

OPC UA Status Code Diagnostic Tree

All 127 normative status codes, grouped by what actually went wrong — with the first thing to do for each. Built from the reference at opcuaconnect.com.

Start here: the four questions that resolve most faults

1	Does the connection open at all?	No → Transport and secure channel , then Certificates and PKI . A connection that never opens is almost always the endpoint URL, the port, or the certificate — not the address space.
2	Does it open but refuse you?	→ Security policy and identity . The server is reachable and its certificate is fine; it does not accept the mode, policy or token you offered. GetEndpoints tells you what it will accept.
3	Are you connected but a specific node fails?	→ Address space and nodes , then Read, write and access . Isolate: does Browse find the node? Does Read work but Write fail? That split names the code.
4	Does everything work, but the data is wrong or stops?	→ Subscriptions and monitored items , Field, device and data quality , and the Uncertain family. This is where silent OEE corruption lives — the session is healthy, the numbers are not.

Three rules worth more than the table

Severity is in the name, and it is binary in practice.

Bad means the value is unusable — never fall back to the last Good value without marking the gap. Uncertain means a value arrived but the server does not vouch for it; treating Uncertain as Good is how a dead sensor becomes a month of clean-looking production data.

The code tells you the layer, not the fix.

BadTimeout on a Browse and BadTimeout on a Publish are different problems. Always record which service call produced the code — that halves the search before you read a word of this document.

Codes in the Good family still change your logic.

GoodNoData, GoodMoreData and GoodOverload are successes that mean 'you did not get everything'. Client code that only tests for Bad silently drops data on all three.

Certificates and PKI (18 codes)

Status code	What it means	First thing to do
BadApplicationSignatureInvalid	BadApplicationSignatureInvalid means the client's ActivateSession signature over the server certificate and server nonce did not verify. The client either signed with a key that does not match the certificate it presented, signed over a stale nonce, or presented a different certificate than the one used to open the channel.	In diagnostic order — work down, do not skip ahead. Verify the certificate and key pair <code>openssl x509 -in cert.pem -noout -modulus openssl md5</code> and <code>openssl rsa -in key.pem -noout -modulus openssl md5</code> must match.
BadCertificateChainIncomplete	BadCertificateChainIncomplete means the validator could not build a path from the peer certificate up to a trust anchor because an intermediate CA certificate is missing. Put the missing intermediates in the issuer store, or configure the peer to send its full chain in OpenSecureChannel.	In diagnostic order — work down, do not skip ahead. Identify the issuer of the failing certificate Export the peer certificate from the rejected store and run <code>openssl x509 -in peer.der -inform der -noout -issuer -subject</code> .
BadCertificateHostNameInvalid	BadCertificateHostNameInvalid means the host part of the endpoint URL you dialled does not appear in the server certificate's Subject Alternative Name. Either connect using a name that is in the SAN, or reissue the certificate with every DNS name and IP address clients will actually use.	In diagnostic order — work down, do not skip ahead. Read the SAN out of the server certificate Export the server certificate (UaExpert's Manage Certificates dialog, or the file in the rejected store) and run <code>openssl x509 -in server.der -inform der -noout -text</code> , then look at the X509v3 Subject Alternative Name block.
BadCertificateInvalid	BadCertificateInvalid means the certificate failed a structural check, not a trust check. Modern stacks reject an empty subject or issuer distinguished name outright and this condition is generally not suppressible.	In diagnostic order — work down, do not skip ahead. Parse the certificate outside the stack Read the certificate with a standard X.509 tool and confirm it parses at all.
BadCertificateIssuerRevocationUnknown	BadCertificateIssuerRevocationUnknown means revocation status could not be determined for a CA certificate in the chain, not for the peer certificate itself. Install a current CRL for every CA above the failing certificate, including the root, in the crl folder of the store holding that CA.	In diagnostic order — work down, do not skip ahead. Enumerate the whole chain Print the chain with <code>openssl x509 -in peer.der -inform der -noout -issuer</code> , then repeat for each CA certificate you hold until you reach a self-signed root.
BadCertificateIssuerRevoked	BadCertificateIssuerRevoked means a CA in the chain has itself been revoked, so every certificate below it is untrustworthy. This is not suppressible.	In diagnostic order — work down, do not skip ahead. Identify the revoked CA and confirm the revocation Dump the full chain the peer presented and check each CA certificate's serial against the CRL of its own issuer.
BadCertificateIssuerTimeInvalid	BadCertificateIssuerTimeInvalid means a CA in the chain is expired or not yet valid, not the peer certificate itself. Check the CA's validity dates and the clocks on both machines.	In diagnostic order — work down, do not skip ahead. Dump the whole chain and read every validity period Print not-before and not-after for the application certificate and for each CA above it.
BadCertificateIssuerUseNotAllowed	BadCertificateIssuerUseNotAllowed means a CA certificate in the chain does not assert the key usage bits OPC UA requires. Every CA certificate must carry both keyCertSign and cRLSign; corporate CAs that omit cRLSign trip this.	In diagnostic order — work down, do not skip ahead. Dump the KeyUsage of every CA in the chain For each CA certificate run <code>openssl x509 -in ca.der -inform der -noout -text</code> and read the X509v3 Key Usage and X509v3 Basic Constraints blocks.
BadCertificatePolicyCheckFailed	BadCertificatePolicyCheckFailed means the certificate is structurally fine but fails your cryptographic policy - usually an RSA key below 2048 bits or a SHA-1 signature while SHA-1 rejection is enabled. Reissue the certificate with a 2048-bit or larger key and a SHA-256 signature rather than relaxing the policy.	In diagnostic order — work down, do not skip ahead. Inspect the certificate's key size and signature algorithm Dump the certificate and read the public key length and the signature algorithm.
BadCertificateRevocationUnknown	BadCertificateRevocationUnknown means the validator could not determine the revocation status of the peer certificate, almost always because no CRL for its issuing CA is present in the crl store. Export a current CRL from that CA and install it alongside the CA certificate.	In diagnostic order — work down, do not skip ahead. Confirm which CA needs a CRL Read the issuer of the failing certificate: <code>openssl x509 -in peer.der -inform der -noout -issuer</code> .
BadCertificateRevoked	BadCertificateRevoked means the peer's certificate is listed on a CRL from its issuer. This is not suppressible - you cannot click through it.	In diagnostic order — work down, do not skip ahead. Read the certificate's serial number and confirm it is on the CRL Extract the serial from the rejected certificate and check it against the CRL in your issuer store.

