passed to the system by means of changes in BOMs; these are manual changes or changes that have been initiated by higher-level ERP systems
cadPartNr	Part no. of the fundamental bearer, e.g., number of the "bare" printed circuit board
errorString	Output text according to output value (see table)

TAB. 3-146 *Parameter (out) »getMergeParts«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: general Error
-2	Error: serialNumber not found
-3	Error: station not found
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-147 *Output values for »getMergeParts«*

3.50 getMergedUnits

This function is used to output the mounted semifinished products in an array depending on the current integration level of the passed serial number. For the resolution of the merge structure, both the direction as well as the number of integration levels can be specified here.

Function declaration (CORBA)

```
long getMergedUnits(
     in string stationNr,
     in string serialNumber,
     in long resolveDirection,
     in long resolveLevel,
     out MergedUnitsArray mergedUnitsData,
     out string errorString)
raises (ServerException);

struct MergedUnitsData
     {
        string stationNr;
        string stationDesc;
        string serialNumber;
        string serialNumberPos;
        long level;
        string masterSerialNumber;
        string masterSerialNumberPos;
        string slaveSerialNumber;
        string slaveSerialNumberPos;
        string plantNr;
        string workOrder;
        string partNr;
        string partDesc;
        long bomVersion;
        string bomIndex;
        string bomInfo;
        string cadPartNr;
        string info1;
        string info2;
        string info3;
     };
typedef sequence<MergedUnitsData> MergedUnitsArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
serialNumber	Serial no. of the panel
resolveDirection	Depending on the current integration level of the passed serial number, defines in which direction the merge structure is checked [0; 1; -1]: **0** = Only the **subordinate** integration levels are checked for mounted semifinished products. **1** = Only the **parent** integration levels are checked for mounted semifinished products. **-1** = Both the **subordinate** as well as the **parent** integration levels are checked for mounted semifinished products. ***Advice:*** *How many integration levels are taken into account when resolving is defined with* `resolveLevel`*!*

TAB. 3-148 *Parameter (in) »getMergedUnits«*

Parameter name	Explanation
resolveLevel	Depending on the current integration level of the passed serial number, defines how many integration levels are to be taken into account when resolving [> 0; -1]: **> 0** = Number of integration levels to be checked **-1** = No restriction: all integration levels are taken into account ***Advice:*** *The direction in which the merge structure is resolved is defined with* `resolveDirection`*!*

TAB. 3-148 *Parameter (in) »getMergedUnits«*

Parameter (out)

Parameter name	Explanation
mergedUnitsData	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
stationNr	Station no. of the work station in iTAC.MES.Suite
stationDesc	Station label of the work station in iTAC.MES.Suite
serialNumber	Serial no. of the panel
serialNumberPos	Position of the single panel in the unit-array
level	Display of the integration level relative to the passed serial number **> 0** = Number of integration levels upward **0** = Passed serial number **< 0** = Number of integration levels downward
masterSerialNumber	Higher order serial no. according to BOM structure.
masterSerialNumberPos	Position of the single panel on the unit-array for `masterSerialNumber`
slaveSerialNumber	Ancillary serial no. (built into part) according to BOM structure.
slaveSerialNumberPos	Position of the single panel on the unit-array for `slaveSerialNumber`
plantNr	Display of the internal site number
workOrder	Number of a production order
partNr	Part no. in iTAC.MES.Suite
partDesc	Part description in iTAC.MES.Suite
bomVersion	Bill-of-material version from iTAC.MES.Suite
bomIndex	Bill-of-material version (index) that is passed to the system by means of changes in BOMs; these are manual changes or changes that have been initiated by higher-level ERP systems
bomInfo	Output of the stored BOM information
cadPartNr	Part no. of the fundamental bearer, e.g., number of the "bare" printed circuit board

TAB. 3-149 *Parameter (out) »getMergeParts«*

Parameter name	Explanation
info1 info2 info3	Individual additional information 1 - 3 ***Advice:*** *The parameters are intended for future extensions and are not currently evaluated!*
errorString	Output text according to output value (see table)

TAB. 3-149 *Parameter (out) »getMergeParts«*

Output values

Value	Explanations (errorString)
11	Warning: No Data found
0	OK: [empty]
-3	Error: station not found
-39	Error: serialnumber not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-318	Error: No valid resolve direction

TAB. 3-150 *Output values for »getMergeParts«*

3.51 getNextBookingForStationAndLayer

This function returns the time stamp of the next state booking following a specified point in time. The work step is ascertained on the basis of the passed station and layer.

ADVICE

This API function is influenced by station parameter code no.: 8310. Further information on this topic can be found in the documentation for the ADM Module!

Function declaration (CORBA)

```
long getNextBookingForStationAndLayer(
     in string stationNr,
     in long processLayer,
     in long bookDate,
     out long lastBookDate,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent ***Special case:*** *Depending on the setting of configuration parameter code no. 8310, the work step number (AVO) used in an ERP system can be passed directly instead of the PCB orientation!*
bookDate	Date of the state booking (reference value for the return); the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*

TAB. 3-151 *Parameter (in) »getNextBookingForStationAndLayer«*

Parameter (out)

Parameter name	Explanation
lastbookDate	Date of the next state booking with respect to the reference value from the input; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20* ***Advice:*** *If there is no booking after the specified point in time (cf.* `bookDate`*),* ***0*** *is returned!*
errorString	Output text according to output value (see table)

TAB. 3-152 *Parameter (out) »getNextBookingForStationAndLayer«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-153 *Output values for »getNextBookingForStationAndLayer«*

3.52 getNextMaterialBinNumber

This function returns a unique material bin number that is issued by iTAC.MES.Suite and depends on the part no. passed

Function declaration (CORBA)

```
long getNextMaterialBinNumber(
     in string stationNr,
     in string productNr,
     out string materialBinNr,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
productNr	Part no. in iTAC.MES.Suite

TAB. 3-154 *Parameter (in) »getNextMaterialBinNumber«*

Parameter (out)

Parameter name	Explanation
materialBinNr	Number of the material bin (container no.)
errorString	Output text according to output value (see table)

TAB. 3-155 *Parameter (out) »getNextMaterialBinNumber«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: productNr not found
-2	Error: stationNr not found
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-156 *Output values for »getNextMaterialBinNumber«*

3.53 getNextSerialNumber

This function returns a unique serial no. that is issued by iTAC.MES.Suite.

Function declaration (CORBA)

```
long getNextSerialNumber(
     in string stationNr,
     out string serialNumber,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite

TAB. 3-157 *Parameter (in) »getNextSerialNumber«*

Parameter (out)

Parameter name	Explanation
serialNumber	Serial no. of the panel
errorString	Output text according to output value (see table)

TAB. 3-158 *Parameter (out) »getNextSerialNumber«*

Output values

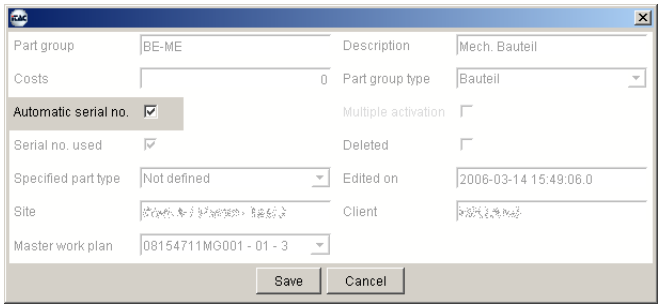
Value	Explanations (errorString)
0	OK: [empty]
-1	Error: No SerialNumber generation is setup for this ProductType and WorkOrderType
-2	Error: No new SerialNumber found
-3	Error: partGroup not found
-4	Error: partNumber not found
-5	Error: stationNr not found
-6	Error: workorder not found
-7	Error: no active workorder found
-21	Error: Serialnumber model range is exceeded
-22	Error: Serialnumber model is invalid
-23	Error: Serialnumber model generation failed
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-159 *Output values for »getNextSerialNumber«*

3.54 getNextSerialNumberForWorkOrder

This function returns the serial numbers created by iTAC.MES.Suite for a WO; the serial numbers are generated based on the deposited serial number model and are irrevocably reserved for the WO. Depending on the part group setting stored in the product of the current WO, two function modes can be distinguished:

1. If the function for the automatic creation of serial numbers is activated for the part group (option field "Automatic serial no." set), all serial numbers required for the WO according to the order quantity are generated at work order creation and are assigned to the WO. The function »getNextSerialNumberForWorkOrder« returns all serial numbers of the WO in this case.



2. If the automatic reservation of serial numbers is deactivated for the part group (option field "Serial no. assignment" not set), a certain number of "new" serial numbers can be created and directly assigned for this WO. The number of the serial numbers to be created is passed via the variable `numberOfRecords`. The function »getNextSerialNumberForWorkOrder« returns all newly created serial numbers in this case.

Function declaration (CORBA)

```
long getNextSerialNumberForWorkOrder(
      in string stationNr,
      in string workOrder,
      inout long numberOfRecords,
      out SerialNumberArray serialNumberArray,
      out string errorString)
raises (ServerException);

struct SerialNumberData
      {
         string serialNumber;
         string serialNumberPos;
      };
typedef sequence<SerialNumberData> SerialNumberArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
workOrder	Number of a production order

TAB. 3-160 *Parameter (in) »getNextSerialNumberForWorkOrder«*

Parameter (inout)

Parameter name	Explanation
numberOfRecords	**in**: Default value for the number of entries in the array ***Advice:*** *Is only evaluated in function mode 2!* **out**: Number of entries in the array returned

TAB. 3-161 *Parameter (inout) »getNextSerialNumberForWorkOrder«*

Parameter (out)

Parameter name	Explanation
serialNumberArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
serialNumber	Serial no. of the panel
serialNumberPos	Position of the single panel in the unit-array
errorString	Output text according to output value (see table)

TAB. 3-162 *Parameter (out) »getNextSerialNumberForWorkOrder«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: No SerialNumber-Generation is Setup for this ProductType and WorkOrderType
-2	Error: No new SerialNumber found ***Advice:*** *-2 is only returned in function mode* ***1****!*
-3	Error: partGroup not found
-4	Error: partNumber not found
-5	Error: stationNr not found
-6	Error: workorder not found
-7	Error: no active workorder found
-21	Error: Serialnumber model range is exceeded
-22	Error: Serialnumber model is invalid
-23	Error: Serialnumber model generation failed
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-163 *Output values for »getNextSerialNumberForWorkOrder«*

3.55 getNumberOfProducts

Production counter: this function returns the number of products that have been produced during the specified period.

Function declaration (CORBA)

```
long getNumberOfProducts(
     in string stationNr,
     in long processLayer,
     in string partNr,
     in long dateFrom,
     in long dateTo,
     in long checkWorkStep,
     out long numberOfProducts,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
partNr	Part no. in iTAC.MES.Suite
dateFrom	Start date for the production counter; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20* ***Special case*** *[-1] passes the current date and 0:00:00 am!*
dateTo	End date for the production counter; the total time is specified as the number of seconds since 01-01-1970 ***Special case:*** *[-1] passes the current date and the current time!*
checkWorkStep	Options for the production counter [0; 1; 2]: **0** = Gets the number of products that have been created within the time period specified *(all)* **1** = Gets the number of products that have been built/ checked at the station passed *(stationNr)* **2** = Gets the number of products that have been built/ checked via the process step of the station passed *(WorkStep)*

TAB. 3-164 *Parameter (in) »getNumberOfProducts«*

Parameter (out)

Parameter name	Explanation
numberOfProducts	Number or products produced
errorString	Output text according to output value (see table)

TAB. 3-165 *Parameter (out) »getNumberOfProducts«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: stationNr not found
-2	Error: `checkWorkStep` not valid [0; 1; 2]
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-166 *Output values for »getNumberOfProducts«*

3.56 getPlacementName

This function returns the name of the mounting program.

Function declaration (CORBA)

```
long getPlacementName(
     in string stationNr,
     out string placementName,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite

TAB. 3-167 *Parameter (in) »getPlacementName«*

Parameter (out)

Parameter name	Explanation
placementName	Name of the mounting program
errorString	Output text according to output value (see table)

TAB. 3-168 *Parameter (out) »getPlacementName«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-169 *Output values for »getPlacementName«*

3.57 getProductInfo

This function returns product information associated with a part number passed. This information can be requested both for the part no. passed and for the structural layers below it. It is possible to select whether exactly one further layer is to be resolved with its entire information, or whether all structures are to be resolved.

Function declaration (CORBA)

```
long getProductInfo(
     in string stationNr,
     in string partNr,
     in long level,
     inout long numberOfRecords,
     out ProductInfoArray productInfos,
     out string errorString)
raises (ServerException);

struct ProductInfoData
     {
        string partNr;
        string partDesc;
        string secPartNr;
        string partGrpNr;
        long level;
        long bomVersion;
        string bomIndex;
        string cadPartNr;
        long long validationDate;
        long long validFrom;
        long long validTo;
     };
typedef sequence<ProductInfoData> ProductInfoArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
partNr	Part no. in iTAC.MES.Suite
level	Specifies the extent of the product information [0; 1; 2; -1; -2]: **0** = Only display information about the part no. passed **1** = Information for part no. passed and for one structural level below, according to the BOM **2** = Information for all structures below part. no., according to the BOM **-1** = Information for part no. passed and one structural level above, according to the BOM **-2** = Information for all structures above part. no., according to the BOM

TAB. 3-170 *Parameter (in) »getProductInfo«*

Parameter (inout)

Parameter name	Explanation
numberOfRecords	**in**: [empty] - value is not evaluated **out**: Number of entries in the array returned

TAB. 3-171 *Parameter (inout) »getProductInfo«*

Parameter (out)

Parameter name	Explanation
productInfos	Array with the following variables:
partNr	Part no. in iTAC.MES.Suite
partDesc	Part description in iTAC.MES.Suite
secPartNr	Customer-specific part no. (customer mat. no.)
partGrpNr	Part group assigned
level	Position of a BOM in the product hierarchy
bomVersion	Bill-of-material version from iTAC.MES.Suite
bomIndex	BOM information
cadPartNr	Part no. of the fundamental bearer, e.g., number of the "bare" printed circuit board
validationDate	Point in time of the last change of the BOM; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
validFrom	Date from which the object is valid, the total time is specified as the number of seconds since 01-01-1970
validTo	Date up to which the object is valid, the total time is specified as the number of seconds since 01-01-1970
errorString	Output text according to output value (see table)

TAB. 3-172 *Parameter (out) »getProductInfo«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: PartNr not found
-2	Error: level not valid [0; 1; 2; -1; -2]
-3	Error: station not found
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-173 *Output values for »getProductInfo«*

3.58 getRecipeData

This function returns the data of a recipe passed. The recipe may be specified either via the identification number of the recipe version or on the basis of the station and part number.

Function declaration (CORBA)

```
long getRecipeData(
     in long recipeVersionId,
     in string stationNr,
     in string partNr,
     in long bomVersion,
     in long processVersion,
     in string revisionIndex,
     out long numberOfRecords,
     out RecipeDataExtendedStructArray recipeData,
     out string errorString)
raises (ServerException);

struct RecipeDataExtendedStruct
     {
        long recipeVersionId;
        double sequentialNumber;
        string name;
        string type;
        string remark;
        string lowerLimit;
        string upperLimit;
        string nominal;
        string tolerance;
        string unit;
        string measureType;
        string lowerActionLimit;
        string upperActionLimit;
        string lowerScrapLimit;
        string upperScrapLimit;
        long weighting;
        long distributionType;
        long confidenceInterval;
     };
typedef sequence<RecipeDataExtendedStruct> RecipeDataExtended-
StructArray;
```

Parameter (in)

Parameter name	Explanation
recipeVersionId	Identification number of the recipe version ***Special case:*** *If the value [-1] is passed, the recipe is determined on the basis of the* `stationNr` *and* `partNr` *variables!*
stationNr	Station no. of the work station in iTAC.MES.Suite ***Special case:*** *Only if [-1] is passed for the* `recipeVersionId` *variable, a valid value must be passed here!*
partNr	Part no. in iTAC.MES.Suite ***Special case:*** *Only if [-1] is passed for the* `recipeVersionId` *variable, a valid value must be passed here!*

TAB. 3-174 *Parameter (in) »getRecipeData«*

Parameter name	Explanation
bomVersion	Bill-of-material version from iTAC.MES.Suite ***Special case:*** *[-1] specifies that this variable will not be analyzed!*
processVersion	Process version of the BOM from iTAC.MES.Suite ***Special case:*** *[-1] specifies that this variable will not be analyzed!*
revisionIndex	Index of changes of the BOM from an external ERP system (SAP) ***Advice:*** *This variable identifies a product version and may be used as an alternative to* **bomVersion** *and* **processVersion***!*

TAB. 3-174 *Parameter (in) »getRecipeData«*

Parameter (out)

Parameter name	Explanation
numberOfRecords	Number of entries in the array returned
recipeData	Array with the following variables:
recipeVersionId	Identification number of the recipe version
sequentialNumber	Testing step no. in the recipe
name	Name of the recipe parameter
type	Specification of the parameter type [V; R; B; D]: **V** = Config **R** = Result **B** = Config-Result **D** = Dynamic
remark	Comment
lowerLimit	Minimum limit value
upperLimit	Maximum limit value
nominal	Nominal value
tolerance	Tolerance value for measurement step
unit	Unit for the value passed
measureType	Specification of the measurement value type [U; T; B; O; H; N; D]: **U** = unknown **T** = text **B** = binary **O** = octal **H** = hexadecimal **N** = decimal EN "." **D** = decimal DE ","
lowerActionLimit	Lower intervention limit
upperActionLimit	Upper intervention limit
lowerScrapLimit	Lower scrap limit
upperScrapLimit	Upper scrap limit
weighting	Weighting for the PAA [0 - 9]:

TAB. 3-175 *Parameter (out) »getRecipeData«*

Parameter name	Explanation
distributionType	Distribution function [0; 1]: **0** = normal distribution **1** = log. normal distribution
confidenceInterval	Confidence region [0; 1; 2; 3]: **0** = 3 Sigma **1** = 4 Sigma **2** = 6 Sigma **3** = 8 Sigma
errorString	Output text according to output value (see table)

TAB. 3-175 *Parameter (out) »getRecipeData«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error
-3	Error: station not found
-60	Error: No recipe header found
-61	Error: Database error while fetching recipe data
-62	Error: Product not found
-63	Error: Invalid session
-64	Error: No recipe data found
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-176 *Output values for »getRecipeData«*

3.59 getRecipeHeaderAndVersion

This function returns the header structure and version information of a recipe passed. The recipe is specified based on the station and part number.

Function declaration (CORBA)

```
long getRecipeHeaderAndVersion(
     in string stationNr,
     in string partNr,
     in long bomVersion,
     in long processVersion,
     in string revisionIndex,
     in boolean onlyActive,
     out long numberOfRecords,
     out RecipeHeaderAndVersionArray recipeHeaders,
     out string errorString)
raises (ServerException);

struct RecipeHeaderAndVersion
     {
        long recipeId;
        string recipeTitle;
        long partId;
        string partNr;
        long bomVersion;
        long processVersion;
        string revisionIndex;
        long processStep;
        long stationId;
        string stationNr;
        long recipeVersionId;
        double recipeVersion;
        string recipeVersionTitle;
        string recipeVersionDescription;
        boolean active;
        string comment;
        long validFrom;
        long validTo;
     };
typedef sequence<RecipeHeaderAndVersion> RecipeHeaderAndVersionArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
partNr	Part no. in iTAC.MES.Suite
bomVersion	Bill-of-material version from iTAC.MES.Suite ***Special case:*** *[-1] specifies that this variable will not be analyzed!*
processVersion	Process version of the BOM from iTAC.MES.Suite ***Special case:*** *[-1] specifies that this variable will not be analyzed!*
revisionIndex	Index of changes of the BOM from an external ERP system (SAP) ***Advice:*** *This variable identifies a product version and may be used as an alternative to* **bomVersion** *and* **processVersion***!*

TAB. 3-177 *Parameter (in) »getRecipeHeaderAndVersion«*

Parameter name	Explanation
onlyActive	Returns only the active recipe [0; 1]: **1** = Only the active recipe is output (TRUE) **0** = All known recipes are output (FALSE)

TAB. 3-177 *Parameter (in) »getRecipeHeaderAndVersion«*

Parameter (out)

Parameter name	Explanation
numberOfRecords	Number of entries in the array returned
recipeHeaderAndVersion	Array with the following variables:
recipeId	Identification number of the recipe header structure
recipeTitle	Title of the recipe header structure
partId	Identification number of the part
partNr	Part no. in iTAC.MES.Suite
bomVersion	Bill-of-material version from iTAC.MES.Suite ***Advice:*** *Value is only output if* revisionIndex *is not used!*
processVersion	Process version of the BOM from iTAC.MES.Suite ***Advice:*** *Value is only output if* revisionIndex *is not used!*
revisionIndex	Index of changes of the BOM from an external ERP system (SAP) ***Advice:*** *Value is only output if* bomVersion *and* processVersion *are not used!*
processStep	Work step number from the work plan
stationId	Identification number of the station
stationNr	Station no. of the work station in iTAC.MES.Suite
recipeVersionId	Identification number of the recipe version
recipeVersion	Recipe version number
recipeVersionTitle	Title of the recipe version
recipeVersionDescription	Description of the recipe version
active	Recipe state [FALSE; TRUE]: **1** = Recipe is active (TRUE) **0** = Recipe is not active (FALSE)
comment	Comment
validFrom	Date from which the object is valid, the total time is specified as the number of seconds since 01-01-1970
validto	Date up to which the object is valid, the total time is specified as the number of seconds since 01-01-1970
errorString	Output text according to output value (see table)

TAB. 3-178 *Parameter (out) »getRecipeHeaderAndVersion«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error
-3	Error: station not found
-60	Error: No recipe header found
-61	Error: Database error while fetching recipe data
-62	Error: Product not found
-63	Error: Invalid session
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-179 *Output values for »getRecipeHeaderAndVersion«*

3.60 getRegisteredBatch

This function returns the material bin currently registered on the passed station.

ADVICE

In the context of bin-based traceability, the »assignBatchNoToWorkorder«, »mergeBatch«, »registerBatch«, »splitBatchNoToSerialNumber« and »unregisterBatch« functions must also be taken into account!

Function declaration (CORBA)

```
long getRegisteredBatch(
     in string stationNr,
     in long processLayer,
     out string batchNumber,
     out long batchComplete,
     out long batchQuantity,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent

TAB. 3-180 *Parameter (in) »getRegisteredBatch«*

Parameter (out)

Parameter name	Explanation
batchNumber	Number of the material bin (container no.)
batchComplete	Complete processing [0; 1]: **1** = Processing fully completed **0** = Processing only interrupted
batchQuantity	Current quantity in the material bin
errorString	Output text according to output value (see table)

TAB. 3-181 *Parameter (out) »getRegisteredBatch«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:** Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-202	Error: container not found

TAB. 3-182 *Output values for »getRegisteredBatch«*

3.61 getRegisteredUser

This function returns the user currently registered at the station.

Function declaration (CORBA)

```
long getRegisteredUser(
     in string stationNr,
     out string userName,
     out string name,
     out string firstname,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite

TAB. 3-183 *Parameter (in) »getRegisteredUser«*

Parameter (out)

Parameter name	Explanation
userName	Login name of the user (user identification)
name	Last name of the user
firstname	First name of the user
errorString	Output text according to output value (see table)

TAB. 3-184 *Parameter (out) »getRegisteredUser«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: user not found
-3	Error: station not found
-50	Error: no user registered at station
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-185 *Output values for »getRegisteredUser«*

3.62 getRequiredEquipmentData

This function returns the required resource – depending on the passed parameters.

ADVICE

In the context of resource management, the »checkEquipmentData«, »getSetupEquipmentData«, »removeEquipmentForWorkorder« and »updateEquipmentData« functions must also be taken into account!

Function declaration (CORBA)

```
long getRequiredEquipmentData(
     in string stationNr,
     in string workOrder,
     in string partNo,
     in long processLayer,
     in string pmGroup,
     out EquipmentCheckDataArray equipmentCheckData,
     out string errorString)
raises (ServerException);

struct EquipmentCheckData
     {
        string equipmentNo;
        string equipmentIndex;
        long checkStatus;
        string equipmentSetupGroup;
        string partNo;
        string partGrp;
        string equipmentExt;
        string equipmentDescription;
        long groupItemType;
        string info1;
        string info2;
        string info3;
     };
typedef sequence<EquipmentCheckData> EquipmentCheckDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
workOrder	Number of a production order ***Special case:** If [-1] is passed here, the work order which is active on the station is used!*
partNo	Part no. in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
pmGroup	Resource group ***Advice:** The wildcard character "*****" can be used during input; pay attention to upper- and lower-case letters!* ***Special case:** If the value [-1] is passed, the resources are returned independent of the resource group!*

TAB. 3-186 *Parameter (in) »getRequiredEquipmentData«*

Parameter (out)

Parameter name	Explanation
equipmentCheckData	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
equipmentNo	Number of the resource in iTAC.MES.Suite
equipmentIndex	Individual designator for the resource
checkStatus	Check status for the resource [0; 1; 2; 3; 4; 5; 6]: **0** = **OK:** Resource is valid and rigged **1** = **Error:** Resource must either be maintained or the point in time of the last permitted use has been exceeded. **2** = **Warning:** Resource must either be maintained or the point in time of the last permitted use has been exceeded. ***Advice:*** *[2] is only output if site parameter code no. 13030 was set to FALSE; otherwise, the value [1] is output!* **3** = **Error:** Resource state is invalid; the resource must have the "available" state **4** = **Error:** Resource is not rigged ***Advice:*** *The affected resource group is output with parameter* equipmentSetupGroup*!* **5** = **Error:** Resource is invalid; period between "valid from" and "valid to" has been exceeded **6** = **Warning:** Resource is invalid; period between "valid from" and "valid to" has been exceeded ***Advice:*** *[5] is only output if site parameter code no. 13030 is set to FALSE!*
equipmentSetup-Group	Name of the resource group in iTAC.MES.Suite; resource groups must be defined as work steps when assigning resources. ***Advice:*** *This parameter is only output if the value [4] is output for* checkStatus*!*
partNo	Part number (device class) on the basis of which the resource was created
partGrp	Part group for the part (device class)
equipmentExt	External resource number
equipmentDescription	Description of the resource
groupItemType	Assignment type for the resource [0; 1]: **0** = Resource was assigned directly **1** = Resource was assigned indirectly via a part (device class) (see partNo)
info1	
info2	Individual additional information 1 - 3
info3	
errorString	Output text according to output value (see table)

TAB. 3-187 *Parameter (out) »getRequiredEquipmentData«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-81	Error: workorder not found
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-188 *Output values for »getRequiredEquipmentData«*

3.63 getResultDataForSerialNumber

This function outputs the booked measurement data for the passed serial number. It is possible to select for the serial number whether the data for a specific work step, for all work steps of the current integration level or for all work steps of the current and the parent integration levels are to be output (`allProductEntries`). In addition to the filtering of the output by measurement step name (`name`) and parameter type (`type`), it is also possible to output the measurement data for all test runs of a work step or only for the last test run (`onlyLastEntry`).

Function declaration (CORBA)

```
long getResultDataForSerialNumber(
     in string stationNr,
     in long processLayer,
     in string serialNumber,
     in long serialNumberPos,
     in string type,
     in string name,
     in long allProductEntries,
     in long onlyLastEntry
     out FailureResultDataArray failureResultData,
     out string errorString)
raises (ServerException);

struct FailureResultData
     {
        string workorder;
        long workstepNo;
        string stationNr;
        long processLayer;
        long seqNumber;
        long nr;
        string name;
        string type;
        string messType;
        string value;
        string min;
        string max;
        string nom;
        string toleranz;
        string unit;
        long failCode;
     };
typedef sequence<FailureResultData> FailureResultDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent ***Advice:*** *The layer is only evaluated, if the variable* `allProductEntries` *has the value* ***0****!*
serialNumber	Serial no. of the panel ***Advice:*** *All relevant order data are ascertained on the basis of the serial number!*

TAB. 3-189 *Parameter (in) »getResultDataForSerialNumber«*

Parameter name	Explanation
serialNumberPos	Position of the single panel in the unit-array ***Advice:*** *If the value [-1] is passed, the position is not evaluated; in this case, each single panel must be specified by means of a unique* `serialNumber` *!*
type	Specification of the parameter type [V; R; B; D]: **V** = Config **R** = Result **B** = Config-Result **D** = Dynamic ***Special case:*** *If the value [-1] is passed, the measurement data are returned independent of the parameter type!*
name	Name of measurement step ***Advice:*** *The wildcard character "*****" can be used during input; pay attention to upper- and lower-case letters!* ***Special case:*** *If the value [-1] is passed, the measurement data are returned independent of the name!*
allProductEntries	Selection of the measurement data to be output [0; 1]: **0** = Measurement data of a work step; the work step is ascertained on the basis of `stationNr` and `processLayer` **1** = Measurement data of the current integration level (work order)
onlyLastEntry	Selection of the test runs to be output [0; 1]: **0** = Measurement data for all test runs **1** = Measurement data of the last test run

TAB. 3-189 *Parameter (in) »getResultDataForSerialNumber«*

Parameter (out)

Parameter name	Explanation
failureResultData	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
`workorder`	Number of a production order
`workstepNo`	Number of the work step
`stationNr`	Station no. of the work station in iTAC.MES.Suite
`processLayer`	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
`seqNumber`	Display of the test run
`nr`	Testing step no. in the recipe
`name`	Name of measurement step

TAB. 3-190 *Parameter (out) »getResultDataForSerialNumber«*

Parameter name	Explanation
`type`	Specification of the parameter type [V; R; B; D]: **V** = Config **R** = Result **B** = Config-Result **D** = Dynamic
`messType`	Format specification for the measurement step: **TXT** = text **BIN** = binary **OCT** = octal **HEX** = hexadecimal **DEN** = decimal EN "." **DDE** = decimal DE ","
`value`	Measured value
`min`	Lower limit for measurement step
`max`	Upper limit for measurement step
`nom`	Nominal value
`toleranz`	Tolerance value for measurement step
`unit`	Unit for the value passed
`failCode`	State information for measurement step [0; 1]: **0** = OK **1** = FAIL
`errorString`	Output text according to output value (see table)

TAB. 3-190 *Parameter (out) »getResultDataForSerialNumber«*

Output values

Value	Explanations (errorString)
`0`	OK: [empty]
`-3`	Error: station not found
`-39`	Error: serialnumber not found
`-50`	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
`-78`	Error: no TR license for this station
`-99`	Error: Server error
`-100`	Error: Corba no Service
`-101`	Error: Corba Exception
`-304`	Error: Invalid value for `processLayer` [0,1,2]
`-314`	Error: Invalid value for `allProductEntries`
`-315`	Error: Invalid value for `onlyLastEntry` [0,1]
`-316`	Error: Invalid value for `type` [V,R,B,D,-1]
`-436`	Error: serialnumber is archived

TAB. 3-191 *Output values for »getResultDataForSerialNumber«*

3.64 getSerialNumberBySnrRef

This function returns an array of serial number and associated placement position that are assigned to a reference serial number.

ADVICE

The `getSerialNumberBySnrRef` *API function is influenced by the "API_GETSNRBYSNRREF_SINGLE_ERROR" station parameter (code no.: 7205). It specifies whether an error message is always issued when a non-reference serial no. is entered (see output value* `-1`*) or respective notices are issued (see output value* `1` *and* `2`*); the latter is the default setting. Further information on this topic can be found in the documentation for the ADM Module!*

Function declaration (CORBA)

```
long getSerialNumberBySnrRef(
     in string stationNr,
     in long processLayer,
     in string serialNumberRef,
     out long numberOfRecords,
     out SerialNumberArray serialNumberArray,
     out string errorString)
raises (ServerException);

struct SerialNumberData
     {
        string serialNumber;
        string serialNumberPos;
     };
typedef sequence<SerialNumberData> SerialNumberArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
serialNumberRef	Serial no. of the first single panel or unit-array; this no. is used as reference serial no.

TAB. 3-192 *Parameter (in) »getSerialNumberBySnrRef«*

Parameter (out)

Parameter name	Explanation
numberOfRecords	Number of entries in the array returned
serialNumberArray	Array with the following variables:
serialNumber	Serial no. of the panel
serialNumberPos	Position of the single panel in the unit-array
errorString	Output text according to output value (see table)

TAB. 3-193 *Parameter (out) »getSerialNumberBySnrRef«*

Output values

Value	Explanations (errorString)
2	OK: Given serialNumber is not a member of a multi-board
1	OK: Given serialNumber is not the reference serialNumber ***Advice:*** *The first serial no. in the array returned is the actual reference serial no.!*
0	OK: [empty]
-1	Error: serialNumberRef not found
-3	Error: station not found
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-436	Error: serialnumber is archived

TAB. 3-194 *Output values for »getSerialNumberBySnrRef«*

3.65 getSerialnumberInfo

This function returns information about the serial number passed and the order it is assigned to.

