# Brainy Data Support Policies

## Contents

1. Introduction ..... 2
2. Why do we charge for maintenance? ..... 2
3. How do we charge for maintenance and support? ..... 2
4. Why do we charge a support fee for each technical contact? ..... 3
5. How do we cater for developers without a maintenance subscription? ..... 3
6. Why use a support request form? ..... 3
7. What information must be provided in the description field? ..... 3
8. Why do you have to provide example code? ..... 4

## 1. Introduction

This document details our policies regarding our maintenance and support subscriptions. It will answer questions such as: why and how we charge for maintenance and support; why we insist on the use of our support request form; how the request form should be used; and why we insist that bug reports are accompanied by working examples.

We hope that this document clarifies our policy positions and addresses your questions. If you have further queries or comments regarding our policies don't hesitate to contact us.

## 2. Why do we charge for maintenance?

Our external components provide highly flexible programable interfaces for adding important functionality to Omnis applications. The flexibility of our external components allows developers to use them in diverse software markets and consequently they can be developed in diverse ways. All our external components are provided with working examples that demonstrate their features and their intended use. However, we cannot anticipate all possible uses and developers will often find themselves operating outside the scope of our example applications and sometimes our external components may not perform as expected.

Some may feel these are software errors that should be corrected free of charge. However, we feel they are the result of technical uncertainties through complexity that is exacerbated through the programable nature of our software and for this we cannot provide a guaranty. Often, the circumstances surrounding such issues are unique to the application. In many cases, when issues arise for which we alter our components we do it to enable developer's to use them in a preferred way, not necessarily in a way that was part of the original design. Furthermore, our less visible work, such as compatibility with multiple major versions of Omnis Studio as well as the support of different platforms and different versions of operating systems, incurs substantial costs.

All developers who subscribe to our maintenance program benefit from this ongoing work, which is solely financed by the maintenance subscription fees.

## 3. How do we charge for maintenance and support?

When our external components are first licensed, we include a twelve months free maintenance subscription. After the initial free period we charge an annual maintenance fee based on a percentage value of the license fee.

Please note that our maintenance subscription does not include a bug reporting facility nor does it include any technical dialogue. We offer a separate technical support subscription (charged per technical contact) which extends the maintenance subscription to include these and other benefits. Please refer to our licensing document for further details regarding the full range of benefits.

Our maintenance and support structure is not that different to that of other software companies. It is common practice for companies to release regular major software upgrades for which upgrade fees apply. If a developer requires software corrections that have been applied in a new major version,

the developer is subject to upgrade fees unless a paid subscription was entered into that includes free upgrades. At Brainy Data we merely differ in that we try to be very responsive to developer's needs and include, besides software corrections, new feature enhancements in minor as well as major version releases. Thus the typical upgrade model of only charging for major versions does not fit our model.

## 4. Why do we charge a support fee for each technical contact?

We have found that companies with multiple developers working on applications involving our external components, often generate a greater number of support requests than is normally the case. Furthermore, communicating with multiple individuals results in additional work such as duplicate inauguration and technical queries, which places additional strain on our resources. We therefore charge a support fee per registered contact.

## 5. How do we cater for developers without a maintenance subscription?

Our demo download pages and public support pages provide free of charge the latest demo software, product documentation, technical notes, release notes and examples. Our free material creates an opportunity for developers who chose not to continue their maintenance subscription to try out and read up about the latest changes and enhancements prior to paying for an upgrade.

Though we are happy to answer some pre-sales questions such as "can the software do this or that?" or "what is included in the license fee?", we are unable to answer technical questions such as "how does this work?" or "how do we do that?" or "I have this problem, can this be fixed in the next version?". Such queries and requests come under the remit of technical support and require a technical support subscription without which we cannot enter into technical dialogue as it takes away valuable resources from developers who have paid for this level of support.

## 6. Why use a support request form?

In order to deal effectively and efficiently with all support requests we require accurate detailed information for each case. As well as ensuring that all vital information is provided, the support request form is linked to our automated case tracking system which increases our efficiency even further. However, our system will only work if all developers use our form. Consequently we will insist on the form being used as we are no longer able to effectively track support cases without using the online form. Queries initiated via email are not compatible with our system. However, after a new case is entered in our system, subsequent dialogue will take place via direct email.

## 7. What information must be provided in the description field?

For us to understand and resolve an issue efficiently, we require appropriate detailed descriptions. If the description is not written correctly we may miss-interpret or fail to reproduce the problem. One successful approach is to break down the description as follows:

- Background info: this can be anything that may be relevant, i.e. what you are trying to do and why you are doing it this way. The latter may help us interpret the former correctly.
- Steps: the detailed steps to reproduce the problem using our example library or a modified example library which should be attached to the form using the “Upload File” box.
- Expected result: a description of what you expected to happen.
- Actual result: a description of what actually happened.

Please always provide all relevant details in this field. Do not refer to emails or attached documents as an alternative to a description. Descriptions such as “details will follow in an email” or “details are attached” are not compatible with our system.

## 8. Why do you have to provide example code?

When it comes to supporting plug-ins for development tools such as Omnis, there is a very thin line between what should be considered consultancy work and what can be covered by the technical support subscription.

When developers encounter a problem, the first action should be to see if it can be replicated in the examples which will have two outcomes:

a. The problem can be replicated in which case it is a simple task to provide us with step-by-step instructions when reporting the problem.

b. The problem cannot be easily replicated so it is probably something intricate in the developer’s library that is causing the problem to surface.

   In such a case, we cannot accept a video or screen shoot instead of a working example, as such media merely displays the symptoms of the problem but does not demonstrate the cause.

   Furthermore, we cannot accept a developer’s applications as this would require us to install, learn and debug an unfamiliar application. This may not be an issue if the library is very basic, but in our experience developer’s libraries are rarely that. Thus, installing and debugging a developer’s own library is not sustainable within the remit of our support program.

   We recommend that the circumstances are identified and re-created within our own examples. This approach has the benefit that it filters out cases of incorrect coding. If this approach proofs too difficult, we may consider installing and debugging a developer’s application, but such work will be subject to consultancy fees.

**Document History**

| | |
|---|---|
| 01 Jan 2017: | changes to maintenance and support subscription benefits |
| 13 May 2015: | grammatical corrections and numbered sections |
| 12 Dec 2014: | further changes to various descriptions |
| 04 Dec 2014: | divergence of maintenance and technical support programs |
| 09 May 2013: | first publication |