Status code	What it means	First thing to do
BadCertificateTimeInvalid	BadCertificateTimeInvalid means the peer certificate's validity window does not contain the current time. Either the certificate really has expired, or one machine's clock is wrong.	In diagnostic order — work down, do not skip ahead. Print the validity window Run openssl x509 -in peer.der -inform der -noout -dates on the certificate from the rejected store.
BadCertificateUntrusted	BadCertificateUntrusted means the peer's application instance certificate is not in your trusted store, so validation stopped at the trust check. Fix it by moving that certificate out of the rejected store into trusted - on both client and server, because each side validates the other independently.	In diagnostic order — work down, do not skip ahead. Work out which side rejected A client-side error names the server certificate; a server-side rejection usually surfaces to the client as a channel that opens then closes, with the real reason only in the server log.
BadCertificateUriInvalid	BadCertificateUriInvalid means the ApplicationUri the peer declares does not match the URI entry in its certificate's Subject Alternative Name. The two must be identical strings.	In diagnostic order — work down, do not skip ahead. Print both strings Read the certificate's URI SAN with openssl x509 -in cert.der -inform der -noout -text and note the URI: entry.
BadCertificateUseNotAllowed	BadCertificateUseNotAllowed means the peer certificate's key usage or extended key usage does not permit how it is being used. An application instance certificate needs digitalSignature and, for RSA policies, key and data encipherment, plus client or server authentication in extendedKeyUsage.	In diagnostic order — work down, do not skip ahead. Read the usage extensions from the failing certificate Run openssl x509 -in cert.der -inform der -noout -text and read X509v3 Key Usage and X509v3 Extended Key Usage.
BadNonceInvalid	BadNonceInvalid means the nonce supplied in the handshake is the wrong length or is not acceptable as a random value. The required length is tied to the negotiated security policy, so a zero-length or fixed nonce sent when a secure policy is in force is rejected.	In diagnostic order — work down, do not skip ahead. Establish which side generates the rejected nonce The client nonce is sent in OpenSecureChannel and in ActivateSession; the server nonce comes back in the responses.
BadSecurityChecksFailed	BadSecurityChecksFailed is the generic verification failure: a signature did not verify, a message did not decrypt, a nonce did not match, or the certificate presented at session creation is not the one the channel was opened with. Check clocks, certificate and private key pairing, and that the client is not switching certificates mid-handshake.	In diagnostic order — work down, do not skip ahead. Establish where in the handshake it fails Enable stack tracing on the client and note whether the last successful step was OpenSecureChannel, CreateSession or ActivateSession.
BadUserSignatureInvalid	BadUserSignatureInvalid means the signature that accompanies your user identity token did not verify at ActivateSession. It applies to X.509 user certificate tokens and to issued tokens that carry a signature.	In diagnostic order — work down, do not skip ahead. Confirm the user certificate and key are a pair openssl x509 -in user.pem -noout -modulus openssl md5 against openssl rsa -in userkey.pem -noout -modulus openssl md5.

Security policy and identity (7 codes)

Status code	What it means	First thing to do
BadIdentityChangeNotSupported	BadIdentityChangeNotSupported means the server will not let you swap the user on an existing session. Re-activating with a different identity is permitted by the standard but many servers refuse it.	In diagnostic order — work down, do not skip ahead. Use one session per identity Open a separate session for each identity you need: an anonymous or service-account session for continuous data collection, and a short-lived authenticated session for operator actions.
BadIdentityTokenInvalid	BadIdentityTokenInvalid is about the shape of the token, not the credentials. The token type or its policy id does not match what the endpoint offers, or its encrypted password was formed with the wrong policy.	In diagnostic order — work down, do not skip ahead. Call GetEndpoints and read the token policies For the endpoint you intend to use, list its UserIdentityTokens entries with their TokenType, PolicyId, SecurityPolicyUri and IssuerEndpointUrl.
BadIdentityTokenRejected	BadIdentityTokenRejected means the token was the right shape and the credentials were wrong or not accepted. Wrong password, unknown user, disabled account, expired issued token, or a user certificate not in the trusted user store.	In diagnostic order — work down, do not skip ahead. Check for a lockout before retrying If several failed attempts have already occurred, further attempts may be pointless and may extend the lockout.

Status code	What it means	First thing to do
BadSecurityModeInsufficient	BadSecurityModeInsufficient means the connection succeeded but this particular operation demands more security than the channel has. Configuration services, trust list updates and nodes carrying an encryption AccessRestriction typically require SignAndEncrypt.	In diagnostic order — work down, do not skip ahead. Check what mode your session is actually using Print the SecurityMode and SecurityPolicyUri of the endpoint the session was created on, not the one in your configuration file.
BadSecurityModeRejected	BadSecurityModeRejected means the server will not accept the MessageSecurityMode you asked for on that endpoint, most often because you requested None on a server that requires Sign or SignAndEncrypt, or asked for a mode that endpoint does not publish. Read the endpoint list and use a mode it actually offers.	In diagnostic order — work down, do not skip ahead. Enumerate the endpoints before configuring anything GetEndpoints with security None against opc.tcp://host:4840 returns every published combination.
BadSecurityPolicyRejected	BadSecurityPolicyRejected means the server does not offer, or has disabled, the security policy your client asked for. Call GetEndpoints with security None, read the SecurityPolicyUri values the server actually advertises, and configure the client to one of them.	In diagnostic order — work down, do not skip ahead. List what the server actually offers Call GetEndpoints with security None against opc.tcp://host:4840 and print SecurityPolicyUri and SecurityMode for every endpoint.
BadUserAccessDenied	BadUserAccessDenied means you are authenticated but not authorised. The user's roles do not grant the operation on that node, or an AccessRestriction requires encryption you are not using.	In diagnostic order — work down, do not skip ahead. Confirm which identity the session is really using Read the server's session diagnostics for your session and check the user identity.