ADVICE

*This API function should **not** be used for new installations. It may only be used if iTAC.MES.Suite is used within **a** site; see also »getSnrInfoData«!*

Function declaration (CORBA)

```
long getSerialnumberInfo(
     in string serialnumber,
     in string serialnumberPos,
     out string partNr,
     out long bomVersion,
     out string bomIndex,
     out string partDesc,
     out string workOrder,
     out long quantity,
     out string state,
     out string cadPartNr,
     out string customerName,
     out string customerPartNr,
     out string attribut1,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
serialnumber	Serial no. of the panel
serialnumberPos	Position of the single panel in the unit-array ***Advice:** If the value [-1] is passed, the position is not evaluated; in this case, each single panel must be specified by means of a unique* `serialNumber` *!*

TAB. 3-195 *Parameter (in) »getSerialnumberInfo«*

Parameter (out)

Parameter name	Explanation
partNr	Part no. in iTAC.MES.Suite
bomVersion	Bill-of-material version from iTAC.MES.Suite
bomIndex	Bill-of-material version (index) that is passed to the system by means of changes in BOMs; these are manual changes or changes that have been initiated by higher-level ERP systems
partDesc	Part description in iTAC.MES.Suite
workOrder	Number of a production order
quantity	Quantity of production order

TAB. 3-196 *Parameter (out) »getSerialnumberInfo«*

Parameter name	Explanation
state	State of the production order [N; A; I; F; R; S; P; D; E]: **N** = new **A** = planned **I** = initialized **F** = created **R** = released **S** = started **P** = stopped **D** = deleted **E** = finished
cadPartNr	Part no. of the fundamental bearer, e.g., number of the "bare" printed circuit board
customerName	Customer name in iTAC.MES.Suite
customerPartNr	Customer material no. in iTAC.MES.Suite
attribut1	Specific parameter from the part master (e.g., **H** for High, **L** for Low)
errorString	Output text according to output value (see table)

TAB. 3-196 *Parameter (out) »getSerialnumberInfo«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: invalid workOrder
-2	Error: serialNumberRef not found
-5	Error: serialNumber not unique
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-436	Error: serialnumber is archived

TAB. 3-197 *Output values for »getSerialnumberInfo«*

3.66 getSerialNumberHistoryData

This function returns all history data associated with a serial no.

ADVICE

Unlike the API function »getSnrHistoryData« this function may be used across sites!

Function declaration (CORBA)

```
long getSerialNumberHistoryData(
     in string stationNr,
     in string serialNumber,
     out string workOrderNr,
     out string SAPCode,
     out string secondPartNr,
     out string partName,
     out string quantity,
     out string lastReportDate,
     out WorkStationDataArray workStationDataArray,
     out ErrorDataArray errorDataArray,
     out TestDataArray testDataArray,
     out string errorString)
raises (ServerException);

struct WorkStationData
     {
        string workOrderNr;
        string resourceNr;
        string resourceDesc;
        string snr;
        string position;
        string state;
        float cycleTime;
        string bookDate;
        string createDate;
     };
typedef sequence<WorkStationData> WorkStationDataArray;

struct ErrorData
     {
        string workOrderNr;
        string resourceNr;
        string snr;
        string position;
        string partName;
        string errorTypeDesc;
        string partNumber;
        string repaired;
        string adapResNumber;
        string serialError;
     };
typedef sequence<ErrorData> ErrorDataArray;

truct TestData
     {
        string snr;
        string position;
        string resourceNr;
        string resourceDesc;
        long asNr;
        long seqNr;
        string name;
```

```
            string value;
            long failCode;
            long step;
            string registrated;
        };
    typedef sequence<TestData> TestDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
serialNumber	Serial no. of the panel

TAB. 3-198 *Parameter (in) »getSetupDataBySerialNumber«*

Parameter (out)

Parameter name	Explanation
workOrderNr	Number of a production order
SAPCode	Part no. in iTAC.MES.Suite
secondPartNr	Customer-specific part no. (Customer material no.)
partName	Part label in iTAC.MES.Suite for individual component
quantity	Quantity of production order
lastReportDate	Date of the last booking; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
workStationDataArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
workOrderNr	Number of a production order
resourceNr	Station no. of the work station in iTAC.MES.Suite
resourceDesc	Station label of the work station in iTAC.MES.Suite
snr	Serial no. under which the measurement data have been booked
position	Position of the single panel
state	State of the serial no. (uploadState) [0; 1; 2]: **0** = Pass **1** = Fail **2** = Scrap
cycleTime	Duration of the check in seconds
bookDate	Date of the last state booking; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
createDate	Date of creation for the order; the total time is specified as the number of seconds since 01-01-1970

TAB. 3-199 *Parameter (out) »getSetupDataBySerialNumber«*

Parameter name	Explanation
errorDataArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
workOrderNr	Number of a production order
resourceNr	Station no. of the work station in iTAC.MES.Suite
snr	Serial no. under which the measurement data have been booked
position	Position of the single panel
partName	Part label in iTAC.MES.Suite for individual component
errorTypeDesc	Description of the error type
partNumber	Part no. in iTAC.MES.Suite
repaired	Repair status as string
adapResNumber	Station no. of the work station that carries out the repair
serialError	State serial error yes or no [J; N]:
testDataArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
snr	Serial no. under which the measurement data have been booked
position	Position of the single panel
resourceNr	Station no. of the work station in iTAC.MES.Suite
resourceDesc	Station label of the work station in iTAC.MES.Suite
asNr	Work step no. of work plan
seqNr	Number of iterations, specifies how often a serial no. has been booked within one work step
name	Name of measurement step
value	Measured value
failCode	State information for measurement step [0; 1]: **0** = OK **1** = FAIL ***Advice:*** *Other integer values can be interpreted either as pseudo failure (code no.: 15250) or without evaluation (code no.: 15260), depending on the station configuration!*
step	Counter for the number of measurements per measurement parameter
registrated	Entry date as string
errorString	Output text according to output value (see table)

TAB. 3-199 *Parameter (out) »getSetupDataBySerialNumber«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-2	Error: wrong station
-3	Error: no workOrder
-39	Error: serialnumber not found
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-436	Error: serialnumber is archived

TAB. 3-200 *Output values for »getSetupDataBySerialNumber«*

3.67 getSerialNumbersForAttribute

This function returns all serial numbers for a specific attribute.

ADVICE

In the context of serial number attributes, the »appendAttributeToSerialNumber«, »createAttribute«, »getAttributesForSerialNumber« and »removeAttributeFromSerialNumber« functions must also be taken into account!

Function declaration (CORBA)

```
long getSerialNumbersForAttribute(
     in string stationNr,
     in string attributeCode,
     in string attributeValue,
     out SerialNumberStructArray serialNrArray,
     out string errorString)
raises (ServerException);

struct SerialNumberStruct
     {
        string serialNumber;
     };
typedef sequence<SerialNumberStruct> SerialNumberStructArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
attributeCode	Attribute code
attributeValue	Value of the attribute

TAB. 3-201 *Parameter (in) »getSerialNumbersForAttribute«*

Parameter (out)

Parameter name	Explanation
serialNrArray	Array with the following variables: ***Advice:** The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
serialNumber	Serial no. of the panel
errorString	Output text according to output value (see table)

TAB. 3-202 *Parameter (out) »getSerialNumbersForAttribute«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-74	Error: no serialnumbers found for attribute
-76	Error: attribute code must be at least one character
-78	Error: no TR license for this station

TAB. 3-203 *Output values for »getSerialNumbersForAttribute«*

Value	Explanations (errorString)
-92	Error: attribute code does not exist
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-203 *Output values for »getSerialNumbersForAttribute«*

3.68 getSetupDataBySerialNumber

This function returns the setup data for a serial number. It is possible to filter the existing setup data for the serial number according to certain components (parts) and according to specific mounting places (see variables `partNo` and `location`). The two filter functions are "AND" coupled such that only setup data that meet both filter conditions are returned.

ADVICE

Mounting places are optional components of a setup; filtering of the output by specific mounting places only functions if the mounting places are actually defined in the setup!

Function declaration (CORBA)

```
long getSetupDataBySerialNumber(
     in string stationNr,
     in string serialNumber,
     in string partNr,
     in string location,
     out SetupCompDataArray setupDataArray,
     out string errorString)
raises (ServerException);

struct SetupCompData
     {
        long stationId;
        string stationNr;
        string stationDesc;
        long snrId;
        string serialNumber;
        string serialNumberRef;
        string workorder;
        long level;
        long startDate;
        long asNr;
        string position;
        long dateFrom;
        long dateTo;
        long partId;
        string partNo;
        string partDesc;
        long skslPositionNr;
        string partName;
        string compName;
        string partOrder;
        string partDateCode;
        string customerNo;
        string customerName;
        string materialBinNr;
        double totalQuantity;
        long state;
        string classification;
        long expirationDate;
        long readyForUseDate;
        string beaStatus;
     };
typedef sequence<SetupCompData> SetupCompDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
serialNumber	Serial no. of the panel
partNr	Part no. in iTAC.MES.Suite ***Advice:*** *Setup data are only returned for this part (components of the individual item)!* ***Special case:*** *If the value [-1] is passed, this filter function is deactivated; all setup data are returned! Note that the output may, in some cases, be restricted by the "mounting place" filter function (see the* `location` *parameter)!*
location	The mounting place conforms to the iTAC.MES.Suite BOM ***Advice:*** *The setup data are only returned for these mounting places; several mounting places must be separated by "|".* ***Special case:*** *If the value [-1] is passed, this filter function is deactivated; all setup data are returned! Note that the output may, in some cases, be restricted by the "part number" filter function (see the* `partNr` *parameter)!*

TAB. 3-204 *Parameter (in) »getSetupDataBySerialNumber«*

Parameter (out)

Parameter name	Explanation
SetupDataArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
`stationId`	Identification number of the station
`stationNr`	Station no. of the work station in iTAC.MES.Suite
`stationDesc`	Station label of the work station in iTAC.MES.Suite
`snrId`	Internal identification number for the serial number
`serialNumber`	Serial no. of the panel
`serialNumberRef`	Serial no. of the first single panel or unit-array; this no. is used as reference serial no.
`workorder`	Number of a production order
`level`	Position of a BOM in the product hierarchy
`startDate`	Start date for the setup (start of production); the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
`asNr`	Work step no. of work plan
`position`	Setup location (e.g. track) of the material that has been set up

TAB. 3-205 *Parameter (out) »getSetupDataBySerialNumber«*

Parameter name	Explanation
dateFrom	Start time of the start-up; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
dateTo	End time for the expiry, the total time is specified as the number of seconds since 01-01-1970
partId	Identification number of the part
partNo	Part no. in iTAC.MES.Suite
partDesc	Part description in iTAC.MES.Suite
skslPositionNr	Position number of the component in the parts list
partName	Part label in iTAC.MES.Suite for individual component
compName	The mounting place conforms to the iTAC.MES.Suite BOM
partOrder	Storage unit or supplier charge
partDateCode	DateCode for the article
customerNo	Specific part no. in iTAC.MES.Suite
customerName	Customer name in iTAC.MES.Suite
materialBinNr	Number of the material bin (container no.)
totalQuantity	Maximum fill level lot
state	Container state [0; 1; 10; 11]: **0** = Container set up **1** = Container set up and activated in conjunction with activated config parameter "CHECK_EXPIRATION": **10** = Container set up and expired **11** = Container set up, activated, and expired
classification	Classification of the container
expirationDate	Point in time from which on the container must no longer be used; the total time is specified as the number of seconds since 01-01-1970
readyForUseDate	Point in time from which on the container may be used; the total time is specified as the number of seconds since 01-01-1970
beaStatus	Current process state [B; C; E; F; L; Q; R; S]: **B** = LOCKED (Container locked) **C** = COMPLETE (container completed) **E** = FINISHED (Container finished) **F** = INIT (new container created) **L** = MRP LOCKED (Container not available for allocation) **Q** = QA LOCKED (Container with QA lock) **R** = READY (Container ready) **S** = IN PROCESS (Container in process)
errorString	Output text according to output value (see table)

TAB. 3-205 *Parameter (out) »getSetupDataBySerialNumber«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-39	Error: serialnumber not found
-77	Error: no setup data found
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-436	Error: serialnumber is archived

TAB. 3-206 *Output values for »getSetupDataBySerialNumber«*

3.69 getSetupEquipmentData

This function outputs all resources rigged on the passed station independent of the active work order.

ADVICE

In the context of resource management, the »checkEquipmentData«, »getRequiredEquipmentData«, »removeEquipmentForWorkorder« and »updateEquipmentData« functions must also be taken into account!

Function declaration (CORBA)

```
long getSetupEquipmentData(
     in string stationNr,
     out EquipmentSetupDataArray equipmentSetupData,
     out string errorString)
raises (ServerException);

struct EquipmentSetupData
     {
        string equipmentNo;
        string equipmentIndex;
        string equipmentNoExt;
        string equipmentDescription;
        string info1;
        string info2;
        string info3;
        long status;
        long secondsBeforeExpiration;
        long usagesBeforeExpiration;
        long failuresBeforeExpiration;
     };
typedef sequence<EquipmentSetupData> EquipmentSetupDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite

TAB. 3-207 *Parameter (in) »getSetupEquipmentData«*

Parameter (out)

Parameter name	Explanation
equipmentSetupData	Array with the following variables: ***Advice:** The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
equipmentNo	Number of the resource in iTAC.MES.Suite
equipmentIndex	Individual designator for the resource
equipmentNoExt	External resource number
equipmentDescription	Description of the resource

TAB. 3-208 *Parameter (out) »getSetupEquipmentData«*

Parameter name	Explanation
info1	
info2	Individual additional information 1 - 3
info3	
status	Resource status [0; 20; 30; 40; 50; 90]: **0** = available **20 =** being repaired **30 =** being maintained **40** = defective **50** = locked **90** = deleted
secondsBeforeExpiration	Next maintenance date in seconds
usagesBeforeExpiration	Number of possible remaining usages before the resource must be maintained
failuresBeforeExpiration	Number of possible remaining faulty usages before the resource must be maintained
errorString	Output text according to output value (see table)

TAB. 3-208 *Parameter (out) »getSetupEquipmentData«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-209 *Output values for »getSetupEquipmentData«*

3.70 getSnrForWorkorderAndWorkstep

This function returns information on the serial numbers which were booked for a work order at a specific work step.

ADVICE

This API function is influenced by station parameter code no.: 8310. Further information on this topic can be found in the documentation for the ADM Module!

Function declaration (CORBA)

```
long getSnrForWorkorderAndWorkstep(
     in string stationNr,
     in long processLayer,
     in string workorder,
     in long workstepNo,
     in long multiplePanel,
     in long state,
     in long numberOfRecords,
     in long confirmFlag
     out SerialNumberBookInfoArray serialNumberBookInfo,
     out string errorString)
raises (ServerException);

struct SerialNumberBookInfo
     {
        string serialNumber;
        long serialNumberPos;
        long state;
        string workorder;
        long workstepNo;
        string stationNr;
        long processLayer;
        string avo;
        long seqNumber;
        string serialNumberRef;
        long serialNumberRefPos;
     };
typedef sequence<SerialNumberBookInfo>
                                    SerialNumberBookInfoArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent ***Advice:*** *If a [-1] is passed, the field is not evaluated!* ***Special case:*** *Depending on the setting of configuration parameter code no. 8310, the work step number (AVO) used in an ERP system can be passed directly instead of the PCB orientation!*
workorder	Number of a production order ***Special case:*** *If [-1] is passed here, the work order which is active on the station is used!* ***Advice:*** *To prevent the result set from becoming too large, the order quantity must be <= 10,000; work orders with an order quantity > 10,000 are rejected!*

TAB. 3-210 *Parameter (in) »getSnrForWorkorderAndWorkstep«*

Parameter name	Explanation
workstepNo	Controls the scope of serial number output with respect to the work step [> 0; 0; -1; -2]: **> 0** = Number of a specific work step (iTAC.MES.Suite) or of the corresponding AVO no. from a parent ERP system for which the serial numbers are to be output **0** = No restriction: all serial numbers of the work order are output (only the last sequence); this mode enables an overview of the progress of all serial numbers in the work order **-1** = All serial numbers which are pending at the current work step for processing are output; the work step is ascertained on the basis of `stationNr` and `processLayer` **-2** = All serial numbers which are booked at the current work step are output; the work step is ascertained on the basis of `stationNr` and `processLayer` ***Advice:*** *This result set can be further restricted using the* `multiplePanel` *and* `state` *parameters!*
multiplePanel	Controls the scope of serial number output with respect to multiple panels [0; > 0]: **0** = No restriction on multiple panels **> 0** = Only multiple panels with a number of panels >= this value are taken into account for the output ***Advice:*** *This result set can be further restricted using the* `workstepNo` *and* `state` *parameters!*
state	Controls the scope of serial number output with respect to the state of the serial numbers [0; 1; 2; -1]: **0** = Only serial numbers with the "PASS" state **1** = Only serial numbers with the "FAIL" state **2** = Only serial numbers with the "SCRAP" state **-1** = No restriction; the serial numbers are output independent of state. ***Advice:*** *This result set can be further restricted with the* `workstepNo` *and* `multiplePanel` *parameters!* In the case of "multiple panels", the total state is decisive here; the total state, in turn, is determined by the worst single panel state (worst case). ***Example:*** *For a four-way panel with three GOOD serial numbers and one FAIL serial number, no serial number would be output in the event of* `state` *=* ***0****.*
numberOfRecords	Default value for the number of entries in the array returned; this restricts the maximum number of returned data sets ***Advice:*** *In the case of a multiple panel, this number can be exceeded by the value "number of panels minus 1"; this ensures that all serial numbers of the multiple panel are output!*
confirmFlag	State of the serial no. (uploadState) [0; 1; 2]: **0** = Only work steps obligated to supply confirmation are taken into account **1** = Fail

TAB. 3-210 *Parameter (in) »getSnrForWorkorderAndWorkstep«*

Parameter (out)

Parameter name	Explanation
serialNumberBookInfo	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
serialNumber	Serial no. of the panel
serialNumberPos	Position of the single panel in the unit-array
state	State of the serial no. (uploadState) [0; 1; 2]: **0** = Pass **1** = Fail **2** = Scrap
workorder	Number of a production order
workstepNo	Number of the work step
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
avo	Work step number in the ERP system (AVO)
seqNumber	Display of the test run
serialNumberRef	Reference serial number: serial number of the 1st single panel or of the unit-array
serialNumberRefPos	Position of the reference serial number
errorString	Output text according to output value (see table)

TAB. 3-211 *Parameter (out) »getSnrForWorkorderAndWorkstep«*

Output values

Value	Explanations (errorString)
> 0	Warning:[> 0]; number of actually transferred entries ***Advice*** *This only occurs if more serial numbers are assigned to a reference serial number than were requested with the* numberOfRecords *parameter!*
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-205	Error: No workstep found for workorder, layer and station
-206	Error: No active workorder found for station
-384	Error: multiplePanel must be greater or equal to [0]

TAB. 3-212 *Output values for »getSnrForWorkorderAndWorkstep«*

Value	Explanations (errorString)
-385	Error: charge size exceeds max. size of 10000
-387	Error: invalid value for `state`: [-1; 0; 1; 2]
-388	Error: invalid value for `workstepNo`: [-2; -1; 0 or valid AVO-No]

TAB. 3-212 *Output values for »getSnrForWorkorderAndWorkstep«*

3.71 getSnrHistoryData

This function returns all history data associated with a serial no.

ADVICE

*This API function should **not** be used for new installations. It may only be used if iTAC.MES.Suite is used within **a** site; see also »getSetupDataBySerialNumber«!*

Function declaration (CORBA)

```
long getSnrHistoryData(
     in string serialNumber,
     out string workOrderNr,
     out string SAPCode,
     out string secondPartNr,
     out string partName,
     out string quantity,
     out string lastReportDate,
     out WorkStationDataArray workStationDataArray,
     out ErrorDataArray errorDataArray,
     out TestDataArray testDataArray,
     out string errorString)
raises (ServerException);

struct WorkStationData
     {
        string workOrderNr;
        string resourceNr;
        string resourceDesc;
        string snr;
        string position;
        string state;
        float cycleTime;
        string bookDate;
        string createDate;
     };
typedef sequence<WorkStationData> WorkStationDataArray;

struct ErrorData
     {
        string workOrderNr;
        string resourceNr;
        string snr;
        string position;
        string partName;
        string errorTypeDesc;
        string partNumber;
        string repaired;
        string adapResNumber;
        string serialError;
     };
typedef sequence<ErrorData> ErrorDataArray;

truct TestData
     {
        string snr;
        string position;
        string resourceNr;
        string resourceDesc;
        long asNr;
        long seqNr;
        string name;
        string value;
```

```
            long failCode;
            long step;
            string registrated;
        };
    typedef sequence<TestData> TestDataArray;
```

Parameter (in)

Parameter name	Explanation
serialNumber	Serial no. of the panel

TAB. 3-213 *Parameter (in) »getSnrHistoryData«*

Parameter (out)

Parameter name	Explanation
workOrderNr	Number of a production order
SAPCode	Part no. in iTAC.MES.Suite
secondPartNr	Customer-specific part no. (Customer material no.)
partName	Part label in iTAC.MES.Suite for individual component
quantity	Quantity of production order
lastReportDate	Date of the last booking; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
workStationDataArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
workOrderNr	Number of a production order
resourceNr	Station no. of the work station in iTAC.MES.Suite
resourceDesc	Station label of the work station in iTAC.MES.Suite
snr	Serial no. under which the measurement data have been booked
position	Position of the single panel
state	State of the serial no. (uploadState) [0; 1; 2]: **0** = Pass **1** = Fail **2** = Scrap
cycleTime	Duration of the check in seconds
bookDate	Date of the last state booking; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
createDate	Date of creation for the order; the total time is specified as the number of seconds since 01-01-1970

TAB. 3-214 *Parameter (out) »getSnrHistoryData«*

Parameter name	Explanation
errorDataArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
workOrderNr	Number of a production order
resourceNr	Station no. of the work station in iTAC.MES.Suite
snr	Serial no. under which the measurement data have been booked
position	Position of the single panel
partName	Part label in iTAC.MES.Suite for individual component
errorTypeDesc	Description of the error type
partNumber	Part no. in iTAC.MES.Suite
repaired	Repair status as string
adapResNumber	Station no. of the work station that carries out the repair
serialError	State serial error yes or no [J; N]:
testDataArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
snr	Serial no. under which the measurement data have been booked
position	Position of the single panel
resourceNr	Station no. of the work station in iTAC.MES.Suite
resourceDesc	Station label of the work station in iTAC.MES.Suite
asNr	Work step no. of work plan
seqNr	Number of iterations, specifies how often a serial no. has been booked within one work step
name	Name of measurement step
value	Measured value
failCode	State information for measurement step [0; 1]: **0** = OK **1** = FAIL ***Advice:*** *Other integer values can be interpreted either as pseudo failure (code no.: 15250) or without evaluation (code no.: 15260), depending on the station configuration!*
step	Counter for the number of measurements per measurement parameter
registrated	Entry date as string
errorString	Output text according to output value (see table)

TAB. 3-214 *Parameter (out) »getSnrHistoryData«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-2	Error: wrong station
-3	Error: no workOrder
-39	Error: serialnumber not found
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-436	Error: serialnumber is archived

TAB. 3-215 *Output values for »getSnrHistoryData«*

3.72 getSnrInfoData

This function returns information about the serial number passed and the order it is assigned to.

ADVICE

Unlike the »getSerialnumberInfo« API function, this function may be used across sites!

Function declaration (CORBA)

```
long getSnrInfoData(
     in string stationNr,
     in string serialnumber,
     in string serialnumberPos,
     out string partNr,
     out long bomVersion,
     out string bomIndex,
     out string partDesc,
     out string workOrder,
     out long quantity,
     out string state,
     out string cadPartNr,
     out string customerName,
     out string customerPartNr,
     out string attribut1,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
serialnumber	Serial no. of the panel
serialnumberPos	Position of the single panel in the unit-array ***Advice:*** *If the value [-1] is passed, the position is not evaluated; in this case, each single panel must be specified by means of a unique* `serialNumber` *!*

TAB. 3-216 *Parameter (in) »getSnrInfoData«*

Parameter (out)

Parameter name	Explanation
partNr	Part no. in iTAC.MES.Suite
bomVersion	Bill-of-material version from iTAC.MES.Suite
bomIndex	Bill-of-material version (index) that is passed to the system by means of changes in BOMs; these are manual changes or changes that have been initiated by higher-level ERP systems
partDesc	Part description in iTAC.MES.Suite
workOrder	Number of a production order
quantity	Quantity of production order

TAB. 3-217 *Parameter (out) »getSnrInfoData«*

Parameter name	Explanation
state	State of the production order [N; A; I; F; R; S; P; D; E]: **N** = new **A** = planned **I** = initialized **F** = created **R** = released **S** = started **P** = stopped **D** = deleted **E** = finished
cadPartNr	Part no. of the fundamental bearer, e.g., number of the "bare" printed circuit board
customerName	Customer name in iTAC.MES.Suite
customerPartNr	Customer material no. in iTAC.MES.Suite
attribut1	Specific parameter from the part master (e.g., **H** for High, **L** for Low)
errorString	Output text according to output value (see table)

TAB. 3-217 *Parameter (out) »getSnrInfoData«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-2	Error: serialNumber not found
-3	Error: station not found
-5	Error: serialNumber not unique
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-436	Error: serialnumber is archived

TAB. 3-218 *Output values for »getSnrInfoData«*

3.73 getSnrUploadInfo

This function returns information about the last state of the serial number passed. This information either relates to the state in the current work step or to the most recently booked state (cf. `checkProcessStep`).

If a passed serial no. cannot be found, it is always checked whether it has been converted into a new serial no. (see chapter 3.132 »switchSerialNumber«). There are two possibilities:

1. The serial no. searched for never existed. The `errorString` with the value `-2` is returned; all other output values remain empty.
2. The serial no. searched for did exist. Here too, the `errorString` with the value `-2` is returned. In addition, however, the "new" serial no. is supplied as output value for `oldSerialNumber`. With this new serial no., the function can then be called again.

ADVICE

The `getSnrUploadInfo` API function is influenced by station parameter code no.: 7209. Further information on this topic can be found in the documentation for the ADM Module!

Function declaration (CORBA)

```
long getSnrUploadInfo(
      in string stationNr,
      in long processLayer,
      in string serialnumber,
      in string serialnumberPos,
      in long checkProcessStep,
      out string partNr,
      out string workOrder,
      out string uploadStationNr,
      out long processStep,
      out long loopCounter,
      out long state,
      out long bookDate,
      out string oldSerialNumber,
      out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent ***Advice:** The layer is only evaluated, if the variable `checkProcessStep` has the value **1**!*
serialnumber	Serial no. of the panel
serialnumberPos	Position of the single panel in the unit-array ***Advice:** If the value [-1] is passed, the position is not evaluated; in this case, each single panel must be specified by means of a unique `serialNumber` !*

TAB. 3-219 *Parameter (in) »getSnrUploadInfo«*

Parameter name	Explanation
checkProcessStep	Determines the content of the status information about the serial no. [0; 1]: **0** = Returns the most recently booked status of the serial number **1** = Returns the status of the serial number from the current work step

TAB. 3-219 *Parameter (in) »getSnrUploadInfo«*

Parameter (out)

Parameter name	Explanation
partNr	Part no. in iTAC.MES.Suite
workOrder	Number of a production order
uploadStationNr	Station no. of the work station that has checked the serial no.
processStep	Work step number from the work plan
loopCounter	Number of bookings against this serial no. and this work step (stations of the associated ERP group) ***Advice:*** *Only bookings of stations of the type "tester" are counted; bookings of repair and diagnosis stations are* ***not*** *counted!*
state	State of the serial no. (uploadState) [0; 1; 2]: **0** = Pass **1** = Fail **2** = Scrap
bookDate	Date of the last state booking; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
oldSerialNumber	Old serial no. that has been replaced by a change ***Special case:*** *If the serial no. passed was not found because it had already been replaced by another serial no., the "new" serial no. is output here!*
errorString	Output text according to output value (see table)

TAB. 3-220 *Parameter (out) »getSnrUploadInfo«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: no workOrder found for serialNumber
-2	Error: serialNumber not found
-3	Error: serialNumber not unique
-4	Error: StationNr not found
-5	Error: Serial number belongs to another station ***Advice:*** *The serial no. passed is assigned to at least two different WOs; however, the current station is not valid in either of these WOs!*
-6	Error: `checkProcessStep` not valid [0; 1]
-7	Error: no processStep found for `stationNr` and `processLayer`
-78	Error: no TR license for this station

TAB. 3-221 *Output values for »getSnrUploadInfo«*

Value	Explanations (errorString)
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-436	Error: serialnumber is archived

TAB. 3-221 *Output values for »getSnrUploadInfo«*

3.74 getStationQuantity

This function queries a station about the current quantities of an order.

Function declaration (CORBA)

```
long getStationQuantity(
     in string stationNr,
     in long processLayer,
     in string workOrder,
     in long funcMode,
     out long qtyWoTotal,
     out long qtyWoStarted,
     out long qtyWoFinished,
     out long qtyStGood,
     out long qtyStReject,
     out long qtyStScrap,
     out long qtyPsGood,
     out long qtyPsReject,
     out long qtyPsScrap,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
workOrder	Number of a production order
funcMode	Mode for quantities to be passed [0; 1; 2]: **0** = All quantities for work step, order, and station **1** = Quantities for order only **2** = Quantities for order and station only

TAB. 3-222 *Parameter (in) »getStationQuantity«*

Parameter (out)

Parameter name	Explanation
qtyWoTotal	Order quantity
qtyWoStarted	Quantity of units that have been started already, i.e. that are currently in production
qtyWoFinished	Quantity of units that have been finished (produced) already
qtyStGood	Unit pass yield for current station and order
qtyStReject	Unit reject quantity for current station and order
qtyStScrap	Unit scrap quantity for current station and order
qtyPsGood	Unit pass yield for current work step and order
qtyPsReject	Unit reject quantity for current work step and order
qtyPsScrap	Unit scrap quantity for current work step and order
errorString	Output text according to output value (see table)

TAB. 3-223 *Parameter (out) »getStationQuantity«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-2	Error: wrong station
-3	Error: no workOrder
-4	Error: error by reading quantity for workorder
-5	Error: error by reading quantity for workorder at the station
-6	Error: error by reading quantity for workorder in the complete process step of the station
-7	Error: wrong function mode
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-224 *Output values for »getStationQuantity«*

3.75 getStationSetup

This function queries a station for the currently set product version and the associated order.

Function declaration (CORBA)

```
long getStationSetup(
     in string stationNr,
     out string partNr,
     out long bomVersion,
     out string bomIndex,
     out string partDesc,
     out string workOrder,
     out long quantity,
     out string state,
     out string cadPartNr,
     out long processLayer,
     out string customerName,
     out string customerPartNr,
     out string attribut1,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite

TAB. 3-225 *Parameter (in) »getStationSetup«*

Parameter (out)

Parameter name	Explanation
partNr	Part no. in iTAC.MES.Suite
bomVersion	Bill-of-material version from iTAC.MES.Suite
bomIndex	Bill-of-material version (index) that is passed to the system by means of changes in BOMs; these are manual changes or changes that have been initiated by higher-level ERP systems
partDesc	Part description in iTAC.MES.Suite
workOrder	Number of a production order
quantity	Quantity of production order
state	State of the production order [N; A; I; F; R; S; P; D; E]: **N** = new **A** = planned **I** = initialized **F** = created **R** = released **S** = started **P** = stopped **D** = deleted **E** = finished
cadPartNr	Part no. of the fundamental bearer, e.g., number of the "bare" printed circuit board

TAB. 3-226 *Parameter (out) »getStationSetup«*

Parameter name	Explanation
`processLayer`	Orientation of the PCB during the work step [0; 1; 2; 3]: **0** = Component side **1** = Solder side **2** = Position independent **3** = Double-sided panel ***Special case:*** *The value [3] is only returned if the option "Double-sided panel" was chosen when the WO was activated!*
`customerName`	Customer name in iTAC.MES.Suite
`customerPartNr`	Customer material no. in iTAC.MES.Suite
`attribut1`	Specific parameter from the part master (e.g., **H** for High, **L** for Low)
`errorString`	Output text according to output value (see table)

TAB. 3-226 *Parameter (out) »getStationSetup«*

Output values

Value	Explanations (errorString)
`0`	OK: [empty]
`-1`	Error: [empty]
`-2`	Error: wrong station
`-3`	Error: no workOrder active
`-78`	Error: no TR license for this station
`-99`	Error: Server error
`-100`	Error: Corba no Service
`-101`	Error: Corba Exception

TAB. 3-227 *Output values for »getStationSetup«*

3.76 getStorage

This function returns storage information on the basis of various search parameters. The available parameters can be combined freely here; at least one search parameter must be specified.