Transport and secure channel (21 codes)

Status code	What it means	First thing to do
BadConnectionClosed	BadConnectionClosed means the transport connection carrying your requests was closed - by the peer, by the network, or by a middlebox. The secure channel and any in-flight requests die with it.	In diagnostic order — work down, do not skip ahead. Correlate with network and server events Line up the timestamp against switch logs, VPN events, server restarts and container events.
BadConnectionRejected	BadConnectionRejected means the client could not open a network connection at all: nothing is listening on that host and port, a firewall dropped or refused it, or the address does not resolve. Establish TCP reachability with nc or Test-NetConnection before touching endpoints, certificates or credentials.	In diagnostic order — work down, do not skip ahead. Test TCP reachability nc -vz plc01 4840 on Linux or macOS, Test-NetConnection plc01 -Port 4840 on Windows.
BadEncodingLimitsExceeded	BadEncodingLimitsExceeded means a message hit a stack encoding limit: MaxMessageSize, MaxChunkCount, MaxArrayLength, MaxStringLength or MaxByteStringLength. It is enforced on both sides, so raise the limit on whichever end is complaining, or split the operation into smaller batches - usually the better fix.	In diagnostic order — work down, do not skip ahead. Establish which side and which limit is complaining The client log usually says whether the failure happened while encoding a request or decoding a response.
BadEndpointUrlInvalid	BadEndpointUrlInvalid means the endpoint URL you supplied is not one the server accepts - wrong scheme, wrong host, wrong port, or an endpoint the server does not advertise. Call GetEndpoints with security None first and connect using an EndpointUrl the server actually returned.	In diagnostic order — work down, do not skip ahead. Prove TCP reachability first Run nc -vz plc01 4840 or Test-NetConnection plc01 -Port 4840.
BadNotConnected	BadNotConnected means the node exists in the address space but has never been wired to a source. It is a configuration state, not a runtime failure: a tag defined without an item path, an alias never bound, or a placeholder node in a model.	In diagnostic order — work down, do not skip ahead. Confirm the node has never had a value Read the node and look at the timestamps.

Status code	What it means	First thing to do
BadProtocolVersionUnsupported	BadProtocolVersionUnsupported means the two ends could not agree on the OPC UA TCP protocol version during the Hello and Acknowledge exchange. In practice it rarely means a real version gap: it means you are talking opc.tcp to something that is not an opc.tcp listener, or a middlebox mangled the handshake.	In diagnostic order — work down, do not skip ahead. Prove what is actually on the port Capture the first packets with the Wireshark OPC UA dissector on the port in question.
BadRequestHeaderInvalid	BadRequestHeaderInvalid means the server rejected the RequestHeader on your service call: a missing or wrong AuthenticationToken, an unusable Timestamp, a bad RequestHandle or TimeoutHint, or an AdditionalHeader it cannot decode. It usually indicates a hand-built request or a client and server disagreeing about encoding.	In diagnostic order — work down, do not skip ahead. Check the session state before blaming the header Confirm the session is created and activated and that you are sending its current authentication token.
BadRequestInterrupted	BadRequestInterrupted means your request did not complete because the transport was interrupted while it was in flight. The server may or may not have executed it.	In diagnostic order — work down, do not skip ahead. Classify the operation before retrying anything Read and Browse are safe to repeat.
BadRequestTimeout	BadRequestTimeout means a single service call ran out of time. The RequestHeader carries a TimeoutHint and the server may abandon work that exceeds it.	In diagnostic order — work down, do not skip ahead. Split the operation before raising any timer Batch Reads and Writes into chunks the server can answer quickly, and browse with MaxReferencesPerNode plus BrowseNext instead of asking for everything at once.
BadRequestTooLarge	BadRequestTooLarge means the server refused your request because its encoded size exceeded the server's receive limit. Split the operation into smaller batches sized from Server_ServerCapabilities_OperationLimits, rather than raising limits.	In diagnostic order — work down, do not skip ahead. Read the server's operation limits Browse Server_ServerCapabilities_OperationLimits and read MaxNodesPerRead, MaxNodesPerWrite, MaxNodesPerBrowse, MaxMonitoredItemsPerCall and the related properties.
BadResponseTooLarge	BadResponseTooLarge means the answer the server assembled will not fit inside the negotiated message limits. Reduce what you ask for: fewer nodes per Read, MaxReferencesPerNode with BrowseNext, MaxNotificationsPerPublish on subscriptions, and continuation points on HistoryRead.	In diagnostic order — work down, do not skip ahead. Bound the request that produced the response Set MaxReferencesPerNode on Browse and follow with BrowseNext.
BadSecureChannelClosed	BadSecureChannelClosed means the UASC secure channel carrying your request no longer exists - it was closed by either peer, or the underlying socket dropped. The session may still be valid: open a new secure channel and re-bind or recreate the session, then transfer subscriptions rather than rebuilding everything.	In diagnostic order — work down, do not skip ahead. Confirm whether the socket or the channel went first Correlate the client fault with the TCP state.
BadSecureChannelIdInvalid	BadSecureChannelIdInvalid means the SecureChannelId in your message header does not match any channel the server holds. It happens after a server restart, after the channel was closed, or when a client sends on a channel that belongs to a different connection.	In diagnostic order — work down, do not skip ahead. Check whether the server restarted Read Server_ServerStatus_StartTime after reconnecting, or check the server log.
BadSecureChannelTokenUnknown	BadSecureChannelTokenUnknown means the channel security token used on your message has expired or is unknown to the peer. Channel tokens are renewable and short-lived by design; the fix is to renew before expiry and to keep client and server clocks in sync so both agree on lifetime.	In diagnostic order — work down, do not skip ahead. Fix the clocks first Put NTP or PTP on the client, the server and any gateway between them, and confirm agreement to within a second.
BadTcpEndpointUrlInvalid	BadTcpEndpointUrlInvalid is the transport-level rejection of the endpoint URL in your Hello message, and in practice it is the path part that is wrong: opc.tcp://host:4840/UA/MyServer versus a bare slash. The host and port are fine - fix the path using the URL returned by GetEndpoints.	In diagnostic order — work down, do not skip ahead. Confirm the socket is open nc -vz host 4840 must succeed.