Function declaration (CORBA)

```
long getStorage(
     in string stationNr,
     in string binLocation,
     in string binLocationBarcode,
     in string partNo,
     in string supplierNo,
     in string chargeNo,
     in string dateCode,
     in string materialBinstate,
     in string classification,
     in AttributeArray attributeArray,
     out StoreInfoArray storeInfoArray,
     out string errorString)
raises (ServerException);

struct AttributeInfo
     {
        string attributeCode;
        string value;
     };
typedef sequence<AttributeInfo> AttributeArray;

struct StoreInfo
     {
        string storageCell;
        string storageCellDesc;
        string storageGroup;
        string storageGroupDesc;
        string storage;
        string storageDesc;
        string materialBinNo;
        string partNo;
        string partDesc;
        double totalQuantity;
        double quantity;
        string unit;
        string supplierNo;
        string supplierName;
        string chargeNo;
        string dateCode;
        long expirationDate;
        char materialBinState;
        double cost;
        long costBase;
        string weNr;
        string classification;
        long createDate;
        long changeDate;
     };
typedef sequence<StoreInfo> StoreInfoArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite

TAB. 3-228 *Parameter (in) »getStorage«*

Parameter name	Explanation
binLocation	Storage location no. in iTAC.MES.Suite ***Advice:*** *The wildcard character "*****" can be used during input; pay attention to upper- and lower-case letters!* ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated!*
binLocationBarcode	Bar code for the storage; must not coincide with the storage no. in iTAC.MES.Suite ***Advice:*** *The wildcard character "*****" can be used during input; pay attention to upper- and lower-case letters!* ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated!*
partNo	Part no. of the container material in iTAC.MES.Suite ***Advice:*** *The wildcard character "*****" can be used during input; pay attention to upper- and lower-case letters!* ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated!*
supplierNo	Supplier number ***Advice:*** *The wildcard character "*****" can be used during input; pay attention to upper- and lower-case letters!* ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated!*
chargeNo	Storage unit or supplier charge ***Advice:*** *The wildcard character "*****" can be used during input; pay attention to upper- and lower-case letters!* ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated!*
dateCode	DateCode for the storage unit ***Advice:*** *The wildcard character "*****" can be used during input; pay attention to upper- and lower-case letters!* ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated!*
materialBinState	Container state [B; C; E; F; L; Q; R; S; -1]: **B** = LOCKED (Container locked) **C** = COMPLETE (container completed) **E** = FINISHED (Container finished) **F** = INIT (new container created) **L** = MRP LOCKED (Container not available for allocation) **Q** = QA LOCKED (Container with QA lock) **R** = READY (Container ready) **S** = IN PROCESS (Container in process) ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated!*
classification	Classification of the container ***Advice:*** *The wildcard character "*****" can be used during input; pay attention to upper- and lower-case letters!* ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated!*
attributeArray	Array with the following variables (OR link):

TAB. 3-228 *Parameter (in) »getStorage«*

Parameter name	Explanation
attributeCode	Attribute code ***Advice:*** *The wildcard character "*****" can be used during input; pay attention to upper- and lower-case letters!* ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated!*
value	Value of the attribute ***Advice:*** *The wildcard character "*****" can be used during input; pay attention to upper- and lower-case letters!* ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated!*

TAB. 3-228 *Parameter (in) »getStorage«*

Parameter (out)

Parameter name	Explanation
storeInfoArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
storageCell	Storage cell no. in iTAC.MES.Suite
storageCellDesc	Storage cell description in iTAC.MES.Suite
storageGroup	Storage groups no. in iTAC.MES.Suite
storageGroupDesc	Storage group description in iTAC.MES.Suite
storage	Storage location no. in iTAC.MES.Suite
storageDesc	Storage location description in iTAC.MES.Suite
materialBinNo	Number of the material bin (container no.)
partNo	Part no. of the container material in iTAC.MES.Suite
partDesc	Part description in iTAC.MES.Suite
totalQuantity	Original storage unit quantity
quantity	Quantity of material container
unit	Unit for the value passed
supplierNo	Supplier number
supplierName	Supplier name
chargeNo	Storage unit or supplier charge
dateCode	DateCode for the storage unit
expirationDate	Point in time from which on the container must no longer be used; the total time is specified as the number of seconds since 01-01-1970

TAB. 3-229 *Parameter (out) »getStorage«*

Parameter name	Explanation
materialBinState	Container state [B; C; E; F; L; Q; R; S]: **B** = LOCKED (Container locked) **C** = COMPLETE (container completed) **E** = FINISHED (Container finished) **F** = INIT (new container created) **L** = MRP LOCKED (Container not available for allocation) **Q** = QA LOCKED (Container with QA lock) **R** = READY (Container ready) **S** = IN PROCESS (Container in process)
cost	Average purchase price [€] of the material (see costBase)
costBase	Material quantity on which the purchase price is based (see cost)
weNr	Receiving number
classification	Classification of the container
createDate	Date of creation for the container; the total time is specified as the number of seconds since 01-01-1970
changeDate	Date of the last state change for the container; the total time is specified as the number of seconds since 01-01-1970
errorString	Output text according to output value (see table)

TAB. 3-229 *Parameter (out) »getStorage«*

Output values

Value	Explanations (errorString)
11	Warning: No Data found
0	OK: [empty]
-3	Error: station not found
-78	Error: no TR license for this station
-91	Error: Not enough values for testing
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-230 *Output values for »getStorage«*

3.77 getWorkorderForLine

This function uses special search criteria to ascertain the production orders relevant for a specific line. Available search criteria are order state, part number, part group, date of creation, and the order number itself. All search criteria are "AND" coupled such that only the work orders that meet all used filter conditions are returned.

Function declaration (CORBA)

```
long getWorkorderForLine(
     in string stationNr,
     in string workorderState,
     in string workorder,
     in string productNo,
     in string partGrp,
     in long startDate,
     out WorkorderForLineArray workorderForLine,
     out string errorString)
raises (ServerException);

struct WorkorderForLine
     {
        string workorder;
        string workorderDesc;
        string productNo;
        double quantity;
        string workorderState;
        string workorderType;
        long bomVersion;
        string revisionIndex;
        long processVersion;
        long workPlanVers;
        long plannedStartDate;
        long plannedShipDate;
        long workorderStartDate;
        string infoTxt1;
        string infoTxt2;
        string infoTxt3;
        string infoTxt4;
        string infoTxt5;
        string infoTxt6;
        long infoNo1;
        long infoNo2;
        long infoNo3;
        long infoNo4;
     };
typedef sequence<WorkorderForLine> WorkorderForLineArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite ***Advice:*** *On basis of the station number, the line is ascertained for which the corresponding production orders are output!*

TAB. 3-231 *Parameter (in) »getWorkorderForLine«*

Parameter name	Explanation
workorderState	State of the production order [N; A; I; F; R; S; P; D; E]: **N** = new **A** = planned **I** = initialized **F** = created **R** = released **S** = started **P** = stopped **D** = deleted **E** = finished ***Advice:*** *The input is optional; if [-1] or [empty] is passed, the field is not evaluated!*
workorder	Number of a production order ***Advice:*** *The input is optional; if [-1] or [empty] is passed, the field is not evaluated!*
productNo	Part no. of the product in iTAC.MES.Suite ***Advice:*** *The input is optional; if [-1] or [empty] is passed, the field is not evaluated!*
partGrp	Part group of the container material in iTAC.MES.Suite ***Advice:*** *The input is optional; if [-1] or [empty] is passed, the field is not evaluated!*
startDate	Date of creation for the order; the total time is specified as the number of seconds since 01-01-1970 ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!*

TAB. 3-231 *Parameter (in) »getWorkorderForLine«*

Parameter (out)

Parameter name	Explanation
workorderForLine	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
workorder	Production order number
workorderDesc	Production order description
productNo	Part no. of the product in iTAC.MES.Suite
quantity	Quantity of production order
workorderState	State of the production order [N; A; I; F; R; S; P; D; E]: **N** = new **A** = planned **I** = initialized **F** = created **R** = released **S** = started **P** = stopped **D** = deleted **E** = finished

TAB. 3-232 *Parameter (out) »getWorkorderForLine«*

Parameter name	Explanation
workorderType	Abbreviation for the work order type (e.g. 01, WO)
bomVersion	Bill-of-material version from iTAC.MES.Suite
revisionIndex	Index of changes of the BOM from an external ERP system (SAP) ***Advice:*** *This variable identifies a product version and may be used as an alternative to* **bomVersion** *and* **processVersion***!*
processVersion	Process version of the BOM from iTAC.MES.Suite
workPlanVers	Work plan version
plannedStartDate	Start date for the work order (planned start of production); the total time is specified as the number of seconds since 01-01-1970, 0:00:00
plannedShipDate	Due date for the order (planned delivery date); the total time is specified as the number of seconds since 01-01-1970, 0:00:00
workorderStartDate	Start date for the work order (actual start of production); the total time is specified as the number of seconds since 01-01-1970, 0:00:00
infoTxt1	Alphanumerical additional information
infoTxt2	
infoTxt3	
infoTxt4	
infoTxt5	
infoTxt6	
infoNo1	Numerical additional information
infoNo2	
infoNo3	
infoNo4	
errorString	Output text according to output value (see table)

TAB. 3-232 *Parameter (out) »getWorkorderForLine«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-75	Error: a required parameter is missing ***Advice:*** *This value is only returned if an empty string is passed for the* stationNr *parameter!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-233 *Output values for »getWorkorderForLine«*

3.78 getWorkordersForAttribute

This function returns all orders for a specific attribute.

ADVICE

In the context of work order attributes, the »appendAttributeToWorkorder«, »createAttribute«, »getAttributesForWorkorder« and »removeAttributeFromWorkorder« functions must also be taken into account!

Function declaration (CORBA)

```
long getWorkordersForAttribute(
     in string stationNr,
     in string attributeCode,
     in string partNo,
     in string attributeValue,
     in string attributeType,
     in string objectKey,
     out ChargeInfoStructArray chargeInfoArray,
     out string errorString)
raises (ServerException);

struct ChargeInfoStruct
     {
        string chargeExt;
        string state;
        long qty;
     };
typedef sequence<ChargeInfoStruct> ChargeInfoStructArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
attributeCode	Attribute code
partNo	Part no. in iTAC.MES.Suite ***Advice:*** *This parameter is optional and enables the querying of work order attributes with a part reference; the part number must be contained in the order BOM. If no part reference is to be checked, the value [-1] must be passed here!*
attributeValue	Value of the attribute ***Advice:*** *The wildcard character "*****" can be used during input; pay attention to upper- and lower-case letters!* ***Special case:*** *If the value [-1] is passed, all work orders are returned independent of the attribute value; pay attention here to the association with the* `attributeType` *and* `objectKey` *parameters!*
attributeType	Attribute type ***Advice:*** *The wildcard character "*****" can be used during input; pay attention to upper- and lower-case letters!* ***Special case:*** *If the value [-1] is passed, all work orders are returned independent of the attribute type; pay attention here to the association with the* `attributeValue` *and* `objectKey` *parameters!*

TAB. 3-234 *Parameter (in) »getWorkordersForAttribute«*

Parameter name	Explanation
objectKey	Key term for the attribute ***Advice:*** *The wildcard character "*****" can be used during input; pay attention to upper- and lower-case letters!* ***Special case:*** *If the value [-1] is passed, all work orders are returned independent of the key term; pay attention here to the association with the* `attributeValue` *and* `attributeType` *parameters!*

TAB. 3-234 *Parameter (in) »getWorkordersForAttribute«*

Parameter (out)

Parameter name	Explanation
chargeInfoArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
chargeExt	Production order number
state	State of the production order [N; A; I; F; R; S; P; D; E]: **N** = new **A** = planned **I** = initialized **F** = created **R** = released **S** = started **P** = stopped **D** = deleted **E** = finished
qty	Quantity of production order
errorString	Output text according to output value (see table)

TAB. 3-235 *Parameter (out) »getWorkordersForAttribute«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-76	Error: attribute code must be at least one character
-78	Error: no TR license for this station
-92	Error: attribute code does not exist
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-221	Error: no workorders found for attribute
-220	Error: Part not found
-222	Error: attribute type not exists

TAB. 3-236 *Output values for »getWorkordersForAttribute«*

3.79 getWorkplanItems

This function returns detailed work step information about the current work plan. In this context it is possible to choose whether the information is to be returned for all stations of the work plan, or only for the station passed.

ADVICE

For this function, either a WO number or a serial number must be passed. If both WO and serial number are being passed, it must be ensured that the serial number is part of the WO.

Function declaration (CORBA)

```
long getWorkplanItems(
     in string stationNr,
     in string workOrder,
     in string serialNumber,
     in long processLayer,
     in long wpFlag,
     out WorkplanDataArray workplanData,
     out string errorString)
raises (ServerException);

struct WorkplanData
     {
        string stationNr;
        string workOrder;
        long workstepNo;
        long processStepNo;
        long voucherNo;
        long threadNo;
        long validFrom;
        long validTo;
        long prevWorkstep;
        long nextWorkstep;
        long prevWorkstepConfirm;
        long nextWorkstepConfirm;
        long qtyWoTotal;
        long pass;
        long scrap;
        long fail;
        long qtyWoOpen;
        long qtyPsScrap;
        long processLayer;
        long separationFlag;
        string erpGroupNo;
        string erpGroupDesc;
        string obligatoryConfirmFlag;
        string setupFlag;
        string workplanProcessStepNo;
        string avo;
     };
typedef sequence<WorkplanData> WorkplanDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
workOrder	Number of a production order ***Advice:*** *If an empty string or [-1] are passed, the field is not evaluated; in these cases, a valid serial number must be passed via* `serialNumber`*!*

TAB. 3-237 *Parameter (in) »getWorkplanItems«*

Parameter name	Explanation
serialNumber	Serial no. of the panel ***Advice:*** *If an empty string or [-1] are passed, the field is not evaluated; in these cases, a valid WO number must be passed via* `workOrder`*!*
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent ***Advice:*** *If a [-1] is passed, the field is not evaluated!*
wpFlag	Possible function modes [0; 1; 2; 3; 4]: **0** = All work steps for all stations included in the work plan are output **1** = All work steps for the station passed are output **2** = The next – not yet booked – work step for the station and serial number is output ***Advice:*** *This mode requires a valid serial number to be passed via* `serialNumber`*!* **3** = The following work step in the work plan with the same layer is output ***Advice:*** *Passing a [-1] for* `processLayer` *is not permitted in this mode!* **4** = The next work step in the work plan with the same layer obligated to supply confirmation is output ***Advice:*** *Passing a [-1] for* `processLayer` *is not permitted in this mode!*

TAB. 3-237 *Parameter (in) »getWorkplanItems«*

Parameter (out)

Parameter name	Explanation
workplanData	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
`stationNr`	Station no. of the work station in iTAC.MES.Suite
`workOrder`	Number of a production order
`workstepNo`	Current work step number
`processStepNo`	Enumerating counter for each station of the ERP group for each work step and work step alternative
`voucherNo`	Voucher number in iTAC.MES.Suite
`threadNo`	Number of the processing thread
`validFrom`	Date from which the object is valid, the total time is specified as the number of seconds since 01-01-1970
`validTo`	Date up to which the object is valid, the total time is specified as the number of seconds since 01-01-1970
`prevWorkstep`	Previous work step number
`nextWorkstep`	Next work step number

TAB. 3-238 *Parameter (out) »getWorkplanItems«*

Parameter name	Explanation
prevWorkstepConfirm	Number of the previous work step with obligatory confirmation
nextWorkstepConfirm	Number of the next work step with obligatory confirmation
qtyWoTotal	Order quantity
pass	Number of "PASS" bookings for the current WO
scrap	Number of "SCRAP" bookings for the current WO
fail	Number of "FAIL" bookings for the current WO
qtyWoOpen	Quantity of units still to be produced in the current work step
qtyPsScrap	Unit scrap quantity for current work step and order
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
separationFlag	Designation for "separating" work step (for multiple panel)
erpGroupNo	ERP group number in iTAC.MES.Suite
erpGroupDesc	ERP group designation in iTAC.MES.Suite
obligatoryConfirmFlag	Obligatory confirmation for current work step (Yes / No) [J; N]
setupFlag	Setup requirement for current work step (Yes / No) [J; N]
workplanProcessStepNo	Consecutive number for the alternatives of a work step
avo	Work step number in the ERP system (AVO)
errorString	Output text according to output value (see table)

TAB. 3-238 *Parameter (out) »getWorkplanItems«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-39	Error: serialnumber not found
-78	Error: no TR license for this station
-80	Error: flag must be in [0;1;2;3;4]
-81	Error: workorder not found
-82	Error: serialnumber belongs to another workorder
-83	Error: serialnumber or workorder required
-84	Error: mode 2 requires a serialnumber
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-239 *Output values for »getWorkplanItems«*

3.80 mdataBomVerify

This function is used to compare an existing mounting program with the current order BOM. The function is generally used at a pre-setup station of an SMD line. If there are discrepancies between mounting program and order BOM, a list with all deviating positions is returned as the result.

ADVICE

In the future, it will also be possible to use this API function to create new BOMs in the system. Used in particular for this purpose is the `mdataBomItemArray` *array; further information on this topic is available upon request!*

When working with this function, it is necessary to differentiate between search and comparison criteria that are passed with the `mdataBomItemArray` array. Search criteria (parameters `partNo` and/or `compName`) are mandatory and are used to assign mounting program data to BOM positions for the comparison. Comparison criteria (all other parameters in the array with the exception of `returnComment`) are optional and enable individual "restriction" of the comparison, since all of the used comparison criteria must be exactly defined in the parts list.

In addition to the successful comparison – the mounting program matches the order BOM (output value (`errorString`) "**`0`**" – the program distinguishes between the four following **error cases**:

1. **Search criteria not found:** If the search criteria passed in the `mdataBomItemArray` array (parameter `partNo` and/or `compName`) are not found for the position, the entered data are returned with an appropriate comment (cf. parameter `returnComment`) in the output array. In this case, the function returns "**`-438`**" as the output value (`errorString`).
2. **Component values are different:** If the values of a component differ between the mounting program and BOM position, the corrected data are returned with an appropriate comment (cf. parameter `returnComment`) in the output array. In this case, the function returns "**`-438`**" as the output value (`errorString`).
3. **Component not found in the BOM**: If individual components from the mounting program data are not found in the order BOM, the entered data are returned with an appropriate comment (cf. parameter `returnComment`) in the output array. In this case, the function returns "**`-438`**" as the output value (`errorString`).
4. **Component not found in the mounting program**: If there are components (BOM positions) in the order BOM that are not contained in the mounting program, these missing positions are returned with an appropriate comment (cf. parameter `returnComment`) in the output array. In this case, the function returns "**`-438`**" as the output value (`errorString`).

Function declaration (CORBA)

```
long mdataBomVerify(
      in string stationNr,
      in long processLayer,
      in string workorder,
      in string productNo,
      in string bareBoardNo,
      in string bomIndex,
      in string bomInfo,
      in long bomValidFrom,
      in string bomVersionERP,
      in long verifyCompNameBased,
```

```
        in long createNewBom,
        inout MdataBomItemArray mdataBomItemArray,
        out string errorString)
    raises (ServerException);

    struct MdataBomItem
        {
           string partNo;
           string partDesc;
           string secPartNo;
           long artGrpNo;
           string compName;
           long alternative;
           string erpPosNo;
           long posTyp;
           double quantity;
           long unit;
           long isProduct;
           string bomVersionERP;
           string bomInfoTxt;
           long processTyp;
           string processGrp;
           long processLayer;
           long traceFlag;
           long workstepNo;
           string returnComment
        };
    typedef sequence<MdataBomItem> MdataBomItemArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Soldering side **2** = Position independent ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated!*
workorder	Number of a production order ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated; in this case, the part (*`productNo`*) must absolutely be passed!*
productNo	Part no. of the product in iTAC.MES.Suite ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated; in this case, the work order (*`workorder`*) must absolutely be passed!*
bareBoardNo	Part no. of the fundamental bearer, e.g., number of the "bare" printed circuit board ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated!*

TAB. 3-240 *Parameter (in) »mdataBomVerify«*

Parameter name	Explanation
bomIndex	Bill-of-material version (index) that is passed to the system by means of changes in BOMs; these are manual changes or changes that have been initiated by higher-level ERP systems ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated!* ***Advice:*** *In the BOM Module this parameter is displayed in the BOM header table in the "Change No." field!*
bomInfo	Individual BOM information (free text) ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated!* ***Advice:*** *In the BOM Module this parameter is displayed in the BOM header table in the "BOM revision" field!*
bomValidFrom	Point in time at which the BOM becomes valid ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated!*
bomVersionERP	BOM version in the ERP system ***Special case:*** *If the value [-1] is passed, this parameter is not evaluated!* ***Advice:*** *In the BOM Module this parameter is displayed in the BOM header table in the "Change index" field!*
verifyCompNameBased	Check of the mounting places [0; 1]: **0** = Mounting places are not checked **1** = Mounting places are checked ***Advice:*** *If the value [1] is passed, the mounting place (*`compName`*) is interpreted as a search criterion in the* `mdataBomItemArray` *array in addition to the part no. (*`partNo`*)!*
createNewBom	BOM creation [0; 1]: **0** = BOM is checked, but not created; all parameters passed in the `mdataBomItemArray` array are interpreted as search or comparison criterion **1** = BOM is created on the basis of the parameters passed in the `mdataBomItemArray` array ***Advice:*** *This parameter is intended for future use; currently, only mode **0** is used!*

TAB. 3-240 *Parameter (in) »mdataBomVerify«*

Parameter (inout)

Parameter name	Explanation
mdataBomItemArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*

TAB. 3-241 *Parameter (inout) »mdataBomVerify«*

Parameter name	Explanation
partNo	**in**: Part no. of the component in iTAC.MES.Suite ***Advice:*** *Unlike the other parameters, input of the part no. as a search criterion is absolutely required! If the* **verifyCompNameBased** *input parameter is passed with [1], the mounting place (*compName*) is also interpreted as a search criterion (OR link).* **out**: In case of an error (see function description), the corrected/missing value is output if necessary
partDesc	**in**: Part description of the component in iTAC.MES.Suite ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!* **out**: In case of an error (see function description), the corrected/missing value is output if necessary
secPartNo	**in**: Customer-specific part no. (customer mat. no.) ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!* **out**: In case of an error (see function description), the corrected/missing value is output if necessary
compName	**in**: Mounting location conformant with iTAC.MES.Suite BOM ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated! If, however, a value of [1] is passed for* **verifyCompNameBased**, *the mounting location must absolutely be passed, as it is interpreted in combination with the* part no. (partNo*) as a search criterion (OR link)!* **out**: In case of an error (see function description), the corrected/missing value is output if necessary
alternative	**in**: Designator for regular or alternative component in the BOM [0; 1]: **0** = Regular component **1** = Alternative component ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!* **out**: In case of an error (see function description), the corrected/missing value is output if necessary
erpPosNo	**in**: Work step no. of the processing process in the ERP system for the component ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!* **out**: In case of an error (see function description), the corrected/missing value is output if necessary
posTyp	**in**: Definition of the component type [11500 - 11502]: **11500** = Raw material **11501** = Semi-finished good **11502** = Finished product ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!* **out**: In case of an error (see function description), the corrected/missing value is output if necessary

TAB. 3-241 *Parameter (inout) »mdataBomVerify«*

Parameter name	Explanation
quantity	**in**: Quantity of the part ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!* **out**: In case of an error (see function description), the corrected/missing value is output if necessary ***Advice:*** *Decimal values are rounded to the nearest whole number!*
unit	**in**: Unit for the value passed ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!* **out**: In case of an error (see function description), the corrected/missing value is output if necessary
isProduct	**in**: Designator for raw material / product [0; 1] **0** = Raw material (component) **1** = Product (unit) ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!* **out**: In case of an error (see function description), the corrected/missing value is output if necessary
bomVersionERP	**in**: BOM version in the ERP system ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!* **out**: In case of an error (see function description), the corrected/missing value is output if necessary ***Advice:*** *In the BOM Module this parameter is displayed in the BOM header table in the "Change index" field!*
bomInfoTxt	**in**: Individual BOM position information (free text) ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!* **out**: In case of an error (see function description), the corrected/missing value is output if necessary ***Advice:*** *In the BOM Module this parameter is displayed in the BOM position table in the "Info text" field!*
processTyp	**in**: Selection of a use type [10300 - 10304]: **10300** = Mounting **10301** = Visual inspection **10302** = Component preparation **10303** = Provision by customer ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!* **out**: In case of an error (see function description), the corrected/missing value is output if necessary
processGrp	**in**: Machine group on which the mounting is carried out ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!* **out**: In case of an error (see function description), the corrected/missing value is output if necessary

TAB. 3-241 *Parameter (inout) »mdataBomVerify«*

Parameter name	Explanation
processLayer	**in**: Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Soldering side **2** = Position independent ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!* **out**: In case of an error (see function description), the corrected/missing value is output if necessary
traceFlag	**in**: Designator for the setup requirement [0; 1]: **0** = No setup required **1** = Setup required ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!* **out**: In case of an error (see function description), the corrected/missing value is output if necessary
workstepNo	**in**: Work step no. in iTAC.MES.Suite ***Advice:*** *The input is optional; if [-1] is passed, the field is not evaluated!* **out**: In case of an error (see function description), the corrected/missing value is output if necessary
returnComment	**in**: [empty] - value is not evaluated **out**: Depending on error case (see function description), the following comments may be returned: in error case **1**: - "a required parameter is missing" in error case **2**: - "data for component are wrong" - "data for mounting place are wrong" in error case **3**: - "mounting place not in the BOM" - "component not in the BOM" in error case **4**: - "mounting place missing" (also for "component missing") ***Advice:*** *Which of the two comments are returned for error cases **2** and **3** is dependent on whether or not the mounting place check is activated (see* **verifyCompNameBased***)!*

TAB. 3-241 *Parameter (inout) »mdataBomVerify«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-242 *Parameter (out) »mdataBomVerify«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-7	Error: no processStep found for stationNr and processLayer
-67	Error: no valid bom version

TAB. 3-243 *Output values for »mdataBomVerify«*

Value	Explanations (errorString)
-68	Error: no valid process version
-81	Error: workorder not found
-91	Error: Not enough values for testing ***Advice:*** *This value is output if no valid part number (*`productNo`*) or order number (*`workorder`*) was passed!*
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-362	Error: Partnumber not found
-418	Error: BOM not found
-422	Error: `verifyCompNameBased` flag not valid [0; 1]
-423	Error: `createNewBom` flag not valid [0; 1]
-438	Error: Bom verify failed

TAB. 3-243 *Output values for »mdataBomVerify«*

3.81 mdataGetBomData

This function returns a specific BOM. Used as reference value is the order number, the serial number or the part number in combination with the BOM version and process version; if multiple values are passed, the order number is used.

ADVICE

This function is based on the older »getBomItems« API function. The »mdataGetBomData« was expanded to facilitate searches for specific process parameters!

The extent of the returned BOM information may be controlled via three control parameters.

- The input value `alternative` specifies whether the alternative BOM components are to be output, too.
- The input value `processBased` specifies whether the process-related BOM (with station no. passed) or the general BOM – valid for all processes – is output.
- The input value `bomType` specifies whether the construction BOM or the scheduling BOM is output:
  Construction BOM = all components are listed according to their mounting place
  Scheduling BOM = identical components are shown in a consolidated manner

Function declaration (CORBA)

```
long mdataGetBomData(
     in string stationNr,
     in string workorder,
     in string serialNumber,
     in string materialNo,
     in long bomVersion,
     in string bomIndex,
     in long processVersion,
     in long processBased,
     in long processType,
     in long alternative,
     in long bomType,
     out MasterDataBomArray masterDataBom,
     out string errorString)
raises (ServerException);

struct MasterDataBom
     {
        string partNo;
        string partDesc;
        string secPartNo;
        string compName;
        long alternative;
        double quantity;
        string unit;
        long isProduct;
        long bomVersion;
        string bomInfo;
        long processType;
        string processGrp;
        long processLayer;
        long traceFlag;
```

```
            string processInfo;
            long workstepNo;
        };
    typedef sequence<MasterDataBom> MasterDataBomArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
workorder	Production order number ***Advice:*** *If the value [-1] is passed, the parameter is not evaluated; in this case a serial number or part number must absolutely be passed (see* `materialNo` *or* `serialNumber`*)!*
serialNumber	Serial no. of the panel ***Advice:*** *If the value [-1] is passed, the parameter is not evaluated; in this case, an order number or part number must absolutely be passed (see* `workorder` *or* `materialNo`*)!*
materialNo	Part no. in iTAC.MES.Suite ***Advice:*** *If the value [-1] is passed, the parameter is not evaluated; in this case, an order number or serial number must absolutely be passed (see* `workorder` *or* `serialNumber`*)! If the part number is used, the BOM version and the process version must absolutely be passed!*
bomVersion	Bill-of-material version from iTAC.MES.Suite ***Special case:*** *If [-1] is passed, the most current version is used automatically. If a* `workorder` *or* `serialNumber` *is passed, this parameter is not evaluated!*
bomIndex	Bill-of-material version (index) that is passed to the system by means of changes in BOMs; these are manual changes or changes that have been initiated by higher-level ERP systems ***Special case:*** *If [-1] is passed, the most current version is used automatically!*
processVersion	Process version of the BOM from iTAC.MES.Suite ***Special case:*** *If [-1] is passed, the most current version is used automatically. If a* `workorder` *or* `serialNumber` *is passed, this parameter is not evaluated!*
processBased	Controls the output with respect to the process [0; 1]: **0** = General BOM **1** = Process-related BOM
processType	Selection of a use type [10300 - 10304]: **10300** = Mounting **10301** = Visual inspection **10302** = Component preparation **10303** = Provision by customer **10304** = Provision to external
alternative	Controls the output for the BOM components [0; 1]: **0** = Without alternatives **1** = With alternatives
bomType	Controls the output for the type of BOM [0; 1]: **0** = Scheduling BOM **1** = Construction BOM

TAB. 3-244 *Parameter (in) »mdataGetBomData«*

Parameter name	Explanation
masterDataBom	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
partNo	Part no. in iTAC.MES.Suite (component)
partDesc	Part description in iTAC.MES.Suite (component)
secPartNo	Customer-specific part no. (customer mat. no.)
compName	The mounting place conforms to the iTAC.MES.Suite BOM
alternative	Designator for regular or alternative component in the BOM [0; 1]: **0** = Regular component **1** = Alternative component
quantity	Quantity of the part ***Advice:*** *Decimal values are rounded to the nearest whole number!*
unit	Unit for the value passed
isProduct	Designator for raw material / product [0; 1]: **0** = Raw material (component) **1** = Product (unit)
bomVersion	BOM version from iTAC.MES.Suite
bomInfo	Output of the stored BOM information
processType	Output of the use type [10300 - 10304]: **10300** = Mounting **10301** = Visual inspection **10302** = Component preparation **10303** = Provision by customer **10304** = Provision to external
processGrp	Machine group, on which the mounting was carried out
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
traceFlag	Designator for the setup requirement [0; 1]: **0** = No setup required **1** = Setup required
processInfo	Output of the stored process information
workstepNo	Work step no. in iTAC.MES.Suite
errorString	Output text according to output value (see table)

TAB. 3-245 *Parameter (out) »mdataGetBomData«*

Output values

Value	Explanations (errorString)
11	Warning: No Data found
0	OK: [empty]
-3	Error: station not found
-39	Error: serialnumber not found
-67	Error: no valid bom version
-68	Error: no valid process version
-81	Error: workorder not found
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-362	Error: Partnumber not found
-419	Error: `processBased` not valid [0; 1]
-420	Error: `alternative` not valid [0; 1]
-421	Error: `bomType` not valid [0; 1]

TAB. 3-246 *Output values for »mdataGetBomData«*

3.82 mdcCreateLog

This function creates a log book entry (type "Log") for the passed station. This can be used to store individual pieces of information in the system in the context of machine data collection (MDC).

ADVICE

Log book entries can be viewed in myiTAC workplace with the "Log book" MDC function. Further information on this topic can be found in the documentation for the PM Module!

Function declaration (CORBA)

```
long mdcCreateLog(
     in string stationNo,
     in long startDate,
     in long endDate,
     in string subject,
     in string text,
     in long forLine,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNo	Station no. of the work station in iTAC.MES.Suite
startDate	Start of the validity period for the entry; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20* ***Special case:*** *[-1] passes the current date and the current time!*
endDate	End of the validity period for the entry; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock ***Special case:*** *[-1] passes 31.12.3000, 23:59:59 o'clock as the end time!*
subject	Subject line for the log book entry (max. 30 characters)
text	Text of the log book entry (max. 250 characters)
forLine	Entry for station or line [0; 1]: **0** = Entry only applies for the passed station **1** = Entry applies for all stations of the line

TAB. 3-247 *Parameter (in) »mdcCreateLog«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-248 *Parameter (out) »mdcCreateLog«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-79	Error: no PM license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-249 *Output values for »mdcCreateLog«*

3.83 mdcGetConditionCodes

This function returns all usable machine conditions and machine messages for the passed station.

ADVICE

The assignment of machine conditions/messages to a station is performed in myiTAC workplace with the "Machine setup" function. Further information on this topic can be found in the documentation for the ADM Module!

Function declaration (CORBA)

```
long mdcGetConditionCodes(
     in string stationNo,
     in long type,
     out MdcConditionCodeArray conditionCodeArray,
     out string errorString)
raises (ServerException);

struct MdcConditionCode
     {
        string name;
        string code;
        string level4;
        string colorRGB;
        string type;
     };
typedef sequence<MdcConditionCode> MdcConditionCodeArray;
```

Parameter (in)

Parameter name	Explanation
stationNo	Station no. of the work station in iTAC.MES.Suite
type	Specifies the machine message type [0; 1; 2]: **0** = Machine conditions and machine messages **1** = Machine conditions only **2** = Machine messages only

TAB. 3-250 *Parameter (in) »mdcGetConditionCodes«*

Parameter (out)

Parameter name	Explanation
conditionCodeArray	Array with the following variables: ***Advice:** The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
name	Name of the machine condition or of the machine message
code	Unique designator of the machine condition or of the machine message
level4	Operating condition description (Level 4) acc. to SEMI E10 (**S**emiconductor **E**quipment and **M**aterials **I**nternational) ***Advice:** This parameter is only output for machine conditions!*

TAB. 3-251 *Parameter (out) »mdcGetConditionCodes«*

Parameter name	Explanation
colorRGB	Color definition (RGB) for the entry; the individual color values are separated by ";": e.g. 000;255;255;000 ***Advice:*** *In addition to the RGB color value definitions, the alpha value (transparency) is defined with the 4th entry!*
type	Machine information type [1; 2]: **1** = Machine condition **2** = Machine message
errorString	Output text according to output value (see table)

TAB. 3-251 *Parameter (out) »mdcGetConditionCodes«*

Output values

Value	Explanations (errorString)
11	Warning: No Data found
0	OK: [empty]
-3	Error: station not found
-79	Error: no PM license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-252 *Output values for »mdcGetConditionCodes«*

3.84 mdcGetLog

This function returns all booked log book entries (type "Log") for the passed station within a specific review period

ADVICE

Log book entries can also be viewed in myiTAC workplace with the "Log book" MDC function. Further information on this topic can be found in the documentation for the PM Module!

Function declaration (CORBA)

```
long mdcGetLog(
     in string stationNo,
     in long fromDate,
     in long toDate,
     out MdcLogArray logArray,
     out string errorString)
raises (ServerException);

struct MdcLog
     {
        long startDate;
        long endDate;
        string subject;
        long txt;
     };
typedef sequence<MdcLog> MdcLogArray;
```

Parameter (in)

Parameter name	Explanation
stationNo	Station no. of the work station in iTAC.MES.Suite
fromDate	Start of the review period for the log book entries; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20* ***Special case:*** *[-1] passes 01.01.1970, 0:00:00 o'clock as the start time!*
toDate	End of the review period for the log book entries; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock ***Special case:*** *[-1] passes 31.12.3000, 23:59:59 o'clock as the end time!*

TAB. 3-253 *Parameter (in) »mdcGetLog«*

Parameter (out)

Parameter name	Explanation
logArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
startDate	Start of the validity period for the log book entry; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock

TAB. 3-254 *Parameter (out) »mdcGetLog«*

Parameter name	Explanation
endDate	End of the validity period for the log book entry; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock
subject	Subject line for the log book entry
txt	Text of the log book entry
errorString	Output text according to output value (see table)

TAB. 3-254 *Parameter (out) »mdcGetLog«*

Output values

Value	Explanations (errorString)
11	Warning: No Data found
0	OK: [empty]
-3	Error: station not found
-79	Error: no PM license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-255 *Output values for »mdcGetLog«*

3.85 mdcGetStationConditions

This function returns all booked machine conditions for the passed station within a specific review period.

ADVICE

Booked machine conditions can also be viewed in myiTAC workplace with the "Analysis/capture" MDC function. Further information on this topic can be found in the documentation for the PM Module!

Function declaration (CORBA)