Status code	What it means	First thing to do
BadTcpMessageTooLarge	BadTcpMessageTooLarge means a message exceeded the size or chunk-count limits negotiated in the Hello and Acknowledge exchange. Either raise MaxMessageSize and MaxChunkCount on both peers, or - usually better - split the operation into batches so individual messages stay small.	In diagnostic order — work down, do not skip ahead. Find the negotiated limits Capture the Hello and Acknowledge on port 4840 with the Wireshark OPC UA dissector, or log them from the stack.
BadTcpMessageTypeInvalid	BadTcpMessageTypeInvalid means a peer received a transport message type it does not accept. In practice it is the reverse-connect signal: a client that does not support ReverseHello returns this code, and per specification the server must then abort its automatic reconnect attempts.	In diagnostic order — work down, do not skip ahead. Confirm the client actually supports reverse connect Check the client's documentation and configuration for a reverse-connect listener.
BadTcpNotEnoughResources	BadTcpNotEnoughResources means the peer could not allocate the memory or buffers needed for your connection or message. It is a capacity failure at the transport layer, common on embedded servers and on clients asking for very large chunk sizes.	In diagnostic order — work down, do not skip ahead. Lower the buffer sizes your client requests Reduce ReceiveBufferSize and SendBufferSize in the client transport configuration to values the server can serve, and lower MaxMessageSize accordingly.
BadTcpSecureChannelUnknown	BadTcpSecureChannelUnknown means the transport layer received a message whose SecureChannelId and TokenId it does not hold. It is the transport-level counterpart of BadSecureChannelIdInvalid and usually means the server restarted, the channel was closed, or a load balancer sent you to a different instance.	In diagnostic order — work down, do not skip ahead. Check for a server restart Read Server_ServerStatus_StartTime after reconnecting, and compare with when your channel was opened.
BadTcpServerTooBusy	BadTcpServerTooBusy is the server refusing a new transport connection because it is at capacity. It is returned early, before any secure channel exists, so it is a load signal rather than a configuration error.	In diagnostic order — work down, do not skip ahead. Stagger the client connects Add a randomised startup delay and bounded exponential backoff with jitter to every client.
BadTimeout	BadTimeout is the general timeout code: something did not answer in time. It is usually raised by your own stack rather than returned by the server, so the first question is which timer fired - a per-request timer, the connect timer, the session timeout or the subscription keep-alive - and only then why.	In diagnostic order — work down, do not skip ahead. Find out which timer fired Turn on stack-level tracing and note the elapsed time.

Sessions (4 codes)

Status code	What it means	First thing to do
BadSessionClosed	BadSessionClosed means the session you are addressing existed but has been closed - by your own CloseSession, by the server on timeout, or by an administrator. Unlike a transient network fault, retrying the same session never works: create and activate a new session, then transfer or rebuild subscriptions.	In diagnostic order — work down, do not skip ahead. Establish who closed it Check your own client log for a CloseSession call first.
BadSessionIdInvalid	BadSessionIdInvalid means the server no longer recognises the SessionId in your request header: the session timed out, was closed, or the server restarted. Stop reusing the handle.	In diagnostic order — work down, do not skip ahead. Read the revised session timeout the server actually granted Log the RevisedSessionTimeout from the CreateSession response, not the value you asked for.
BadSessionNotActivated	BadSessionNotActivated means CreateSession succeeded but ActivateSession never did, so the session carries no verified user identity and the server refuses all other services. Send ActivateSession with a valid client signature and UserIdentityToken before any Read, Browse or CreateSubscription call.	In diagnostic order — work down, do not skip ahead. Confirm ActivateSession was actually sent and what it returned Turn on stack-level tracing or capture with the Wireshark OPC UA dissector on port 4840.

Status code	What it means	First thing to do
BadTooManySessions	BadTooManySessions means the server hit its configured session limit and refused CreateSession. Either the server is genuinely at capacity, or - far more often - dead sessions are still counted because clients reconnect without closing them.	In diagnostic order — work down, do not skip ahead. Count the live sessions on the server Browse Server_ServerDiagnostics_SessionsDiagnosticsSummary and read SessionDiagnosticsArray.

Subscriptions and monitored items (15 codes)

Status code	What it means	First thing to do
BadContinuationPointInvalid	BadContinuationPointInvalid means the continuation point you sent is no longer valid. Servers release them on a timeout, when the session ends, or when their pool is needed elsewhere.	In diagnostic order — work down, do not skip ahead. Page promptly and do not block between pages Issue the next BrowseNext immediately and process the results afterwards, rather than processing each page before continuing.
BadDeadbandFilterInvalid	BadDeadbandFilterInvalid means the deadband in your DataChangeFilter is not usable. Percent deadband requires an EURange property on the node and a value between 0 and 100; absolute deadband requires a non-negative value on a numeric item.	In diagnostic order — work down, do not skip ahead. Read EURange before requesting a percent deadband Browse the Variable for a HasProperty reference to EURange and read it.
BadEventFilterInvalid	BadEventFilterInvalid rejects the EventFilter as a whole rather than one operand. Usual causes are an empty select clause, a where clause whose operator and operand counts do not match, or every select clause failing at once.	In diagnostic order — work down, do not skip ahead. Subscribe with the minimal known-good filter first Create an event item on the Server object with select clauses for EventId, EventType, SourceNode, SourceName, Time, ReceiveTime, Message and Severity taken from BaseEventType, and no where clause.
BadFilterNotAllowed	BadFilterNotAllowed means you attached a filter to a monitored item that cannot carry one. The classic case is a DataChangeFilter on a non-value attribute, or an EventFilter on a data item and vice versa.	In diagnostic order — work down, do not skip ahead. Confirm which attribute the item monitors Log the AttributeId of every item you create.
BadFilterOperandInvalid	BadFilterOperandInvalid means one operand inside your filter is malformed. In an EventFilter that is nearly always a SimpleAttributeOperand whose TypeDefinitionId or BrowsePath does not resolve, or an ElementOperand whose index points at a nonexistent element.	In diagnostic order — work down, do not skip ahead. Read the per-operand results, not just the overall status The response carries an EventFilterResult with a status per select clause and per where-clause element and operand.
BadMonitoredItemFilterUnsupported	BadMonitoredItemFilterUnsupported means the server understood your filter but does not implement it. It is a capability gap, not a syntax error - typically percent deadband, AggregateFilter on a subscription, or the richer content filter operators.	In diagnostic order — work down, do not skip ahead. Establish what the server supports once, in a probe At commissioning, create test items with absolute deadband, percent deadband and a simple where clause, and record which are accepted.
BadMonitoredItemIdInvalid	BadMonitoredItemIdInvalid means the monitored item handle you passed does not exist in that subscription. It comes from stale handles after a rebuild, from handles used against the wrong subscription, or from an item whose creation actually failed.	In diagnostic order — work down, do not skip ahead. Check every per-item result from CreateMonitoredItems Walk the results array and pair each StatusCode with the request you sent.
BadMonitoringModeInvalid	BadMonitoringModeInvalid means the MonitoringMode you asked for is not valid or not accepted for that item. The three legal values are Disabled, Sampling and Reporting.	In diagnostic order — work down, do not skip ahead. Confirm which of the three modes you are actually sending Log the enum by name, not as an integer, at the point of the call.
BadNoContinuationPoints	BadNoContinuationPoints means the server's pool of continuation points is exhausted, usually because clients hold points they never finished or released. Release abandoned points, page one traversal at a time per session, and reduce how much paging you need by filtering the browse.	In diagnostic order — work down, do not skip ahead. Release every continuation point you abandon Add a release path to your traversal code: on exception, on cancellation and on shutdown, call BrowseNext with releaseContinuationPoints set for any point you hold.