```
long mdcGetStationConditions(
     in string stationNo,
     in long fromDate,
     in long toDate,
     out MdcConditionArray conditionArray,
     out string errorString)
raises (ServerException);

struct MdcCondition
     {
        string colorRGB;
        string level4;
        string conditionCode;
        string conditionDesc;
        string conditionName;
        long startDate;
        long toDate;
        string txt;
     };
typedef sequence<MdcCondition> MdcConditionArray;
```

Parameter (in)

Parameter name	Explanation
stationNo	Station no. of the work station in iTAC.MES.Suite
fromDate	Start of the review period; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20* ***Special case:*** *[-1] passes 01.01.1970, 0:00:00 o'clock as start time!*
toDate	End of the review period; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock ***Special case:*** *[-1] passes 31.12.3000, 23:59:59 o'clock as the end time!*

TAB. 3-256 *Parameter (in) »mdcGetStationConditions«*

Parameter (out)

Parameter name	Explanation
conditionArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
colorRGB	Color definition (RGB) for the entry; the individual color values are separated by ";": e.g. 000;255;255;000 ***Advice:*** *In addition to the RGB color value definitions, the alpha value (transparency) is defined with the 4th entry!*
level4	Operating condition description (Level 4) acc. to SEMI E10 (**S**emiconductor **E**quipment and **M**aterials **I**nternational)
conditionCode	Unique designator of the machine condition
conditionDesc	Description of the machine condition
conditioName	Name of the machine condition
startDate	Start of the validity period; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock
endDate	End of the validity period; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock
txt	Text for the machine condition
errorString	Output text according to output value (see table)

TAB. 3-257 *Parameter (out) »mdcGetStationConditions«*

Output values

Value	Explanations (errorString)
11	Warning: No Data found
0	OK: [empty]
-3	Error: station not found
-79	Error: No PM license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-258 *Output values for »mdcGetStationConditions«*

3.86 mdcGetStationMessages

This function returns all booked machine messages for the passed station within a specific review period.

ADVICE

Booked machine messages can also be viewed in myiTAC workplace with the "Analysis/capture" MDC function. Further information on this topic can be found in the documentation for the PM Module!

Function declaration (CORBA)

```
long mdcGetStationMessages(
     in string stationNo,
     in long fromDate,
     in long toDate,
     out MdcMessageArray messageArray,
     out string errorString)
raises (ServerException);

struct MdcMessage
     {
        string colorRGB;
        string conditionCode;
        string conditionDesc;
        string conditionName;
        long startDate;
        long endDate;
        string txt;
     };
typedef sequence<MdcMessage> MdcMessageArray;
```

Parameter (in)

Parameter name	Explanation
stationNo	Station no. of the work station in iTAC.MES.Suite
fromDate	Start of the review period; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20* ***Special case:*** *[-1] passes 01.01.1970, 0:00:00 o'clock as start time!*
toDate	End of the review period; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock ***Special case:*** *[-1] passes 31.12.3000, 23:59:59 o'clock as the end time!*

TAB. 3-259 *Parameter (in) »mdcGetStationMessages«*

Parameter (out)

Parameter name	Explanation
messageArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
colorRGB	Color definition (RGB) for the entry; the individual color values are separated by ";": e.g. 000;255;255;000 ***Advice:*** *In addition to the RGB color value definitions, the alpha value (transparency) is defined with the 4th entry!*
conditionCode	Unique designator of the machine message
conditionDesc	Description of the machine message
conditioName	Name of the machine message
startDate	Start of the validity period; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock
endDate	End of the validity period; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock
txt	Text for the machine condition
errorString	Output text according to output value (see table)

TAB. 3-260 *Parameter (out) »mdcGetStationMessages«*

Output values

Value	Explanations (errorString)
11	Warning: No Data found
0	OK: [empty]
-3	Error: station not found
-79	Error: no PM license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-261 *Output values for »mdcGetStationMessages«*

3.87 mdcUploadStationCondition

This function enables the booking of machine conditions or machine messages.

ADVICE

Booked machine conditions or messages can be viewed in myiTAC workplace with the "Analysis/capture" MDC function. Further information on this topic can be found in the documentation for the PM Module!

Because only one machine condition can be active on a station at a given point in time, the machine conditions that lie in the validity period passed here are overwritten. This does not apply for machine messages, as multiple machine messages may be active at a given point in time on a station.

Application type "already active machine condition": If the validity ranges of the passed machine condition overlap with an active machine condition, the active entry is overwritten. Depending on the setting for the `startDate` and `endDate` parameters, the validity range is shortened accordingly. Depending on where the validity ranges overlap, the validity range of the existing entry ends earlier (on the `startDate` of the new entry) or the validity range of the existing entry begins later (on the `endDate` of the new entry).

Function declaration (CORBA)

```
long mdcUploadStationCondition(
     in string stationNo,
     in string conditionCode,
     in long startDate,
     in long endDate,
     in string text,
     in long bookingTarget,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNo	Station no. of the work station in iTAC.MES.Suite
conditionCode	Unique designator of the machine condition or of the machine message
startDate	Start of the validity period for the entry; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20* ***Special case:*** *[-1] passes the current date and current time; for the application type "already active machine condition", see description further above.*
endDate	End of the validity period for the entry; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock ***Special case:*** *[-1] passes 31.12.3000, 23:59:59 o'clock as the end time; for the application type "already active machine condition", see description further above.*
text	Text for machine condition or for machine message (max. 200 characters)
bookingTarget	Entry for station or line [0; 1]: **0** = Entry only applies for the passed station **1** = Entry applies for all stations of the line

TAB. 3-262 *Parameter (in) »mdcUploadStationCondition«*

Parameter (out)

Parameter name	Explanation
`errorString`	Output text according to output value (see table)

TAB. 3-263 *Parameter (out) »mdcUploadStationCondition«*

Output values

Value	Explanations (errorString)
`0`	OK: [empty]
`-3`	Error: station not found
`-79`	Error: no PM license for this station
`-99`	Error: Server error
`-100`	Error: Corba no Service
`-101`	Error: Corba Exception
`-410`	Error: Given condition is not mapped to the given station ***Advice:*** *The assignment of machine conditions and machine messages to stations is performed in the machine setup in the ADM Module!*
`-411`	Error: Given condition is not a line condition ***Advice:*** *The assignment of machine conditions and machine messages to lines is performed in the machine setup in the ADM Module!*
`-412`	Error: Condition not found
`-413`	Error: Update of station condition is not allowed

TAB. 3-264 *Output values for »mdcUploadStationCondition«*

3.88 mergeBatch

This function installs the units which are contained in a material bin and which are not yet serialized either in a parent serial number (product) or in a different bin of a parent work order. Independent of the application type, a change in integration level (work order change) must always be performed when executing this function.

Application type "from bin in serial number": If the `duplicateSerialNumber` parameter has the value **0**, only the reference serial nos. of two products are merged. If the value of the `duplicateSerialNumber` parameter is **1**, every sub-number (single panel) is merged with a master according to a specified order. Take special notice here that scrapped serial numbers of a multiple panel can only be merged with another product if server parameter code no.: 2032 is deactivated. Reason: If the parameter is active, all scrapped serial numbers are automatically deactivated, making product merging impossible.

ADVICE

This API function is influenced by station parameter code nos.: 7206 and 7208. In addition, the "Work order generation" station setting ("Setup" tab) must be activated. Further information on this topic can be found in the documentation for the ADM Module! In the context of bin-based traceability, the »assignBatchNoToWorkorder«, »getRegisteredBatch«, »registerBatch«, »splitBatchNoToSerialNumber« and »unregisterBatch« functions must also be taken into account!

Function declaration (CORBA)

```
long mergeBatch(
     in string stationNr,
     in string serialNumberRef,
     in string serialNumberPos,
     in string batchNumber,
     in long processLayer,
     in double usedBatchQuantity,
     in long duplicateSerialNumber,
     in long ignoreBatchComplete,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
serialNumberRef	• **"from bin in serial number"**: Serial number in which a non-serialized unit is to be installed; serial numbers from multiple panels can also be used here in connection with the `serialNumberPos` and `duplicateSerialNumber` parameters. • **"from bin in bin"**: Number of the material bin into which the unit is to be inserted (number of units corresponds to the `usedBatchQuantity` parameter)
serialNumberPos	Position of the single panel in the unit-array ***'Advice:** If the value [-1] is passed, the position is not evaluated; in this case, each single panel must be specified by means of a unique* `serialNumberRef`*!* ***Advice:** In the "from bin in bin" application type, a "1" must be passed here!*
batchNumber	Number of the material bin (container no.) ***Advice:** The non-serialized units are removed from this bin!*

TAB. 3-265 *Parameter (in) »mergeBatch«*

Parameter name	Explanation
`processLayer`	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
`usedBatchQuantity`	• **"from bin in serial number"**: Quantity by which the bin quantity is reduced on each function call (normally "1"). ***Special case:*** *If the value [-1] is passed here, the quantity is ascertained from the BOM and, if applicable, calculated using the number of panels (*`duplicateSerialNumber` *= 1)!* • **"from bin in bin"**: Number of units which are to be taken over into the new bin
`duplicateSerialNumber`	Referencing to a single panel [0; 1]: **0** = Action is valid for the specified serial no. only (a position of the multiple printed panel) **1** = Action is carried out on all stored single panels, based on the reference serial number provided ***Advice:*** *In the "from bin in bin" application type, this parameter is not evaluated!*
`ignoreBatchComplete`	Bin check [0; 1]: **0** = Incomplete material bins cannot be used **1** = Incomplete material bins can be used

TAB. 3-265 *Parameter (in) »mergeBatch«*

Parameter (out)

Parameter name	Explanation
`errorString`	Output text according to output value (see table)

TAB. 3-266 *Parameter (out) »mergeBatch«*

Output values

Value	Explanations (errorString)
`0`	OK: [empty]
`-3`	Error: station not found
`-11`	Error: merge is invalid
`-12`	Error: Slave is not finished
`-17`	Error: serialnumber is locked
`-39`	Error: serialnumber not found
`-50`	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
`-78`	Error: no TR license for this station
`-99`	Error: Server error
`-100`	Error: Corba no Service
`-101`	Error: Corba Exception
`-202`	Error: container not found
`-358`	Error: materialBin registered at an other station
`-359`	Error: materialBin not completed at an other station
`-361`	Error: Container not registered at station

TAB. 3-267 *Output values for »mergeBatch«*

Value	Explanations (errorString)
-370	Error: serialNumber is not a serialNumberRef
-374	Error: SerialNo FAIL on last workstep
-375	Error: SerialNo SCRAP on last workstep
-376	Error: Master Serialnumber not found
-377	Error: Slave serialnumber not found
-380	Error: Different quantities Master/Slave
-381	Error: more than one valid SerialNo for merge
-382	Error: Container and serialnumber belongs to an other workorder
-383	Error: Container and serialnumber belongs to the same workorder

TAB. 3-267 *Output values for »mergeBatch«*

3.89 mergeParts

The function merges two products (merge slave product into master product). It is checked first whether the work progress is OK and the association of the product versions is permitted. If the value of the parameter `duplicateSerialNumber` is **0**, only the reference serial numbers of two products are merged.

If the value of the `duplicateSerialNumber` parameter is **1**, every sub-number (single panel) is merged with a master according to a specified order. Take special notice here that scrapped serial numbers of a multiple panel can only be merged with another product if server parameter code no.: 2032 is deactivated. Reason: If the parameter is active, all scrapped serial numbers are automatically deactivated, making product merging impossible.

ADVICE

The `mergeParts` API function is influenced by the "API_Check_merge_to_Stkl" (code no.: 7206) and "API_IGNORE_SNR_UNIQUE_FOR_MERGE J/N" (code no.: 7208) station parameters. In addition, the "Work order generation" station setting ("Setup" tab) must be activated. Further information on this topic can be found in the documentation for the ADM Module!

Function declaration (CORBA)

```
long mergeParts(
     in string stationNr,
     in long processLayer,
     in long duplicateSerialNumber,
     in string serialNumberMaster,
     in string serialNumberSlave,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
duplicateSerialNumber	Referencing to a single panel [0; 1]: **0** = Action is valid for the specified serial no. only (a position of the multiple printed panel) **1** = Action is carried out on all stored single panels, based on the reference serial number provided
serialNumberMaster	Serial no. which is retained during a product merging (see `serialNumberSlave`)
serialNumberSlave	Serial no. which is deleted during a product merging (see `serialNumberMaster`)

TAB. 3-268 *Parameter (in) »mergeParts«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-269 *Parameter (out) »mergeParts«*

Output values

Value	Explanations (errorString)
2	OK: … but serialNumber slave is SCRAP at last workstep
1	OK: … but serialNumber slave is FAIL at last workstep
0	OK: [empty]
-1	Error: [empty]
-2	Error: merge is not valid
-3	Error: station not found
-5	Error: Slave serialnumber not found
-6	Error: amount of master and slave-serialNumbers is not equal
-7	Error: serialNumber slave is not finished
-17	Error: Serialnumber Slave is locked
-18	Error: Serialnumber Master is locked
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-270 *Output values for »mergeParts«*

3.90 queryRecipeData

This function returns values for machine specifications.

ADVICE

The `queryRecipeData` *API function is influenced by the "Create level for new failure type" station parameter (code no.: 15201). This parameter is used to specify which level – i.e. violation of which limit value – is to be applied in the creation of failure types. This in turn influences the values that are output here. Depending on the setting, deviations for the intervention and scrap limits can be output in addition to the deviations for min. and max. (default). Further information on this topic can be found in the User's Guide.*

Function declaration (CORBA)

```
long queryRecipeData(
     in string stationNr,
     in string workOrder,
     inout long numberOfRecords,
     inout RecipeDataArray recipeDataArray,
     out string errorString)
raises (ServerException);

struct RecipeData
     {
        string name;
        string value;
     };
typedef sequence<RecipeData> RecipeDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
workOrder	Number of a production order

TAB. 3-271 *Parameter (in) »queryRecipeData«*

Parameter (inout)

Parameter name	Explanation
numberOfRecords	**in**: Default value for the number of entries in the array ***Special case:*** *To ascertain the identification number of the recipe, 1 must be passed!* Positive values correspond to the number of RecipeData returned. These must be identified via the name in the `RecipeDataArray`. For this, `numberOfRecords` must receive a positive input value. Two types of calls are possible: • **Return of the entire recipe** if `numberOfRecords` = 0, all RecipeData is returned. • **Search for specific parameters** if `numberOfRecords` > 0, only the RecipeData that match the default value (`name`) and the value passed for `numberOfRecords` are returned. **out**: Number of entries in the array returned
recipeDataArray	Array with the following variables:

TAB. 3-272 *Parameter (inout) »queryRecipeData«*

Parameter name	Explanation
name	**in**: Concrete specification for the output values (optional; otherwise [empty]); the specification enables the output of specific machine specifications (recipes) ***Special case:*** *To ascertain the identification number of the recipe,* ***$itac.rc_id*** *must be passed!* **out**: Name of the machine specification
value	**in**: [empty] - value is not evaluated **out**: Value for machine specification ***Special case:*** *If the additional function "Return of the* `recipeId`*" is active, the identification number is output here!*

TAB. 3-272 *Parameter (inout) »queryRecipeData«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to the output value or [empty] for fault-free execution of the function

TAB. 3-273 *Parameter (out) »queryRecipeData«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-2	Error: wrong station
-3	Error: wrong workOrder
-4	Error: wrong process
-5	Error: no recipe-data available
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-274 *Output values for »queryRecipeData«*

Additional function "Return of the `recipeId`" - This function may also be used to determine the identification number (`recipeId`) of the current recipe. To do this, "1" must be passed for the **`numberOfRecords`** variable and the value "$itac.rc_id" for the **`name`** variable. The `recipeID` is required, for example, as input value for the »activateRecipe« API functions.

3.91 queryTestData

This function makes measurement data available.

ADVICE

The `queryTestData` *API function is influenced by station parameter code no.: 7209. Further information on this topic can be found in the documentation for the ADM Module!*

Function declaration (CORBA)

```
long queryTestData(
     in string stationNr,
     in long processLayer,
     in string serialNumber,
     inout long numberOfRecords,
     inout ResultDataArray resultDataArray,
     out long loopCounter,
     out float cycleTime,
     out string errorString)
raises (ServerException);

struct ResultData
     {
        string name;
        string value;
        long failCode;
     };
typedef sequence<ResultData> ResultDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
serialNumber	Serial no. of the panel

TAB. 3-275 *Parameter (in) »queryTestData«*

Parameter (inout)

Parameter name	Explanation
numberOfRecords	**in**: Positive values correspond to the number of ResultData returned. These must be identified via the name in the `ResultDataArray`. If `numberOfRecords` = **-1**, all ResultData with error code **>0** are returned. **out**: Number of entries in the array returned
resultDataArray	Array with the following variables:
`name`	**in:** Concrete specification for the output values (optional; otherwise [empty]); the specification enables the output of specific measurement steps (ResultData) **out**: Measurement step name
`value`	**in**: [empty] - value is not evaluated **out**: Measured value

TAB. 3-276 *Parameter (inout) »queryTestData«*

Parameter name	Explanation
failCode	**in**: [empty] - value is not evaluated **out**: State information for measurement step [0; 1]: **0** = OK **1** = FAIL ***Advice:*** *Other integer values can be interpreted independent of the station configuration either as pseudo failure (code no.: 15250) or without evaluation (code no.: 15260)!*

TAB. 3-276 *Parameter (inout) »queryTestData«*

Parameter (out)

Parameter name	Explanation
loopCounter	Number of bookings against this serial no. and this work step (stations of the associated ERP group) ***Advice:*** *Only bookings of stations of the type "tester" are counted; bookings of repair and diagnosis stations are **not** counted!*
cycleTime	Duration of the check in seconds
errorString	Output text according to output value (see table)

TAB. 3-277 *Parameter (out) »queryTestData«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: serialNumber not found
-2	Error: wrong station
-3	Error: no data found for serialNumber
-4	Error: [empty]
-6	Error: station is not valid for this serialNumber
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-278 *Output values for »queryTestData«*

3.92 registerBatch

This function registers a material bin at a station. From this point in time until unregistering occurs (»unregisterBatch«), the setup data at the station are assigned to the registered material bin. Unlike continuously serialized manufacturing, this condition is responsible for the larger gray area during tracking. Here, different containers for a given raw material can be rigged during the bin registration and unregistration periods. It is then no longer possible to uniquely assign the units contained in the bins.

During registration, a check is performed to determine whether the material bin was completely processed by all previous stations. Use the `ignoreBatchComplete` parameter to individually decide whether incompletely processed material bins may be used.

ADVICE

In the context of bin-based traceability, the »assignBatchNoToWorkorder«, »getRegisteredBatch«, »mergeBatch«, »splitBatchNoToSerialNumber« and »unregisterBatch« functions must also be taken into account!

Function declaration (CORBA)

```
long registerBatch(
     in string stationNr,
     in string batchNumber,
     in long processLayer,
     in long ignoreBatchComplete,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
batchNumber	Number of the material bin (container no.)
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
ignoreBatchComplete	Bin check [0; 1]: **0** = Incomplete material bins cannot be used **1** = Incomplete material bins can be used

TAB. 3-279 *Parameter (in) »registerBatch«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-280 *Parameter (out) »registerBatch«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:** Value is only returned, if the station parameter code no. 8300 is active!*

TAB. 3-281 *Output values for »registerBatch«*

Value	Explanations (errorString)
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-202	Error: container not found
-205	Error: No workstep found for workorder, layer and station
-213	Error: Container state not valid
-357	Error: materialBin already registered
-358	Error: materialBin registered at an other station
-359	Error: materialBin not completed at an other station
-360	Error: No workorder active at station

TAB. 3-281 *Output values for »registerBatch«*

3.93 registerUser

This function registers a user at the station.

ADVICE

A user is unregistered using the »unregisterUser« API function. With each state booking, users registered in this way are booked by name in the system (key word "personalized machine booking"). In the unit history in the TR Module, the user registered at the time of booking can be identified (see field "user name")!

Function declaration (CORBA)

```
long registerUser(
     in string stationNr,
     in string userName,
     in string password,
     in string client,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
userName	Login name of the user (user identification)
password	Password of the user
client	Abbreviation for the client

TAB. 3-282 *Parameter (in) »registerUser«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-283 *Parameter (out) »registerUser«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: user or password not found
-3	Error: station not found
-4	Error: other user already registered to this station
-5	Error: this user already registered to this station
-6	Error: user not unique
-51	Error: password is invalid
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-284 *Output values for »registerUser«*

3.94 registerUserAtLine

This function logs a user onto or off of a line (type "M") and thereby enables staff-related time acquisition. These staff work times can be evaluated later in the PM Module with regard to line utilization.

Function declaration (CORBA)

```
long registerUserAtLine(
     in string stationNr,
     in string userId,
     in string password,
     in long registrationType,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
userId	Login name of the user (user identification)
password	Password of the user
registrationType	Login / logoff [0; 1]: **0** = User is logged off **1** = User is logged in

TAB. 3-285 *Parameter (in) »registerUserAtLine«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-286 *Parameter (out) »registerUserAtLine«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-79	Error: no PM license for this station
-98	Error: unknown user
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-416	Error: This team has no permission for line
-417	Error: Invalid registration type (start 1, stop 0)
-437	Error: Line not found

TAB. 3-287 *Output values for »registerUserAtLine«*

3.95 removeAttributeFromMaterialBin

This function deletes an attribute for a container number.

ADVICE

In the context of container attributes, the »appendAttributesToMaterialBin«, »createAttribute«, »getAttributesForMaterialBin« and »getMaterialBinsForAttribute« functions must also be taken into account!

Function declaration (CORBA)

```
long removeAttributeFromMaterialBin(
     in string stationNr,
     in string materialBinNr,
     in string attributeCode,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
materialBinNr	Number of the material bin (container no.)
attributeCode	Attribute code

TAB. 3-288 *Parameter (in) »removeAttributeFromMaterialBin«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-289 *Parameter (out) »removeAttributeFromMaterialBin«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-76	Error: attribute code must be at least one character
-78	Error: no TR license for this station
-92	Error: attribute code does not exist
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-402	Error: no attributes found for material bin
-434	Error: material bin was not found

TAB. 3-290 *Output values for »removeAttributeFromMaterialBin«*

3.96 removeAttributeFromSerialNumber

This function deletes an attribute for a serial number.

ADVICE

In the context of serial number attributes, the »appendAttributeToSerialNumber«, »createAttribute«, »getAttributesForSerialNumber« and »getSerialNumbersForAttribute« functions must also be taken into account!

Function declaration (CORBA)

```
long removeAttributeFromSerialNumber(
     in string stationNr,
     in string serialNumber,
     in string attributeCode,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
serialNumber	Serial no. of the panel
attributeCode	Attribute code

TAB. 3-291 *Parameter (in) »removeAttributeFromSerialNumber«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-292 *Parameter (out) »removeAttributeFromSerialNumber«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-39	Error: serialnumber not found
-72	Error: attribute not found for serialnumber
-76	Error: attribute code must be at least one character
-78	Error: no TR license for this station
-92	Error: attribute code does not exist
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-293 *Output values for »removeAttributeFromSerialNumber«*

3.97 removeAttributeFromWorkorder

This function deletes a specific attribute for a work order. If the order number is not known, a serial number can be used instead to ascertain the work order.

ADVICE

In the context of work order attributes, the »appendAttributeToWorkorder«, »createAttribute«, »getAttributesForWorkorder« and »getWorkordersForAttribute« functions must also be taken into account!

Function declaration (CORBA)

```
long removeAttributeFromWorkorder(
     in string stationNr,
     in string chargeExt,
     in string serialNumber,
     in string attributeCode,
     in string objectKey,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
chargeExt	Production order number ***Advice:*** *If the value [-1] is passed, the parameter is not evaluated; in this case, it is absolutely necessary that a serial number be passed (see* `serialNumber`*)!*
serialNumber	Serial no. of the panel ***Advice:*** *If the value [-1] is passed, the parameter is not evaluated; in this case, it is absolutely necessary that an order number be passed (see* `chargeExt`*)!*
attributeCode	Attribute code
objectKey	Key term for the attribute ***Special case:*** *If the value [-1] is passed, all entries are deleted independent of the* `objectKey`*!*

TAB. 3-294 *Parameter (in) »removeAttributeFromWorkorder«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-295 *Parameter (out) »removeAttributeFromWorkorder«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-76	Error: attribute code must be at least one character
-78	Error: no TR license for this station
-81	Error: workorder not found
-92	Error: attribute code does not exist

TAB. 3-296 *Output values for »removeAttributeFromWorkorder«*

Value	Explanations (errorString)
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-218	Error: attribute not found for workorder

TAB. 3-296 *Output values for »removeAttributeFromWorkorder«*

3.98 removeEquipmentForWorkorder

This function removes all resources rigged on a station for a work order; resources for other work orders remain rigged.

ADVICE

In the context of resource management, the »checkEquipmentData«, »getRequiredEquipmentData«, »getSetupEquipmentData« and »updateEquipmentData« functions must also be taken into account!

Function declaration (CORBA)

```
long removeEquipmentForWorkorder(
     in string stationNr,
     in string workOrder,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
workOrder	Number of a production order ***Special case:*** *If the value [-1] is passed here, the work order which is active on the station is used!*

TAB. 3-297 *Parameter (in) »removeEquipmentForWorkorder«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-298 *Parameter (out) »removeEquipmentForWorkorder«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-310	Error: Station is not part of given workorder

TAB. 3-299 *Output values for »removeEquipmentForWorkorder«*

3.99 removeMergeParts

This function takes the serial number passed to it and removes it from the master product. This requires that the serial number is built into another product as a slave, i.e., that a product merger has taken place in the past. In standard mode (see advice), the original slave serial no. is reactivated after a successful unmerge.

ADVICE

The **removeMergeParts** *API function is influenced by the "API_UNMERGEPARTS deactivate (M)aster, (S)lave or (N)othing" station parameter (code no.: 7207). Further information on this topic can be found in the documentation for the ADM Module!*

Function declaration (CORBA)

```
long removeMergeParts(
     in string stationNr,
     in long processLayer,
     in string serialNumberSlave,
     in string textInfo,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
serialNumberSlave	Serial no. that is deleted if the products are merged
textInfo	Compulsory field: Information regarding the removal from the assembly (e.g. name, reason for fault)

TAB. 3-300 *Parameter (in) »removeMergeParts«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-301 *Parameter (out) »removeMergeParts«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-2	Error: unmerge is not valid
-3	Error: station not found
-5	Error: no workOrder found for serialNumber
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-302 *Output values for »removeMergeParts«*

3.100 reportRmQuantity

This functions books the quantities passed into PM Module. The specification of the station no., store, and order no. determines the work step for which the acknowledgment is to take place.

ADVICE

For this function, the station must be of type "M"!

Function declaration (CORBA)

```
long reportRmQuantity(
     in string stationNr,
     in long processLayer,
     in string workOrder,
     in double passQty,
     in double failQty,
     in double scrapQty,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
workOrder	Number of a production order
passQty	Number of "pass" bookings
failQty	Number for fail
scrapQty	Number for scrap

TAB. 3-303 *Parameter (in) »reportRmQuantity«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-304 *Parameter (out) »reportRmQuantity«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: workorder not found
-2	Error: stationNr not found
-3	Error: quantity not valid
-4	Error: no data fund for workOrder, stationNr
-5	Error: could not report quantity data to RM
-79	Error: no PM license for this station

TAB. 3-305 *Output values for »reportRmQuantity«*

Value	Explanations (errorString)
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-305 *Output values for »reportRmQuantity«*

3.101 setMaterialBinLocation

This function places the passed material bin at a specific storage; if the material bin is already at a different storage, it is moved.

Function declaration (CORBA)

```
long setMaterialBinLocation(
     in string stationNr,
     in string materialBinNr,
     in string binLocation,
     in string binLocationBarcode,
     in long transactionType,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
materialBinNr	Number of the material bin (container no.)
binLocation	Storage location no. in iTAC.MES.Suite
binLocationBarcode	Bar code for the storage; must not coincide with the storage no. in iTAC.MES.Suite
transactionType	Code for the movement type from iTAC.MES.Suite (see movement types ML Module)

TAB. 3-306 *Parameter (in) »setMaterialBinLocation«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-307 *Parameter (out) »setMaterialBinLocation«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-2	Error: stationNr not found
-3	Error: materialBinNr not found
-4	Error: binLocation not found
-6	Error: `transactionType` not valid
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-308 *Output values for »setMaterialBinLocation«*

3.102 setMaterialBinState

This function changes the state of the material container passed.

Function declaration (CORBA)

```
long setMaterialBinState(
     in string stationNr,
     in string materialBinNr,
     in string state,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
materialBinNr	Number of the material bin (container no.)
state	Container state [B; E; Q, R; S]: **B** = LOCKED (Container locked) **E** = FINISHED (Container finished) **Q** = QA LOCKED (container QA locked) **R** = READY (Container ready) **S** = IN PROCESS (Container in process) ***Note:*** *If the state **E** is passed, the bin is set to "empty", i.e., the residual quantity of the container is set to 0. Moreover, the container is removed from the current storage location, which in turn is the prerequisite for the creation of a new container with the same container no.! If the state **B** or **Q** is passed, the text "State changed by API" is automatically entered in the lock history!*

TAB. 3-309 *Parameter (in) »setMaterialBinState«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-310 *Parameter (out) »setMaterialBinState«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: materialBinNr not found
-2	Error: `State` not valid [B; E; Q; R; S]
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-311 *Output values for »setMaterialBinState«*

3.103 setProductionCycleTime

This function enables an order-based adjustment of the default value "TE_M" (time requirement of the work step for the station).

Function declaration (CORBA)

```
long setProductionCycleTime(
     in string stationNo,
     in string workorder,
     in long cycleTime,
     in long processLayer,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNo	Station no. of the work station in iTAC.MES.Suite ***Advice:*** *The affected work step is ascertained on the basis of the passed station!*
workorder	Number of a production order ***Special case:*** *If [-1] is passed, the work order which is active on the station is used!*
cycleTime	Time requirement of the work step for the station in seconds (TE_M) ***Advice:*** *This value only applies for the current work order!*
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent

TAB. 3-312 *Parameter (in) »setProductionCycleTime«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-313 *Parameter (out) »setProductionCycleTime«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-79	Error: no PM license for this station
-81	Error: workorder not found
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-308	Error: workplan not found

TAB. 3-314 *Output values for »setProductionCycleTime«*

3.104 setSetupCycleTime

This function enables an order-based adjustment of the default value "TR_M" (setup time of the station for the work step).

Function declaration (CORBA)

```
long setSetupCycleTime(
     in string stationNo,
     in string workorder,
     in long cycleTime,
     in long processLayer,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNo	Station no. of the work station in iTAC.MES.Suite ***Advice:*** *The affected work step is ascertained on the basis of the passed station!*
workorder	Number of a production order ***Special case:*** *If [-1] is passed, the work order which is active on the station is used!*
cycleTime	Setup time of the station for the work step in seconds (TE_R) ***Advice:*** *This value only applies for the current work order!*
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent

TAB. 3-315 *Parameter (in) »setSetupCycleTime«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-316 *Parameter (out) »setSetupCycleTime«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-79	Error: no PM license for this station
-81	Error: workorder not found
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-308	Error: workplan not found

TAB. 3-317 *Output values for »setSetupCycleTime«*

3.105 setupActivation

Depending on the parameter `activateFlag` – this function permits the deletion, activation, or deactivation of the setup at a station.

ADVICE

This API function is influenced by station parameter code no.: 13120. Further information on this topic can be found in the documentation for the ADM Module!

Function declaration (CORBA)

```
long setupActivation(
     in string stationNr,
     in long processLayer,
     in string workorder,
     in long activateFlag,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2; 3]: **0** = Component side **1** = Solder side **2** = Position independent **3** = Double-sided panel ***Advice:*** *If the value [3] is passed, the setup data for the component and solder side are combined and considered as a whole!*
workorder	Number of a production order (required for the "activate setup" mode only)
activateFlag	Change "setup" mode [0; 1; 2]: **0** = Activate setup **1** = Deactivate setup **2** = Delete setup

TAB. 3-318 *Parameter (in) »setupActivation«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-319 *Parameter (out) »setupActivation«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: general Error
-2	Error: stationNr not found
-3	Error: no setup found for station
-4	Error: workorder not found
-5	Error: no active workorder found for station
-6	Error: `activateFlag` not valid [0; 1; 2]

TAB. 3-320 *Output values for »setupActivation«*

Value	Explanations (errorString)
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-320 *Output values for »setupActivation«*

3.106 setWorkOrderFromSerialNumber

The function changes the order at a station to the order of the serial number passed. The most recent order of the serial no. is used.

Function declaration (CORBA)

```
long setWorkOrderFromSerialNumber(
      in string stationNr,
      in string serialNumber,
      out string partNr,
      out string partDesc,
      out long bomVersion,
      out string bomIndex,
      out string cadPartNr,
      out long processLayer,
      out string workOrder,
      out long quantity,
      out string state,
      out string customerName,
      out string customerPartNr,
      out string attribut1,
      out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
serialNumber	Serial no. of the panel

TAB. 3-321 *Parameter (in) »setWorkOrderFromSerialNumber«*

Parameter (out)

Parameter name	Explanation
partNr	Part no. in iTAC.MES.Suite
partDesc	Part description in iTAC.MES.Suite
bomVersion	Bill-of-material version from iTAC.MES.Suite
bomIndex	Bill-of-material version (index) that is passed to the system by means of changes in BOMs; these are manual changes or changes that have been initiated by higher-level ERP systems
cadPartNr	Part no. of the fundamental bearer, e.g., number of the "bare" printed circuit board
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
workOrder	Number of a production order
quantity	Quantity of production order

TAB. 3-322 *Parameter (out) »setWorkOrderFromSerialNumber«*

Parameter name	Explanation
state	State of the production order [N; A; I; F; R; S; P; D; E]: **N** = new **A** = planned **I** = initialized **F** = created **R** = released **S** = started **P** = stopped **D** = deleted **E** = finished
customerName	Customer name in iTAC.MES.Suite
customerPartNr	Customer material no. in iTAC.MES.Suite
attribut1	Specific parameter from the part master (e.g., **H** for High, **L** for Low)
errorString	Output text according to output value (see table)

TAB. 3-322 *Parameter (out) »setWorkOrderFromSerialNumber«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-2	Error: wrong station
-3	Error: no workorder found for SerialNr
-4	Error: wrong process
-5	Error: no recipe-data available
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-323 *Output values for »setWorkOrderFromSerialNumber«*

3.107 shipActivateShippingLotAtKap

This function is used to assign lots to a station (shipping station). A lot can only be assigned to one shipping station at any one time; a plausibility check is carried out before the assignment.

Function declaration (CORBA)

```
long shipActivateShippingLotAtKap(
     in string stationNr,
     in string lotNr,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
lotNr	Identification no. of the lot

TAB. 3-324 *Parameter (in) »shipActivateShippingLotAtKap«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-325 *Parameter (out) »shipActivateShippingLotAtKap«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: lot not assigned
-2	Error: lotNr not found
-3	Error: station not found
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-326 *Output values for »shipActivateShippingLotAtKap«*

3.108 shipAddChildLotToParentLot

This function packs a lot (childLot) into another lot (parentLot).