Status code	What it means	First thing to do
BadNoSubscription	BadNoSubscription means you called Publish on a session that has no subscriptions at all. It is the normal response to a Publish issued before CreateSubscription, after the last subscription was deleted, or on a fresh session after a reconnect that lost them.	In diagnostic order — work down, do not skip ahead. Order the calls correctly Activate the session, call CreateSubscription, check its status, call CreateMonitoredItems, then start publishing.
BadSubscriptionIdInvalid	BadSubscriptionIdInvalid means the server has no subscription with that id on this session. It is almost always a subscription the server already deleted - lifetime expiry, session loss, or a reconnect that did not transfer it.	In diagnostic order — work down, do not skip ahead. Treat the code as a rebuild trigger, not a retry condition Retrying Publish with the same id will never succeed.
BadTooManyMonitoredItems	BadTooManyMonitoredItems means the server hit its monitored item limit, either per subscription or in total. Read MaxMonitoredItemsPerSubscription from ServerCapabilities, spread items across subscriptions grouped by publishing interval, and remove items nobody consumes.	In diagnostic order — work down, do not skip ahead. Read the advertised limits before you design Browse the Server object to ServerCapabilities and read the maximum monitored items values it publishes.
BadTooManyPublishRequests	BadTooManyPublishRequests is the server telling you to reduce the number of outstanding Publish requests. Publish is a long poll, so clients pipeline several to avoid gaps, and servers cap how many they will hold.	In diagnostic order — work down, do not skip ahead. Set a hard cap on outstanding Publish requests Configure the client's maximum outstanding Publish count explicitly rather than leaving it dynamic.
BadTooManySubscriptions	BadTooManySubscriptions means the server hit its subscription limit. The usual cause is a client that uses a different publishing interval for every group of tags, because each distinct publishing interval forces another subscription.	In diagnostic order — work down, do not skip ahead. Count your distinct publishing intervals Inventory the configuration and list the unique publishing intervals.
GoodSubscriptionTransferred	GoodSubscriptionTransferred is not an error - it is what a correct reconnect looks like. Your subscription has been moved to a new session and will keep delivering there.	In diagnostic order — work down, do not skip ahead. Accept it and do not rebuild Log it at information level and continue.

Address space and nodes (21 codes)

Status code	What it means	First thing to do
BadArgumentsMissing	BadArgumentsMissing means your Call supplied fewer inputArguments than the method declares. OPC UA methods are positional and strict: every declared input must be present, in order, with no omissions and no reliance on defaults.	In diagnostic order — work down, do not skip ahead. Read the method's InputArguments property Browse the Method node for its HasProperty reference to InputArguments and read the value: an array of Argument structures giving Name, DataType, ValueRank and Description for each input, in order.
BadAttributeIdInvalid	BadAttributeIdInvalid means you asked for an attribute the node cannot have. Every node carries NodeId, NodeClass, BrowseName and DisplayName; the rest are class-specific.	In diagnostic order — work down, do not skip ahead. Read NodeClass before anything else NodeClass is present on every node.
BadBoundNotFound	BadBoundNotFound means the server could not find a bounding value needed to answer your read: an at-time or interpolated request needs a recorded value before and after each requested instant, and one side is missing. Widen the search window, or accept the SimpleBounds behaviour instead of interpolation.	In diagnostic order — work down, do not skip ahead. Find the real start of the archive Do a raw read with a very early startTime, numValuesPerNode set to one and no endTime, to get the oldest recorded value for the node.
BadBrowseDirectionInvalid	BadBrowseDirectionInvalid means the BrowseDirection value is not one of the defined ones: Forward, Inverse or Both. It is almost always an out-of-range integer from hand-built request code or a configuration field, not a server problem.	In diagnostic order — work down, do not skip ahead. Use the SDK's named enum Pass the library's BrowseDirection constant rather than an integer.

Status code	What it means	First thing to do
BadDataEncodingInvalid	BadDataEncodingInvalid means the DataEncoding QualifiedName in your Read or monitored item does not belong to that node's DataType. Leave DataEncoding empty to get the server's default, or browse HasEncoding from the DataType node to find the encoding names it really offers.	In diagnostic order — work down, do not skip ahead. Leave DataEncoding unset and retry An empty DataEncoding tells the server to use its default for that node.
BadDataEncodingUnsupported	BadDataEncodingUnsupported means the encoding you asked for is a valid concept but this server will not produce it for that node. Fall back to the server's default encoding, which over opc.tcp is UA Binary, and let your client decode the structure from DataTypeDefinition rather than dictating the wire format.	In diagnostic order — work down, do not skip ahead. Remove the encoding request and use the server default Leave DataEncoding empty on both the ReadValueId and the monitored item create request.
BadIndexRangeInvalid	BadIndexRangeInvalid means the IndexRange string you supplied is malformed or cannot apply to that attribute. IndexRange uses zero-based start:end with commas between dimensions, for example 2:5 or 0:3,1:2.	In diagnostic order — work down, do not skip ahead. Drop the IndexRange and confirm the base operation works Read the node with no IndexRange.
BadIndexRangeNoData	BadIndexRangeNoData means the IndexRange parsed correctly but selected nothing: the array is shorter than the range you asked for, or is currently empty. Read the array without a range to see its real length, then clamp the range to that length.	In diagnostic order — work down, do not skip ahead. Read the node with no IndexRange and look at the actual length The returned Variant carries the real element count.
BadInvalidArgument	BadInvalidArgument means the argument count was right but at least one value was rejected: wrong data type, wrong array rank, out-of-range, or semantically invalid for the machine's current state. Read the inputArgumentResults array in CallMethodResult; it tells you which argument, position by position.	In diagnostic order — work down, do not skip ahead. Read inputArgumentResults, not just the statusCode CallMethodResult carries a StatusCode per input argument plus diagnostic infos.
BadMethodInvalid	BadMethodInvalid means the methodId you passed to Call is not a Method of the objectId you passed with it. The pairing is what fails, not the node itself.	In diagnostic order — work down, do not skip ahead. Browse the object instance for its methods Browse the objectId with browseDirection Forward, referenceTypeId HasComponent, includeSubtypes true and nodeClassMask restricted to Method.
BadNoMatch	BadNoMatch means the request was well formed but nothing satisfied it: a browse path element matched no target, or a query filter matched no node. Diagnose it exactly like BadNotFound - check the QualifiedName namespace, the reference type of the path element, and whether the member is optional on this instance.	In diagnostic order — work down, do not skip ahead. Browse the parent and list the actual children Read BrowseName - with its namespace index - and NodeClass for every child of the node where the path stops.
BadNodeClassInvalid	BadNodeClassInvalid means the node exists but is the wrong kind of node for what you asked. Calling a node that is not a Method, subscribing to events on a node that is not an event notifier, or adding a node with a class that does not match its definition all land here.	In diagnostic order — work down, do not skip ahead. Read NodeClass on the exact node you are targeting One Read settles it.
BadNodeIdInvalid	BadNodeIdInvalid means the NodeId itself is malformed, not merely absent: a null or empty identifier, a namespace index beyond the server's NamespaceArray, or a string that does not parse as a NodeId. Compare it with BadNodeIdUnknown, which means the NodeId was valid but no such node exists.	In diagnostic order — work down, do not skip ahead. Print the NodeId exactly as your stack parsed it Log the parsed NodeId, not the source string.
BadNodeIdUnknown	BadNodeIdUnknown means the NodeId was well formed but the server has no such node. The classic cause is a shifted namespace index after a PLC program download or a server upgrade.	In diagnostic order — work down, do not skip ahead. Read NamespaceArray and map URI to index at connect Read the NamespaceArray property on the Server object at every connect and rebuild your index map before using any cached NodeId.