Function declaration (CORBA)

```
long shipAddChildLotToParentLot(
     in string stationNr,
     in string childLot,
     in string parentLot,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
childLot	Identification no. of the lot ***Advice:*** *A* `childLot` *is the lot that is or has been packed into a* `parentLot`*!*
parentLot	Identification no. of the lot ***Advice:*** *A* `parentLot` *is the lot into which* `childLots` *are or have been packed!*

TAB. 3-327 *Parameter (in) »shipAddChildLotToParentLot«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-328 *Parameter (out) »shipAddChildLotToParentLot«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-12	Error: parentlot is not configured for this lot
-21	Error: child lot not found
-22	Error: addLotNr not unique
-24	Error: child lot is already assigned to this lot
-25	Error: child lot is already assigned to another lot
-34	Error: child lot was not completed
-41	Error: child lot not found
-48	Error: parent lot is already completed
-49	Error: parent lot is already finished
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-329 *Output values for »shipAddChildLotToParentLot«*

3.109 shipAddSnrToShippingLot

This function is used to assign the serial no. of a product to a lot at a shipping station; before the assignment is carried out, the validity of the product and of the serial no. are checked.

ADVICE

*By default, the system is configured such that only **one** product in one BOM version (pure with identical product version) may be packed into a lot. However, it is possible to reconfigure the system to permit different BOM versions of a product to be packed into one lot. The packing of different products into one lot (unpure) is not currently supported!*

Function declaration (CORBA)

```
long shipAddSnrToShippingLot(
     in string stationNr,
     in string lotNr,
     in string serialNumber,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
lotNr	Identification no. of the lot ***Advice:** If `-1` is entered, the lot (identification no.) that is currently assigned to the station is used automatically!*
serialNumber	Serial no. of the panel

TAB. 3-330 *Parameter (in) »shipAddSnrToShippingLot«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-331 *Parameter (out) »shipAddSnrToShippingLot«*

Output values

Value	Explanations (errorString)
2	OK: serialnumber is assigned to lot and it is the last serialnumber for the lot
1	OK: serialnumber is assigned to lot and the first serialnumber has changed the lot status and also the qty., can be changed with individual lotsize
0	OK: serialnumber is assigned to lot
-1	Error: serialnumber not found
-2	Error: serialnumber not unique
-3	Error: station not found
-4	Error: serialnumber is already assigned to this lot
-5	Error: serialnumber is assigned to another lot
-9	Error: serialnumber has not yet been booked at the previous station
-10	Error: serialnumber not valid
-11	Error: lot not found

TAB. 3-332 *Output values for »shipAddSnrToShippingLot«*

Value	Explanations (errorString)
-12	Error: PartNo is not configured for this lot
-13	Error: PartNo is not same than the other serialNumbers
-14	Error: BomVersion is not same than the other serialNumbers
-15	Error: lotQuantity already reached
-16	Error: serialNumber not finished
-17	Error: serialNumber not pass
-30	Error: lot not registered at this station
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-332 *Output values for »shipAddSnrToShippingLot«*

3.110 shipCheckSnrAddToShippingLot

This function checks whether a serial no. may be added to a lot

ADVICE

*By default, the system is configured such that only **one** product in one BOM version (pure with identical product version) may be packed into a lot. However, it is possible to reconfigure the system to permit different BOM versions of a product to be packed into one lot. The packing of different products into one lot (unpure) is not currently supported!*

Function declaration (CORBA)

```
long shipCheckSnrAddToShippingLot(
     in string stationNr,
     in string lotNr,
     in string serialNumber,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
lotNr	Identification no. of the lot ***Advice:** If `-1` is entered, the lot (identification no.) that is currently assigned to the station is used automatically!*
serialNumber	Serial no. of the panel

TAB. 3-333 *Parameter (in) »shipCheckSnrAddToShippingLot«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-334 *Parameter (out) »shipCheckSnrAddToShippingLot«*

Output values

Value	Explanations (errorString)
2	OK: serialnumber can be assigned to lot and it will be the last serialnumber for the lot
1	OK: serialnumber can be assigned to lot and will be the first serialnumber in the lot
0	OK: serialnumber can be assigned to lot
-1	Error: serialnumber not found
-2	Error: serialnumber not unique
-3	Error: serialnumber found but not active
-4	Error: serialnumber is already assigned to this lot
-5	Error: serialnumber is assigned to another lot
-10	Error: serialnumber not valid
-11	Error: lot not found
-12	Error: PartNo is not configured for this lot
-13	Error: PartNo is not same than the other serialNumbers

TAB. 3-335 *Output values for »shipCheckSnrAddToShippingLot«*

Value	Explanations (errorString)
-14	Error: BomVersion is not same than the other serialNumbers
-15	Error: lotQuantity already reached
-16	Error: serialNumber not finished
-17	Error: serialNumber not pass
-18	Error: lot is already completed, no add is allowed
-19	Error: lot is already finished, no add is allowed
-30	Error: lot not registered at this station
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-335 *Output values for »shipCheckSnrAddToShippingLot«*

3.111 shipCheckSnrFromShippingLot

This function is used to check whether a certain serial no. is assigned to a certain lot and whether the assignment is still valid. A previously valid assignment could, for example, subsequently become invalid if the state of the serial no. changes from "Pass" to "Scrap" as the result of a test.

Function declaration (CORBA)

```
long shipCheckSnrFromShippingLot(
     in string stationNr,
     in string lotNr,
     in string serialNumber,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
lotNr	Identification no. of the lot ***Advice:*** *If* **-1** *is entered, the lot (identification no.) that is currently assigned to the station is used automatically!*
serialNumber	Serial no. of the panel

TAB. 3-336 *Parameter (in) »shipCheckSnrFromShippingLot«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-337 *Parameter (out) »shipCheckSnrFromShippingLot«*

Output values

Value	Explanations (errorString)
0	OK: serialnumber is assigned to lot
-1	Error: serialnumber not found
-2	Error: serialnumber not unique
-3	Error: serialnumber found but not active
-4	Error: serialnumber not assigned to a lot
-5	Error: serialnumber is assigned to another lot
-10	Error: serialnumber not valid
-11	Error: lot not found
-12	Error: PartNo is not configured for this lot
-13	Error: PartNo is not same than the other serialNumbers
-14	Error: BomVersion is not same than the other serialNumbers
-15	Error: lotQuantity already reached
-16	Error: serialNumber not finished
-17	Error: serialNumber not pass
-18	Error: stationNr not found
-30	Error: lot not registered at this station
-78	Error: no TR license for this station

TAB. 3-338 *Output values for »shipCheckSnrFromShippingLot«*

Value	Explanations (errorString)
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-338 *Output values for »shipCheckSnrFromShippingLot«*

3.112 shipCompleteLot

This function specifies the system behavior in the case of a lot whose maximum fill level has not been reached when it is closed.

Function declaration (CORBA)

```
long shipCompleteLot(
     in string stationNr,
     in string lotNr,
     in long allowLessQty,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
lotNr	Identification no. of the lot ***Advice:*** *If* `-1` *is entered, the lot (identification no.) that is currently assigned to the station is used automatically!*
allowLessQty	max. fill level for lot [0; 1]: **0** = If the maximum fill level is not reached, an error message is output **1** = Even if the maximum fill level was not reached, lots can be closed (State: complete). The system stores the information that this lot contains less than its maximum fill level despite having the state "complete".

TAB. 3-339 *Parameter (in) »shipCompleteLot«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-340 *Parameter (out) »shipCompleteLot«*

Output values

Value	Explanations (errorString)
0	OK: lot completed
-11	Error: lot not found
-18	Error: stationNr not found
-19	Error: lotNr not found
-21	Error: lot has less quantity ***Advice:*** *Only if input* `allowLessQty` = `0`
-22	Error: lot was not updated
-23	Error: lot is already finished
-24	Error: lot is already completed
-25	Error: lot not of type shipping
-30	Error: lot not registered at this station
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-341 *Output values for »shipCompleteLot«*

3.113 shipDeactivateShippingLotAtKap

This function cancels the association between lot and the station (shipping station) (see chapter 3.107 »shipActivateShippingLotAtKap«).

Function declaration (CORBA)

```
long shipDeactivateShippingLotAtKap(
     in string stationNr,
     in string lotNr,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
lotNr	Identification no. of the lot ***Advice:*** *If* **-1** *is entered, the lot (identification no.) that is currently assigned to the station is used automatically!*

TAB. 3-342 *Parameter (in) »shipDeactivateShippingLotAtKap«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-343 *Parameter (out) »shipDeactivateShippingLotAtKap«*

Output values

Value	Explanations (errorString)
0	OK: lot separated successfully
-1	Error: lot not separated
-2	Error: lotNr not found
-3	Error: station not found
-30	Error: lot not registered at this station
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-344 *Output values for »shipDeactivateShippingLotAtKap«*

3.114 shipGetChildLotsForParentLot

This function returns all lots (ChildLots) that are packed into a parent lot (ParentLot).

Function declaration (CORBA)

```
long shipGetChildLotsForParentLot(
     in string stationNr,
     in string parentLot,
     out long numberOfRecords,
     out ShippingLotArray shippingLots,
     out string errorString)
raises (ServerException);

struct ShippingLotStruct
     {
        string partNo;
        string partDesc;
        long meh;
        string custCode;
        string customerName;
        string extCustCode;
        long startDate;
        long endDate;
        string snrMatExt;
        double meTotal;
        double meRest;
        string beaStatus;
        string artBezPart;
        string artPart;
     };
typedef sequence<ShippingLotStruct> ShippingLotArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
parentLot	Identification no. of the lot ***Advice:*** *A* `parentLot` *is the lot into which* `childLots` *are or have been packed!*

TAB. 3-345 *Parameter (in) »shipGetChildLotsForParentLot«*

Parameter (out)

Parameter name	Explanation
numberOfRecords	Number of entries in the array returned
shippingLots	Array with the following variables:
`partNo`	Part no. in iTAC.MES.Suite ***Advice:*** *In this case, this is the part no. of the lot!*
`partDesc`	Part description in iTAC.MES.Suite ***Advice:*** *In this case, this is the part description of the lot!*

TAB. 3-346 *Parameter (out) »shipGetChildLotsForParentLot«*

Parameter name	Explanation
meh	Designator for the quantity unit in the lot: **11000** = (mm) millimeter **11001** = (") inch **11002** = (stk) piece **11003** = (cm) centimeter **11004** = (m) meter **11005** = (g) gram **11006** = (kg) kilogram **11007** = (ml) milliliter **11008** = (l) liter ***Advice:*** *If necessary, quantity units may be adapted and amended in the system (details upon request)!*
custCode	Designator of the customer
customerName	Customer name in iTAC.MES.Suite
extCustCode	Designator of the customer in an external system
startDate	Start date for the setup (start of production); the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
endDate	Last booking for the setup; the total time is specified as the number of seconds since 01-01-1970
snrMatExt	Identification no. in external system
meTotal	Maximum fill level lot
meRest	Current fill level lot
beaStatus	Current process state [B; C; E; F; L; Q; R; S]: **B** = LOCKED (Container locked) **C** = COMPLETE (container completed) **E** = FINISHED (Container finished) **F** = INIT (new container created) **L** = MRP LOCKED (Container not available for allocation) **Q** = QA LOCKED (Container with QA lock) **R** = READY (Container ready) **S** = IN PROCESS (Container in process)
artBezPart	Part description in iTAC.MES.Suite ***Advice:*** *Here, this is the part desc. of the associated serial no.*
artPart	Part no. in iTAC.MES.Suite ***Advice:*** *Here, this is the part no. of the associated serial no.*
errorString	Output text according to output value (see table)

TAB. 3-346 *Parameter (out) »shipGetChildLotsForParentLot«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-41	Error: child lot not found
-42	Error: parent lot not found
-78	Error: no TR license for this station

TAB. 3-347 *Output values for »shipGetChildLotsForParentLot«*

Value	Explanations (errorString)
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-347 *Output values for »shipGetChildLotsForParentLot«*

3.115 shipGetLotFromSerialNo

This function returns information about the lot containing the serial number passed.

ADVICE

Use the »shipGetSnrDataForShippingLot« function to output all serial numbers that are packaged in the lot returned here!

Function declaration (CORBA)

```
long shipGetLotFromSerialNo(
     in string stationNr,
     in string serialNumber,
     out ShippingLotStruct shippingLot,
     out string errorString)
raises (ServerException);

struct ShippingLotStruct
     {
        string partNo;
        string partDesc;
        long meh;
        string custCode;
        string customerName;
        string extCustCode;
        long startDate;
        long endDate;
        string snrMatExt;
        double meTotal;
        double meRest;
        string beaStatus;
        string artBezPart;
        string artPart;
     };
typedef sequence<ShippingLotStruct> ShippingLotArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
serialNumber	Serial no. of the panel

TAB. 3-348 *Parameter (in) »shipGetLotFromSerialNo«*

Parameter (out)

Parameter name	Explanation
shippingLot	Structure with the following variables:
partNo	Part no. in iTAC.MES.Suite ***Advice:** In this case, this is the part no. of the lot!*
partDesc	Part description in iTAC.MES.Suite ***Advice:** In this case, this is the part description of the lot!*

TAB. 3-349 *Parameter (out) »shipGetLotFromSerialNo«*

Parameter name	Explanation
meh	Designator for the quantity unit in the lot: **11000** = (mm) millimeter **11001** = (") inch **11002** = (stk) piece **11003** = (cm) centimeter **11004** = (m) meter **11005** = (g) gram **11006** = (kg) kilogram **11007** = (ml) milliliter **11008** = (l) liter ***Advice:*** *If necessary, quantity units may be adapted and amended in the system (details upon request)!*
custCode	Designator of the customer
customerName	Customer name in iTAC.MES.Suite
extCustCode	Designator of the customer in an external system
startDate	Start date for the setup (start of production); the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
endDate	Last booking for the setup; the total time is specified as the number of seconds since 01-01-1970
snrMatExt	Identification no. in external system
meTotal	Maximum fill level lot
meRest	Current fill level lot
beaStatus	Current process state [B; C; E; F; L; Q; R; S]: **B** = LOCKED (Container locked) **C** = COMPLETE (container completed) **E** = FINISHED (Container finished) **F** = INIT (new container created) **L** = MRP LOCKED (Container not available for allocation) **Q** = QA LOCKED (Container with QA lock) **R** = READY (Container ready) **S** = IN PROCESS (Container in process)
artBezPart	Part description in iTAC.MES.Suite ***Advice:*** *Here, this is the part desc. of the associated serial no.*
artPart	Part no. in iTAC.MES.Suite ***Advice:*** *Here, this is the part no. of the associated serial no.*
errorString	Output text according to output value (see table)

TAB. 3-349 *Parameter (out) »shipGetLotFromSerialNo«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-39	Error: serialnumber not found
-40	Error: serialnumber not found in lot
-41	Error: childLot not found
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-350 *Output values for »shipGetLotFromSerialNo«*

3.116 shipGetShippingLotForLotNo

This function returns detailed information about a lot.

Function declaration (CORBA)

```
long shipGetShippingLotForLotNo(
     in string stationNr,
     in string lotNr,
     out ShippingLotStruct shippingLot,
     out string errorString)
raises (ServerException);

struct ShippingLotStruct
     {
        string partNo;
        string partDesc;
        long meh;
        string custCode;
        string customerName;
        string extCustCode;
        long startDate;
        long endDate;
        string snrMatExt;
        double meTotal;
        double meRest;
        string beaStatus;
        string artBezPart;
        string artPart;
     };
typedef sequence<ShippingLotStruct> ShippingLotStruct;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
lotNr	Identification no. of the lot ***Advice:*** *If* `-1` *is entered, the lot (identification no.) that is currently assigned to the station is used automatically!*

TAB. 3-351 *Parameter (in) »shipGetShippingLotForLotNo«*

Parameter (out)

Parameter name	Explanation
shippingLot	Structure with the following variables:
partNo	Part no. in iTAC.MES.Suite ***Advice:*** *In this case, this is the part no. of the lot!*
partDesc	Part description in iTAC.MES.Suite ***Advice:*** *In this case, this is the part description of the lot!*

TAB. 3-352 *Parameter (out) »shipGetShippingLotForLotNo«*

Parameter name	Explanation
meh	Designator for the quantity unit in the lot: **11000** = (mm) millimeter **11001** = (") inch **11002** = (stk) piece **11003** = (cm) centimeter **11004** = (m) meter **11005** = (g) gram **11006** = (kg) kilogram **11007** = (ml) milliliter **11008** = (l) liter ***Advice:*** *If necessary, quantity units may be adapted and amended in the system (details upon request)!*
custCode	Designator of the customer
customerName	Customer name in iTAC.MES.Suite
extCustCode	Designator of the customer in an external system
startDate	Start date for the setup (start of production); the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
endDate	Last booking for the setup; the total time is specified as the number of seconds since 01-01-1970
snrMatExt	Identification no. in external system
meTotal	Maximum fill level lot
meRest	Current fill level lot
beaStatus	Current process state [B; C; E; F; L; Q; R; S]: **B** = LOCKED (Container locked) **C** = COMPLETE (container completed) **E** = FINISHED (Container finished) **F** = INIT (new container created) **L** = MRP LOCKED (Container not available for allocation) **Q** = QA LOCKED (Container with QA lock) **R** = READY (Container ready) **S** = IN PROCESS (Container in process)
artBezPart	Part description in iTAC.MES.Suite ***Advice:*** *Here, this is the part desc. of the associated serial no.*
artPart	Part no. in iTAC.MES.Suite ***Advice:*** *Here, this is the part no. of the associated serial no.*
errorString	Output text according to output value (see table)

TAB. 3-352 *Parameter (out) »shipGetShippingLotForLotNo«*

Output values

Value	Explanations (errorString)
1	Warning: lot no is not unique
0	OK: [empty]
-1	Error: no shipping lot for this lot no
-3	Error: station not found
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-353 *Output values for »shipGetShippingLotForLotNo«*

3.117 shipGetShippingLotsForPartNo

This function retrieves and returns lots that contain certain parts. When entering the part no. to search for, a part schema (part no. pattern) may be used instead. By using the wild card character "*****" (e.g., **123***), it is also possible to retrieve the lots of several parts in one step.

Function declaration (CORBA)

```
long shipGetShippingLotsForPartNo(
     in string stationNr,
     in string partNoPattern,
     in long onlyActive,
     out long numberOfRecords,
     out ShippingLotArray shippingLots,
     out string errorString)
raises (ServerException);

struct ShippingLotStruct
     {
        string partNo;
        string partDesc;
        long meh;
        string custCode;
        string customerName;
        string extCustCode;
        long startDate;
        long endDate;
        string snrMatExt;
        double meTotal;
        double meRest;
        string beaStatus;
        string artBezPart;
        string artPart;
     };
typedef sequence<ShippingLotStruct> ShippingLotArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
partNoPattern	Part schema (part no. pattern) for the input of multiple parts ***Advice:*** *The wild card character "*****" may be used in the input!*
onlyActive	Number of lot nos. to be returned for the part no. specified [0; 1]: **0** = Only those lot nos. are returned that neither have the state "complete" nor the state "finished" **1** = All lot nos. found are returned

TAB. 3-354 *Parameter (in) »shipGetShippingLotsForPartNo«*

Parameter (out)

Parameter name	Explanation
numberOfRecords	Number of entries in the array returned
shippingLots	Array with the following variables:
partNo	Part no. in iTAC.MES.Suite ***Advice:*** *In this case, this is the part no. of the lot!*
partDesc	Part description in iTAC.MES.Suite ***Advice:*** *In this case, this is the part description of the lot!*

TAB. 3-355 *Parameter (out) »shipGetShippingLotsForPartNo«*

Parameter name	Explanation
meh	Designator for the quantity unit in the lot: **11000** = (mm) millimeter **11001** = (") inch **11002** = (stk) piece **11003** = (cm) centimeter **11004** = (m) meter **11005** = (g) gram **11006** = (kg) kilogram **11007** = (ml) milliliter **11008** = (l) liter ***Advice:*** *If necessary, quantity units may be adapted and amended in the system (details upon request)!*
custCode	Designator of the customer
customerName	Customer name in iTAC.MES.Suite
extCustCode	Designator of the customer in an external system
startDate	Start date for the setup (start of production); the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
endDate	Last booking for the setup; the total time is specified as the number of seconds since 01-01-1970
snrMatExt	Identification no. in external system
meTotal	Maximum fill level lot
meRest	Current fill level lot
beaStatus	Current process state [B; C; E; F; L; Q; R; S]: **B** = LOCKED (Container locked) **C** = COMPLETE (container completed) **E** = FINISHED (Container finished) **F** = INIT (new container created) **L** = MRP LOCKED (Container not available for allocation) **Q** = QA LOCKED (Container with QA lock) **R** = READY (Container ready) **S** = IN PROCESS (Container in process)
artBezPart	Part description in iTAC.MES.Suite ***Advice:*** *Here, this is the part desc. of the associated serial no.*
artPart	Part no. in iTAC.MES.Suite ***Advice:*** *Here, this is the part no. of the associated serial no.*
errorString	Output text according to output value (see table)

TAB. 3-355 *Parameter (out) »shipGetShippingLotsForPartNo«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: lots not found
-3	Error: station not found
-78	Error: no TR license for this station

TAB. 3-356 *Output values for »shipGetShippingLotsForPartNo«*

Value	Explanations (errorString)
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-356 *Output values for »shipGetShippingLotsForPartNo«*

3.118 shipGetSnrDataForShippingLot

This function retrieves and returns the serial nos. of products that have been packed into a certain lot.

Function declaration (CORBA)

```
long shipGetSnrDataForShippingLot(
     in string stationNr,
     in string lotNr,
     out long numberOfRecords,
     out SerialNumberStructArray serialNrArray,
     out string errorString)
raises (ServerException);

struct SerialNumberStruct
     {
        string serialNumber;
     };
typedef sequence<SerialNumberStruct> SerialNumberStructArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
lotNr	Identification no. of the lot ***Advice:*** *If* **-1** *is entered, the lot (identification no.) that is currently assigned to the station is used automatically!*

TAB. 3-357 *Parameter (in) »shipGetSnrDataForShippingLot«*

Parameter (out)

Parameter name	Explanation
numberOfRecords	Number of entries in the array returned
serialNrArray	Array with the following variables:
serialNumber	Serial no. of the panel
errorString	Output text according to output value (see table)

TAB. 3-358 *Parameter (out) »shipGetSnrDataForShippingLot«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: serialnumber check with unknown result
-2	Error: lotNr not found
-3	Error: station not found
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-359 *Output values for »shipGetSnrDataForShippingLot«*

3.119 shipMoveAllChildLotsToNewParentLot

This function moves all lots from the lot currently active at the station to a new lot (newParentLot).

Function declaration (CORBA)

```
long shipMoveAllChildLotsToNewParentLot(
     in string stationNr,
     in string newParentLot,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
newParentLot	Identification no. of the new lot ***Advice:*** *A* `parentLot` *is the lot into which* `childLots` *are or have been packed!*

TAB. 3-360 *Parameter (in) »shipMoveAllChildLotsToNewParentLot«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-361 *Parameter (out) »shipMoveAllChildLotsToNewParentLot«*

Output values

Value	Explanations (errorString)
> 0	OK: [> 0] A value larger than 0 indicates the number of lots shifted successfully
-1	Error: [empty]
-3	Error: station not found
-18	Error: Lot is already completed, operation is not allowed
-19	Error: Lot is already finished, operation is not allowed
-42	Error: parent lot not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-362 *Output values for »shipMoveAllChildLotsToNewParentLot«*

3.120 shipMoveChildLotsFromLotToLot

This function moves the lots passed (childLots) from one lot (actParentLot) to another lot (newParentLot).

Function declaration (CORBA)

```
long shipMoveChildLotsFromLotToLot(
     in string stationNr,
     in long numberOfRecords,
     in StringStructArray childLots,
     in string actParentLot,
     in string newParentLot,
     out string errorString)
raises (ServerException);

struct StringStruct
     {
        string value;
     };
typedef sequence<StringStruct> StringStructArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
numberOfRecords	Default value for the number of entries in the array
childLots	Array with the following variables:
value	Identification no. of the lot ***Advice:*** *A* childLot *is the lot that is or has been packed into a* parentLot*!*
actParentLot	Identification no. of the current lot ***Advice:*** *A* parentLot *is the lot into which* childLots *are or have been packed!*
newParentLot	Identification no. of the new lot ***Advice:*** *A* parentLot *is the lot into which* childLots *are or have been packed!*

TAB. 3-363 *Parameter (in) »shipMoveChildLotsFromLotToLot«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-364 *Parameter (out) »shipMoveChildLotsFromLotToLot«*

Output values

Value	Explanations (errorString)
0	OK: ChildLot moved
-1	Error: [empty]
-3	Error: station not found
-18	Error: Lot is already completed, operation is not allowed
-19	Error: Lot is already finished, operation is not allowed
-41	Error: child lot not found
-42	Error: parent lot not found

TAB. 3-365 *Output values for »shipMoveChildLotsFromLotToLot«*

Value	Explanations (errorString)
-43	Error: new parent lot not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-365 *Output values for »shipMoveChildLotsFromLotToLot«*

3.121 shipMoveChildLotToNewParentLot

This function moves the lot passed (childLot) from the lot currently active at the station to a new lot (newParentLot).

Function declaration (CORBA)

```
long shipMoveChildLotToNewParentLot(
     in string stationNr,
     in string childLot,
     in string newParentLot,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
childLot	Identification no. of the lot ***Advice:*** *A* `childLot` *is the lot that is or has been packed into a* `parentLot`*!*
newParentLot	Identification no. of the new lot ***Advice:*** *A* `parentLot` *is the lot into which* `childLots` *are or have been packed!*

TAB. 3-366 *Parameter (in) »shipMoveChildLotToNewParentLot«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-367 *Parameter (out) »shipMoveChildLotToNewParentLot«*

Output values

Value	Explanations (errorString)
0	OK: ChildLot moved
-1	Error: [empty]
-3	Error: station not found
-18	Error: Lot is already completed, operation is not allowed
-19	Error: Lot is already finished, operation is not allowed
-41	Error: child lot not found
-42	Error: parent lot not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-368 *Output values for »shipMoveChildLotToNewParentLot«*

3.122 shipRemoveAllChildLotFromParentLot

This function removes all lots (`childLot`) from a lot (`parentLot`).

Function declaration (CORBA)

```
long shipRemoveAllChildLotFromParentLot(
     in string stationNr,
     in string parentLot,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
parentLot	Identification no. of the lot ***Advice:*** *A* `parentLot` *is the lot into which* `childLots` *are or have been packed!*

TAB. 3-369 *Parameter (in) »shipRemoveAllChildLotFromParentLot«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-370 *Parameter (out) »shipRemoveAllChildLotFromParentLot«*

Output values

Value	Explanations (errorString)
> 0	OK: int count of the childLots removed from the shippinglot
-1	Error: childLots were not removed
-3	Error: station not found
-18	Error: Lot is already completed, operation is not allowed
-19	Error: Lot is already finished, operation is not allowed
-42	Error: parent lot not found
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-371 *Output values for »shipRemoveAllChildLotFromParentLot«*

3.123 shipRemoveChildLotFromLot

This function removes a lot (childLot) from a lot (parentLot).

Function declaration (CORBA)

```
long shipRemoveChildLotFromLot(
     in string stationNr,
     in string childLot,
     in string parentLot,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
childLot	Identification no. of the lot ***Advice:*** *A* `childLot` *is the lot that is or has been packed into a* `parentLot`*!*
parentLot	Identification no. of the lot ***Advice:*** *A* `parentLot` *is the lot into which* `childLots` *are or have been packed!*

TAB. 3-372 *Parameter (in) »shipRemoveChildLotFromLot«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-373 *Parameter (out) »shipRemoveChildLotFromLot«*

Output values

Value	Explanations (errorString)
0	OK: childLot removed
-1	Error: [empty]
-3	Error: station not found
-19	Error: Lot is already finished, operation is not allowed
-41	Error: child lot not found
-42	Error: parent lot not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-374 *Output values for »shipRemoveChildLotFromLot«*

3.124 shipRemoveSnrFromShippingLot

This function is used to remove a specified serial no. from a specified lot; the fill level (number of serial nos. in the lot) is reduced accordingly. If the lot has a current fill level of "0", the state of the container is reset to "init" and is thus freely available again.

Function declaration (CORBA)

```
long shipRemoveSnrFromShippingLot(
     in string stationNr,
     in string lotNr,
     in string serialNumber,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
lotNr	Identification no. of the lot ***Advice:*** *If* **-1** *is entered, the lot (identification no.) that is currently assigned to the station is used automatically!*
serialNumber	Serial no. of the panel

TAB. 3-375 *Parameter (in) »shipRemoveSnrFromShippingLot«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-376 *Parameter (out) »shipRemoveSnrFromShippingLot«*

Output values

Value	Explanations (errorString)
> 0	OK: No. of remaining serial no. in the lot
0	OK: serialnumber removed
-1	Error: serialnumber not found
-2	Error: lotNr not found
-3	Error: station not found
-30	Error: lot not registered at this station
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-377 *Output values for »shipRemoveSnrFromShippingLot«*

3.125 shipReopenShippingLot

With this function, a lot that is already closed (state "COMPLETE") can be reopened for processing (state "READY"). For example, a serial no. could then be removed from the lot with the function *»shipRemoveSnrFromShippingLot«*.

Function declaration (CORBA)

```
long shipReopenShippingLot(
     in string stationNr,
     in string lotNr,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
lotNr	Identification no. of the lot ***Advice:*** *If* **-1** *is entered, the lot (identification no.) that is currently assigned to the station is used automatically!*

TAB. 3-378 *Parameter (in) »shipReopenShippingLot«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-379 *Parameter (out) »shipReopenShippingLot«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-11	Error: lot not found
-32	Error: lot was not reopened
-33	Error: lot is not completed
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-380 *Output values for »shipReopenShippingLot«*

3.126 shipReplaceShippingLot

This function completes a lot (state "finished"), and a new and empty lot (state "init") is created on the system with the same identification no. Detailed information is returned for the new lot.

The requirement for completing the lot is that the filling has finished ("complete" state) and that the lot is not registered with another station. However, the lot does not need to be registered with the current station.