Status code	What it means	First thing to do
BadNotExecutable	BadNotExecutable means the method is present and correctly addressed but the server will not run it now. Executable reflects the machine's own state; UserExecutable reflects your permissions.	In diagnostic order — work down, do not skip ahead. Read Executable and UserExecutable on the method node These are two distinct attributes on the Method NodeClass.
BadNotFound	BadNotFound is the generic search failure: a browse path element, a discovery lookup or a query found nothing. In day-to-day work it almost always comes from TranslateBrowsePathsToNodeIds, and the cause is almost always a BrowseName whose namespace is wrong or a path element that does not exist on this instance.	In diagnostic order — work down, do not skip ahead. Browse one level at a time to the failing element Start at your anchor and browse forward, reading BrowseName on each child including its namespace index.
BadObjectDeleted	BadObjectDeleted means the node you are addressing existed and has since been removed. Address spaces are not static: job, result, file and alarm objects come and go, and reconfiguration deletes nodes wholesale.	In diagnostic order — work down, do not skip ahead. Subscribe to model change events Create an event monitored item on the Server object and take GeneralModelChangeEvent seriously: when the server reports that nodes were added, deleted or their references changed, invalidate the affected part of your cache and re-resolve.
BadReferenceTypeIdInvalid	BadReferenceTypeIdInvalid means the NodeId you gave as a reference type filter is not a ReferenceType node in that server. Usually it is a hardcoded numeric id from a different server, a namespace index that shifted, or an ObjectType passed where a ReferenceType belongs.	In diagnostic order — work down, do not skip ahead. Resolve reference types by namespace URI and BrowseName at connect Read the server's NamespaceArray, find the index for the namespace URI you need, and build the NodeId from that index.
BadStructureMissing	BadStructureMissing means a required structured parameter was absent from the request the client sent. In practice it points at a client that omitted a mandatory field - a filter, a request header member, a settings structure - rather than at anything wrong on the server.	In diagnostic order — work down, do not skip ahead. Capture the failing request on the wire Wireshark decodes OPC UA binary requests field by field.
BadTooManyArguments	BadTooManyArguments means your Call sent more inputArguments than the method's InputArguments property declares. The argument list is positional and fixed-length, so extras are an error rather than being ignored.	In diagnostic order — work down, do not skip ahead. Read InputArguments and count it Browse the Method node's HasProperty reference to InputArguments and read the Argument array.
BadTypeMismatch	BadTypeMismatch means the Variant you wrote does not match the target node's declared DataType, ValueRank or ArrayDimensions. Read the node's DataType, ValueRank and ArrayDimensions attributes first, then build the Variant to match exactly - Int16 is not Int32, and a one-element array is not a scalar.	In diagnostic order — work down, do not skip ahead. Read the node's DataType, ValueRank and ArrayDimensions attributes Do not assume from the DisplayName or from what the value looks like.

Read, write and access (9 codes)

Status code	What it means	First thing to do
BadAggregateNotSupported	BadAggregateNotSupported means the server does historical access but not the aggregate you asked for. Part 13 defines a long list; servers implement a subset.	In diagnostic order — work down, do not skip ahead. Browse the AggregateFunctions folder Under Server, ServerCapabilities, AggregateFunctions the server lists every aggregate it actually implements, as nodes with BrowseNames you can read.
BadInvalidState	BadInvalidState means the request was well formed but the target is not in a state that permits it right now. It is normally a state machine, a file handle or a session lifecycle refusing an out-of-order operation, so read the current state and drive the transition first.	In diagnostic order — work down, do not skip ahead. Read the current state before acting Browse the owning object for its state machine and read CurrentState.

Status code	What it means	First thing to do
BadNotImplemented	BadNotImplemented means the code path for what you asked simply does not exist in this server build. Treat it as a profile and facet question rather than a defect: read ServerProfileArray and the ServerCapabilities short codes to see what the product genuinely claims, and design to that.	In diagnostic order — work down, do not skip ahead. Establish what the server claims Read ServerProfileArray and Server/ServerCapabilities, including the short capability codes.
BadNotReadable	BadNotReadable means the node's AccessLevel or UserAccessLevel withholds the CurrentRead bit, so neither Read nor a data-change subscription will return a value. Check UserAccessLevel against AccessLevel: a write-only command node is denied by design, an anonymous session is denied by policy.	In diagnostic order — work down, do not skip ahead. Read AccessLevel and UserAccessLevel. If AccessLevel lacks CurrentRead the node is write-only in the model and no credential will change that.
BadNotSupported	BadNotSupported means the server understood the request and declines the specific option in it, rather than the whole service. It is usually a conformance gap, so read ServerProfileArray and the ServerCapabilities codes to see what the product claims before treating it as a bug.	In diagnostic order — work down, do not skip ahead. Isolate the option that causes it. Strip the request back to its simplest legal form and add one option at a time.
BadNotWritable	BadNotWritable means the node's AccessLevel or UserAccessLevel does not grant write permission. Read both attributes: AccessLevel is the node's capability, UserAccessLevel is what your identity gets.	In diagnostic order — work down, do not skip ahead. Read AccessLevel and UserAccessLevel on the node. Both are Byte bit masks carrying CurrentRead and CurrentWrite among others.
BadNothingToDo	BadNothingToDo means you sent a service request with an empty operation array: a Read with no nodes, a Write with no values, a CreateMonitoredItems with no items. The server refuses rather than returning an empty success, so fix the code that built the empty list.	In diagnostic order — work down, do not skip ahead. Guard every service call on a non-empty list. Check the length of the operation array immediately before sending, and skip the call when it is zero.
BadOutOfRange	BadOutOfRange means the value is the right type but outside the permitted range: beyond the node's EURange or InstrumentRange, outside an enumeration's defined values, or outside the legal range of a service parameter. Read the node's range properties and clamp before writing.	In diagnostic order — work down, do not skip ahead. Read the node's range properties. Browse HasProperty from the variable and read EURange and, where present, InstrumentRange.
BadWriteNotSupported	BadWriteNotSupported means the server accepts writes to that node but not the specific DataValue you sent - typically because you included a StatusCode or SourceTimestamp. Most servers only accept the Value field on Write.	In diagnostic order — work down, do not skip ahead. Send only the Value field in the DataValue. Build a fresh DataValue containing the Variant and nothing else.

Server state and capacity (12 codes)

Status code	What it means	First thing to do
BadDecodingError	BadDecodingError means the receiving side could not parse the bytes it was given. It is almost always a type mismatch between what the sender encoded and what the receiver expected: a stale structure definition, an unknown encoding id, or a truncated message.	In diagnostic order — work down, do not skip ahead. Narrow it to one node or one service. Read nodes individually rather than in a batch until you find the one that fails.
BadInternalError	BadInternalError means the server hit a fault of its own while handling your request. It is a server-side defect or misconfiguration, not a client error.	In diagnostic order — work down, do not skip ahead. Reduce the request to a minimal reproducer. Cut nodes, parameters and options until you have the smallest request that still fails.
BadOutOfMemory	BadOutOfMemory means the server could not allocate what your request needed. On embedded servers it is usually the size of what you asked for; on software servers it is usually accumulated load from too many subscriptions, monitored items or sessions.	In diagnostic order — work down, do not skip ahead. Shrink the failing request first. Halve the batch: fewer nodes per read, fewer monitored items per CreateMonitoredItems call, a shorter history window, a browse with MaxReferencesPerNode set and BrowseNext for the rest.

Status code	What it means	First thing to do
BadResourceUnavailable	BadResourceUnavailable means the server needs something outside itself to answer you and cannot reach it: a database, a file, a device driver, a licence, a worker thread. Look at what the server depends on rather than at your request, which is usually correct.	In diagnostic order — work down, do not skip ahead. Identify which resource the failing branch depends on Map the failing NodeIds to their source: a device, a database, a file store or an external service.
BadServerHalted	BadServerHalted means the server application is present and answering, but is not in a state where it will serve requests - stopped, failed, suspended or held for maintenance. Read Server_ServerStatus to see the actual ServerState and ShutdownReason instead of reconnecting in a loop against a server that is deliberately halted.	In diagnostic order — work down, do not skip ahead. Read the server status before anything else Read Server_ServerStatus and inspect State, StartTime, SecondsTillShutdown and ShutdownReason.
BadServerNotConnected	BadServerNotConnected means the operation was attempted with no live connection to the server behind it. On a client it usually means code called a service before connecting or after disconnecting.	In diagnostic order — work down, do not skip ahead. Check the client's own connection state first Read the state from the SDK before blaming the network.
BadServerTooBusy	BadServerTooBusy is the server refusing a service call at the application layer because it is at capacity. Unlike BadTcpServerTooBusy, which refuses the transport handshake, this arrives on an established session.	In diagnostic order — work down, do not skip ahead. Honour the refusal Wait and retry with an increasing, randomised delay.
BadServiceUnsupported	BadServiceUnsupported means the server does not implement that OPC UA service at all. It is a conformance statement, not a fault: read ServerProfileArray and the ServerCapabilities short codes to see which service sets the product actually claims, then design around what is there.	In diagnostic order — work down, do not skip ahead. Read ServerProfileArray before writing any client code Browse the Server object and read ServerProfileArray.
BadShutdown	BadShutdown means the operation was abandoned because an application is going down - usually the server, sometimes your own client. It is normally a planned event, not a fault.	In diagnostic order — work down, do not skip ahead. Read the shutdown fields while you still can Subscribe to or read Server_ServerStatus - State, SecondsTillShutdown and ShutdownReason.
BadTooManyMatches	BadTooManyMatches means your request matched more targets than the server will return. It comes from under-specified browse paths and broad queries: an empty target name, a path element that matches every child, or a filter with nothing to narrow it.	In diagnostic order — work down, do not skip ahead. Make every path element fully qualified Give each RelativePathElement an explicit target QualifiedName with its namespace index, an explicit ReferenceType, and an explicit direction.
BadTooManyOperations	BadTooManyOperations means one request carried more operations than the server allows for that service. Servers publish per-service caps such as MaxNodesPerRead, MaxNodesPerWrite, MaxNodesPerBrowse and MaxMonitoredItemsPerCall in ServerCapabilities.	In diagnostic order — work down, do not skip ahead. Read the operation limits at connect Browse to ServerCapabilities under the Server object and read the OperationLimits properties: MaxNodesPerRead, MaxNodesPerWrite, MaxNodesPerBrowse, MaxNodesPerTranslateBrowsePathsToNodeIds, MaxNodesPerMethodCall, MaxNodesPerHistoryReadData, MaxMonitoredItemsPerCall.
BadUnexpectedError	BadUnexpectedError is the server saying something went wrong that it could not classify. It carries no diagnostic value on its own, so the work is entirely in correlating it with the server log and reducing the trigger to a minimal reproducible request.	In diagnostic order — work down, do not skip ahead. Get the server log for the exact timestamp This code is not actionable from the client side.