Function declaration (CORBA)

```
long shipReplaceShippingLot(
     in string stationNr,
     in string lotNr,
     out ShippingLotStruct shippingLot,
     out string errorString)
raises (ServerException);

struct ShippingLotStruct
     {
        string partNo;
        string partDesc;
        long meh;
        string custCode;
        string customerName;
        string extCustCode;
        long startDate;
        long endDate;
        string snrMatExt;
        double meTotal;
        double meRest;
        string beaStatus;
        string artBezPart;
        string artPart;
     };
typedef sequence<ShippingLotStruct> ShippingLotStruct;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
lotNr	Identification no. of the lot ***Advice:*** *If* `-1` *is entered, the lot (identification no.) that is currently assigned to the station is used automatically!*

TAB. 3-381 *Parameter (in) »shipReplaceShippingLot«*

Parameter (out)

Parameter name	Explanation
shippingLot	Structure with the following variables:
partNo	Part no. in iTAC.MES.Suite ***Advice:*** *In this case, this is the part no. of the lot!*
partDesc	Part description in iTAC.MES.Suite ***Advice:*** *In this case, this is the part description of the lot!*

TAB. 3-382 *Parameter (out) »shipReplaceShippingLot«*

Parameter name	Explanation
meh	Designator for the quantity unit in the lot: **11000** = (mm) millimeter **11001** = (") inch **11002** = (stk) piece **11003** = (cm) centimeter **11004** = (m) meter **11005** = (g) gram **11006** = (kg) kilogram **11007** = (ml) milliliter **11008** = (l) liter ***Advice:*** *If necessary, quantity units may be adapted and amended in the system (details upon request)!*
custCode	Designator of the customer
customerName	Customer name in iTAC.MES.Suite
extCustCode	Designator of the customer in an external system
startDate	Start date for the setup (start of production); the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
endDate	Last booking for the setup; the total time is specified as the number of seconds since 01-01-1970
snrMatExt	Identification no. in external system
meTotal	Maximum fill level lot
meRest	Current fill level lot
beaStatus	Current process state [B; C; E; F; L; Q; R; S]: **B** = LOCKED (Container locked) **C** = COMPLETE (container completed) **E** = FINISHED (Container finished) **F** = INIT (new container created) **L** = MRP LOCKED (Container not available for allocation) **Q** = QA LOCKED (Container with QA lock) **R** = READY (Container ready) **S** = IN PROCESS (Container in process)
artBezPart	Part description in iTAC.MES.Suite ***Advice:*** *Here, this is the part desc. of the associated serial no.*
artPart	Part no. in iTAC.MES.Suite ***Advice:*** *Here, this is the part no. of the associated serial no.*
errorString	Output text according to output value (see table)

TAB. 3-382 *Parameter (out) »shipReplaceShippingLot«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-33	Error: lot is not completed
-37	Error: lot is activated at another station
-78	Error: no TR license for this station

TAB. 3-383 *Output values for »shipReplaceShippingLot«*

Value	Explanations (errorString)
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-383 *Output values for »shipReplaceShippingLot«*

3.127 smtConsumption

This function processes the consumption data from inserters and is used exclusively by the API-based "SMT Connector" service.

ADVICE

The SMT Connector includes various setup functions which are based on API functions. This (SMT) function calls the »uploadMaterialBinBookingsExt« API function in the background. In the context of the SMT Connector, the »smtEventSetup«, »smtSerialNumberSetup«, and »smtSetupPreparation« functions must also to be taken into account!

Function declaration (CORBA)

```
long smtConsumption(
     in string stationNr,
     in long processLayer,
     in string lineName,
     in string setupName,
     in string productName,
     in string bomVersion,
     in string placementRecipe,
     in boolean assignSerialnumbers,
     in boolean bookSerialnumbers,
     in boolean activateSetup,
     inout SmtSerialNoArray serialNumbers,
     inout SmtEventArray events,
     inout SmtPlacementArray placements,
     in boolean ignoreContainerProblems,
     in boolean createWorkorder,
     in boolean activateWorkorder,
     in string workOrder,
     in double cycleTime,
     out string errorString)
raises (ServerException);

struct SmtSerialNo
     {
        string serialNumber;
        long serialNumberPosition;
        long serialNumberState;
        long validationFlag;
        long processFlag;
     };
typedef sequence<SmtSerialNo> SmtSerialNoArray;

struct SmtEvent
     {
        long eventId;
        long eventDate;
        long validationFlag;
        long processFlag;
        string position;
        string feederBank;
        string feederId;
        long placementId;
        long qtyRemain;
        long qtyPlaced;
        long qtyWasted;
        string materialBinNr;
        boolean createMaterialBin;
        string partNo;
        double quantity;
```

```
        string supplier;
        string lotNo;
        string dateCode;
    };
typedef sequence<SmtEvent> SmtEventArray;

struct SmtPlacement
    {
        long placementId;
        string compName;
        string xPosition;
        string yPosition;
        long panelPosition;
    };
typedef sequence<SmtPlacement> SmtPlacementArray;
```

Parameter (in, inout)

The same structures are used for all SMT functions (»smtConsumption«, »smtEventSetup«, »smtSerialNumberSetup« and »smtSetupPreparation«). Each individual function only needs a partial quantity of the input parameters specified in the declaration for processing. A detailed description of the parameters is not provided here (for details, see documentation **SMT_Connector_EN.pdf**).

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-384 *Parameter (out) »smtConsumption«*

Output values

Value	Explanations (errorString)
7	Warning: Decrement amount of material to a negative number ***Advice:** Value is only returned if site parameter code no. 7210 was set to "2"; for "1", "-392" is returned!*
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:** Value is only returned, if the station parameter code no. 8300 is active!*
-62	Error: Product not found
-75	Error: a required parameter is missing
-78	Error: no TR license for this station
-81	Error: workorder not found
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-202	Error: container not found
-203	Error: product is not retrograde
-204	Error: Product belongs not to given material bin
-392	Error: Decrement amount of material to a negative number

TAB. 3-385 *Output values for »smtConsumption«*

3.128 smtEventSetup

This function processes setup events for inserters and is used exclusively by the API-based "SMT Connector" service.

ADVICE

The SMT Connector includes various setup functions which are based on API functions. Depending on the process, this (SMT) function calls the »getPlacementName«, »updateMaterialSetup«, »changeFeederBank« and »setupActivation« API functions in the background. In the context of the SMT Connector, the »smtConsumption«, »smtSerialNumberSetup«, and »smtSetupPreparation« functions must also to be taken into account!

Function declaration (CORBA)

```
long smtEventSetup(
      in string stationNr,
      in long processLayer,
      in string lineName,
      in string setupName,
      in string productName,
      in string bomVersion,
      in string placementRecipe,
      in boolean assignSerialnumbers,
      in boolean bookSerialnumbers,
      in boolean activateSetup,
      inout SmtSerialNoArray serialNumbers,
      inout SmtEventArray events,
      inout SmtPlacementArray placements,
      in boolean ignoreContainerProblems,
      in boolean createWorkorder,
      in boolean activateWorkorder,
      in string workOrder,
      in double cycleTime,
      out string errorString)
raises (ServerException);

struct SmtSerialNo
      {
         string serialNumber;
         long serialNumberPosition;
         long serialNumberState;
         long validationFlag;
         long processFlag;
      };
typedef sequence<SmtSerialNo> SmtSerialNoArray;

struct SmtEvent
      {
         long eventId;
         long eventDate;
         long validationFlag;
         long processFlag;
         string position;
         string feederBank;
         string feederId;
         long placementId;
         long qtyRemain;
         long qtyPlaced;
         long qtyWasted;
         string materialBinNr;
         boolean createMaterialBin;
         string partNo;
```

```
        double quantity;
        string supplier;
        string lotNo;
        string dateCode;
    };
typedef sequence<SmtEvent> SmtEventArray;

struct SmtPlacement
    {
        long placementId;
        string compName;
        string xPosition;
        string yPosition;
        long panelPosition;
    };
typedef sequence<SmtPlacement> SmtPlacementArray;
```

Parameter (in, inout)

The same structures are used for all (SMT) functions (»smtConsumption«, »smtEventSetup«, »smtSerialNumberSetup« and »smtSetupPreparation«). Each individual function only needs a partial quantity of the input parameters specified in the declaration for processing. A detailed description of the parameters is not provided here (for details, see documentation **SMT_Connector_EN.pdf**).

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-386 *Parameter (out) »smtEventSetup«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-387 *Output values for »smtEventSetup«*

3.129 smtSerialNumberSetup

This function processes serial number-related setup data and is used exclusively by the API-based "SMT Connector" service.

ADVICE

The SMT Connector includes various setup functions which are based on API functions.Depending on the process, this (SMT) function calls the »getNextBookingForStationAndLayer«, »getSerialnumberInfo«, »assignSerialNumberForProductOrWorkorder«, »getMaterialSetupDataForKapAndPeriod«, »getMaterialBinInfo«, »createNewMaterialBin«, »getPlacementName«, »setupActivation«, »updateSetupData«, »updateSetupDataHistory« and »uploadStateAndFailureData« API functions in the background. In the context of the SMT Connector, the »smtConsumption«, »smtEventSetup«, and »smtSetupPreparation« functions must also to be taken into account!

Function declaration (CORBA)

```
long smtSerialNumberSetup(
      in string stationNr,
      in long processLayer,
      in string lineName,
      in string setupName,
      in string productName,
      in string bomVersion,
      in string placementRecipe,
      in boolean assignSerialnumbers,
      in boolean bookSerialnumbers,
      in boolean activateSetup,
      inout SmtSerialNoArray serialNumbers,
      inout SmtEventArray events,
      inout SmtPlacementArray placements,
      in boolean ignoreContainerProblems,
      in boolean createWorkorder,
      in boolean activateWorkorder,
      in string workOrder,
      in double cycleTime,
      out string errorString)
raises (ServerException);

struct SmtSerialNo
      {
         string serialNumber;
         long serialNumberPosition;
         long serialNumberState;
         long validationFlag;
         long processFlag;
      };
typedef sequence<SmtSerialNo> SmtSerialNoArray;

struct SmtEvent
      {
         long eventId;
         long eventDate;
         long validationFlag;
         long processFlag;
         string position;
         string feederBank;
         string feederId;
         long placementId;
         long qtyRemain;
         long qtyPlaced;
         long qtyWasted;
```

```
            string materialBinNr;
            boolean createMaterialBin;
            string partNo;
            double quantity;
            string supplier;
            string lotNo;
            string dateCode;
         };
      typedef sequence<SmtEvent> SmtEventArray;

      struct SmtPlacement
         {
            long placementId;
            string compName;
            string xPosition;
            string yPosition;
            long panelPosition;
         };
      typedef sequence<SmtPlacement> SmtPlacementArray;
```

Parameter (in, inout)

The same structures are used for all SMT functions (»smtConsumption«, »smtEventSetup«, »smtSerialNumberSetup« and »smtSetupPreparation«). Each individual function only needs a partial quantity of the input parameters specified in the declaration for processing. A detailed description of the parameters is not provided here (for details, see documentation **SMT_Connector_EN.pdf**).

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-388 *Parameter (out) »smtSerialNumberSetup«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service

TAB. 3-389 *Output values for »smtSerialNumberSetup«*

3.130 smtSetupPreparation

This function processes setup preparation data for inserters and is used exclusively by the API-based "SMT Connector" service.

ADVICE

The SMT Connector includes various setup functions which are based on API functions. In the context of the SMT Connector, the »smtConsumption«, »smtEventSetup«, and »smtSerialNumberSetup« functions must also to be taken into account!

Function declaration (CORBA)

```
long smtSetupPreparation(
     in string stationNr,
     in long processLayer,
     in string lineName,
     in string setupName,
     in string productName,
     in string bomVersion,
     in string placementRecipe,
     in boolean assignSerialnumbers,
     in boolean bookSerialnumbers,
     in boolean activateSetup,
     inout SmtSerialNoArray serialNumbers,
     inout SmtEventArray events,
     inout SmtPlacementArray placements,
     in boolean ignoreContainerProblems,
     in boolean createWorkorder,
     in boolean activateWorkorder,
     in string workOrder,
     in double cycleTime,
     out string errorString)
raises (ServerException);

struct SmtSerialNo
     {
        string serialNumber;
        long serialNumberPosition;
        long serialNumberState;
        long validationFlag;
        long processFlag;
     };
typedef sequence<SmtSerialNo> SmtSerialNoArray;

struct SmtEvent
     {
        long eventId;
        long eventDate;
        long validationFlag;
        long processFlag;
        string position;
        string feederBank;
        string feederId;
        long placementId;
        long qtyRemain;
        long qtyPlaced;
        long qtyWasted;
        string materialBinNr;
        boolean createMaterialBin;
        string partNo;
        double quantity;
        string supplier;
```

```
        string lotNo;
        string dateCode;
    };
typedef sequence<SmtEvent> SmtEventArray;

struct SmtPlacement
    {
        long placementId;
        string compName;
        string xPosition;
        string yPosition;
        long panelPosition;
    };
typedef sequence<SmtPlacement> SmtPlacementArray;
```

Parameter (in, inout)

The same structures are used for all SMT functions (»smtConsumption«, »smtEventSetup«, »smtSerialNumberSetup« and »smtSetupPreparation«). Each individual function only needs a partial quantity of the input parameters specified in the declaration for processing. A detailed description of the parameters is not provided here (for details, see documentation **SMT_Connector_EN.pdf**).

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-390 *Parameter (out) »smtSetupPreparation«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-391 *Output values for »smtSetupPreparation«*

3.131 splitBatchNoToSerialNumber

This function copies the units which are contained in a material bin and which have not yet been serialized to serial numbers (1:1). Before doing this, the necessary serial numbers must first be created; these must match the product version of the bin »assignSerialNumberToWorkOrder«. Thus, it is possible to create n individual items from one bin. In doing so, all relevant information for the bin is transferred to the serial number.

Requirements for this are that the work order assigned to the bin (»assignBatchNoToWorkorder«) must correspond to the work order of the serialized part and that the used serial number must already be known.

ADVICE

In the context of bin-based traceability, the »assignBatchNoToWorkorder«, »getRegisteredBatch«, »mergeBatch«, »registerBatch« and »unregisterBatch« functions must also be taken into account!

Function declaration (CORBA)

```
long splitBatchNoToSerialNumber(
     in string stationNr,
     in string serialNumberRef,
     in string serialNumberPos,
     in string batchNumber,
     in long processLayer,
     in double usedBatchQuantity,
     in long duplicateSerialNumber,
     in long ignoreBatchComplete,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
serialNumberRef	Serial number into which a non-serialized unit is to be copied; in connection with the `serialNumberPos` and `duplicateSerialNumber` parameters, serial numbers of multiple panels can also be used here.
serialNumberPos	Position of the single panel in the unit-array ***'Advice:** If the value [-1] is passed, the position is not evaluated; in this case, each single panel must be specified by means of a unique* `serialNumberRef`*!*
batchNumber	Number of the material bin (container no.) ***Advice:** The non-serialized units are removed from this bin!*
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent

TAB. 3-392 *Parameter (in) »splitBatchNoToSerialNumber«*

Parameter name	Explanation
`usedBatchQuantity`	Quantity by which the bin quantity is reduced on each function call (normally "1"). ***Special case:*** *If the value [-1] is passed here, the quantity is ascertained from the BOM and, if applicable, calculated using the number of panels (`duplicateSerialNumber` = 1)!*
`duplicateSerialNumber`	Referencing to a single panel [0; 1]: **0** = Action is valid for the specified serial no. only (a position of the multiple printed panel) **1** = Action is carried out on all stored single panels, based on the reference serial number provided
`ignoreBatchComplete`	Bin check [0; 1]: **0** = Incomplete material bins cannot be used **1** = Incomplete material bins can be used

TAB. 3-392 *Parameter (in) »splitBatchNoToSerialNumber«*

Parameter (out)

Parameter name	Explanation
`errorString`	Output text according to output value (see table)

TAB. 3-393 *Parameter (out) »splitBatchNoToSerialNumber«*

Output values

Value	Explanations (errorString)
`0`	OK: [empty]
`-3`	Error: station not found
`-11`	Error: merge is invalid
`-12`	Error: Slave is not finished
`-17`	Error: serialnumber is locked
`-39`	Error: serialnumber not found
`-50`	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
`-78`	Error: no TR license for this station
`-99`	Error: Server error
`-100`	Error: Corba no Service
`-101`	Error: Corba Exception
`-202`	Error: container not found
`-358`	Error: materialBin registered at an other station
`-359`	Error: materialBin not completed at an other station
`-370`	Error: serialNumber is not a serialNumberRef
`-374`	Error: SerialNo FAIL on last workstep
`-375`	Error: SerialNo SCRAP on last workstep
`-376`	Error: Master Serialnumber not found
`-377`	Error: Slave serialnumber not found
`-380`	Error: Different quantities Master/Slave
`-381`	Error: more than one valid SerialNo for merge

TAB. 3-394 *Output values for »splitBatchNoToSerialNumber«*

Value	Explanations (errorString)
-382	Error: Container and serialnumber belongs to an other workorder
-383	Error: Container and serialnumber belongs to the same workorder

TAB. 3-394 *Output values for »splitBatchNoToSerialNumber«*

3.132 switchSerialNumber

This function resets the current serial number of a product. Before the serial number is changed, it is checked whether the switch is permitted.

ADVICE

The variable `duplicateSerialNumber` *is not available in the C++API!*

Function declaration (CORBA)

```
long switchSerialNumber(
     in string stationNr,
     in long processLayer,
     in long duplicateSerialNumber,
     in string serialNumberOld,
     in string serialNumberNew,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
duplicateSerialNumber	Referencing to a single panel [0; 1]: **0** = Action is valid for the specified serial no. only (a position of the multiple printed panel) **1** = Action is carried out on all stored single panels, based on the reference serial number provided
serialNumberOld	Serial no. that is replaced when the product serial no. is changed (see `serialNumberNew`)
serialNumberNew	Serial no. that is used when the product serial no. is changed (see `serialNumberOld`)

TAB. 3-395 *Parameter (in) »switchSerialNumber«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-396 *Parameter (out) »switchSerialNumber«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-2	Error: `SerialNumberNew` is already active
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-397 *Output values for »switchSerialNumber«*

3.133 switchSerialNumberbySnrRef

This function converts a serial number of a single panel into a multiple panel. The serial number for which the "switch" is to be performed is specified with the aid of the reference serial number and the panel position.

Function declaration (CORBA)

```
long switchSerialNumberBySnrRef(
     in string stationNr,
     in long processLayer,
     in string serialNumberRef,
     in string serialNumberPos,
     in string serialNumberNew,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
serialNumberRef	Serial no. of the first single panel or unit-array; this no. is used as reference serial no.
serialNumberPos	Position of the single panel in the unit-array
serialNumberNew	Serial no. that is used when the product serial no. is changed

TAB. 3-398 *Parameter (in) »switchSerialNumberbySnrRef«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-399 *Parameter (out) »switchSerialNumberbySnrRef«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-2	Error: SerialNumberNew is already active
-3	Error: station not found
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-400 *Output values for »switchSerialNumberbySnrRef«*

3.134 testPaa

This function carries out a PAA (Part Average Analysis) for the serial number passed, which permits an early detection of (sporadic) failures in the field. Here, the measurement data of the individual item are compared with those of other individual items, inspected for unusual features (anomalies), and a characteristic risk figure is calculated. As a result, this function shows whether or not the individual item is an outlier and carries a higher risk of failure.

ADVICE

This function is designed to post-test individual items already manufactured; if you wish to carry out the PAA directly in the production process, the »uploadResultDataAndRecipe« or »uploadStateAndResultData« API functions may be used for this purpose!
The **testPaa** *API function is influenced by station parameter code nos.: 7210 - 7216. Further information on this topic can be found in the documentation for the ADM Module!*

Function declaration (CORBA)

```
long testPaa(
     in string stationNr,
     in string serialNumber,
     in long processLayer,
     in long model,
     out long resultPaa,
     out string infoText,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
serialNumber	Serial no. of the panel
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
model	Determines, to which other serial numbers the passed serial number is to be compared to [0; 1; 2]: **0** = End of band (default); the measurement data of the serial number passed are only compared to the data of previously produced individual items **1** = Middle of band; the measurement data of the passed serial number are compared in equal parts to data of individual items produced previously and in future **2** = Band start (default); the measurement data of the serial number passed are only compared to the data of subsequently produced individual items ***Advice:*** *The number of individual items to be compared (PAA interval) is specified in the work plan!*

TAB. 3-401 *Parameter (in) »testPaa«*

Parameter (out)

Parameter name	Explanation
resultPaa	PAA result [0; 1]: **0** = OK **1** = Anomaly
infoText	Output of an information text
errorString	Output text according to output value (see table)

TAB. 3-402 *Parameter (out) »testPaa«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-16	Error: anomaly
-39	Error: serialnumber not found
-78	Error: no TR license for this station
-86	Error: PAA test not configured for station
-88	Error: PAA test not configured for workplan
-90	Error: PAA test not configured for recipe
-91	Error: Not enough values for testing
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-230	Error: Invalid value for input parameter: model

TAB. 3-403 *Output values for »testPaa«*

3.135 testSpa

This function carries out an SPA (Statistical Process Analysis) for the serial number passed (individual item), which permits an early detection of systematic failures in the field. The function calculates the process capability indices Cp and Cpk for the individual item and compares them to the values of other individual items. As a result, the function returns pre-interpreted Cp and Cpk values, hence permitting a classification as to whether or not the individual item is an outlier. The results of this calculation affect the overall PAA result.

ADVICE

This function is designed to post-test individual items already manufactured; if you wish to carry out the SPA in conjunction with the PAA directly in the production process, the »uploadResultDataAndRecipe« or »uploadStateAndResultData« API function may be used for this purpose!
The **testSpa** *API function is influenced by station parameter code nos.: 7210 - 7216 and 7250 - 7256. Further information on this topic can be found in the documentation for the ADM Module!*

Function declaration (CORBA)

```
long testSpa(
     in string stationNr,
     in string serialNumber,
     in long processLayer,
     in long model,
     out long resultCp,
     out long resultCpk,
     out string infoText,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
serialNumber	Serial no. of the panel
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
model	Determines, to which other serial numbers the passed serial number is to be compared to [0; 1; 2]: **0** = End of band (default); the measurement data of the serial number passed are only compared to the data of previously produced individual items **1** = Middle of band; the measurement data of the passed serial number are compared in equal parts to data of individual items produced previously and in future **2** = Band start (default); the measurement data of the serial number passed are only compared to the data of subsequently produced individual items ***Advice:*** *The number of individual items to be compared (PAA interval) is specified in the work plan!*

TAB. 3-404 *Parameter (in) »testSpa«*

Parameter (out)

Parameter name	Explanation
resultCp	Result of the Cp calculation [0; 1; 2]: **0** = OK (green) **1** = Conditional OK (yellow) **2** = Not OK (red)
resultCpk	Result of the Cpk calculation [0; 1; 2] **0** = OK (green) **1** = Conditional OK (yellow) **2** = Not OK (red)
infoText	Output of an information text
errorString	Output text according to output value (see table)

TAB. 3-405 *Parameter (out) »testSpa«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-39	Error: serialnumber not found
-78	Error: no TR license for this station
-85	Error: SPA test not configured for station
-87	Error: SPA test not configured for workplan
-89	Error: SPA test not configured for recipe
-91	Error: Not enough values for testing
-93	Error: CP violation
-94	Error: CPK violation
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-230	Error: Invalid value for input parameter: model

TAB. 3-406 *Output values for »testSpa«*

3.136 trGetLockedSerialNumbers

This function returns all locked serial numbers for a specific lock group and/or a work order for a review period. All search criteria are "AND" coupled such that only the serial numbers that meet all used filter conditions are returned.

Function declaration (CORBA)

```
long trGetLockedSerialNumbers(
     in string stationNo,
     in long dateFrom,
     in long dateTo,
     in string lockName,
     in string workOrder,
     out LockInformationArray lockInformation,
     out string errorString)
raises (ServerException);

struct LockInformation
     {
        string serialNumber;
        string serialNumberPos;
        string lockName;
        string workOrder;
        long serialNumberLockCount;
     };
typedef sequence<LockInformation> LockInformationArray;
```

Parameter (in)

Parameter name	Explanation
stationNo	Station no. of the work station in iTAC.MES.Suite
dateFrom	Start of the review period; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20* ***Special case:*** *If the values for* `dateFrom` *and* `dateTo` *are identical, the review period is not evaluated and there are, thus, no time restrictions!*
dateTo	End of the review period; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock ***Special case:*** *If the values for* `dateFrom` *and* `dateTo` *are identical, the review period is not evaluated and there are, thus, no time restrictions!*
lockName	Lock group ***Advice:*** *If an empty string is passed, the parameter is not evaluated; pay attention to upper- and lower-case letters!*
workOrder	Number of a production order ***Advice:*** *If an empty string is passed, the parameter is not evaluated!*

TAB. 3-407 *Parameter (in) »trGetLockedSerialNumbers«*

Parameter (out)

Parameter name	Explanation
lockInformation	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
serialNumber	Serial no. of the panel
serialNumberPos	Position of the single panel in the unit-array
lockName	Lock group
workOrder	Number of a production order
serialNumberLock-Count	Number of currently active locks for the serial number
errorString	Output text according to output value (see table)

TAB. 3-408 *Parameter (out) »trGetLockedSerialNumbers«*

Output values

Value	Explanations (errorString)
11	Warning: No Data found
0	OK: [empty]
-3	Error: station not found
-81	Error: workorder not found
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-431	Error: lock name not found

TAB. 3-409 *Output values for »trGetLockedSerialNumbers«*

3.137 trGetSerialNumberLockHistory

This function can output the lock history for a specific serial number or the affected serial numbers can be output with lock history for a review period, a specific lock group and/or a work order. All search criteria are "AND" coupled such that only the serial numbers that meet all used filter conditions are returned.

Function declaration (CORBA)

```
long trGetSerialNumberLockHistory(
     in string stationNo,
     in string serialNumber,
     in string serialNumberPos,
     in long dateFrom,
     in long dateto,
     in string lockName,
     in string workOrder,
     in long onlyLocked,
     out LockHistoryArray lockHistory,
     out string errorString)
raises (ServerException);

struct LockHistory
     {
        string serialNumber;
        string serialNumberPos;
        string lockName;
        string workOrder;
        long lockDate;
        string lockUser;
        string lockStation;
        string lockInfo;
        long unlockDate;
        string unlockUser;
        string unlockStation;
        string unlockInfo;
        long serialNumberLockCount;
     };
typedef sequence<LockHistory> LockHistoryArray;
```

Parameter (in)

Parameter name	Explanation
stationNo	Station no. of the work station in iTAC.MES.Suite
serialNumber	Serial no. of the panel ***Advice:*** *If an empty string is passed, the parameter is not evaluated; in this case a review period (see* `dateFrom` *and* `dateTo`*), work order (see* `workOrder`*) or lock group (see* `lockName`*) must absolutely be passed!*
serialNumberPos	Position of the single panel in the unit-array ***Advice:*** *If the value [-1] is passed, the position is not evaluated; in this case, each single panel must be specified by means of a unique* `serialNumber` *!*
dateFrom	Start of the review period; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20* ***Special case:*** *If the values for* `dateFrom` *and* `dateTo` *are identical, the review period is not evaluated; in this case a serial number (see* `serialNumber`*) must absolutely be passed!*

TAB. 3-410 *Parameter (in) »trGetSerialNumberLockHistory«*

Parameter name	Explanation
dateTo	End of the review period; specification of the total time in sec. since 01.01.1970, 0:00:00 o'clock ***Special case:** If the values for `dateFrom` and `dateTo` are identical, the review period is not evaluated; in this case a serial number (`serialNumber`) must absolutely be passed!*
lockName	Lock group ***Advice:** If an empty string is passed, the parameter is not evaluated; pay attention to upper- and lower-case letters!*
workOrder	Number of a production order ***Advice:** If an empty string is passed, the parameter is not evaluated!*
onlyLocked	Specifies which serial numbers are output [0; 1]: **1** = Only currently locked serial numbers are output **0** = Currently and previously locked serial numbers are output

TAB. 3-410 *Parameter (in) »trGetSerialNumberLockHistory«*

Parameter (out)

Parameter name	Explanation
lockHistory	Array with the following variables: ***Advice:** The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension **Size**!*
`serialNumber`	Serial no. of the panel
`serialNumberPos`	Position of the single panel in the unit-array
`lockName`	Lock group
`workOrder`	Number of a production order
`lockDate`	Date of locking
`lockUser`	Name of the user who locked the serial number
`lockStation`	Name of the station that locked the serial number (see API function *»trLockSerialNumbers«*)
`lockInfo`	Lock information (reason for locking)
`unlockDate`	Date of unlocking
`unlockUser`	Name of the user who unlocked the serial number
`unlockStation`	Name of the station that unlocked the serial number (see API function *»trUnlockSerialNumbers«*)
`unlockInfo`	Lock information (reason for unlocking)
`serialNumberLock-Count`	Number of currently active locks for the serial number
errorString	Output text according to output value (see table)

TAB. 3-411 *Parameter (out) »trGetSerialNumberLockHistory«*

Output values

Value	Explanations (errorString)
11	Warning: No Data found
0	OK: [empty]
-3	Error: station not found
-81	Error: workorder not found
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-431	Error: lock name not found

TAB. 3-412 *Output values for »trGetSerialNumberLockHistory«*

3.138 trLockSerialNumbers

This function locks the passed serial numbers.

Function declaration (CORBA)

```
long trLockSerialNumbers(
     in string stationNo,
     in string lockName,
     in string lockInformation,
     in SerialNumberArray serialNumberArray,
     out string errorString)
raises (ServerException);

struct SerialNumberData
     {
        string serialNumber;
        string serialNumberPos;
     };
typedef sequence<SerialNumberData> SerialNumberArray;
```

Parameter (in)

Parameter name	Explanation
stationNo	Station no. of the work station in iTAC.MES.Suite
lockName	Lock group
lockInformation	Lock information (reason for locking)
serialNumberArray	Array with the following variables:
serialNumber	Serial no. of the panel
serialNumberPos	Position of the single panel in the unit-array

TAB. 3-413 *Parameter (in) »trLockSerialNumbers«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-414 *Parameter (out) »trLockSerialNumbers«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-39	Error: serialnumber not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-430	Error: serialnumber is already locked under this lock name
-431	Error: lock name not found

TAB. 3-415 *Output values for »trLockSerialNumbers«*

3.139 trUnlockSerialNumbers

This function unlocks the passed serial numbers.

Function declaration (CORBA)

```
long trUnlockSerialNumbers(
     in string stationNo,
     in long fullUnlock,
     in string lockName,
     in string lockInformation,
     in SerialNumberArray serialNumberArray,
     out string errorString)
raises (ServerException);

struct SerialNumberData
     {
        string serialNumber;
        string serialNumberPos;
     };
typedef sequence<SerialNumberData> SerialNumberArray;
```

Parameter (in)

Parameter name	Explanation
stationNo	Station no. of the work station in iTAC.MES.Suite
fullUnlock	Specifies which serial numbers are unlocked [0; 1]: **1** = All passed serial numbers are unlocked **0** = Only serial numbers from the passed lock group are unlocked (see lockName)
lockName	Lock group
lockInformation	Lock information (reason for locking)
serialNumberArray	Array with the following variables:
serialNumber	Serial no. of the panel
serialNumberPos	Position of the single panel in the unit-array

TAB. 3-416 *Parameter (in) »trUnlockSerialNumbers«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-417 *Parameter (out) »trUnlockSerialNumbers«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-39	Error: serialnumber not found
-50	Error: no user registered at station ***Advice:** Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service

TAB. 3-418 *Output values for »trUnlockSerialNumbers«*

Value	Explanations (errorString)
-101	Error: Corba Exception
-431	Error: lock name not found
-432	Error: serialnumber not locked

TAB. 3-418 *Output values for »trUnlockSerialNumbers«*

3.140 unregisterBatch

This function unregisters a material bin from a station. The `batchComplete` parameter is used to specify whether the work step was interrupted as a result of the unregistration or if it was completely processed; this affects the registration procedure of material bins (»registerBatch«). A state booking is performed for the material bin on each unregistration.

ADVICE

This API function is influenced by station parameter code no.: 4016.Further information on this topic can be found in the documentation for the ADM Module! In the context of bin-based traceability, the »assignBatchNoToWorkorder«, »getRegisteredBatch«, »mergeBatch«, »registerBatch« and »splitBatchNoToSerialNumber« functions must also be taken into account!

Function declaration (CORBA)

```
long unregisterBatch(
     in string stationNr,
     in string batchNumber,
     in long processLayer,
     in double batchQuantity,
     in long batchComplete,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
batchNumber	Number of the material bin (container no.)
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
batchQuantity	Current quantity in the material bin ***Advice:*** *This quantity is captured with each function call in iTAC.MES.Suite (PDC booking). In order for the quantity bookings to be evaluated in the QM Module, configuration parameter code no. 4016 must be active!*
batchComplete	Complete processing [0; 1]: **1** = Processing fully completed **0** = Processing only interrupted

TAB. 3-419 *Parameter (in) »unregisterBatch«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-420 *Parameter (out) »unregisterBatch«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-7	Error: no workstep found for station and processLayer
-39	Error: serialnumber not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-79	Error: no PM license for this station
-81	Error: workorder not found
-82	Error: serialnumber belongs to another workorder
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-202	Error: container not found
-205	Error: No workstep found for workorder, layer and station
-206	Error: No active workorder found for station
-359	Error: materialBin not completed at an other station
-361	Error: Container not registered at station
-363	Error: Quantity not allowed
-370	Error: serialNumber is not a serialNumberRef
-371	Error: serialNumber not unique
-373	Error: serialNumberState is not valid [0; 1; 2]

TAB. 3-421 *Output values for »unregisterBatch«*

3.141 unregisterUser

This function unregisters the user from the station; as only the registered user may unregister, the password must be provided too.

ADVICE

A user is registered using the »registerUser« API function!

Function declaration (CORBA)

```
long unregisterUser(
     in string stationNr,
     in string userName,
     in string password,
     in string client,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
userName	Login name of the user (user identification)
password	Password of the user
client	Abbreviation for the client

TAB. 3-422 *Parameter (in) »unregisterUser«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-423 *Parameter (out) »unregisterUser«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: user or password not found
-3	Error: station not found
-4	Error: other user already registered to this station
-6	Error: user not unique
-50	Error: no user registered at station
-51	Error: password is invalid
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-424 *Output values for »unregisterUser«*

3.142 updateEquipmentData

This function defines which resources are assigned to the station. The **setupFlag** parameter can be used to define whether the passed resources are to be assigned to (rigged) or removed from (stripped down) the station.

ADVICE

In the context of resource management, the »checkEquipmentData«, »getRequiredEquipmentData«, »getSetupEquipmentData« and »removeEquipmentForWorkorder« functions must also be taken into account!

Function declaration (CORBA)

```
long updateEquipmentData(
     in string stationNr,
     in long setupFlag
     in EquipmentPrepareDataArray equipmentPrepareData,
     out string errorString)
raises (ServerException);

struct EquipmentPrepareData
     {
        string equipmentNo;
        string equipmentIndex;
     };
typedef sequence<EquipmentPrepareData> EquipmentPrepareDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
setupFlag	Change "setup" mode [0; 1]: **0** = Rig resource **1** = Strip down resource
equipmentPrepareData	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
equipmentNo	Number of the resource in iTAC.MES.Suite
equipmentIndex	Individual designator for the resource ***Special case:*** *If the resource number is unique without the addition of the index, an empty string or the value [-1] can be passed here!*

TAB. 3-425 *Parameter (in) »updateEquipmentData«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-426 *Parameter (out) »updateEquipmentData«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-6	Error: `setupFlag` not valid [0; 1]
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-305	Error: Equipment not found
-306	Error: Equipment result inconsistent ***Advice:*** *Value is only returned if an empty string or the value [-1] is passed for the* `equipmentIndex` *parameter!*
-307	Error: Equipment not available
-317	Error: EquipmentNo found more than once ***Advice:*** *Value is only returned if an empty string or the value [-1] is passed for the* `equipmentIndex` *parameter!*
-353	Error: Equipment is expired
-354	Error: Equipment is invalid
-355	Error: Equipment has invalid status

TAB. 3-427 *Output values for »updateEquipmentData«*

3.143 updateMaterialSetup

With this function, it is possible to update the setup for individual setup locations for a station. In other words, it can be used to add or update individual setup data.

ADVICE

This API function is influenced by station parameter code nos.: 13110 and 13130. Further information on this topic can be found in the documentation for the ADM Module. Unlike the »updateSetupData« API function, this function can also process mounting places and panel positions!

Function declaration (CORBA)

```
long updateMaterialSetup(
     in string stationNr,
     in long processLayer,
     inout MaterialSetupDataArray materialSetupData,
     inout CompPositionArray compPositions,
     out string errorString)
raises (ServerException);

struct MaterialSetupData
     {
        string position;
        string materialBinNr;
        string stationNr;
        string productNr;
        string supplierCode;
        string supplierName;
        string workOrder;
        string placementName;
        string partNr;
        long compReference;
        long setupDate;
        long endDate;
        long returnCode;
        string returnComment;
        string dateCode;
        boolean removeSetup;
     };
typedef sequence<MaterialSetupData> MaterialSetupDataArray;

struct CompPosition
     {
        long compReference;
        string compName;
        long panelPosition;
     };
typedef sequence<CompPosition> CompPositionArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent

TAB. 3-428 *Parameter (in) »updateMaterialSetup«*

Parameter (inout)

Parameter name	Explanation
materialSetupData	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
position	Setup location (e.g. track) of the material that has been set up
materialBinNr	Number of the material bin (container no.)
stationNr	Station no. of the work station in iTAC.MES.Suite
productNr	Part no. of the product in iTAC.MES.Suite
supplierCode	Supplier number
supplierName	Supplier name
workOrder	Production order number
placementName	Name of the mounting program
partNr	Part no. of the container material in iTAC.MES.Suite
compReference	Reference value to array compPositionArray
setupDate	Start date for the setup (start of production); the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20* ***Special case:*** *[-1] passes the current date and the current time!*
endDate	Point in time at which the container was stripped down; the total time is specified as the number of seconds since 01-01-1970 ***Special case:*** *[-1] passes the current date and the current time!*
returnCode	**in**: [empty] - value is not evaluated **out**: "errorString" for each array returned: **2**; **-1**; **-2**; **-7**; **-10**; **-11**; **-101** ***Advice:*** *See parameter* returnComments*!*
returnComment	**in**: [empty] - value is not evaluated **out**: Explanation "errorString": **2** = KompNames not set **-1** = StationId [...] not found **-2** = ChargeExt [...] not found **-7** = LabelId [...] not found" **-10** = Invalid Daterange **-11** = Invalid Position **-101** = ServerExceptionMessage
dateCode	DateCode for the storage unit
removeSetup	Identifies whether the container was stripped down [0; 1]: **1** = Container was already stripped down (TRUE) **0** = Container is still rigged (FALSE)

TAB. 3-429 *Parameter (inout) »updateMaterialSetup«*

Parameter name	Explanation
compPositionArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
compReference	Reference value to array materialSetupData
compName	The mounting place conforms to the iTAC.MES.Suite BOM
panelPosition	Panel position

TAB. 3-429 *Parameter (inout) »updateMaterialSetup«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-430 *Parameter (out) »updateMaterialSetup«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-431 *Output values for »updateMaterialSetup«*

3.144 updateMaterialSetupHistory

This function permits the retrospective update of setup information for a station in the history. This may be required if setup information from a period before the introduction of the iTAC.MES.Suite system are to be captured.