Field, device and data quality (7 codes)

Status code	What it means	First thing to do
BadDataUnavaila ble	BadDataUnavailable means the historian expected to have data for your time range but cannot reach it right now: an offline archive volume, a detached cold store, or a backend the history layer cannot query. It is a temporary infrastructure fault, unlike GoodNoData which says nothing was ever recorded.	In diagnostic order — work down, do not skip ahead. Narrow the window to find the boundary Binary-search the start time.
BadDeviceFailure	BadDeviceFailure means the server reached the device and the device reported a fault. Communication is working; the equipment is not.	In diagnostic order — work down, do not skip ahead. Confirm the fault is at the device, not at the link If other tags from the same device are Good, the communication path is fine and the device is reporting a genuine fault.
BadNoCommunication	BadNoCommunication means the OPC UA server is healthy and talking to you, but it cannot reach the device behind the tag and has no cached value to offer. The session is not the problem.	In diagnostic order — work down, do not skip ahead. Check the scope of the failure Read a few Server object nodes such as Server_ServerStatus_State and Server_ServerStatus_CurrentTime.
BadOutOfService	BadOutOfService means the source has been deliberately taken out of service, typically for maintenance or because a function block was disabled. It is an intentional state, not a fault.	In diagnostic order — work down, do not skip ahead. Establish whether this was intentional Check maintenance records, the operator log and any change window in progress.
BadSensorFailure	BadSensorFailure means the device is fine but the sensor feeding the value has failed - a broken thermocouple, an open loop, a signal outside the valid input range. The value is not usable.	In diagnostic order — work down, do not skip ahead. Establish the scope If exactly one measurement is bad, look at that sensor and its wiring.
BadWaitingForInitialData	BadWaitingForInitialData means the monitored item exists but the server has not yet obtained a first value from the source. It is a normal transient state at subscription creation.	In diagnostic order — work down, do not skip ahead. Wait before reacting Give the item at least a couple of sampling intervals plus one publishing interval before treating this as a problem.
UncertainNoCommunicationLastUsableValue	UncertainNoCommunicationLastUsableValue is not a failure of your client. It names the cause explicitly: communication with the source is down and this is the last usable reading.	In diagnostic order — work down, do not skip ahead. Confirm the scope of the loss List which tags degraded and at what timestamp.

History and aggregates (3 codes)

Status code	What it means	First thing to do
BadAggregateInvalidInputs	BadAggregateInvalidInputs means the server understood your aggregate but could not compute it for that interval: the underlying raw values were Bad, too sparse, or of a type the aggregate cannot handle. Check the raw data for the same window first, then the aggregate configuration's TreatUncertainAsBad and PercentDataGood settings.	In diagnostic order — work down, do not skip ahead. Read the raw values for one failing interval Take a single interval that returned the code and do a raw read of exactly that window with returnBounds false.
BadHistoryOperationInvalid	BadHistoryOperationInvalid means the server supports historical access but rejected the details structure you sent: an inconsistent time range, a mutually exclusive combination of start time, end time and numValuesPerNode, or a details type that does not match the operation. Fix the request parameters, not the server configuration.	In diagnostic order — work down, do not skip ahead. Log the exact details structure you sent Print startTime, endTime, numValuesPerNode, returnBounds and the details type name before the call.
BadHistoryOperationUnsupported	BadHistoryOperationUnsupported means the server accepted your HistoryRead or HistoryUpdate call but does not implement that particular history operation. Check the Server object's HistoryServerCapabilities and ServerProfileArray for the HistoricalRawData2022 or AggregateHistorical2022 facet before assuming the archive exists.	In diagnostic order — work down, do not skip ahead. Read the server's declared history capability Browse Server, ServerCapabilities, HistoryServerCapabilities and read AccessHistoryDataCapability and AccessHistoryEventsCapability.

Uncertain — the value is suspect (5 codes)

Status code	What it means	First thing to do
UncertainEngineeringUnitsExceeded	UncertainEngineeringUnitsExceeded is not an error. The value is outside the EURange declared on the node, so the server hands it over with a warning rather than clamping or rejecting it.	In diagnostic order — work down, do not skip ahead. Read EURange and InstrumentRange before anything else. Read the EURange property on the node and, where present, the instrument range property.
UncertainInitialValue	UncertainInitialValue is not an error. It is the server saying this is a starting value it has not yet confirmed from the source.	In diagnostic order — work down, do not skip ahead. Expect it and hold the item pending. Design the client so the initial notification is treated as state rather than as data.
UncertainLastUsableValue	UncertainLastUsableValue is not an error. The server is deliberately telling you the number is the last good reading and is now stale.	In diagnostic order — work down, do not skip ahead. Read SourceTimestamp and compute the age. The status says the value is stale; SourceTimestamp says how stale.
UncertainSensorNotAccurate	UncertainSensorNotAccurate is not an error. The reading is real but the instrument is outside the range or conditions where it can be trusted - saturated, out of calibration, or past a diagnostic threshold.	In diagnostic order — work down, do not skip ahead. Read the value together with EURange and the engineering unit. Compare the reading against the EURange property and the EUInformation on the node.
UncertainSubstituteValue	UncertainSubstituteValue is not an error. Somebody configured a fallback value and the server is now serving it because the real one is unavailable.	In diagnostic order — work down, do not skip ahead. Treat the value as absent, not as data. This is the strongest rule in this family.

Good — not an error, but it changes your logic (5 codes)

Status code	What it means	First thing to do
GoodClamped	GoodClamped is not an error - but the server changed your value. A write outside a declared limit was applied at the limit instead of being rejected.	In diagnostic order — work down, do not skip ahead. Always read back after a write and compare. Treat a write as complete only when a read of the same node returns the value you intended.
GoodCompletesAsynchronously	GoodCompletesAsynchronously is not an error and not a completion. The server accepted your request and will finish it later, so the output arguments you got back are not final.	In diagnostic order — work down, do not skip ahead. Branch on the status, not on the absence of an error. Check for this specific status explicitly.
GoodMoreData	GoodMoreData is not an error. The call succeeded and returned as much as fit; there is more.	In diagnostic order — work down, do not skip ahead. Loop on the continuation point. After each HistoryRead, check for a continuation point on each node's result.
GoodNoData	GoodNoData is not an error. The history query worked and the answer is that nothing was recorded in that range.	In diagnostic order — work down, do not skip ahead. Confirm the query window in UTC. Print the start and end timestamps you actually sent, in UTC, and confirm the order.
GoodOverload	GoodOverload is not an error, and that is the problem: it reports a Good severity while telling you samples were lost. A monitored item queue overflowed and the server dropped values.	In diagnostic order — work down, do not skip ahead. Instrument the overflow indication and alarm on it. The overflow condition is carried as a flag inside the StatusCode rather than as a separate code, so a consumer that only compares against Good will never see it.

Where to go next

Every code in this document has a full page at opcuaconnect.com/en/errors/ — what you actually see, why it happens, how to fix it, what it looks like in the log, how to confirm the fix, and how to stop it happening again.

If the fault turned out to be commissioning rather than code, the **OPC UA Server Acceptance Test Pack** is the checklist that catches it before sign-off. If it turned out your OEE numbers were built on the wrong nodes, the **Machine Data to OEE Mapping Kit** is the one that fixes that. Both are free at opcuaconnect.com/get/.

opcuaconnect.com is an educational reference
and is not affiliated with the OPC Foundation. Everything on it is free to read and free to cite.