Should the retrospective update of a setup in the history lead to overlapping validity periods, the basic rule applies that the period currently being updated prevails. Setups that are overlapped by it have their period of validity adjusted. This ensures that there was always only one valid setup at any one time.

ADVICE

Unlike the »updateSetupDataHistory« API function, this function can also process mounting places and panel positions!

Function declaration (CORBA)

```
long updateMaterialSetupHistory(
     in string stationNr,
     in long processLayer,
     inout MaterialSetupDataArray materialSetupData,
     inout CompPositionArray compPositions,
     out string errorString)
raises (ServerException);

struct MaterialSetupData
     {
        string position;
        string materialBinNr;
        string stationNr;
        string productNr;
        string supplierCode;
        string supplierName;
        string workOrder;
        string placementName;
        string partNr;
        long compReference;
        long setupDate;
        long endDate;
        long returnCode;
        string returnComment;
        string dateCode;
        boolean removeSetup;
     };
typedef sequence<MaterialSetupData> MaterialSetupDataArray;

struct CompPosition
     {
        long compReference;
        string compName;
        long panelPosition;
     };
typedef sequence<CompPosition> CompPositionArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent

TAB. 3-432 *Parameter (in) »updateMaterialSetupHistory«*

Parameter (inout)

Parameter name	Explanation
materialSetupData	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
position	Setup location (e.g. track) of the material that has been set up
materialBinNr	Number of the material bin (container no.)
stationNr	Station no. of the work station in iTAC.MES.Suite
productNr	Part no. of the product in iTAC.MES.Suite
supplierCode	Supplier number
supplierName	Supplier name
workOrder	Production order number
placementName	Name of the mounting program
partNr	Part no. of the container material in iTAC.MES.Suite
compReference	Reference value to array compPositionArray
setupDate	Start date for the setup (start of production); the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
endDate	Point in time at which the container was stripped down; the total time is specified as the number of seconds since 01-01-1970
returnCode	**in**: [empty] - value is not evaluated **out**: "errorString" for each array returned: **2**; **-1**; **-2**; **-7**; **-10**; **-11**; **-101** ***Advice:*** *See parameter* returnComments*!*
returnComment	**in**: [empty] - value is not evaluated **out**: Explanation "errorString": **2** = KompNames not set **-1** = StationId [...] not found **-2** = ChargeExt [...] not found **-7** = LabelId [...] not found" **-10** = Invalid Daterange **-11** = Invalid Position **-101** = ServerExceptionMessage

TAB. 3-433 *Parameter (inout) »updateMaterialSetupHistory«*

Parameter name	Explanation
dateCode	DateCode for the storage unit
removeSetup	Identifies whether the container was stripped down: **1** = Container was already stripped down (TRUE) **0** = Container is still rigged (FALSE)
compPositionArray	Array with the following variables: ***Advice:*** *The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
compReference	Reference value to array materialSetupData
compName	The mounting place conforms to the iTAC.MES.Suite BOM
panelPosition	Panel position

TAB. 3-433 *Parameter (inout) »updateMaterialSetupHistory«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-434 *Parameter (out) »updateMaterialSetupHistory«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-81	Error: workorder not found
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-216	Error: at least one material bin was not found
-217	Error: at least one component name made a problem

TAB. 3-435 *Output values for »updateMaterialSetupHistory«*

3.145 updateSetupData

Depending on the parameter `setupFlag`, this function permits the deletion or preparation of the setup for every setup location (e.g., track) of a station.

ADVICE

This API function is influenced by station parameter code nos.: 13110 and 13130. Further information on this topic can be found in the documentation for the ADM Module. Unlike the »updateMaterialSetup« API function, this function cannot process mounting places and panel positions!

Function declaration (CORBA)

```
long updateSetupData(
     in string stationNr,
     in long processLayer,
     in long setupFlag,
     in long numberOfRecords,
     in SetupPrepareDataArray setupPrepareDataArray,
     out string errorString)
raises (ServerException);

struct SetupPrepareData
     {
        string position;
        string materialBinNr;
     };
typedef sequence<SetupPrepareData> SetupPrepareDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
setupFlag	Change "setup" mode [0; 1]: **0** = Prepare setup **1** = Delete setup
numberOfRecords	Default value for the number of entries in the array
setupPrepareDataArray	Array with the following variables:
position	Setup location (e.g. track) of the material that has been set up
materialBinNr	Number of the material bin (container no.)

TAB. 3-436 *Parameter (in) »updateSetupData«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-437 *Parameter (out) »updateSetupData«*

Output values

Value	Explanations (errorString)
2	Warning:setup not activated
0	OK: [empty]
-1	Error: general Error
-2	Error: stationNr not found
-3	Error: no setup found for station
-6	Error: `setupFlag` not valid [0; 1]
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-438 *Output values for »updateSetupData«*

3.146 updateSetupDataBySnr

This function sets up the part number of a semifinished product (slave serial number) on a station for a work order. The function only works together with a leading ERP system, since the material lot information of the semifinished product must first be passed by a parent ERP system.

ADVICE

This API function is influenced by station parameter code no.: 13130.Further information on this topic can be found in the documentation for the ADM Module!

Function declaration (CORBA)

```
long updateSetupDataBySnr(
     in string stationNr,
     in long processLayer,
     in string workorder,
     in string serialNumberSlave,
     in string position,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent ***Special case:** If [-1] is passed, the orientation of the PCB of the active work step is used!*
workorder	Number of a production order ***Special case:** If [-1] is passed, the work order which is active on the station is used!*
serialNumberSlave	Serial number of the semifinished product (slave serial no.) ***Advice:** The part number of the semifinished product is rigged for the passed work order (`workorder`) on the passed station (`stationNr`) and slot (`position`)!*
position	Setup location (e.g. track) of the material that has been set up ***Special case:** If [-1] is passed, the part number of the semifinished product is used as setup location!*

TAB. 3-439 *Parameter (in) »updateSetupDataBySnr«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-440 *Parameter (out) »updateSetupDataBySnr«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: [empty]
-3	Error: station not found
-6	Error: `processLayer` has an invalid value
-75	Error: a required parameter is missing
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-205	Error: No workstep found for workorder, layer and station
-206	Error: No active workorder found for station
-207	Error: Material charge for setup placement not found
-208	Error: No material charge found
-209	Error: No workorder found for serialnumber

TAB. 3-441 *Output values for »updateSetupDataBySnr«*

3.147 updateSetupDataHistory

This function permits the retrospective update of setup information for a station in the history. This may be required if setup information from a period before the introduction of the iTAC.MES.Suite system are to be captured.

Should the retrospective update of a setup in the history lead to overlapping validity periods, the basic rule applies that the period currently being updated prevails. Setups that are overlapped by it have their period of validity adjusted. This ensures that there was always only one valid setup at any one time.

Function declaration (CORBA)

```
long updateSetupDataHistory(
     out long numberOfRecords,
     inout SetupHistoryDataArray setupHistoryDataArray,
     out string errorString)
raises (ServerException);

struct SetupHistoryData
     {
        string stationNr;
        string workOrder;
        long workStep;
        string position;
        string materialBinNr;
        string kompInfo;
        long dateFrom;
        long dateTo;
        long returnCode;
        string returnComment;
     };
typedef sequence<SetupHistoryData> SetupHistoryDataArray;
```

Parameter (inout)

Parameter name	Explanation
SetupHistoryDataArray	Array with the following variables:
stationNr	Station no. of the work station in iTAC.MES.Suite
workOrder	Number of a production order
workStep	Number of the work step
position	Setup location (e.g. track) of the material that has been set up
materialBinNr	Number of the material bin (container no.)
kompInfo	Mounting places of the respective components ***Advice:*** *If several mounting places are affected, they must be separated by a ";" (e.g. "R20;R24")!*
dateFrom	Start time of the start-up; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
dateTo	End time for the expiry, the total time is specified as the number of seconds since 01-01-1970
returnCode	**in**: [empty] - value is not evaluated **out**: "errorString" for each array returned: **2**; **-1**; **-2**; **-7**; **-10**; **-11**; **-101** ***Advice:*** *See parameter* returnComments*!*

TAB. 3-442 *Parameter (inout) »updateSetupDataHistory«*

Parameter name	Explanation
returnComment	**in**: [empty] - value is not evaluated **out**: Explanation "errorString": **2** = KompNames not set **-1** = StationId [...] not found **-2** = ChargeExt [...] not found **-7** = LabelId [...] not found" **-10** = Invalid Daterange **-11** = Invalid Position **-101** = ServerExceptionMessage

TAB. 3-442 *Parameter (inout) »updateSetupDataHistory«*

Parameter (out)

Parameter name	Explanation
numberOfRecords	Number of entries in the array returned
errorString	Output text according to output value (see table)

TAB. 3-443 *Parameter (out) »updateSetupDataHistory«*

Output values

Value	Explanations (errorString)
***Advice:** The value for the* errorString *is computed from all return arrays! As a rule, the worst* returnCode *is used (worst case).*	
1	Warning:component name problem at index
0	OK: [empty]
-1	Error: station not found at index
-2	Error: materialBin not found at index
-3	Error: workOrder not found at index
-10	Error: unknown problem at index
-50	Error: no user registered at station ***Advice:** Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-444 *Output values for »updateSetupDataHistory«*

3.148 uploadFailureslip

This function books a fault note for a serial number; the function can also be used to select whether a state booking should occur at the same time. In addition to the test step number, the fault note which is passed also contains information on the location (mounting place) where the failure occurred.

ADVICE

This API function is influenced by station parameter code nos.: 8310 and 16000. In addition, the station settings for "Upload Settings" and "Setup" control this function. Further information on this topic can be found in the documentation for the ADM Module!

Function declaration (CORBA)

```
long uploadFailureslip(
     in string stationNr,
     in long processLayer,
     in string serialNumber,
     in string serialnumberPos,
     in string partNr,
     in long state,
     in long bookDate,
     in FailureslipArray failureslip,
     in long checkActiveWorkorder,
     out string errorString)
raises (ServerException);

struct Failureslip
     {
        string compName;
        string testStepName;
        string testStepDescription;
     };
typedef sequence<Failureslip> FailureslipArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent ***Special case:*** *Depending on the setting of configuration parameter code no. 8310, the work step number (AVO) used in an ERP system can be passed directly instead of the PCB orientation!*
serialNumber	Serial no. of the panel
serialNumberPos	Position of the single panel in the unit-array ***Advice:*** *If the value [-1] is passed, the position is not evaluated; in this case, each single panel must be specified by means of a unique* `serialNumber` *!*
partNr	Part no. in iTAC.MES.Suite ***Advice:*** *The input is optional; if [-1] or [empty] is passed, the field is not evaluated; otherwise, the passed part no. is checked against the serial number!*

TAB. 3-445 *Parameter (in) »uploadFailureslip«*

Parameter name	Explanation
state	State information for the fault note [0; 1]: **0** = Pass **1** = Fail ***Advice:*** *If the value [-1] is passed, no state booking is performed!*
bookDate	Date of the state booking; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20* ***Special case:*** *[-1] passes the current date and the current time!*
failureslip	Array with the following variables:
compName	Fault location (mounting place) is consistent with the iTAC.MES.Suite BOM
testStepName	Name of the faulty test step
testStepDescription	Description of the test step ***Advice:*** *This entry may consist of multiple lines; lines are separated by "\\n"; a line may not contain more than one thousand characters!*
checkActivWorkOrder	Checking the current order of the station by comparing it with the order that is specified via the current serial no. [0; 1]: **0** = Is not checked **1** = Is checked

TAB. 3-445 *Parameter (in) »uploadFailureslip«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-446 *Parameter (out) »uploadFailureslip«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-3	Error: station not found
-7	Error: no workstep found for station and processLayer
-39	Error: serialnumber not found
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-71	Error: checkActiveWorkOrder must be in [0;1]
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-206	Error: No active workorder found for station
-214	Error: Invalid value for state: [-1,0,1]

TAB. 3-447 *Output values for »uploadFailureslip«*

3.149 uploadMaterialBinBookingsExt

This function is used to pass material movement data (e.g. quantity changes through consumption) from the production line to the parent ERP system via iTAC.MES.Suite.

ADVICE

This API function is affected by station parameter code no.: 8310 as well as by site parameter code no.: 7210. In addition, this function is controlled by the station settings for "Upload Settings". Further information on this topic can be found in the documentation for the ADM Module!

In general, both positive as well as negative quantity bookings can be processed. On a function call, multiple quantity bookings can be passed – each quantity booking is processed as a unique data set in an array. In the event of a transmission error, only the faulty data set is returned together with a corresponding error message.

Depending on the used input parameters, it is possible to distinguish between functions for material movement **with** or **without** container reference. If a container number is passed (`materialBinNr`), the quantity changes are booked to the container. In connection with the part number (`partNr`), a check can be performed to determine whether the part on the container matches the part number passed. If only the part number is specified, only a material movement is logged – without a container being changed with regard to quantity.

Function declaration (CORBA)

```
long uploadMaterialBinBookingsExt(
     in string stationNr,
     inout MaterialBinBookingsExtArray materialBinData,
     out string errorString)
raises (ServerException);

struct MaterialBinBookingsExt
     {
        long counter;
        string partNr;
        string materialBinNr;
        double quantity;
        long transact;
        string workorder;
        long processLayer;
        string stationNoBook;
        long bookDate;
        long errorcode;
        string errortxt;
        double qtyPlaced;
        double qtyWasted;
     };
typedef sequence<MaterialBinBookingsExt>
                                   MaterialBinBookingsExtArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite

TAB. 3-448 *Parameter (in) »uploadMaterialBinBookingsExt«*

Parameter (inout)

Parameter name	Explanation
materialBinBookings	Array with the following variables: ***Advice:** The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension* **Size***!*
counter	**in**: Sequential number for each data set in the array **out**: Number of the incorrectly transferred data set ***Advice:** The data sets of an array must be uniquely numbered!*
partNr	**in**: Part no. in iTAC.MES.Suite ***Advice:** The part no. may optionally be passed with the material bin. In this case a [-1] or [empty] must be passed here and the container number in the field* materialBinNr *must be known to the system!* **out**: Part no. of the faulty data set ***Advice:** Is only returned in the event of an error!*
materialBinNr	**in**: Number of the material bin (container no.) ***Advice:** The input is optional; if [-1] or [empty] is passed, the field is not evaluated!* **out**: Container no. of the faulty data set ***Advice:** Is only returned in the event of an error!*
quantity	**in**: Material quantity; this quantity is booked in connection with materialBinNr as container quantity change and without passing the materialBinNr as a material quantity change. ***Advice:** Both negative as well as positive quantity movements can be processed. Positive values increase the material stock. Negative values reduce it. If this booking causes the material stock to become negative, error code = 1 is returned!* **out**: Material quantity of the faulty data set ***Advice:** Is only returned in the event of an error!*
transact	**in**: Unique designator (ID) of the material movement type ***Advice:** If [0] or [-1] is passed, mapping to the ID of the (default) material movement type occurs (customer-specific function, details upon request)!* **out**: Material movement type of the faulty data set ***Advice:** Is only returned in the event of an error!*
workorder	**in**: Number of a production order ***Special case:** If [0] is passed, the booking is performed without association with a work order! If [-1] is passed, the work order which is active on the station is used; in this case, it must be ensured that the active work order also matches the booking (key word: time-delayed collective booking)!* **out**: Order number of the faulty data set ***Advice:** Is only returned in the event of an error!*

TAB. 3-449 *Parameter (inout) »uploadMaterialBinBookingsExt«*

Parameter name	Explanation
processLayer	**in**: Orientation of the PCB during the work step: **0** = Component side **1** = Solder side **2** = Orientation independent **out**: Orientation of the PCB of the faulty data set ***Advice:*** *Is only returned in the event of an error!* ***Special case:*** *Depending on the setting of configuration parameter code no. 8310, it is possible to directly process the work step number (AVO) passed directly from an ERP system instead of the PCB orientation!*
stationNoBook	**in**: Station number on which booking occurs **out**: Station number of the faulty data set ***Advice:*** *Is only returned in the event of an error!*
bookDate	**in**: Date of the state booking; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Special case:*** *[-1] passes the current date and current time!* **out**: Booking date of the faulty data set ***Advice:*** *Is only returned in the event of an error!*
errorcode	**in**: [empty] - value is not evaluated **out**: **1** (for return text, see errortxt) **0** (for return text, see errortxt) **-62** (for return text, see errortxt) **-75** (for return text, see errortxt) **-81** (for return text, see errortxt) **-99** (for return text, see errortxt) **-202** (for return text, see errortxt) **-203** (for return text, see errortxt) **-204** (for return text, see errortxt)
errortxt	**in**: [empty] - value is not evaluated **out**: For errorcode **1**: Decrement amount of material to a negative number For errorcode **-62**: Product not found For errorcode **-75**: A required parameter is missing For errorcode **-81**: Workorder not found For errorcode **-99**: Server error For errorcode **-202**: Container not found For errorcode **-203**: Product is not retrograde For errorcode **-204**: Product belongs not to given material bin
qtyPlaced	**in**: Sum of mounted components from container **out**: Sum of the faulty data set ***Advice:*** *Is only returned in the event of an error!*
qtyWasted	**in**: Sum of lost components from container **out**: Sum of the faulty data set ***Advice:*** *Is only returned in the event of an error!*

TAB. 3-449 *Parameter (inout) »uploadMaterialBinBookingsExt«*

Parameter (out)

Parameter name	Explanation
`errorString`	Output text according to output value (see table)

TAB. 3-450 *Parameter (out) »uploadMaterialBinBookingsExt«*

Output values

Value	Explanations (errorString)
`7`	Warning: Decrement amount of material to a negative number ***Advice:*** *Value is only returned if site parameter code no. 7210 was set to "2"; for "1", "-392" is returned!*
`0`	OK: [empty]
`-3`	Error: station not found
`-50`	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
`-78`	Error: no TR license for this station
`-99`	Error: Server error
`-100`	Error: Corba no Service
`-101`	Error: Corba Exception
`-392`	Error: Decrement amount of material to a negative number

TAB. 3-451 *Output values for »uploadMaterialBinBookingsExt«*

3.150 uploadMaterialBookings

This function is used to pass material movement data (e.g. quantity changes through consumption) from the production line to the parent ERP system via iTAC.MES.Suite.

ADVICE

This API function is affected by station parameter code no.: 8310 as well as by site parameter code no.: 7210. In addition, this function is controlled by the station settings for "Upload Settings". Further information on this topic can be found in the documentation for the ADM Module!

In general, both positive as well as negative quantity bookings can be processed. On a function call, multiple quantity bookings can be passed – each quantity booking is processed as a unique data set in an array. In the event of a transmission error, only the faulty data set is returned together with a corresponding error message.

Depending on the used input parameters, it is possible to distinguish between functions for material movement **with** or **without** container reference. If a container number is passed (`materialBinNr`), the quantity changes are booked to the container. In connection with the part number (`partNr`), a check can be performed to determine whether the part on the container matches the part number passed. If only the part number is specified, only a material movement is logged – without a container being changed with regard to quantity.

Function declaration (CORBA)

```
long uploadMaterialBookings(
     in string stationNr,
     inout MaterialBinBookingsArray materialBinBookings,
     out string errorString)
raises (ServerException);

struct MaterialBinBookings
     {
        long counter;
        string partNr;
        string materialBinNr;
        double quantity;
        long transact;
        string workorder;
        long processLayer;
        string stationNoBook;
        long bookDate;
        long errorcode;
        string errortxt;
     };
typedef sequence<MaterialBinBookings> MaterialBinBookingsArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite

TAB. 3-452 *Parameter (in) »uploadMaterialBookings«*

Parameter (inout)

Parameter name	Explanation
materialBinBookings	Array with the following variables: ***Advice:** The number of entries in the array returned is ascertained directly from the structure. When using the C programming language, note that the number is output in a separate variable directly preceding the array; the name of the variable corresponds to the name of the array with the extension **Size**!*
counter	**in**: Sequential number for each data set in the array **out**: Number of the incorrectly transferred data set ***Advice:** The data sets of an array must be uniquely numbered!*
partNr	**in**: Part no. in iTAC.MES.Suite ***Advice:** The part no. may optionally be passed with the material bin. In this case a [-1] or [empty] must be passed here and the container number in the field* materialBinNr *must be known to the system!* **out**: Part no. of the faulty data set ***Advice:** Is only returned in the event of an error!*
materialBinNr	**in**: Number of the material bin (container no.) ***Advice:** The input is optional; if [-1] or [empty] is passed, the field is not evaluated!* **out**: Container no. of the faulty data set ***Advice:** Is only returned in the event of an error!*
quantity	**in**: Material quantity; this quantity is booked in connection with materialBinNr as container quantity change and without passing the materialBinNr as a material quantity change. ***Advice:** Both negative as well as positive quantity movements can be processed. Positive values increase the material stock. Negative values reduce it. If this booking causes the material stock to become negative, error code = 1 is returned!* **out**: Material quantity of the faulty data set ***Advice:** Is only returned in the event of an error!*
transact	**in**: Unique designator (ID) of the material movement type ***Advice:** If [0] or [-1] is passed, mapping to the ID of the (default) material movement type occurs (customer-specific function, details upon request)!* **out**: Material movement type of the faulty data set ***Advice:** Is only returned in the event of an error!*
workorder	**in**: Number of a production order ***Special case:** If [0] is passed, the booking is performed without association with a work order! If [-1] is passed, the work order which is active on the station is used; in this case, it must be ensured that the active work order also matches the booking (key word: time-delayed collective booking)!* **out**: Order number of the faulty data set ***Advice:** Is only returned in the event of an error!*

TAB. 3-453 *Parameter (inout) »uploadMaterialBookings«*

Parameter name	Explanation
processLayer	**in**: Orientation of the PCB during the work step: **0** = Component side **1** = Solder side **2** = Orientation independent **out**: Orientation of the PCB of the faulty data set ***Advice:*** *Is only returned in the event of an error!* ***Special case:*** *Depending on the setting of configuration parameter code no. 8310, it is possible to directly process the work step number (AVO) passed directly from an ERP system instead of the PCB orientation!*
stationNoBook	**in**: Station number on which booking occurs **out**: Station number of the faulty data set ***Advice:*** *Is only returned in the event of an error!*
bookDate	**in**: Date of the state booking; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Special case:*** *[-1] passes the current date and current time!* **out**: Booking date of the faulty data set ***Advice:*** *Is only returned in the event of an error!*
errorcode	**in**: [empty] - value is not evaluated **out**: **1** (for return text, see errortxt) **0** (for return text, see errortxt) **-62** (for return text, see errortxt) **-75** (for return text, see errortxt) **-81** (for return text, see errortxt) **-99** (for return text, see errortxt) **-202** (for return text, see errortxt) **-203** (for return text, see errortxt) **-204** (for return text, see errortxt)
errortxt	**in**: [empty] - value is not evaluated **out**: For errorcode **1**: Decrement amount of material to a negative number For errorcode **-62**: Product not found For errorcode **-75**: A required parameter is missing For errorcode **-81**: Workorder not found For errorcode **-99**: Server error For errorcode **-202**: Container not found For errorcode **-203**: Material bookings only allowed on retrograde configured products For errorcode **-204**: Product belongs not to given material bin

TAB. 3-453 *Parameter (inout) »uploadMaterialBookings«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-454 *Parameter (out) »uploadMaterialBookings«*

Output values

Value	Explanations (errorString)
7	Warning: Decrement amount of material to a negative number ***Advice:*** *Value is only returned if site parameter code no. 7210 was set to "2"; for "1", "-392" is returned!*
0	OK: [empty]
-3	Error: station not found
-75	Error: a required parameter is missing
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-392	Error: Decrement amount of material to a negative number

TAB. 3-455 *Output values for »uploadMaterialBookings«*

3.151 uploadResultDataAndRecipe

This function books individual products including state and measurement data in a work step conformant manner. It also automatic generates an associated testing plan. The measurement data must contain the data in the testing plan, so that this function can automatically assign or create a testing plan.

ADVICE

In the QM Module, only measurement data that are assigned to a testing plan can be analyzed! The **uploadResultDataAndRecipe** *API function is influenced by station parameter code nos.: 4015, 7202, 7210 - 7215, 8100, 8200, 8310, 8400, 15200, 15201, 15210, 15250, 15260 and 16000. In addition, this function is controlled by the station settings for "Upload Settings", "Setup", and "QA Settings". Further information on this topic can be found in the documentation for the ADM Module!*

If the passed serial number is unknown, a suitable work order may be assigned based on the part number (**produktNo**) and BOM version (**produktRev**) passed. If no suitable work order can be found or if the order quantity has already been reached, a new work order may be created (station parameter code number 4015) if the "automatic work order creation" is active.

If a **-1** is passed for **serialNumberState**, measurement data can be passed without a new state booking. This makes sense if failure data have already been booked via another "Upload" function and a measurement data set is to be created for this booking without its own state booking. The requirement for this, however, is that a previous state booking must be found. Otherwise, the upload aborts with an error (-395).

Function declaration (CORBA)

```
long uploadResultDataAndRecipe(
     in string stationNr,
     in long processLayer,
     in string serialNumberRef,
     in string serialNumberRefPos,
     in long serialNumberState,
     in string produktNo,
     in string produktRev,
     in long long bookDate,
     in float cycleTime,
     in long numberOfRecords,
     in ResultDataExtendedStructArray resultDataExtArray,
     out string errorString)
raises (ServerException);

struct ResultDataExtendedStruct
     {
        float nr;
        string name;
        string value;
        string min;
        string max;
        string nom;
        string toleranz;
        string unit;
        long failCode;
        string messtyp;
     };
typedef sequence<ResultDataExtendedStruct> ResultDataExtended-
StructArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent ***Special case:*** *Depending on the setting of configuration parameter code no. 8310, the work step number (AVO) used in an ERP system can be passed directly instead of the PCB orientation!*
serialNumberRef	Serial no. of the first single panel or unit-array; this no. is used as reference serial no.
serialNumberRefPos	Position of the serial no. of the first single panel or unit-array
serialNumberState	State of the product associated with the serial no. [0; 1; 2; -1]: **0** = Pass **1** = Fail **2** = Scrap **-1** = No state booking ***Advice****: If the value* ***-1*** *is passed, no state booking is carried out; the* `loopCounter` *remains unchanged and the most recently booked state remains valid!*
produktNo	Part no. of the product in iTAC.MES.Suite ***Advice:*** *This field is only evaluated if the serial numbers passed are unknown (see explanation above)!*
produktRev	Bill-of-material version from iTAC.MES.Suite ***Advice:*** *This field is only evaluated if the serial numbers passed are unknown (see explanation above)!* ***Special case:*** *If the automatic work order creation is active (code number 4015), the value [-1] would assign the current BOM version to the new work order!*
bookDate	Date of the state booking; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20* ***Special case:*** *[-1] passes the current date and the current time!*
cycleTime	Duration of the check in seconds
numberOfRecords	Default value for the number of entries in the array
resultDataExtArray	Array with the following variables:
`nr`	Testing step no. in the recipe
`name`	Name of measurement step ***Advice:*** *The string passed must not be empty; if it is,* `-7` *is returned!*

TAB. 3-456 *Parameter (in) »uploadResultDataAndRecipe«*

Parameter name	Explanation
value	Measured value; the following format definitions apply: **Binary** (Example: 0b110010) `0b[0-1]*` **Octal** (Example: 0o173210) `0o[0-7]*` **Hexadecimal** (Example: 0x05FB7A `0x[0-9A-Fa-f]*` **Decimal EN** (Example: 1366.665) `[0-9]*.[0-9]*` **Decimal DE** (Example: 1366,665) `[0-9]*,[0-9]*` **Text** (Example: Hello world) `.*`
min	Lower limit for measurement step ***Advice:*** *Alternatively, the limit value can also be defined with the* `nom` *and* `toleranz` *parameters!*
max	Upper limit for measurement step ***Advice:*** *Alternatively, the limit value can also be defined with the* `nom` *and* `toleranz` *parameters!*
nom	Nominal value ***Advice:*** *Alternatively, the limit value can also be defined with the* `min` *and* `max` *parameters; the* `min` *and* `max` *parameters override this setting!*
toleranz	Tolerance value for measurement step ***Advice:*** *Alternatively, the limit value can also be defined with the* `min` *and* `max` *parameters; the* `min` *and* `max` *parameters override this setting!*
unit	Unit for the value passed
failCode	State information for measurement step [0; 1]: **0** = OK **1** = FAIL ***Advice:*** *Other integer values can be interpreted either as pseudo failure (code no.: 15250) or without evaluation (code no.: 15260), depending on the station configuration!*
messtyp	Type of the measurement step; optional with format specification **CO** = Config **RE** = Result **CR** = Config-Result **DY** = Dynamic Supported formats: **TXT** = text **BIN** = binary **OCT** = octal **HEX** = hexadecimal **DEN** = decimal EN "." **DDE** = decimal DE "," When a format specification is passed, it is attached to the type (with separator "**|**", e.g., **CR|HEX**). If no format is passed, the format is detected automatically. ***Advice:*** *If no or a different format is entered, the measurement step type "Config-Result" with automatic format detection is used!*

TAB. 3-456 *Parameter (in) »uploadResultDataAndRecipe«*

Parameter (out)

Parameter name	Explanation
`errorString`	Output text according to output value (see table)

TAB. 3-457 *Parameter (out) »uploadResultDataAndRecipe«*

Output values

Value	Explanations (errorString)
`0`	OK: [empty]
`-1`	Error: no active workOrder found
`-2`	Error: serialNumber not found
`-3`	Error: station not found
`-4`	Error: serialNumber is assigned to another workOrder
`-5`	Error: `serialNumberState` not valid [0; 1; 2; -1]
`-6`	Error: station is not valid for this serialNumber
`-7`	Error: Teststep names not unique
`-10`	Error: general error while upload state
`-11`	Error: general error while upload result data
`-12`	Error: Max sequence reached ***Advice:*** *This value depends on the default for "max. recursion" in the upload settings!*
`-15`	Error: serialNumber not unique
`-16`	Error: fail because of anomaly ***Advice:*** *Value is only returned, if the station parameter code no. 7210 is active!*
`-17`	Error: serialnumber is locked
`-62`	Error: Product not found
`-67`	Error: no valid bom version
`-78`	Error: no TR license for this station
`-91`	Error: Not enough values for testing ***Advice:*** *Value is only returned if PAA functionality is used (the station parameter code no. 7210 must be active)!*
`-93`	Error: CP violation
`-94`	Error: CPK violation
`-99`	Error: Server error
`-100`	Error: Corba no Service
`-101`	Error: Corba Exception
`-210`	Error: no unique workstep for this process layer
`-395`	Error: no previous booking found for this serialnumber and this workstep ***Advice:*** *Value is only returned if* `-1` *was passed for the* `serialNumberState` *input parameter!*
`-436`	Error: serialnumber is archived

TAB. 3-458 *Output values for »uploadResultDataAndRecipe«*

3.152 uploadState

This function books individual products with their state in a work step conformant manner. It sets a stamp for the component lot capture. This function is the universal function for the booking of a product.

ADVICE

`duplicateSerialNumber` = ***1*** *permits the booking of an entire panel! The* **`uploadState`** *API function is influenced by station parameter code nos.: 7202, 7320, 8310, 8400 and 16000 as well as site parameter 18200. In addition, this function is controlled by the station settings for "Upload Settings". Further information on this topic can be found in the documentation for the ADM Module!*

Function declaration (CORBA)

```
long uploadState(
     in string stationNr,
     in long processLayer,
     in string serialNumberRef,
     in string serialNumberRefPos,
     in long serialNumberState,
     in long duplicateSerialNumber,
     in long checkActiveWorkOrder,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
`stationNr`	Station no. of the work station in iTAC.MES.Suite
`processLayer`	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent ***Special case:*** *Depending on the setting of configuration parameter code no. 8310, the work step number (AVO) used in an ERP system can be passed directly instead of the PCB orientation!*
`serialNumberRef`	Serial no. of the first single panel or unit-array; this no. is used as reference serial no.
`serialNumberRefPos`	Position of the serial no. of the first single panel or unit-array
`serialNumberState`	State of the product associated with the serial no. [0; 1; 2]: **0** = Pass **1** = Fail **2** = Scrap
`duplicateSerialNumber`	Referencing to a single panel [0; 1]: **0** = Action is valid for the specified serial no. only (a position of the multiple printed panel) **1** = Action is carried out on all stored single panels, based on the reference serial number provided
`checkActivWorkOrder`	Checking the current order of the station by comparing it with the order that is specified via the current serial no. [0; 1]: **0** = Is not checked **1** = Is checked

TAB. 3-459 *Parameter (in) »uploadState«*

Parameter (out)

Parameter name	Explanation
`errorString`	Output text according to output value (see table)

TAB. 3-460 *Parameter (out) »uploadState«*

Output values

Value	Explanations (errorString)
`4`	Warning:at least one serial no. is scrap ***Advice:*** *Value is only returned if a multiple panel is booked in which at least one but less than all single panels have been scrapped. In addition, the check for scrap bookings must be activated with site parameter code no. 18200!*
`0`	OK: [empty]
`-1`	Error: no active workOrder found
`-2`	Error: serialNumber not found
`-3`	Error: station not found
`-4`	Error: serialNumber is assigned to another workOrder
`-5`	Error: `serialNumberState` not valid [0; 1; 2]
`-6`	Error: station is not valid for this serialNumber
`-7`	Error: serialNumber is not a serialNumberRef
`-9`	Error: Max sequence reached ***Advice:*** *This value depends on the default for "max. recursion" in the upload settings!*
`-10`	Error: General server error
`-15`	Error: serialNumber not unique
`-17`	Error: serialnumber is locked
`-50`	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
`-70`	Error: `duplicateSerialNumber` must be in [0; 1]
`-71`	Error: `checkActiveWorkOrder` must be in [0;1]
`-78`	Error: no TR license for this station
`-99`	Error: Server error
`-100`	Error: Corba no Service
`-101`	Error: Corba Exception
`-210`	Error: no unique workstep for this process layer
`-375`	Error: SerialNo SCRAP on last workstep ***Advice:*** *Value is only returned if a single panel or all panels of a multiple panel have been scrapped. In addition, the check for scrap bookings must be activated with site parameter code no. 18200!*
`-436`	Error: serialnumber is archived

TAB. 3-461 *Output values for »uploadState«*

3.153 uploadStateAndFailData

This function books attributive error data. Defined failure characteristics are passed together with the specification of the location (mounting place) where the failure occurred.

ADVICE

`duplicateSerialNumber` = ***1*** *permits the booking of an entire panel! This API function is influenced by station parameter code nos.: 7202, 7209, 7320, 8310, 8400 and 16000 as well as site parameter 18200. In addition, this function is controlled by the station settings for "Upload Settings". Further information on this topic can be found in the documentation for the ADM Module!*
This function should no longer be used for new installations as its functionality has been extended by Release 4.1 and is made available via another API function (see »uploadStateAndFailureData«)!

If a **-1** is passed for **`serialNumberState`**, failure data can be passed without a new state booking. This makes sense if failure data are to be updated without state booking. The requirement for this, however, is that a previous state booking must be found for the work step. Otherwise, the upload aborts with an error (`-395`).

Function declaration (CORBA)

```
long uploadStateAndFailData(
     in string stationNr,
     in long processLayer,
     in string artikel,
     in string serialNumber,
     in string serialNumberPos,
     in string nutzenBez,
     in long serialNumberState,
     in long duplicateSerialNumber,
     in FailDataArray fehlerDatenArray,
     in float cycleTime,
     out long loopCounter,
     out string errorString)
raises (ServerException);

struct FailData
     {
        string fehlerOrt;
        string fehlerArt;
        long lage;
        long repariert;
     };
typedef sequence<FailData> FailDataArray;
```

Parameter (in)

Parameter name	Explanation
`stationNr`	Station no. of the work station in iTAC.MES.Suite
`processLayer`	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent ***Special case:*** *Depending on the setting of configuration parameter code no. 8310, the work step number (AVO) used in an ERP system can be passed directly instead of the PCB orientation!*

TAB. 3-462 *Parameter (in) »uploadStateAndFailData«*

Parameter name	Explanation
artikel	Part no. in iTAC.MES.Suite
serialNumber	Serial no. of the panel
serialNumberPos	Position of the single panel in the unit-array ***Advice:*** *If the value [-1] is passed, the position is not evaluated; in this case, each single panel must be specified by means of a unique* `serialNumber` *!*
nutzenBez	Description of the panel position ***Advice:*** *This field is used to create new serial nos., i.e., applies only to process steps during which the serial no. is made known for the first time - no mandatory field!*
serialNumberState	State of the product associated with the serial no. [0; 1; 2; -1]: **0** = Pass **1** = Fail **2** = Scrap **-1** = No state booking ***Advice****: If the value* ***-1*** *is passed, no state booking is carried out; the* `loopCounter` *remains unchanged and the most recently booked state remains valid!*
duplicateSerialNumber	Referencing to a single panel [0; 1]: **0** = Action is valid for the specified serial no. only (a position of the multiple printed panel) **1** = Action is carried out on all stored single panels, based on the reference serial number provided
fehlerDatenArray	Array with the following variables:
`fehlerOrt`	Fault location (mounting place)
`fehlerArt`	Failure type ***Advice:*** *The failure types must be created in advance in iTAC.MES.Suite and must be assigned to the station (see ADM Module)!*
`lage`	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
`repariert`	Repair state [0; 1]: **0** = Not repaired **1** = Repaired
cycleTime	Duration of the check in seconds

TAB. 3-462 *Parameter (in) »uploadStateAndFailData«*

Parameter (out)

Parameter name	Explanation
loopCounter	Number of bookings against this serial no. and this work step (stations of the associated ERP group) ***Advice:*** *Only bookings of stations of the type "tester" are counted; bookings of repair and diagnosis stations are* ***not*** *counted!*
errorString	Output text according to output value (see table)

TAB. 3-463 *Parameter (out) »uploadStateAndFailData«*

Output values

Value	Explanations (errorString)
4	Warning:at least one serial no. is scrap ***Advice:*** *Value is only returned if a multiple panel is booked in which at least one but less than all single panels have been scrapped. In addition, the check for scrap bookings must be activated with site parameter code no. 18200!*
0	OK: [empty]
-1	Error: no active workOrder found
-2	Error: serialNumber not found
-3	Error: station not found
-4	Error: serialNumber is assigned to another workOrder
-5	Error: `serialNumberState` not valid [0; 1; 2]
-6	Error: station is not valid for this serialNumber
-8	Error: Component-name not valid
-9	Error: Errortype not valid
-11	Error: General server error
-12	Error: Max sequence reached ***Advice:*** *This value depends on the default for "max. recursion" in the upload settings!*
-15	Error: serialNumber not unique
-17	Error: serialnumber is locked
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-210	Error: no unique workstep for this process layer
-375	Error: SerialNo SCRAP on last workstep ***Advice:*** *Value is only returned if a single panel or all panels of a multiple panel have been scrapped. In addition, the check for scrap bookings must be activated with site parameter code no. 18200!*
-395	Error: no previous booking found for this serialnumber and this workstep ***Advice:*** *Value is only returned if* `-1` *was passed for the* `serialNumberState` *input parameter!*
-436	Error: serialnumber is archived

TAB. 3-464 *Output values for »uploadStateAndFailData«*

3.154 uploadStateAndFailureData

This function books attributive error data. Defined failure characteristics are passed together with the specification of the location (mounting place) where the failure occurred.

ADVICE

`duplicateSerialNumber` = ***1*** *permits the booking of an entire panel! This API function is influenced by station parameter code nos.: 4015, 7202, 7209, 7320, 8310, 8400 and 16000 as well as site parameter 18200. In addition, the station settings for "Upload Settings" and "Setup" control this function. Further information on this topic can be found in the documentation for the ADM Module!*
This API function should be used for new installations, as the functionality has been extended compared to the function »uploadStateAndFailData«!

If the passed serial number is unknown, a suitable work order may be assigned based on the part number (**`productNo`**) and BOM version (**`productRev`**) passed. If no suitable work order can be found or if the order quantity has already been reached, a new work order may be created (station parameter code number 4015) if the "automatic work order creation" is active.

If a **-1** is passed for **`serialNumberState`**, failure data can be passed without a new state booking. This makes sense if failure data are to be updated without state booking. The requirement for this, however, is that a previous state booking must be found for the work step. Otherwise, the upload aborts with an error (`-395`).

Function declaration (CORBA)

```
long uploadStateAndFailureData(
     in string stationNr,
     in long processLayer,
     in string productNo,
     in string productRev,
     in string serialNumber,
     in string serialNumberPos,
     in string posDescr,
     in long serialNumberState,
     in long duplicateSerialNumber,
     in CompFailDataArray compFailDataArray,
     in float cycleTime,
     in long bookDate,
     out long loopCounter,
     out string errorString)
raises (ServerException);

struct CompFailData
     {
        string compName;
        string compPartNo;
        string failureType;
        string failureCause;
        string infoText;
        long processLayer;
        long repair;
     };
typedef sequence<CompFailData> CompFailDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent ***Special case:*** *Depending on the setting of configuration parameter code no. 8310, the work step number (AVO) used in an ERP system can be passed directly instead of the PCB orientation!*
productNo	Part no. of the product in iTAC.MES.Suite ***Advice:*** *This field is only evaluated if the serial numbers passed are unknown (see explanation above)!*
productRev	Bill-of-material version from iTAC.MES.Suite ***Advice:*** *This field is only evaluated if the serial numbers passed are unknown (see explanation above)!* ***Special case:*** *If the automatic work order creation is active (code number 4015), the value [-1] would assign the current BOM version to the new work order!*
serialNumber	Serial no. of the panel
serialNumberPos	Position of the single panel in the unit-array ***Advice:*** *If the value [-1] is passed, the position is not evaluated; in this case, each single panel must be specified by means of a unique* `serialNumber` *!*
posDescr	Description of the panel position ***Advice:*** *This field is used to create new serial nos., i.e., applies only to process steps during which the serial no. is made known for the first time - no mandatory field!*
serialNumberState	State of the product associated with the serial no. [0; 1; 2; -1]: **0** = Pass **1** = Fail **2** = Scrap **-1** = No state booking ***Advice****: If the value* ***-1*** *is passed, no state booking is carried out; the* `loopCounter` *remains unchanged and the most recently booked state remains valid!*
duplicateSerialNumber	Referencing to a single panel [0; 1]: **0** = Action is valid for the specified serial no. only (a position of the multiple printed panel) **1** = Action is carried out on all stored single panels, based on the reference serial number provided
compFailDataArray	Array with the following variables:
`compName`	Fault location (mounting place) is consistent with the iTAC.MES.Suite BOM
`compPartNo`	Part number of the faulty component is consistent with the iTAC.MES.Suite BOM
`failureType`	Failure type ***Advice:*** *The failure types must be created in advance in iTAC.MES.Suite and must be assigned to the station (see ADM Module)!*

TAB. 3-465 *Parameter (in) »uploadStateAndFailureData«*

Parameter name	Explanation
failureCause	Failure cause; reason that led to the failure ***Advice:*** *The failure causes must be created in advance in iTAC.MES.Suite and must be assigned to the station (see ADM Module)!*
infoText	Output of an information text
processLayer	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent
repair	Repair state [0; 1]: **0** = Not repaired **1** = Repaired
cycleTime	Duration of the check in seconds
bookDate	Date of the state booking; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20* ***Special case:*** *[-1] passes the current date and the current time!*

TAB. 3-465 *Parameter (in) »uploadStateAndFailureData«*

Parameter (out)

Parameter name	Explanation
loopCounter	Number of bookings against this serial no. and this work step (stations of the associated ERP group) ***Advice:*** *Only bookings of stations of the type "tester" are counted; bookings of repair and diagnosis stations are* ***not*** *counted!*
errorString	Output text according to output value (see table)

TAB. 3-466 *Parameter (out) »uploadStateAndFailureData«*

Output values

Value	Explanations (errorString)
4	Warning:at least one serial no. is scrap ***Advice:*** *Value is only returned if a multiple panel is booked in which at least one but less than all single panels have been scrapped. In addition, the check for scrap bookings must be activated with site parameter code no. 18200!*
0	OK: [empty]
-1	Error: no active workOrder found
-2	Error: serialNumber not found
-3	Error: station not found
-4	Error: serialNumber is assigned to another workOrder
-5	Error: serialNumberState not valid [0; 1; 2]
-6	Error: station is not valid for this serialNumber
-7	Error: repair flag is ambiguous, data has booked with repair flag N
-8	Error: Component-name not valid
-9	Error: Errortype not valid

TAB. 3-467 *Output values for »uploadStateAndFailureData«*

Value	Explanations (errorString)
-10	Error: An error occurred during reading error types for ppm default values
-11	Error: General server error
-12	Error: Max sequence reached ***Advice:*** *This value depends on the default for "max. recursion" in the upload settings!*
-15	Error: serialNumber not unique
-17	Error: serialnumber is locked
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-210	Error: no unique workstep for this process layer
-375	Error: SerialNo SCRAP on last workstep ***Advice:*** *Value is only returned if a single panel or all panels of a multiple panel have been scrapped. In addition, the check for scrap bookings must be activated with site parameter code no. 18200!*
-395	Error: no previous booking found for this serialnumber and this workstep ***Advice:*** *Value is only returned if a* `-1` *was passed for the* `serialNumber-State` *input parameter!*
-436	Error: serialnumber is archived

TAB. 3-467 *Output values for »uploadStateAndFailureData«*

3.155 uploadStateAndResultData

This function books individual products with their state and measurement data in a work step conformant manner. It sets a stamp for the component lot capture. This is the universal function for booking a product onto a process step and for setting the time stamp for the lot capture.

ADVICE

`duplicateSerialNumber` = ***1*** *permits the booking of an entire panel! The* **uploadStateAndResultData** *API function is influenced by station parameter code nos.: 7202, 7209 - 7215, 7320, 8200, 8310, 8400, 15250, 15260 and 16000 as well as site parameter 18200. In addition, this function is controlled by the station settings for "Upload Settings", and "QA Settings". Further information on this topic can be found in the documentation for the ADM Module!*

If a **-1** is passed for **`serialNumberState`**, measurement data can be passed without a new state booking. This makes sense if failure data have already been booked via another "Upload" function and a measurement data set is to be created for this booking without its own state booking. The requirement for this, however, is that a previous state booking must be found. Otherwise, the upload aborts with an error (`-395`).

Function declaration (CORBA)

```
long uploadStateAndResultData(
     in string stationNr,
     in long processLayer,
     in string serialNumber,
     in string serialNumberPos,
     in long serialNumberState,
     in long duplicateSerialNumber,
     in long numberOfRecords,
     in ResultDataArray resultDataArray,
     in float cycleTime,
     out long loopCounter,
     out string errorString)
raises (ServerException);

struct ResultData
     {
        string name;
        string value;
        long failCode;
     };
typedef sequence<ResultData> ResultDataArray;
```

Parameter (in)

Parameter name	Explanation
`stationNr`	Station no. of the work station in iTAC.MES.Suite
`processLayer`	Orientation of the PCB during the work step [0; 1; 2]: **0** = Component side **1** = Solder side **2** = Position independent ***Special case:*** *Depending on the setting of configuration parameter code no. 8310, the work step number (AVO) used in an ERP system can be passed directly instead of the PCB orientation!*
`serialNumber`	Serial no. of the panel

TAB. 3-468 *Parameter (in) »uploadStateAndResultData«*

Parameter name	Explanation
serialNumberPos	Position of the single panel in the unit-array ***Advice:*** *If the value [-1] is passed, the position is not evaluated; in this case, each single panel must be specified by means of a unique* `serialNumber` *!*
serialNumberState	State of the product associated with the serial no. [0; 1; 2; -1]: **0** = Pass **1** = Fail **2** = Scrap **-1** = No state booking ***Advice****: If the value* ***-1*** *is passed, no state booking is carried out; the* `loopCounter` *remains unchanged and the most recently booked state remains valid!*
duplicateSerialNumber	Referencing to a single panel [0; 1]: **0** = Action is valid for the specified serial no. only (a position of the multiple printed panel) **1** = Action is carried out on all stored single panels, based on the reference serial number provided
numberOfRecords	Default value for the number of entries in the array
resultDataArray	Array with the following variables:
`name`	Name of measurement step
`value`	Measured value
`failCode`	State information for measurement step [0; 1]: **0** = OK **1** = FAIL ***Advice:*** *Other integer values can be interpreted either as pseudo failure (code no.: 15250) or without evaluation (code no.: 15260), depending on the station configuration!*
cycleTime	Duration of the check in seconds

TAB. 3-468 *Parameter (in) »uploadStateAndResultData«*

Parameter (out)

Parameter name	Explanation
loopCounter	Number of bookings against this serial no. and this work step (stations of the associated ERP group) ***Advice:*** *Only bookings of stations of the type "tester" are counted; bookings of repair and diagnosis stations are* ***not*** *counted!*
errorString	Output text according to output value (see table)

TAB. 3-469 *Parameter (out) »uploadStateAndResultData«*

Output values

Value	Explanations (errorString)
4	Warning:at least one serial no. is scrap ***Advice:*** *Value is only returned if a multiple panel is booked in which at least one but less than all single panels have been scrapped. In addition, the check for scrap bookings must be activated with site parameter code no. 18200!*
0	OK: [empty]
-1	Error: no active workOrder found

TAB. 3-470 *Output values for »uploadStateAndResultData«*

Value	Explanations (errorString)
-2	Error: serialNumber not found
-3	Error: station not found
-4	Error: serialNumber is assigned to another workOrder
-5	Error: `serialNumberState` not valid [0; 1; 2; -1]
-6	Error: station is not valid for this serialNumber
-7	Error: no active recipe found
-8	Error: resultdata does not match to active recipe
-10	Error: general error while upload state
-11	Error: general error while upload result data
-12	Error:Max sequence reached ***Advice:*** *This value depends on the default for "max. recursion" in the upload settings!*
-15	Error: serialNumber not unique
-16	Error: fail because of anomaly ***Advice:*** *Value is only returned, if the station parameter code no. 7210 is active!*
-17	Error: serialnumber is locked
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-91	Error: Not enough values for testing ***Advice:*** *Value is only returned if PAA functionality is used (the station parameter code no. 7210 must be active)!*
-93	Error: CP violation
-94	Error: CPK violation
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-210	Error: no unique workstep for this process layer
-375	Error: SerialNo SCRAP on last workstep ***Advice:*** *Value is only returned if a single panel or all panels of a multiple panel have been scrapped. In addition, the check for scrap bookings must be activated with site parameter code no. 18200!*
-395	Error: no previous booking found for this serialnumber and this workstep ***Advice:*** *Value is only returned if a* `-1` *was passed for the* `serialNumber-State` *input parameter!*
-436	Error: serialnumber is archived

TAB. 3-470 *Output values for »uploadStateAndResultData«*

3.156 uploadStationResult

This function logs process data associated with a station. Measurement data are stored for the period specified; the values that were stored previously are set to invalid.

Function declaration (CORBA)

```
long uploadStationResult(
     in string stationNr,
     in long numberOfRecords,
     in ResultDataExtendedStructArray resultDataArray,
     in long startDate,
     in long createRecipe,
     in long snrTrace,
     out string errorString)
raises (ServerException);

struct ResultDataExtendedStruct
     {
        float nr;
        string name;
        string value;
        string min;
        string max;
        string nom;
        string toleranz;
        string unit;
        long failCode;
        string messtyp;
     };
typedef sequence<ResultDataExtendedStruct> ResultDataExtended-
StructArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
numberOfRecords	Default value for the number of entries in the array
resultDataExtArray	Array with the following variables:
nr	Testing step no. in the recipe
name	Name of measurement step
value	Measured value; the following format definitions apply: **Binary** (Example: 0b110010) `0b[0-1]*` **Octal** (Example: 0o173210) `0o[0-7]*` **Hexadecimal** (Example: 0x05FB7A) `0x[0-9A-Fa-f]*` **Decimal EN** (Example: 1366.665) `[0-9]*.[0-9]*` **Decimal DE** (Example: 1366,665) `[0-9]*,[0-9]*` **Text** (Example: Hello world) `.*`
min	Lower limit for measurement step ***Advice:*** *Alternatively, the limit value can also be defined with the* `nom` *and* `toleranz` *parameters!*
max	Upper limit for measurement step ***Advice:*** *Alternatively, the limit value can also be defined with the* `nom` *and* `toleranz` *parameters!*

TAB. 3-471 *Parameter (in) »uploadStationResult«*

Parameter name	Explanation
nom	Nominal value ***Advice:*** *Alternatively, the limit value can also be defined with the* `min` *and* `max` *parameters; the* `min` *and* `max` *parameters override this setting!*
toleranz	Tolerance value for measurement step ***Advice:*** *Alternatively, the limit value can also be defined with the* `min` *and* `max` *parameters; the* `min` *and* `max` *parameters override this setting!*
unit	Unit for the value passed
failCode	State information for measurement step [0; 1]: **0** = OK **1** = FAIL ***Advice:*** *Other integer values can be interpreted either as pseudo failure (code no.: 15250) or without evaluation (code no.: 15260), depending on the station configuration!*
messtyp	Type of the measurement step; optional with format specification **CO** = Config **RE** = Result **CR** = Config-Result **DY** = Dynamic Supported formats: **TXT** = text **BIN** = binary **OCT** = octal **HEX** = hexadecimal **DEN** = decimal EN "." **DDE** = decimal DE "," When a format specification is passed, it is attached to the type (with separator "**|**", e.g., **CR|HEX**). If no format is passed, the format is detected automatically. ***Advice:*** *If no or a different format is entered, the measurement step type "Config-Result" with automatic format detection is used!*
startDate	Start date for the setup (start of production); the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20* **Special cases:** **a.)** If the value [-1] is passed, the current date and the current time are used as the start date! **b.)** If the value passed is between [1] and [31536000], i.e., within the period "one year", it is used as offset; in this case, the start date corresponds to the current date/time plus the period passed here

TAB. 3-471 *Parameter (in) »uploadStationResult«*

Parameter name	Explanation
`createRecipe`	System behavior "recipe creation" [0; 1; 2]: **0** = In general, the active recipe is used and all measured values associated with this recipe are stored. If not all measurement points are available, the residual measurement points are added automatically (via the recipe) and are filled with the values of the previous booking. If there are measurement points that do not exist in the recipe, they are not stored in the system (they expire). **1** = It is assumed that the passed measurement points contain a complete measurement log! If no suitable recipe can be found, a new recipe is created. **2** = The current recipe for the station is ascertained. If one or more of the measured values passed are not included with their parameters, a new recipe is created as a copy of the active recipe and extended by the new value(s).
`snrTrace`	System behavior "serial number link" [0; 1]: **0** = Serial numbers are **not** linked with the previous process data. **1** = All serial numbers which have a state booking for the passed station in the previous time interval are linked with the process data.

TAB. 3-471 *Parameter (in) »uploadStationResult«*

Parameter (out)

Parameter name	Explanation
`errorString`	Output text according to output value (see table)

TAB. 3-472 *Parameter (out) »uploadStationResult«*

Output values

Value	Explanations (errorString)
`0`	OK: [empty]
`-1`	Error: create Processdata
`-2`	Error: stationNr not found
`-3`	Error: `snrTrace` not valid
`-4`	Error: `createRecipe` not valid
`-5`	Error: no recipe found/created
`-6`	Error: `resultData` is empty - not valid
`-7`	Error: `resultData` is not similar to recipeData
`-78`	Error: no TR license for this station
`-99`	Error: Server error
`-100`	Error: Corba no Service
`-101`	Error: Corba Exception

TAB. 3-473 *Output values for »uploadStationResult«*

3.157 uploadStationState

This function logs the current condition/state of a station. The state of the station from the point of time specified is stored; the previously defined states become invalid.

Function declaration (CORBA)

```
long uploadStationState(
     in string stationNr,
     in long date,
     in string stationState,
     in long withHistory,
     in string infoText,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
date	Start date for the status logging; the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20* ***Special case:*** *If the value [-1] is passed, the current date and the current time are used as the start date!*
stationState	Wage type defined in iTAC.MES.Suite (e.g., 100) that must be booked by the station
withHistorie	System behavior "history booking" [0; 1]: **0** = Only the current state is booked without the previous bookings being stored in a history. **1** = The previous bookings for the passed station are stored in the overhead cost table as fault time / down-time. The booking time is used as the start-up date, and the new point in time minus one second as expiry date.
infotext	Input of an arbitrary information text

TAB. 3-474 *Parameter (in) »uploadStationState«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-475 *Parameter (out) »uploadStationState«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: general Error
-2	Error: stationNr not found
-3	Error: `withHistory` not valid
-6	Error: `stationState` not valid
-7	Error: `date` is not valid

TAB. 3-476 *Output values for »uploadStationState«*

Value	Explanations (errorString)
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-476 *Output values for »uploadStationState«*

3.158 uploadStationWasteTime

This function books downtimes to a station. These times are booked in the overhead cost table as fault time / downtime for the passed station (see also function »uploadStationState«).

Function declaration (CORBA)

```
long uploadStationWasteTime(
     in string stationNr,
     in long dateFrom,
     in long dateTo,
     in string stationState,
     in string infoText,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
dateFrom	Start time of the downtime (start-up); the total time is specified as the number of seconds since 01-01-1970, 0:00:00 ***Example:*** *1100000000 sec. (1.1 bn) correspond to 09-11-2004, 12:33:20*
dateTo	End time for the downtime (finish), the total time is specified as the number of seconds since 01-01-1970
stationState	Wage type defined in iTAC.MES.Suite (e.g., 100) that must be booked by the station
infotext	Input of an arbitrary information text

TAB. 3-477 *Parameter (in) »uploadStationWasteTime«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-478 *Parameter (out) »uploadStationWasteTime«*

Output values

Value	Explanations (errorString)
0	OK: [empty]
-1	Error: general Error
-2	Error: stationNr not found
-6	Error: `stationState` not valid
-7	Error: `dateFrom/dateTo` is not valid
-50	Error: no user registered at station ***Advice:*** *Value is only returned, if the station parameter code no. 8300 is active!*
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception

TAB. 3-479 *Output values for »uploadStationWasteTime«*

3.159 verifyMerge

This function checks whether a slave serial number may be installed according to BOM in the product version currently registered at a station. It is also checked whether the slave serial number has been completely assembled and is not in the "fail" state.

ADVICE

This API function is influenced by station parameter code nos.: 7203, 7217, 7218 and 13010. Further information on this topic can be found in the documentation for the ADM Module!
If a check is to be performed independent of the current integration level, the »verifyMergeProduct« function must be used.

Function declaration (CORBA)

```
long verifyMerge(
     in string stationNr,
     in string serialNumberSlave,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
serialNumberSlave	Serial no. that is deleted if the products are merged

TAB. 3-480 *Parameter (in) »verifyMerge«*

Parameter (out)

Parameter name	Explanation
errorString	Output text according to output value (see table)

TAB. 3-481 *Parameter (out) »verifyMerge«*

Output values

Value	Explanations (errorString)
2	OK: Serialnumber Slave - State SCRAP
1	OK: Serialnumber Slave - State FAIL
0	OK: [empty]
-1	Error: serialNumber not found
-2	Error: merge is not valid
-3	Error: station not found
-4	Error: no active workOrder found
-5	Error: Serialnumber Slave not finished
-6	Error: Serialnumber Slave is scrap at one of the previous workstep
-7	Error: Serialnumber Slave is fail at one of the previous workstep
-8	Error: Serialnumber Slave not pass
-17	Error: Serialnumber Slave is locked
-78	Error: no TR license for this station
-99	Error: Server error

TAB. 3-482 *Output values for »verifyMerge«*

Value	Explanations (errorString)
-100	Error: Corba no Service
-101	Error: Corba Exception
-393	Error: materialBin exists but is empty ***Advice:*** *This value is only output if station parameter code no. 7217 is activated for the station!*
-394	Error: Serialnumber is not complete ***Advice:*** *This value is only output if station parameter code no. 7218 is activated for the station (activates/deactivates the check to determine whether all components are mounted)!*

TAB. 3-482 *Output values for »verifyMerge«*

3.160 verifyMergeBySnrRef

This function checks for a multiple panel and the single panels contained therein whether a slave serial no. can be mounted into the currently registered product version at a station.

ADVICE

The **verifyMergeBySnrRef** *API function is influenced by station parameter code nos.: 7203, 7209, 7218 and 13010. Further information on this topic can be found in the documentation for the ADM Module!*

Function declaration (CORBA)

```
long verifyMergeBySnrRef(
     in string stationNr,
     in string serialNumberSlave,
     out long numberOfRecords,
     out StateDataArray testDataArray,
     out string errorString)
raises (ServerException);

struct StateData
     {
        string serialNumber;
        string serialNumberPos;
        long loopCounter;
        long state;
     };
typedef sequence<StateData> StateDataArray;
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
serialNumberSlave	Serial no. that is deleted if the products are merged

TAB. 3-483 *Parameter (in) »verifyMergeBySnrRef«*

Parameter (out)

Parameter name	Explanation
numberOfRecords	Number of entries in the array returned
testDataArray	Array with the following variables:
serialNumber	Serial no. of the panel
serialNumberPos	Position of the single panel in the unit-array
loopCounter	Number of bookings against this serial no. and this work step (stations of the associated ERP group) ***Advice:*** *Only bookings of stations of the type "tester" are counted; bookings of repair and diagnosis stations are* ***not*** *counted!*
state	State of the serial no. (uploadState) [0; 1; 2; -1]: **0** = Pass **1** = Fail **2** = Scrap ***Special case:*** *[-1] means unchecked!*
errorString	Output text according to output value (see table)

TAB. 3-484 *Parameter (out) »verifyMergeBySnrRef«*

Output values

Value	Explanations (errorString)
***Advice:** The value for the* `errorString` *is calculated from all single panels! Here, the worst single panel state determines the total panel state (worst case).*	
`2`	OK: Serialnumber Slave - State SCRAP
`1`	OK: Serialnumber Slave - State FAIL
`0`	OK: [empty]
`-1`	Error: serialNumber not found
`-2`	Error: merge is not valid
`-3`	Error: station not found
`-4`	Error: no active workOrder found
`-5`	Error: Serialnumber Slave not finished
`-6`	Error: serialnumber slave is scrap at one of the previous workstep
`-7`	Error: serialnumber slave is fail at one of the previous workstep
`-17`	Error: Serialnumber Slave is locked
`-78`	Error: no TR license for this station
`-99`	Error: Server error
`-100`	Error: Corba no Service
`-101`	Error: Corba Exception
`-394`	Error: Serialnumber is not complete ***Advice:** This value is only output if station parameter code no. 7218 is activated for the station (activates/deactivates the check to determine whether all components are mounted)!*

TAB. 3-485 *Output values for »verifyMergeBySnrRef«*

3.161 verifyMergeProduct

This function checks whether it is permitted to install the unit with the serial number passed (`serialNumberSlave`) into the product (part number of the product passed and product version). It is also checked whether the slave serial number has been completely assembled and is not in the "fail" state.

If the mounting check is to be logged in the system, this function can also be used to trigger a state booking (see variable `booking`).

ADVICE

This API function is influenced by station parameter code nos.: 7203, 7217 and 7218. Further information on this topic can be found in the documentation for the ADM Module!

Unlike the »verifyMerge« function, which only performs a check for the next integration level, this checks whether the slave serial no. may be mounted anywhere within the product structure tree; i.e. multiple integration levels can be skipped.

EXAMPLE

a.) This is a product with five integration levels.
b.) The slave serial no. is successfully mounted on the lowest level.
The »verifyMergeProduct« function can now be called with the upper four integration levels and returns the output value "0" for each!

Function declaration (CORBA)

```
long verifyMergeProduct(
     in string stationNr,
     in string productId,
     in long bomVersion,
     in string bomIndex,
     in string serialNumberSlave,
     in long booking,
     out string errorString)
raises (ServerException);
```

Parameter (in)

Parameter name	Explanation
stationNr	Station no. of the work station in iTAC.MES.Suite
productId	Part no. of the product in iTAC.MES.Suite
bomVersion	Bill-of-material version from iTAC.MES.Suite ***Special case:*** *If [-1] is passed, the most current version is used automatically!*

TAB. 3-486 *Parameter (in) »verifyMergeProduct«*

Parameter name	Explanation
`bomIndex`	Bill-of-material version (index) that is passed to the system by means of changes in BOMs; these are manual changes or changes that have been initiated by higher-level ERP systems ***Special case:*** *If [-1] is passed, the most current version is used automatically!*
`serialNumberSlave`	Serial no. that is deleted if the products are merged
`booking`	Check with state booking [0; 1]: **0** = Check **without** state booking **1** = Check **with** state booking ***Advice:*** *If the value [1] is passed, the function attempts to execute a state booking for the slave serial no. after the check. For this purpose, the specified station must be contained in the work plan of the slave serial no. (e.g. as last work step "mounting check"). On the basis of this state booking, the system logs whether the »verifyMergeProduct« function was called for the slave serial no.!*

TAB. 3-486 *Parameter (in) »verifyMergeProduct«*

Parameter (out)

Parameter name	Explanation
`errorString`	Output text according to output value (see table)

TAB. 3-487 *Parameter (out) »verifyMergeProduct«*

Output values

Value	Explanations (errorString)
`4`	State REJECT or serialno. not active ***Advice:*** *This output value is returned only if the* `booking` *variable is set to "1"!*
`3`	Not valid for this Station. Part needs to be seen at one of the previous Stations ***Advice:*** *This output value is returned only if the* `booking` *variable is set to "1"!*
`2`	Not valid for this Station. Part needs to be seen at the previous Station first ***Advice:*** *This output value is returned only if the* `booking` *variable is set to "1"!*
`1`	State "FAIL" ***Advice:*** *This output value is returned only if the* `booking` *variable is set to "1"!*
`0`	OK: [empty]
`-1`	Error: Merge is not valid
`-2`	Error: serialNumber not found
`-3`	Error: station not found
`-4`	Error: no valid bom found
`-5`	Error: serialnumber slave not finished
`-6`	Error: `productID` not found
`-7`	Error: serialnumber slave is scrap at one of the previous workstep
`-8`	Error: serialnumber slave is fail at one of the previous workstep
`-9`	Error: serialnumber not valid for uploadState ***Advice:*** *This output value is returned only if the* `booking` *variable is set to "1"!*
`-17`	Error: serialnumber slave is locked

TAB. 3-488 *Output values for »verifyMergeProduct«*

Value	Explanations (errorString)
-78	Error: no TR license for this station
-99	Error: Server error
-100	Error: Corba no Service
-101	Error: Corba Exception
-393	Error: materialBin exists but is empty ***Advice:*** *This value is only output if station parameter code no. 7217 is activated for the station!*
-394	Error: Serialnumber is not complete ***Advice:*** *This value is only output if station parameter code no. 7218 is activated for the station (activates/deactivates the check to determine whether all components are mounted)!*

TAB. 3-488 *Output values for »verifyMergeProduct«*

Index

M

Q

R

S

T

U

V